PROCEEDINGS

of the 23rd International Society for Music Information Retrieval Conference

ISMIR 2022

Bengaluru, India

(Hybrid Conference)

December 04-08, 2022

Editors

Preeti Rao, Hema Murthy, Ajay Srinivasamurthy,
Rachel Bittner, Rafael Caro Repetto, Masataka Goto,
Xavier Serra, Marius Miron

ISMIR 2022 was organized by the International Society for Music Information Retrieval, Women in Music Information Retrieval (WiMIR) and a diverse international committee of organizers.

Website: https://ismir2022.ismir.net

ISMIR 2022 logo designed by Mrinali Kamath and graphics designed by Vivek Vasudev

Edited by:
Preeti Rao *(Indian Institute of Technology Bombay, Mumbai, India)*
Hema Murthy *(Indian Institute of Technology Madras, Chennai, India)*
Ajay Srinivasamurthy *(Amazon Alexa, Bengaluru, India)*
Rachel Bittner *(Spotify, Paris, France)*
Rafael Caro Repetto *(Institute of Ethnomusicology, Kunstuniversität Graz, Austria)*
Masataka Goto *(National Institute of Advanced Industrial Science and Technology, Tsukuba, Japan)*
Xavier Serra *(Pompeu Fabra University, Barcelona, Spain)*
Marius Miron *(Pompeu Fabra University, Barcelona, Spain)*

ISBN: 978-1-7327299-2-6

Title: Proceedings of the 23rd International Society for Music Information Retrieval Conference, Bengaluru, India, Dec 4-8, 2022.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee, provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page.

© 2022 International Society for Music Information Retrieval

ISMIR 2022 Organizers

ISMIR

ISMIR 2022 SPONSORS

Platinum Sponsors

Spotify

moises

Gold Sponsors

Indian Speech Communication Association Registered 2019 IndSCA Bangalore

Adobe

deezer

utopia

SiriusXM pandora

smule

YAMAHA

chordify

TikTok

Silver Sponsors

steinberg

soundwide

Supporters

ACRCloud

cochl.

School of Information Sciences
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

WIMIR SPONSORS

Patrons

Contributors

Supporters

Organizing Team

Conference Chairs

General Chairs
Preeti Rao (Indian Institute of Technology Bombay, Mumbai, India)
Hema Murthy (Indian Institute of Technology Madras, Chennai, India)
Ajay Srinivasamurthy (Amazon Alexa, Bengaluru, India)

Scientific Program
Rachel Bittner (Spotify, Paris, France)
Rafael Caro Repetto (Institute for Ethnomusicology, Kunstuniversität Graz, Austria)
Masataka Goto (National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan)
Xavier Serra (Universitat Pompeu Fabra, Barcelona, Spain)

Tutorial
Vipul Arora (Indian Institute of Technology Kanpur, Kanpur, India)
Keunwoo Choi (Gaudio Lab, Seoul, South Korea)
Sri Rama Murty K. (Indian Institute of Technology Hyderabad, Hyderabad, India)

Late-Breaking / Demo
Sanjeel Parekh (Telecom, Paris, France)
Siddharth Gururani (NVIDIA, Santa Clara, USA)

Music Program
Kaustuv Kanti Ganguli (Zayed University, Abu Dhabi, UAE)
Sumitra Ranganathan (Krea University, Sri City, AP, India)

Publications
Marius Miron (Universitat Pompeu Fabra, Barcelona, Spain)

Diversity and Inclusion
Ranjani H G (Ericsson R&D, Bengaluru, India)
Xiao Hu (University of Hong Kong, Hong Kong, China)
Makarand Velankar (Cummins College, Pune, India)

Sponsorship/Industry
Joe Cheri Ross (LinkedIn, Bengaluru, India)
Pratibha Moogi (IBM Research, Bengaluru, India)
Rishikesh Daoo (Dolby Laboratories, San Francisco, USA)

Local Organization, Technology, and Virtual Platform
Chandra Sekhar Seelamantula (Indian Institute of Science, Bengaluru, India)
Swapnil Gupta (Bengaluru, India)
Sharath Adavanne (Freshworks Inc, Bengaluru, India)

Local Initiatives/Outreach Chair
Vinoo Alluri (International Institute of Information Technology, Hyderabad, India)

Website
Chitralekha Gupta (National University of Singapore, Singapore)
Ashvala Vinay (Georgia Institute of Technology, Atlanta, USA)

Social Media/Publicity
Albin Andrew Correya (Moodagent, Copenhagen, Denmark)
Vishnu Srinivasa Murthy (Vellore Institute of Technology, Vellore, India)

Social Program
Oriol Nieto (Adobe, San Francisco, USA)
Kaustuv Kanti Ganguli (Zayed University, Abu Dhabi, UAE)

Volunteers

Abhut Vipin Bhardwaj
Adithi Shankar Sivasankar
Dilip Harish
Elena Georgieva
Gowriprasad R
Jyoti Narang
Jom Koriakose
Kavya Ranjan Saxena
Milind Barman
Mohit Jain
Nithya Shikarpur
Parampreet Singh
Prashant Mishra
Pratap Kygonahalli
Rajesh R
Recep Oğuz Araz
Rhythm Jain
Rohit M A
Sangsaptak Pal
Sharvaree Sinkar
Shreyas Nadkarni
Suraj Jaiswal
Thishyan Raj T
Tirna Ghosh
Utkarsh Gupta
Vayyavuru Venkatesh
Venkatakrishnan V K
Vibha Masti
Vinod Subramanian
Vishruth Veerendranath
Yuxi Xiao

Program Committee

Meta-Reviewers

Alexander Lerch, Georgia Institute of Technology
Alexander Schindler, Austrian Institute of Technology
Andre Holzapfel, KTH Royal Institute of Technology
Andreas Arzt, Apple
Andrew McLeod, École Polytechnique Fédérale De Lausanne
Audrey Laplante, Université de Montréal
Bob L. T. Sturm, KTH Royal Institute of Technology
Brian McFee, New York University
Bryan Pardo, Northwestern University
Cheng-i Wang, Smule, Inc.
Chris Donahue, Stanford University
Christof Weiss, International Audio Laboratories Erlangen
Claire Arthur, Georgia Institute of Technology
Cory McKay, Marianopolis College
Cynthia C. S. Liem, Delft University of Technology
Dmitry Bogdanov, Universitat Pompeu Fabra
Douglas Turnbull, Ithaca College
Emilia Gomez, Universitat Pompeu Fabra
Emmanouil Benetos, Queen Mary University of London
Eva Zangerle, University of Innsbruck
Fabien Gouyon, Pandora/SiriusXM
Florence Leve, Université de Picardie Jules Verne - Lab. MIS - Algomus
Gaël Richard, LTCI, Paris Tech
Geoffroy Peeters, LTCI - Télécom Paris, IP Paris
George Fazekas, Queen Mary University of London
George Tzanetakis, University of Victoria
Gerhard Widmer, Johannes Kepler University
Guangyu Xia, New York University Shanghai
Jin Ha Lee, University of Washington
Johan Pauwels, Queen Mary University of London
John Ashley Burgoyne, University of Amsterdam
Jordi Pons, Dolby Laboratories
Juan J. Bosch, Spotify
Juhan Nam, Korea Advanced Institute of Science & Technology
Justin Salamon, Adobe Research
Kahyun Choi, Indiana University of Bloomington
Katherine M. Kinnaird, Smith College
Kazuyoshi Yoshii, Kyoto University
Li Su, Academia Sinica
Magdalena Fuentes, New York University
Mark B Sandler, Queen Mary University of London
Matt McVicar, Apple
Meinard Müller, International Audio Laboratories Erlangen
Michael Mandel, Brooklyn College, CUNY
Mitsunori Ogihara, University of Miami
Mohamed Sordo, Pandora
Nicholas J. Bryan, Adobe Research
Olivier Lartillot, RITMO, University of Oslo
Oriol Nieto, Pandora
Ozgur Izmirli, Connecticut College
Peter Knees, Vienna University of Technology
Peter Van Kranenburg, Utrecht University; Meertens Institute
Romain Hennequin, Deezer Research
Sankalp Gulati, Orbi Health Pvt Ltd
Sebastian Ewert, Spotify

Sebastian Stober, Otto von Guericke University
Simon Dixon, Queen Mary University of London
Stephen Downie, University of Illinois
Tom Collins, University of York; MAIA, Inc.
Vinoo Alluri, IIIT - Hyderabad
Xiao Hu, The University of Hong Kong; Shenzhen Institute of Research and Innovation
Ye Wang, National University of Singapore
Yi-Hsuan Yang, Academia Sinica
Zhiyao Duan, University of Rochester

Reviewers

Adam Roberts, Google Brain
Adrián Barahona-Ríos, University of York
Akira Maezawa, Yamaha Corporation
Alessio Degani, UNIBS
Alexandra Uitdenbogerd, RMIT
Alia Morsi, Universitat Pompeu Fabra
Alice Cohen-Hadria, IRCAM
Amanda E Krause, James Cook University
Amelie Anglade, Freelance MIR Consulting
Amruta Vidwans, Georgia Institute of Technology
Andres Ferraro, Universitat Pompeu Fabra
Aneesh Vartakavi, Gracenote
Angelica Ribeiro, Universidade de São Paulo
Anna Jordanous, School of Computing, University of Kent
Anna Xambó, De Montfort University
Antoine Liutkus, INRIA
Antonio Pertusa, University of Alicante
Ashis Pati, Georgia Institute of Technology
Axel Roebel, IRCAM
Ben Hayes, Queen Mary University of London
Benjamin Martin, Deezer
Benno Weck, Universitat Pompeu Fabra
Berit Janssen, Utrecht University
Bochen Li, University of Rochester
Bo-Yu Chen, National Taiwan University; Academia Sinica
Bruna Wundervald, Maynooth University
Bruno Di Giorgi, Apple
Bryony Buck, University of Nottingham
Camille Noufi, Stanford University
Carlos Guedes, New York University Abu Dhabi
Charles Inskip, University College London
Chenyu Gao, Northwestern Polytechnical University
Chia-Hao Chung, National Taiwan University
Chih-Wei Wu, Netflix, Inc.
Ching-Yu Chiu, Academia Sinica
Chitralekha Gupta, National University of Singapore
Chris Hubbles, NorthEast Document Conservation Center
Christian J. Steinmetz, Queen Mary University of London
Christoph Finkensiep, École Polytechnique Fédérale de Lausanne
Christopher A Ick, New York University
Chung-Che Wang, National Taiwan University
Cong Jin, Communication University of China
Cyrus Vahidi, Queen Mary University of London
Dan Brown, University of Waterloo
Daniel Harasim, École Polytechnique Fédérale de Lausanne
Daniel Stoller, Spotify
Daniel Wolff, Utopia Music
Dasaem Jeong, Sogang University
David Lewis, University of Oxford
David Martins de Matos, INESC-ID Lisboa
David Rizo, Universidad de Alicante
David M. Weigl, University of Music and Performing Arts Vienna
Delia Fano Yela, European Commission
Diego Furtado Silva, Universidade Federal de São Carlos
Dogac Basaran, Audible Magic
Dominique Fourer, IBISC - Univ. Evry/Paris-Saclay
Eleanor Selfridge Field, Stanford University
Eli Stine, Oberlin Conservatory
Emilia Parada-Cabaleiro, Johannes Kepler University
Emir Demirel, Spotify
Fabian-Robert Stöter, Audioshake.ai
Fabio Morreale, University of Auckland
Fanjie Li, University of Hong Kong
Federico Simonetta, Università di Milano
Filip Korzeniowski, Moises.ai
Florian Thalmann, Kyoto University
François G Germain, Mitsubishi Electric Research Laboratories
Frank Kurth, Fraunhofer FKIE
Frederic Font, Universitat Pompeu Fabra
Furkan Yesiler, ByteDance
Gabriel Meseguer Brocal, IRCAM
Gaëtan Hadjeres, Sony Computer Science Laboratories
Genís Plaja-Roglans, Universitat Pompeu Fabra
Geoffray Bonnin, Lorriane Research laboratory in Computer Science and its Applications
George Sioros, University of Plymouth
Gerard Roma, University of Huddersfield
Gianluca Micchi, ByteDance
Gilberto Bernardes, INESC TEC; University of Porto
Gissel Velarde, Consultant
Gordon Wichern, Mitsubishi Electric Research Laboratories
Guillaume Doras, IRCAM
Hao-Wen Dong, UC San Diego
Helena Cuesta, Time Machine Capital 2 / DAACI
Hendrik Schreiber, International Audio Laboratories Erlangen

Hyeongi Moon, Gaudio Lab.
Hyeong-Seok Choi, Seoul National University
Igor Vatolkin, TU Dortmund
Il-Young Jeong, Cochlear.ai
Iran R Roman, New York University
Jaehun Kim, Delft University of Technology
Jake Drysdale, Birmingham City University
Jakob Abeßer, Fraunhofer IDMT
Jan Hajič, jr., Charles University
Jan Van Balen, Spotify
Jeremy Tzi Dong Ng, The University of Hong Kong
Jianyu Fan, Simon Fraser University
Jiawen Huang, Queen Mary University of London
Jonathan Driedger, Chordify
Jong Wook Kim, OpenAI
Jongpil Lee, Neutune
Jorge Calvo-Zaragoza, University of Alicante
Jorge Herrera, Shazam
Jose J. Valero-Mas, University of Alicante
Juan P Bello, New York University
Juan S. Gómez-Cañón, Universitat Pompeu Fabra
Ju-Chiang Wang, ByteDance
Julio Carabias, University of Jaen
Junyan Jiang, Carnegie Mellon University
Katerina Kosta, ByteDance
Katharina Hoedt, Johannes Kepler University Linz
Kaustuv Kanti Ganguli, Zayed University
Ke Chen, University of California San Diego
Kento Watanabe, National Institute of Advanced Industrial Science and Technology (AIST)
Keunwoo Choi, Gaudio Lab
Klaus Frieler, University of Music Franz Lizt Weimar
Kosetsu Tsukuda, National Institute of Advanced Industrial Science and Technology (AIST)
Kosmas Kritsis, Athena Research Center
Kyle J Worrall, University of York
Kyungyun Lee, Gaudio Lab
Lam Pham, Austrian Institute of Technology
Laura Cros Vila, KTH Royal Institute of Technology
Leandro Balby Marinho, Federal University of Campina Grande
Lele Liu, Queen Mary University of London
Lorenzo Porcaro, Universitat Pompeu Fabra
Louis Bigo, Université de Lille
Lucas S Maia, Federal University of Rio de Janeiro
Lucas Nascimento Ferreira, University of Alberta
Luis Joglar-Ongay, SonoSuite
Luiz W P Biscainho, University of Rio de Janeiro
Luwei Yang, Alibaba Group
Manuel Moussallam, Deezer
Marcelo Caetano, National Center for Scientific Research
Marco A Martinez Ramirez, Sony Group Corporation
Maria Navarro, University of Salamanca
Mark Cartwright, New Jersey Institute of Technology
Mark R H Gotham, Technische Universität Dortmund
Mark Levy, Apple
Marko Stamenovic, Bose.
Marko Tkalcic, Free University of Bozen Bolzano
Martín Rocamora, Universidad de la República
Matevž Pesek, University of Ljubljana
Mathieu Lagrange, Ecole Centrale de Nantes
Matija Marolt, University of Ljubljana
Matthew Davies, Pandora/SiriusXM
Maximilian Mayerl, University of Innsbruck
Maximilian Schmitt, University of Augsburg
Meijun Liu, Fudan University
Maximos Kaliakatsos-Papakostas, Athena Research and Innovation Centre
Michael Krause, International Audio Laboratories Erlangen
Michael Taenzer, Fraunhofer IDMT
Michael Vötter, Universität Innsbruck
Minz Won, ByteDance
Nathaniel Condit-Schultz, Georgia Institute of Technology
Nazif Can Tamer, Universitat Pompeu Fabra
Nick Gang, Apple
Nicola Montecchio, Spotify
Nicolas Bouillot, Société des Arts Technologiques Montréal
Olga Slizovskaia, Voctro Labs S.L.
Ondřej Cífka, Université de Montpellier
Pablo Alonso-Jiménez, Universitat Pompeu Fabra
Pablo Zinemanas, Universitat Pompeu Fabra
Paolo Bientinesi, Umeå Universitet
Patricio López-Serrano, Zplane Development
Patricio Ovalle, Independent researcher
Pedro D. Pestana, Catholic University
Pedro Pereira Sarmento, Centre for Digital Music
Petri Toiviainen, University of Jyväskylä
Peyman Heydarian, University of Waikato
Philip Tovstogan, Universitat Pompeu Fabra
Phillip B Kirlin, Rhodes College
Pierre Laffitte, Moodagent
Piyush Papreja, Arizona State University
Qiuqiang Kong, ByteDance
Radha Manisha Kopparti, City University of London
Ranjani H G, Ericsson India
Recep Oğuz Araz, Universitat Pompeu Fabra
Reinier de Valk, Moodagent
Remi Mignot, IRCAM
Robert Lieck, École Polytechnique Fédérale De Lausanne
Robert Rowe, New York University
Roger Dannenberg, School of Computer Science, Carnegie Mellon University
Roman B. Gebhardt, Cyanite; TU Berlin
Rongfeng Li, Beijing University of Posts and Telecommunications
Samuel P Goree, Indiana University
Sandy Manolios, Delft University of Technology
Sangeon Yong, Korea Advanced Institute of Science & Technology
Sangeun Kum, Neutune
Sanjeel Parekh, Telecom Paris
Sanna Wager, Amazon
Satoru Fukayama, National Institute of Advanced Industrial Science and Technology (AIST)
Scott Wisdom, Google
Sebastian Rosenzweig, International Audio Laboratories Erlangen
Serkan Sulun, INESC TEC
Sertan Şentürk, Kobalt Music Group

Sharath Adavanne, Tampere University of Technology
Shih-Lun Wu, National Taiwan University
Shlomo Dubnov, UC San Diego
Shreyan Chowdhury, Johannes Kepler University
Shuo Zhang, Bose
Shuqi Dai, Carnegie Mellon University
Siddharth Gururani, Georgia Institute of Technology
Silvan Peter, Johannes Kepler University
Simon Durand, Spotify
Simon J Schwär, International Audio Laboratories Erlangen
So Yeon Park, Stanford University
Stefan Lattner, Sony Computer Science Laboratories
Stefanie Acevedo, University of Connecticut
Sungkyun Chang, Cochlear.ai
Sylvain Le Groux, Voicea.ai; Cisco Systems
Taegyun Kwon, KAIST
Tetsuro Kitahara, Nihon University
Thomas Prätzlich, Learnfield GmbH
Tiago Tavares, UNICAMP
Tian Cheng, National Institute of Advanced Industrial Science and Technology (AIST)
Timothy R de Reuse, McGill University
Timothy Tsai, Harvey Mudd College
Tomoyasu Nakano, National Institute of Advanced Industrial Science and Technology (AIST)
Tsung-Ping Chen, Academia Sinica
Venkata S Viraraghavan, Tata Consultancy Services
Viet-Anh Tran, Deezer
Vinod Subramanian, Queen Mary University of London
Vinutha T T.P., Shah & Anchor Kutchhi Engineering College
Vipul Arora, IIT Kanpur
Vsevolod E Eremenko, Universitat Pompeu Fabra
WeiHsiang Liao, Sony Group Corporation
Xavier Favory, Utopia Music
Yaolong Ju, McGill University
Yichi Zhang, University of Rochester
Yigitcan Özer, International Audio Laboratories Erlangen
Ying Que, University of Hong Kong
Yi-Wen Liu, National Tsing Hua University
Youngmoo Kim, Drexel University
Yu Wang, New York University
Yuan-Pin Lin, National Sun Yat-sen University
Yudong Zhao, Queen Mary University of London
Yu-Fen Huang, Academia Sinica
Yu-Hua Chen, National Taiwan University
Yujia Yan, University of Rochester
Zafar Rafii, Gracenote
Zeyu Jin, Adobe Research
Ziyu Wang, New York University Shanghai

Preface

Welcome to ISMIR 2022, the 23rd International Society for Music Information Retrieval Conference. ISMIR is the world's leading research forum on processing, searching, organizing, and accessing music-related data. Our community reflects a diversity of scientific disciplines, seniority levels, professional affiliations, and cultural backgrounds. This year's ISMIR conference is being hosted in Bengaluru, India, making it the first ever ISMIR conference to take place in India. It comes at a very appropriate time, when we are seeing a steadily growing interest in MIR in both academic institutions and in start-ups in the country. Due to the changing global landscape as a result of the COVID-19 pandemic, the 23rd ISMIR conference became the first hybrid ISMIR conference, with both in-person and virtual attendees and presenters. While this posed unique challenges, we took this as an opportunity to bring back the benefits of a physical conference, while maintaining the expanded reach and improved inclusivity that the past two virtual editions of ISMIR have provided. The organizing team, who themselves came together from across the globe, welcomes you to ISMIR 2022.

I. Scientific Program

The Scientific Program Chairs (SPC) of this year's ISMIR are Masataka Goto (National Institute of Advanced Industrial Science and Technology (AIST), Japan), Rachel Bittner (Spotify, France), Rafael Caro Repetto (Kunstuniversität Graz, Austria), and Xavier Serra (Universitat Pompeu Fabra, Spain). Together with the General Chairs and with the participation of many members of the community, mainly as meta-reviewers and reviewers, the SPC coordinated the call for papers, the review process of the conference, and the organization of the scientific sessions.

The core of the ISMIR 2022 Scientific Program is the 113 papers that will be presented during the conference. A total of 305 abstracts were registered of which 264 were submitted as full papers eligible for review. In keeping with the practices of the previous years, a two-tier double-blind review process was conducted involving a total of 241 reviewers and 64 meta-reviewers. Meta-reviewers were instructed to complete a full review of each of their assigned papers, in addition to the final meta-review summarizing the individual reviews. Each paper was assigned to a single meta-reviewer and at least 3 reviewers, with every paper receiving at least 4 reviews in total. A reviewer training session given by Meinhard Muller and Ashley Burgoyne was organized before the review process started. Each meta-reviewer was responsible for between 2 and 5 papers, and each reviewer was responsible for no more than 5 papers, with the average reviewer being responsible for 3.4 papers. The initial reviewing phase was followed by a discussion period, in which reviewers and meta-reviewers could discuss and revise their assessments of each paper. Meta-reviewers were then instructed to summarize the discussion and reviews in the final report, and provide a final recommendation. The SPC rendered final decisions for each paper. The SPC would like to express their thanks to the ISMIR community of reviewers for their wholehearted support to this critical aspect of a successful ISMIR technical program.

Table 1 summarizes the submitted papers by subject area together with the corresponding accepted proportion. Figure 1 illustrates the number of papers accepted with at least one contributing author from each region. Geographic affiliations were inferred from self-reported author affiliations and email addresses. Finally, Table 2 summarizes the publication statistics over the 23-year-history of the conference.

Table 1: Papers submitted and accepted by subject area

Subject Area	Submitted	Accepted	Accept %
Applications	22	5	23
Domain knowledge	50	22	44

Evaluation, datasets and reproducibility	26	11	42
Human-centered MIR	16	3	19
MIR fundamentals and methodology	18	10	56
MIR tasks	94	45	48
Musical features and properties	35	15	43
Philosophical and ethical discussions	3	2	67
Total	**264**	**113**	**42.8**

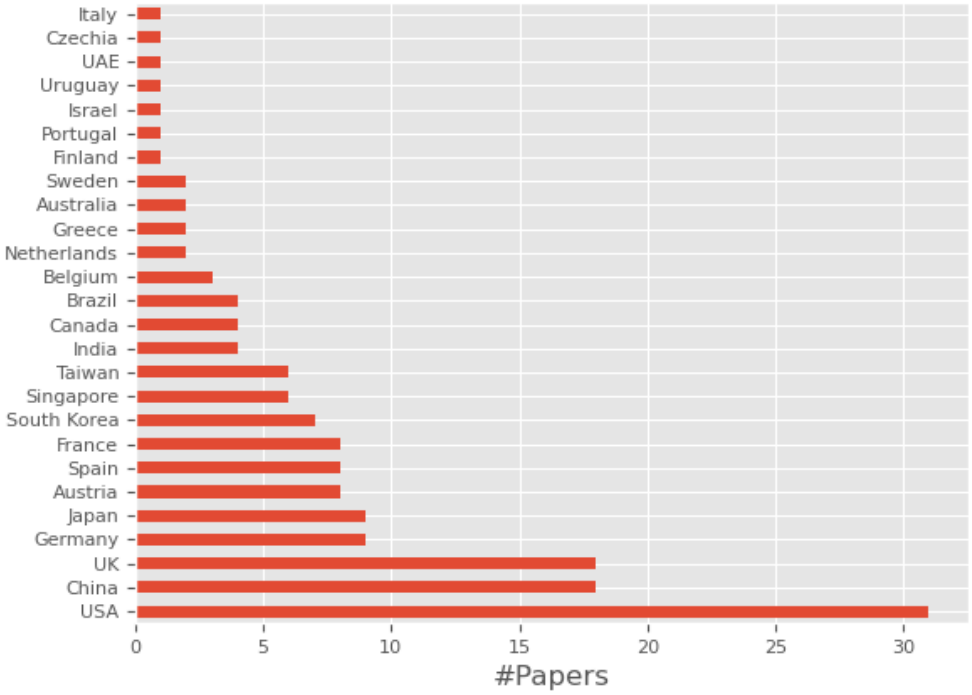


Figure 1: Number of papers accepted with at least one contributing author affiliated to an institution from each country

Table 2: Summary of publication statistics over the 23-year-history of the ISMIR conference

Year	**Location**	**Oral**	**Poster**	**Total**	**Authors**	**Unique Authors**	**Authors / Paper**	**Unique Authors / Paper**
2000	Plymouth	19	16	35	68	63	1.9	1.8
2001	Indiana	25	16	41	100	86	2.4	2.1
2002	Paris	35	22	57	129	117	2.3	2.1
2003	Baltimore	26	24	50	132	111	2.6	2.2
2004	Barcelona	61	44	105	252	214	2.4	2.0

2005	London	57	57	114	316	233	2.8	2.0
2006	Victoria	59	36	95	246	198	2.6	2.1
2007	Vienna	62	65	127	361	267	2.8	2.1
2008	Philadelphia	24	105	105	296	253	2.8	2.4
2009	Kobe	38	85	123	375	292	3.0	2.4
2010	Utrecht	24	86	110	314	263	2.0	2.4
2011	Miami	36	97	133	395	322	3.0	2.4
2012	Porto	36	65	101	324	264	3.2	2.6
2013	Curitiba	31	67	98	395	236	3.0	2.4
2014	Taipei	33	73	106	343	271	3.2	2.6
2015	Málaga	24	90	114	370	296	3.2	2.6
2016	New York	25	88	113	341	270	3.0	2.4
2017	Suzhou	24	73	97	324	248	3.3	2.6
2018	Paris			104	337	265	3.2	2.5
2019	Delft			114	390	315	3.4	2.8
2020	Montréal / Virtual			115	426	343	3.7	3.0
2021	Virtual			104	334	269	3.2	2.6
2022	Bengaluru (Hybrid)			113	423	355	3.8	3.0

Cultural and Social Diversity in MIR (Special Call)

In order to promote the study of music traditions that are still under-represented in MIR, there was a Special Call for Papers on "Cultural and Social Diversity in MIR," continuing the spirit of ISMIR 2021's Special Call for Papers on "Cultural Diversity in MIR." These traditions might not only present music characteristics that would require novel approaches even for standard MIR tasks, but can represent under-studied musical functions and communities. In this regard, this Special Call also encouraged the study of new groups of music makers and users, as well as the development of tools that benefit beyond mainstream music communities. Equally, it encouraged studies from a cross-cultural perspective that cater to this musical and cultural diversity. During the submission process, authors had to indicate that their paper was submitted to this Special Call.

The submissions to the Special Call underwent the same review process as the papers in the main track, with the same number of reviews and a similar number of bids per submission, and with meta-reviewers who were carefully chosen to oversee the review process. In all, 41 papers were submitted to this call, of which 14 were accepted and

verified by the SPC to match the topics of the call. Table 3 shows the distribution of submitted papers across subject areas together with the proportion of accepted papers in each for this special call.

Table 3: Special Call for Papers on "Cultural Diversity in MIR": Papers submitted and accepted by subject area

Subject Area	Submitted	Accepted	Accept %
Applications	5	0	0%
Domain knowledge	14	4	29.6%
Evaluation, datasets, and reproducibility	4	2	50%
Human-centered MIR	2	1	50%
MIR fundamentals and methodology	1	0	0%
MIR tasks	12	4	33.3%
Musical features and properties	2	2	100%
Philosophical and ethical discussions	1	1	100%
Total	**41**	**14**	**34.1%**

Figure 2 depicts the number of Special Call papers accepted with at least one contributing author from each of the specified regions of the world. The geographic affiliations were inferred from self-reported author affiliations. We note an increase of regional diversity in the papers submitted to this year's Special Call, with 7 papers co-authored by researchers from institutions located in 2 or 3 different countries.

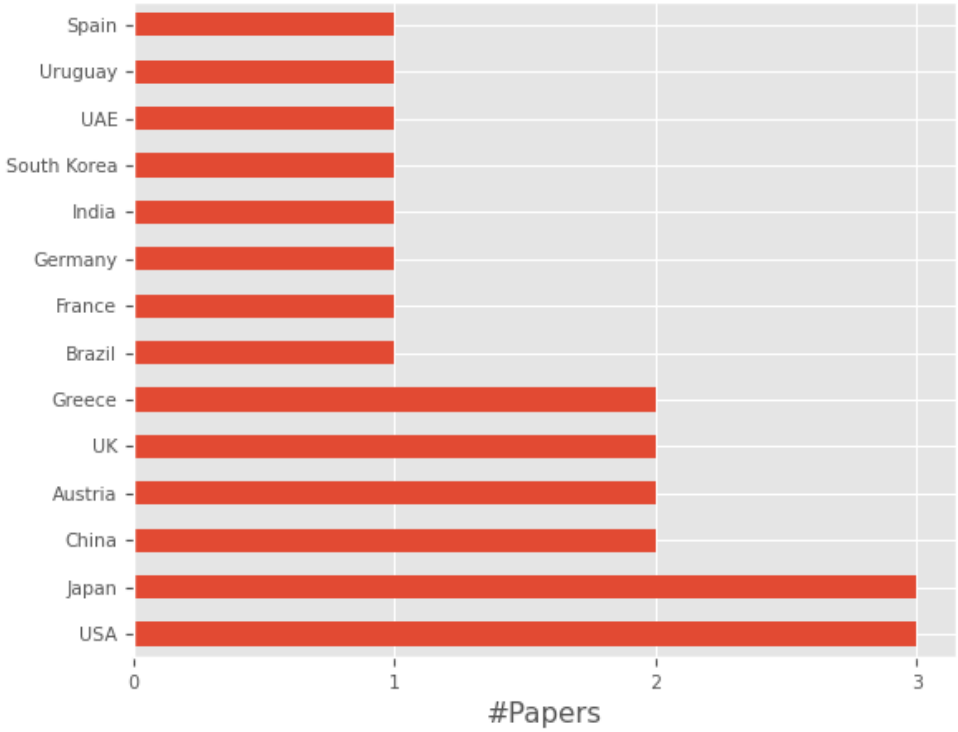


Figure 2: Number of papers in the special call accepted with at least one contributing author affiliated to an institution from each country.

Best Paper Awards

Best paper candidates were selected from the 113 accepted papers. The SPC first short-listed 39 papers based on reviewers' and meta-reviewers' nominations as well as the paper review scores. Of these, the SPC nominated 7 paper candidates under four categories: the Best Paper Award (5 candidates), the Best Student Paper Award (5 candidates), the Best Special Call Paper Award (2 candidates), and the Brave New Idea Award (3 candidates), based on their own judgement of the paper attributes as well as the detailed reviewer comments. This year, the nominations for the Best Paper Award and the Best Student Paper Award are the same. Moreover, we added a criterion,
"Pioneering proposals: This paper proposes a novel topic, task or application. Since this is intended to encourage brave new ideas and challenges, papers rated *Strongly Agree* and *Agree* can be highlighted, but please do not penalize papers rated *Disagree* or *Strongly Disagree*."
to the review form, and thus the Brave New Idea Award was introduced this year to encourage papers with high marks for this criterion. The nominations below are listed in the order of Paper ID for each category. The award winners are underlined.

Best Paper Award Nominations

Yixiao Zhang, Junyan Jiang, Gus Xia, and Simon Dixon, Interpreting Song Lyrics with an Audio-Informed Pre-trained Language Model

Oleg Lesota, Emilia Parada-Cabaleiro, Stefan Brandl, Elisabeth Lex, Navid Rekabsaz, and Markus Schedl, Traces of Globalization in Online Music Consumption Patterns and Results of Recommendation Algorithms

Simeon Rau, Frank Heyen, Stefan Wagner, and Michael Sedlmair, Visualization for AI-Assisted Composing

Mathilde Abrassart and Guillaume Doras, And What If Two Musical Versions Don't Share Melody, Harmony, Rhythm, or Lyrics?

Lele Liu, Qiuqiang Kong, Veronica Morfi, and Emmanouil Benetos, Performance MIDI-to-Score Conversion by Neural Beat Tracking

Best Student Paper Award Nominations

Yixiao Zhang, Junyan Jiang, Gus Xia, and Simon Dixon, Interpreting Song Lyrics with an Audio-Informed Pre-trained Language Model

Oleg Lesota, Emilia Parada-Cabaleiro, Stefan Brandl, Elisabeth Lex, Navid Rekabsaz, and Markus Schedl, Traces of Globalization in Online Music Consumption Patterns and Results of Recommendation Algorithms

Simeon Rau, Frank Heyen, Stefan Wagner, and Michael Sedlmair, Visualization for AI-Assisted Composing

Mathilde Abrassart and Guillaume Doras, And What If Two Musical Versions Don't Share Melody, Harmony, Rhythm, or Lyrics?

Lele Liu, Qiuqiang Kong, Veronica Morfi, and Emmanouil Benetos, Performance MIDI-to-Score Conversion by Neural Beat Tracking

Best Special Call Paper Award Nominations

Martin Clayton, Preeti Rao, Nithya Shikarpur, Sujoy Roychowdhury, and Jin Li, Raga Classification from Vocal Performances Using Multimodal Analysis

Oleg Lesota, Emilia Parada-Cabaleiro, Stefan Brandl, Elisabeth Lex, Navid Rekabsaz, and Markus Schedl, Traces of Globalization in Online Music Consumption Patterns and Results of Recommendation Algorithms

Brave New Idea Award Nominations

Yixiao Zhang, Junyan Jiang, Gus Xia, and Simon Dixon, Interpreting Song Lyrics with an Audio-Informed Pre-trained Language Model

Oleg Lesota, Emilia Parada-Cabaleiro, Stefan Brandl, Elisabeth Lex, Navid Rekabsaz, and Markus Schedl, Traces of Globalization in Online Music Consumption Patterns and Results of Recommendation Algorithms

Jaidev Shriram, Makarand Tapaswi, and Vinoo Alluri, Sonus Texere! Automated Dense Soundtrack Construction for Books Using Movie Adaptations

The final selections were made by specially appointed judges drawn from experienced researchers and were announced during the conference.

Best Reviewer Awards

Based on the scores provided by meta-reviewers on the quality of individual reviews, in relation to the number of papers reviewed by each reviewer, the SPC selected a total of 5 awardees listed below:

Katharina Hoedt
Maximilian Schmitt
Thomas Prätzlich
Silvan Peter
Reinier de Valk

II. Tutorials

T1: **An Introduction to Symbolic Music Processing in Python with Partitura**
Carlos Cancino-Chacón, Francesco Foscarin, Emmanouil Karystinaios, Silvan David Peter

T2: **Computational Methods for Supporting Corpus-Based Research on Indian Art Music**
Thomas Nuttall, Genís Plaja-Roglans, Lara Pearson, Brindha Manickavasakan, Ajay Srinivasamurthy, Kaustuv Kanti Ganguli

T3: **Designing Controllable Synthesis System for Musical Signals**
Hyeong-Seok Choi, Yusong Wu

T4: **Few-Shot and Zero-Shot Learning for Musical Audio**
Yu Wang, Hugo Flores García, Jeong Choi

T5: **Deep learning for Automatic Mixing**
Christian J. Steinmetz, Soumya Sai Vanka, Gary Bromham, Marco A. Martínez Ramírez

T6: **Trustworthy MIR: Creating MIR applications with values**
Christine Bauer, Andrés Ferraro, Emilia Gómez, Lorenzo Porcaro

III. Keynotes

TM Krishna
Karnatik Musician, Author & Activist
Evolution of Performance and Aesthetics in Indian Art Music

Richa Singh
Professor and Head, Dept. of Computer Science and Engineering, Indian Institute of Technology Jodhpur
Adventures of AI: Deepfake and Bias in Audio Processing

IV. Special Sessions

To complement the regular scientific program there were six special sessions, organized as panel discussions, aimed at covering various trending topics of relevance to the ISMIR community. Two of the sessions had a hybrid format and three were fully online.

1. **Enhancing music listening with MIR** (Hybrid)

Moderator: Xavier Serra
Panelists: Anna Gatzioura, Fabien Gouyon, Thomas Lidy, Hugo Rodrigues

In this panel we will discuss the research challenges and opportunities related to the development of new MIR technologies and services to support music listening.

2. **Enhancing music creativity with MIR** (Hybrid)

Moderator: Jan Van Balen
Panelists: Georgi Dzhambazov, Dorien Herremans, Oriol Nieto, Akira Maezawa, Igor Pereira

While audio technology has always had an important role in music production, it is now recognised that MIR tools can provide for workflows that enhance music creativity at every stage of the journey. The panel will discuss the possibilities and challenges of this exciting partnership between music computing and creativity.

3. **Ethics and MIR** (Online)

Moderators: Andre Holzapfel, Fabio Morreale, Bob Sturm

This special session will discuss an action plan towards a code of ethics for the ISMIR community. A code of ethics represents a specific list of values and behaviors that a research community either endorses or objects to. Codes of ethics have been established on the general level of engineering associations (IEEE, ACM), but also more specifically by research communities such as NIME (https://www.nime.org/ethics/).

Whereas ISMIR has seen a series of tutorials on ethics and values, and guidelines have been proposed (https://ismir.net/resources/ethics/), these attempts have not yet manifested into an official code of ethics. Does ISMIR need such a code? What is the function of the code? How can we establish and maintain such a code? What are the main ethical concerns regarding ISMIR research and practice?

4. **PhD in MIR: Challenges and Opportunities** (Online)

Moderator: Meinard Mueller

Music information retrieval (MIR) is an exciting research field related to different disciplines, including signal processing, machine learning, information retrieval, psychology, musicology, and the digital humanities. This diversity opens up many opportunities for challenging, interdisciplinary, and fascinating research projects at the intersection of engineering and humanities. However, younger researchers can also feel overwhelmed by the variety and complexity of MIR research questions. In this session, we will have an informal exchange of ideas and experiences, inviting doctoral candidates and more experienced MIR researchers. Responding to questions from the audience, we hope this interactive session will be helpful for current PhD students and students considering a PhD in MIR.

5. **TISMIR: the open journal of the ISMIR society** (Online)

Moderator: Emilia Gómez

Transactions of the International Society for Music Information Retrieval https://transactions.ismir.net/ was established in 2018 to complement the ISMIR conference proceedings and provide a vehicle for the dissemination of the highest quality and most substantial scientific research in MIR. TISMIR retains the Open Access model of the ISMIR Conference proceedings, encourages reproducibility of the published research papers, and maintains a low publication cost.

Almost 5 years later, this ISMIR 2022 is devoted to discuss and brainstorm on the current status and future perspectives of the journal with a series of TISMIR recent and potential authors, reviewers and editors. We will address the following questions, and others proposed by participants:

What do you appreciate more about TISMIR?

What is the link and complementarity to the ISMIR conference?

Which are the main challenges/limitations that need to be addressed?

How to make TISMIR competitive as a journal in the current publication landscape?

How to engage with more community members in order to make TISMIR a success?

Which are future avenues for conference vs journal outlets in the ISMIR field?

V. Diversity & Inclusion (D&I)

The ISMIR 2022 conference took a broad view of Diversity and Inclusion (D&I). Under the leadership of the conference D&I Chairs, in collaboration with the organizing team at large, ISMIR 2022 offered a variety of initiatives intended to make the conference a positive, welcoming, and supportive environment for a diverse range of presenters and attendees.

The initiatives under ISMIR 2022 D&I included:

1. A new New-to-ISMIR paper mentoring program which was designed for members new to the ISMIR community (early-stage researchers, researchers from areas underrepresented in ISMIR, and/or researchers from other allied fields who wish to submit their work to ISMIR conference). It is envisioned that the program becomes a regular part of the D&I initiatives planned by the society
2. As a part of commitments to foster D&I and to support early-stage researchers, the ISMIR society, WiMIR, local organizers have provided financial support in terms of registration waivers enabled by the generous support of the sponsors. Partial or full waivers have been extended to applicants interested in in-person or virtual participation
3. WiMIR plenary session of ISMIR 2022 invited a panel of four women researchers who are exploring new frontiers and themes in music research, to motivate the audience on their recent research topic, share the challenges and insights and have an interactive session with the audience
4. ISMIR 2022 is made accessible to the local community through open sessions, welcoming the local community to register for free and participate in select sessions

Finally, the ISMIR conference Code of Conduct has been updated to align with this year's hybrid format to provide a harassment-free experience for all participants.

New-to-ISMIR paper mentoring program

The New-to-ISMIR paper mentoring program is designed for members new to ISMIR (early-stage researchers in MIR or researchers from allied fields who wish to consider submitting their work to an ISMIR conference) to share their advanced-stage work-in-progress ISMIR paper drafts with senior members of the ISMIR community (mentors who volunteered for the program) to obtain focused reviews and constructive feedback. The program supplements the generic submission guidelines. The program is being run as a pilot in 2022, closely aligned with the ISMIR 2022 paper submissions deadlines.

We thank the following 24 invited senior members, who were kind enough to sign up as mentors, for their valuable time to review the submitted works with constructive feedback:

Emmanouil Benetos	Cory McKay	David Sears
Jin Ha Lee	Chris Donahue	Ichiro Fujinaga
Cheng-i Wang	Philippe Esling	Juhan Nam
Oriol Nieto	Alexander Lerch	Cynthia Liem
Mitsunori Ogihara	Arthur Flexer	George Fazekas
Carlos Cancino-Chacón	Geoffroy Peeters	Christof Weiss
Justin Salamon	Vipul Arora	Sankalp Gulati
Blair Kaneshiro	Jordan Smith	Xiao Hu

There was an enthusiastic 30 mentee sign-up in the pilot program. 10 out of 15 mentored submissions were submitted to ISMIR 2022. Two of these submissions have been accepted in ISMIR 2022.

Feedback from both mentors and mentees has been collected to establish best practices for future editions of the paper mentoring program.

Women in Music Information Retrieval (WiMIR)

Women in Music Information Retrieval (WiMIR) is a group of people dedicated to promoting the role of, and increasing opportunities for, women in the MIR field. WiMIR's initiatives started as informal gatherings around breakfast or lunch during ISMIR conferences (2011–2014), and moved to formal WiMIR events included in the conference program (2015–today) garnering a high turnout of both women and allies. These events provide occasions for people to network and to discuss several important issues ranging from mentorship and conference support, to improving the representation of women and, more broadly, diversity in the community. In 2018, WiMIR started hosting its own workshop as a satellite event, in which attendees of all genders participated. These workshops aim to offer participants an opportunity for networking, put the spotlight on technical work done by women in the field, and foster collaboration between women and allies by proposing group work led by project guides to try to solve small research problems or to undertake new research projects that could lead to longer-term collaborations. The ISMIR 2022 D&I Chairs gratefully acknowledge the support of this year's WiMIR sponsors, whose contributions support women in the field as well as the broader D&I efforts of this year's conference.

WiMIR Plenary Session

WiMIR plenary session of ISMIR 2022 was a hybrid session, featuring a panel of four women researchers who were invited to motivate the audience about their recent topic of research, while sharing the associated challenges in their journey and valuable insights from their experience. The session also featured an interactive conversation with the audience.

Moderators: Xiao Hu (online), Ranjani H G (in-person)
Panelists:
Xiao Hu (The University of Hong Kong), "Music for learning and well-being"

Emilia Parada-Cabaleiro (Johannes Kepler University), “Working in MIR with a diverse background: A personal view”
Chitralekha Gupta (National University of Singapore), “Automated singing quality analysis: Overview and challenges”
Shahar Elisa (Spotify), “Research on the industrial lane”

Open sessions

As a part of D&I efforts, we had the following events open to the general public and within the IISc community.

The late breaking/demo (LBD) session was proposed to be an open session in ISMIR 2022, and invites students and researchers in the local community. It enabled interested participants to register (without cost) and attend the late-breaking/demo session of the conference and interact with students, experts and senior members of the ISMIR research community.

ISMIR 2022 music track session was proposed to be an open session as a part of outreach to the local community to as a part of efforts to encourage more submissions in future ISMIR editions. The ISMIR 2022 music concert featured a Jugalbandi vocal concert to highlight the commonalities, differences of the two forms of Indian art music - Hindustani and Carnatic music to the ISMIR community. The concert was open to the local community.

Financial support

ISMIR 2022 provided partial and full registration waivers to authors, students, women and other underrepresented minorities in MIR, attendees from low-income countries, new to ISMIR and unaffiliated attendees. The grants included Author Grants (for both accepted full paper and music submissions), WiMIR grants (based on the applicants’ eligibility criteria) to encourage student participants, authors for both in-person or virtual participants. Overall, 25 author grants and more than 50 WiMIR grants have been offered. Almost all applicants have been offered registration grants. Registration waivers were offered to all volunteers who signed up for ISMIR 2022. This is the first edition of the hybrid version of ISMIR conference. Since there were uncertainties around travel due to the pandemic, ISMIR 2022 did not consider providing any travel and accommodation grants.

IV. Late Breaking/Demo Session

Late Breaking/Demo (LBD) session was organised in a hybrid format this year with both local and remote sessions. A light screening process was carried out for each submission by the LBD chairs. Owing to visa application processing times, an earlier submission date was recommended for people requiring visas. To clearly distinguish LBD extended abstracts from main conference papers, a new watermark was introduced on the abstract template. Keeping in line with the cultural and social diversity theme of the conference, the chairs reached out to groups working on Indian music inviting submissions. The local LBD session was an open session for students and independent researchers from the local community, allowing them to attend the event for free (subject to venue capacity constraints). This enabled participants to interact with students, experts and senior members of the ISMIR research community.

VI. Music

The idea of the music program this year was to bring together composers, technologists and performers, demonstrating application of music information retrieval, computational musicology, human computer interaction, machine accompaniment, call-response, or ethical hacking. Keeping in line with the cultural and social diversity theme, submissions on non-Eurogenetic music that could present cultural diversity were also encouraged. The submissions were evaluated on the basis of audio-visual presentation, submission statement, concept clarity while capturing and retaining audience attention. Here is a list of music submissions that were presented:

M1: **Hindustronic Live**
Carlos Guedes

M2: **Conformity 16 for autonomous piano and large ensemble**
Jason Palamara

M3: **Wings for Solo Clarinet and Automated Accompaniment Video Animation**
Kaitlin Pet, Nikki Pet, Christopher Raphael

M4: **AI Phantasy**
Panayiotis Kokoras

M5: **A song with yati Patterns - Visual representation through Kolam**
Saroja TK, Sujatha TKL, Chandrakanth Mamillapalli

M6: **Fantastic AI Sinawi**
Danbinaerin Han, Hannah Park, Chaeryeong Oh, Dasaem Jeong

M7: **Mukti - Kahan Re Aaya Tu (मुक्ति - कहाँ रे आया तू)**
Jyoti Narang, Thomas Nuttall

M8: **The Oratory of Saint Philip Neri**
Luke Dzwonczyk

M9: **Beatboxing with a homespun Sound box**
Ranaprathap Ponnam

M10: **Confluence of Carnatic and Western Music using Grahabedha and Carnatic Gamakas**
Tallapragada Shanmukha Sreevatsa, Suswara Pochampally (Equal contribution)

M11: **Recurrent Variations for String Orchestra**
Hendrik Vincent Koops
M12: 'b_dot_io': an Audio-Visual Miniature for Saxophone and Computer
Mark Hanslip

M13: **Bloom for cello and live electronics**
Austin A Franklin

Music was alive and kicking at hybrid ISMIR 2022!

VII. Industry Presentations Session

With MIR technology playing a prominent role in the present-day industry, we had ISMIR 2022 sponsors making brief presentations about their company profile, the nature of their R&D and the opportunities available to students interested in the field. The participating sponsors were: Spotify, Moises, Adobe, Deezer, Utopia music, Pandora, Smule, Yamaha, and Chordify.

IX. Social Program

After two years of Virtual ISMIR conference, ISMIR 2022 was the first hybrid edition that enabled in-person participants. ISMIR 2022 social program was designed to be engaging to the in-person participants and aimed to

showcase the rich cultural and musical heritage of India. The social program included a performance of Kalidasa's play Mālavikāgnimitram by Dhaatu Puppet Theater (https://www.dhaatupuppets.org/), a Jugalbandi vocal Indian art music concert by Kaustuv Kanti Ganguli and Vignesh Ishwar, an Indian street food themed welcome reception, a banquet dinner in a museum that showcased a model village capturing Karnataka's distinct rural life in all its quaint glory and the traditional ISMIR jam session.

VIII. Satellite Events

In addition to the main conference, five satellite events of ISMIR 2022 were available to participants:

1. **WiMIR workshop**: The WiMIR Workshop comprised two days of talks by eminent researchers in the WiMIR community, and an opportunity to network, socialize, and have discussions with peers ahead of the ISMIR conference. The Workshop was, as ever, free and open to all members of the MIR community. More information: https://wimir.wordpress.com/2022/08/03/wimir-workshop-2022/

2. Music, Mind, Movement and Technology (MMMT) workshop: The MMMT workshop was a hybrid satellite workshop around the 23rd International Society for Music Information Retrieval Conference (ISMIR 2022). MMMT was an attempt to increase the dialogue between the fields of Music Information Retrieval and Music Cognition. This hybrid two-day workshop on Dec 2-3, 2022 brought together leading international researchers for a series of talks highlighting interdisciplinary research and facilitating interaction and exchange of ideas around various themes. More information: https://ismir2022.ismir.net/satellites/mmmt

3. **Indian Music Experience**: As a satellite event for ISMIR 2022, we organized an Indian music experience workshop on the 9th Dec, 2022. The day-long physical-only workshop was hosted at the Indian Music Experience Museum (IME), which is India's first interactive music museum and involved a museum visit, workshops on Indian art music and a music exhibition. More information: https://ismir2022.ismir.net/satellites/ime

4. **Music HackDay India**: The Music Tech Community India is an open community of musicians, developers, researchers and artists with an aim to collaborate, share knowledge and bridge the gap between media, arts and technology as a community. The community conducts talks, workshops, seminars and are currently curating tutorials, resources and interviews with artists, startup founders and researchers. As a satellite event of ISMIR 2022, the community organized a 2-day MusicHackDay India 2022 event on 10-11 Dec, 2022 that included a hackathon, algorave concert, networking, and panel discussions. The event was open to public (with prior registration) and organized in Bengaluru, with a possibility of virtual participation. MusicHackDay India 2022 included the following events:

    - HAMR (Hacking Audio Music Research): Extending on the tradition of past ISMIR conferences, MusicHackDay India included a hackathon with prizes for the best hacks.
    - Algorave concert: Live coding, audio-visual performance from our friends from the Algorave India community, coupled with a social event
    - Networking: A great opportunity to connect the burgeoning tech and music community in Bengaluru, India with the diverse ISMIR community.
    - Panel discussions/Interviews: A panel discussion between individuals from different dimensions of music tech from the Academia, Business and the Musicians.

    More information: https://musichackdayindia.github.io/

5. **CompMusic workshop**: CompMusic Workshop 2022 was a 5-day workshop and a satellite event of ISMIR 2022 to introduce the field of Computational Musicology while focusing on the study of Carnatic Music. By combining theoretical lectures with hands-on labs, the workshop was aimed at giving the participants the conceptual framework and practical tools needed to analyze and understand music signals using a variety of computational methodologies. The workshop was aimed at undergraduate or graduate

students doing engineering, music, or social sciences, without prior experience in the topic but highly interested in acquiring the computational and musicological competencies needed to study music, in particular Carnatic Music. The workshop took place at the Indian Institute of Technology Madras, Chennai during the Madras Music Season, which is the largest music festival of Carnatic Music. In the evenings, the participants were able to attend a wide variety of Carnatic Music concerts. More information: https://compmusic.upf.edu/workshop-ismir-2022

IX. Acknowledgements

We are happy to present to you the proceedings of ISMIR 2022. The conference program was made possible thanks to the hard work of many people, including the ISMIR 2022 conference chairs, the joint host institutes of IIT Bombay, IISc Bangalore and IIT Madras, the administrative staff of the National Science Seminar Complex (NSSC) at IISc, ISMIR Board members, volunteers, and the many reviewers and meta-reviewers from the program committee.

We would also like to thank our sponsors, whose generous support made this conference possible:

Platinum sponsors

- Spotify
- Moises

Gold sponsors

- Adobe
- Deezer
- Utopia Music
- Pandora
- IndSCA
- Steinberg Media Technology
- Smule
- Yamaha
- Chordify
- TikTok

Silver sponsors

- Steinberg Media Technology
- iZotope/Soundwide

Supporters

- ACR Cloud
- Cochl
- School of Information Sciences, University of Illinois Urbana-Champaign

We would like to thank the sponsors that explicitly chose to sponsor WiMIR, its grants, and its initiatives:

Patrons

- Spotify
- Deezer

Contributors

- Adobe

- Pandora
- Soundwide
- TikTok

Supporters

- Steinberg

ISMIR 2022 would not have been possible without the exceptional contributions of our community in response to our call for participation. The biggest acknowledgment goes to you, the presenters and the (in-person and remote) participants.

Rachel Bittner
Rafael Caro Repetto
Masataka Goto
Xavier Serra
Scientific Program Chairs

Preeti Rao
Hema Murthy
Ajay Srinivasamurthy
General Chairs

Contents

Keynote Talks

Keynote Talk - 1

Evolution of Performance and Aesthetics in Indian Art Music

TM Krishna
Karnatik Musician, Author & Activist

Abstract

Indian art music continues to evolve in current performance practice, while staying within the framework provided by some of the immutable axiomatic concepts that define the music culture. The changes that lead the evolution of the music culture are guided by practitioners and influenced by the evolving socio-cultural, socio-political or performance and aesthetic considerations. In this talk, we focus on the evolution of Indian art music from the perspective of performance and aesthetics, highlighting some important milestones around the melodic and rhythmic systems in Indian art music. Focusing on recent developments and our own influences on performance practice and aesthetics, we discuss our effort and approaches to create more inclusive roles in music composition and performance. We further aim to provide concrete examples and formulations of the abstractions in current performance and aesthetics. We propose thoughts and ideas that can help current MIR formulations and solutions to go beyond the limiting assumptions based on current music performance practices and (often rigid) structures, and focus on the music abstractions that are more fundamental to our understanding, appreciation and analysis of Indian art music.

Biography

TM Krishna, is one of the pre-eminent vocalists in the rigorous Karnatik tradition of India's classical music. As a public intellectual, Krishna speaks and writes about issues affecting the human condition and about matters cultural. As a vocalist, he has made path-breaking innovations in both the style and substance of his concerts. His award-winning book, A Southern Music – The Karnatik Story, published by Harper Collins in 2013 was a first-of-its-kind philosophical, aesthetic and socio-political exploration of Karnatik music. TM Krishna has partnered with individuals and collectives working at the intersections of social change, a new politics for contemporary India, a fresh new imagining of the wider universe of the Arts. In 2016, TM Krishna received the prestigious Ramon Magsaysay Award in recognition of "his forceful commitment as artist and advocate to art’s power to heal India’s deep social divisions".

Keynote Talk - 2

Adventures of AI: Deepfake and Bias in Audio Processing

Richa Singh
Professor and Head, Dept. of Computer Science and Engineering, Indian Institute of Technology Jodhpur

Abstract

The increasing capabilities for machine learning algorithms is enabling the usage of ML models for a variety of tasks including for creativity such as generating new music and modifying existing music. Similar applications are present in different kinds of audio signals such as voice biometrics, speaker and speech recognition. However, these technologies that support creativity can also be used for malicious purposes. Deepfake audios are one such technology which enable flawlessly altering existing audio signals or creating new signals from any given text. Audio can also be integrated with videos to provide a complete multimodal experience, which can be purely synthetic and fake. While there is significant research ongoing in image and video, the space of detecting these anomalies in audio processing is relatively unaddressed. We will discuss some of these possible adventures of machine learning in audio processing and the research efforts that we are undertaking to detect them. In addition, we will also discuss the bias and fairness issues in audio processing where we will highlight "out of distribution" behavior of popular approaches and some strategies to address them.

Biography

Richa Singh received her Ph.D. degree in computer science from West Virginia University, Morgantown, USA, in 2008. She is currently a Professor and Head at Department of CSE, IIT Jodhpur. She has co-edited the book Deep Learning in Biometrics and has delivered keynote talks/tutorials on deep learning, trusted AI, and domain adaptation in NVIDIA GTC 2021, BIOSIG2021, ICCV 2017, AFGR 2017, and IJCNN 2017. Her areas of interest are pattern recognition, machine learning, and biometrics. She is a Fellow of IEEE, IAPR and AAIA, and a Senior Member of ACM. She was a recipient of the Kusum and Mohandas Pai Faculty Research Fellowship at the IIIT-Delhi, the FAST Award by the Department of Science and Technology, India, and several best paper and best poster awards in international conferences. She is/was served as the Program Co-Chair of CVPR2022, ICMI2022, IJCB2020, AFGR2019 and BTAS 2016, and a General Co-Chair of FG 2021 and ISBA 2017. She is also the Vice President (Publications) of the IEEE Biometrics Council and an Associate Editor-in-Chief of Pattern Recognition.

Tutorials

Tutorial 1

An Introduction to Symbolic Music Processing in Python with Partitura

Carlos Cancino-Chacón, Francesco Foscarin, Emmanouil Karystinaios, Silvan David Peter

Abstract

Symbolic music formats (e.g., MIDI, MusicXML/MEI) can provide a variety of high-level musical information like note pitch and duration, key/time signature, beat/downbeat position, etc. Such data can be used as both input/training data and as ground truth for MIR systems.

This tutorial aims to provide an introduction to symbolic music processing for a broad MIR audience, with a particular focus on showing how to extract relevant MIR features from symbolic musical formats in a fast, intuitive, and scalable way. We do this with the aid of the Python package Partitura. To target different kinds of symbolic data, we use an extended version of the ASAP Dataset, a multi-modal dataset that contains MusicXML scores, MIDI performances, audio performances, and score-to-performance alignments.

The tutorial will be structured in four parts: The first part provides an introduction to the topic of symbolic music processing. The second, third, and fourth parts are hands-on tutorials that showcase the structure of the Partitura package (including its relation to other popular Python packages for symbolic music processing), how to extract common MIR features, and how to work with symbolic multimodal datasets, respectively.

The motivation behind this tutorial is to promote research on symbolic music processing in the MIR community. Therefore, we target a broad audience of researchers without requiring prior knowledge of this particular area. For the hands-on parts of the tutorial, we presuppose some practical experience with the Python language, but we will provide well-documented step-by-step access to the code in the form of Google Colab notebooks, which will be made publicly available after the tutorial. Furthermore, some familiarity with the basic concepts of statistics and machine learning is useful.

This work receives funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement No 101019375 (Whither Music?).

Biographies of Presenters

Carlos Cancino-Chacón is an Assistant Professor at the Institute of Computational Perception, Johannes Kepler University, Linz, Austria, and a Guest Researcher at the RITMO Centre for Interdisciplinary Studies in Rhythm, Time and Motion, University of Oslo, Norway. His research focuses on studying expressive music performance, music cognition, and music theory with machine learning methods. He received a doctoral degree in Computer Science at the Institute of Computational Perception of the Johannes Kepler University Linz, a M.Sc. degree in Electrical Engineering and Audio Engineering from the Graz University of Technology, a degree in Physics from the National Autonomous University of Mexico, and a degree in Piano Performance from the National Conservatory of Music of Mexico.

Francesco Foscarin is a postdoctoral researcher at the Institute of Computational Perception, Johannes Kepler University, Linz, Austria. He completed his Ph.D. at CNAM Paris on music transcription, with a focus on the production of musical scores, and holds classical and jazz piano degrees from the Conservatory of Vicenza. His research interests include post-hoc explainability techniques for DL models, grammar-based parsing of hierarchical chord structures, piano comping generation for jazz music, and voice separation in symbolic music.

Emmanouil Karystinaios is a Ph.D. student at the Institute of Computational Perception, Johannes Kepler University, Linz, Austria. His research topics encompass graph neural networks, music structure segmentation, and automated music

analysis. He holds an M.Sc. degree in Mathematical Logic from Paris Diderot University, an M.A. in Composition from Paris Vincennes University, and an integrated M.A. in Musicology from the Aristotle University of Thessaloniki.

Silvan David Peter is a University Assistant at the Institute of Computational Perception, Johannes Kepler University, Linz, Austria. His research interests are the evaluation of and interaction with computational models of musical skills. He holds an M.Sc. degree in Mathematics from the Humboldt University of Berlin.

Tutorial 2

Computational Methods For Supporting Corpus-Based Research On Indian Art Music

Thomas Nuttall, Genís Plaja-Roglans, Lara Pearson, Brindha Manickavasakan

Abstract

Culture-aware approaches to computational musicology and music information research (MIR) have been shown to be effective for a musically relevant analysis of a music culture. Projects such as CompMusic (2011-2017), MusicalBridges (2018-2022) or the initiatives funded by SPARC (2019-2022) have demonstrated the importance of considering sociocultural specifics of a music tradition to effectively define research problems, collect data and propose methods for analysis. These projects have made particularly notable contributions to the analysis of Indian Art Music (IAM), leading to a collective body of bespoke computational methods for analyzing these traditions.

Through this tutorial we aim to compile and present such works, making openly available a number of software tools and materials developed by MIR researchers working on the two main IAM traditions, Carnatic and Hindustani. The content will be organized into five sections: (1) datasets and corpora, (2) melodic analysis, (3) rhythmic analysis, (4) timbral analysis and (5) structural analysis. Each topic will include an introduction covering the basic musical concepts required to understand its constituent tasks, followed by a practical presentation of the materials and software tools compiled.

This tutorial is the result of an ongoing collaborative effort involving many contributors. The software will be available in Python through a single Github repository, containing clear and reproducible implementations of the presented methodologies. A Jupyter WebBook will be the main tutorial reference, in which we will introduce all the materials, contextualize the software tools, and include Jupyter Notebook examples for most of the research tasks covered.

Biographies of Presenters

Thomas Nuttall is a Research Engineer in the Music Technology Group (MTG) of Universitat Pompeu Fabra in Barcelona, Spain. His research focus is on melodic pattern analysis in musical traditions under-represented in the music computation and computational musicology fields, such as Arab-Andalusian or Indian Art Music, and on building tools that bridge the gap between the music information retrieval and musicology research communities.

Genís Plaja-Roglans is a Ph.D student in the Music Technology Group (MTG) of Universitat Pompeu Fabra in Barcelona, Spain. His research focus is on creation of bespoke machine learning models for the understanding of musical traditions under-represented in Music Information Research, currently focusing on Carnatic and Hindustani music. Recent work includes vocal melody estimation, singing voice source separation and repeated pattern discovery.

Lara Pearson is a musicologist at the Max Planck Institute for Empirical Aesthetics (MPIEA), Frankfurt am Main, Germany. Her work explores bodily and movement dimensions of music experience and meaning, often combining sonic and kinetic analyses. Her stylistic focus lies in South Indian music practices, in particular Carnatic music. She has also published on cross-cultural aesthetics, cultural heritage, music notation and the concept of improvisation.

Brindha Manickavasakan is a Carnatic Music vocalist, and is among the foremost, popular young performing Carnatic musicians in India. She has been performing for the past 21 years, and is currently learning from Vidushi Suguna Varadachari. She is an ‘A’ graded vocal artist of All India Radio. Brindha holds a Master’s degree in Biostatistics from Georgetown University, USA, a Master’s degree in Music and is a PhD candidate in Music from Madras University with a thesis on the contribution of Tañjāvūr K Ponnayyā Piḷḷai. She is a constant feature in all the major sabhas in Chennai, and performs regularly across India and abroad.

Tutorial 3
Designing Controllable Synthesis System for Musical Signals

Hyeong-Seok Choi, Yusong Wu

Abstract

Advances in deep learning and signal processing research have made it possible to generate signals that at times can be difficult to distinguish from real samples. Despite the realistic output the models can produce, however, the controllability of the models is still constrained because of the black-box-like nature of many models.

In this tutorial, we aim to introduce considerations researchers can take into account for a better end-user experience. We would like to focus in particular on how to design deep generative models with intuitive control of music audio signals, specifically vocal and instrumental performance. To this end, we will first present a broad review of up-to-date generative models for singing voices and musical instrument performance. Then, we will share our own research results and insights regarding both the implicit and explicit controllability of the deep learning models. In the section on presenting controllable models for instrumental performance synthesis, we will include a walk-through of the building, training, and control of the DDSP and MIDI-DDSP models via Jupyter (Colab) Notebook with Python and Tensorflow.

The target audience for this tutorial is researchers who are interested in deep generative models for monophonic signals, especially for singing voice and musical instruments. We expect the audience to have a basic understanding of machine learning concepts for audio signal processing.

Biographies of Presenters

Hyeong-Seok Choi received his PhD from Seoul National University, South Korea, in 2022, with a thesis titled, “A Controllable Generation of Signals from Self-Supervised Representations” under the supervision of Prof. Kyogu Lee. His recent research interest is mainly in representation learning and controllable synthesis of speech and singing voices. He co-founded the audio technology startup company, Supertone, where he has been working as the lead of their research team. He contributed to the winning of the CES 2022 Innovation Awards Honoree: Software & Mobile Apps by proposing a real-time voice conversion technology.

Yusong Wu is a final-year research master at the University of Montreal and Mila in Montreal, Canada. He is co-advised by Prof. Aaron Courville and Prof. Cheng-Zhi Anna Huang and will become a Ph.D. student under the same advisors shortly. His research focuses on making better generative models for music creativity. His recent work ``MIDI-DDSP: Detailed Control of Musical Performance via Hierarchical Modeling”, collaborating with Google Magenta, was accepted by ICLR 2022 for oral presentation.

Tutorial 4

Few-Shot and Zero-Shot Learning for Musical Audio

Yu Wang, Hugo Flores García, Jeong Choi

Abstract

While deep neural networks achieved promising results in many MIR tasks, they typically require a large amount of labeled data for training. Rare, fine-grained, or newly emerged classes (e.g. a rare musical instrument, a new music genre) where large-scale data collection is hard or simply impossible are often considered out-of-vocabulary and unsupported by MIR systems. To address this, few-shot learning (FSL) and zero-shot learning (ZSL) are learning paradigms that aim to train a model that can learn a new concept based on just a handful of labeled examples (few-shot) or some auxiliary information (zero-shot), mimicking human ability. By doing so, the trained model is no longer limited to a pre-defined and fixed set of classes but ideally can generalize to any class of interest with the cost of little human intervention. In addition, few-shot and zero-shot models naturally incorporate human input without asking for significant effort, making them useful tools when developing MIR systems that can be customized by individual users.

In this tutorial, we will go over

- FSL/ZSL foundations - Task definition and existing approaches.
- Recent advances of FSL/ZSL in MIR - Techniques and contributions in recent studies. We will also discuss the remaining challenges and future directions.
- Coding examples - Showcasing the training and evaluation pipeline of FSL and ZSL models on specific MIR tasks. Code and references to the tools and datasets will be provided.

We aim for this tutorial to be useful to researchers and practitioners in the ISMIR community who are facing labeled data scarcity issues, looking for new interaction paradigms between users and MIR systems, or generally interested in the techniques and applications of FSL and ZSL. We assume the audience is familiar with the basic machine learning concepts.

Biographies of Presenters

Yu Wang Yu is a Ph.D. candidate in Music Technology at the Music and Audio Research Laboratory at New York University, working under Prof. Juan Pablo Bello. Her research interests focus on machine learning and signal processing for music and general audio. Specifically, she is interested in adaptive and interactive machine listening with minimal supervision. She has interned with Adobe Research and Spotify. Before joining MARL in 2017, she was in the Music Recording and Production program at the Institute of Audio Research. She holds two M.S. degrees in Materials Science & Engineering from Massachusetts Institute of Technology (2015) and National Taiwan University (NTU) (2012), and a B.S. in Physics from NTU (2010). Yu is a guitar player and also enjoys sound engineering. Japanese math rock is her current favorite music genre.

Hugo Flores García is a Ph.D. student in Computer Science at Northwestern University, working under Prof. Bryan Pardo in the Interactive Audio Lab. Hugo's research interests lie at the intersection of machine learning, signal processing, and human computer interaction for music and audio. Hugo has previously worked on a deep learning framework for Audacity, an open source audio editor, and will be a research intern at Spotify and Descript during the latter half of 2022. Hugo holds an B.S. in Electrical Engineering from Georgia Southern University (2020). He is a jazz guitarist, and can be seen playing with various groups local to the Chicago area. Hugo enjoys augmenting musical instruments with technology, as well as making interactive music and art in SuperCollider and Max/MSP.

Jeong Choi is a machine learning researcher at Naver, where he leads NOW AI team that's working on a multi-modal recommendation system for a video streaming service, Naver NOW. Before joining Naver, he was a researcher at

NCSOFT, working on a recommendation system in a music game FUSER. He also interned at Deezer Research. He received a M.S. in Culture Technology at Korea Advance Institute of Science and Technology, under the supervision of Prof. Juhan Nam. His research interest is on representational learning of various signals that can further contribute to diverse music recommendation strategies. Previously, he pursued a long music career as a composer and a bassist. His passion for music research originates from the experience. He also received a M.S. and a B.E. in Digital Media at Ajou University, and majored in French at Daewon Foreign Language High School.

Tutorial 5

Deep learning for automatic mixing

Christian J. Steinmetz, Soumya Sai Vanka, Gary Bromham, Marco A. Martínez Ramírez

Abstract

Mixing is a central task within audio post-production where expert knowledge is required to deliver professional quality content, encompassing both technical and creative considerations. Recently, deep learning approaches have been introduced that aim to address this challenge by generating a cohesive mixture of a set of recordings as would an audio engineer. These approaches leverage large-scale datasets and therefore have the potential to outperform traditional approaches based on expert systems, but bring their own unique set of challenges. In this tutorial, we will begin by providing an introduction to the mixing process from the perspective of an audio engineer, along with a discussion of the tools used in the process from a signal processing perspective. We will then discuss a series of recent deep learning approaches and relevant datasets, providing code to build, train, and evaluate these systems. Future directions and challenges will be discussed, including new deep learning systems, evaluation methods, and approaches to address dataset availability. Our goal is to provide a starting point for researchers working in MIR who have little to no experience in audio engineering so they can easily begin addressing problems in this domain. In addition, our tutorial may be of interest to researchers outside of MIR, but with a background in audio engineering or signal processing, who are interested in gaining exposure to current approaches in deep learning.

Biographies of Presenters

Christian J. Steinmetz is PhD researcher working with Prof. Joshua D. Reiss within the Centre for Digital Music at Queen Mary University of London. He researches applications of machine learning in audio with a focus on differentiable signal processing. Currently, his research revolves around high fidelity audio and music production, which involves enhancing audio recordings, intelligent systems for audio engineering, as well as applications that augment and extend creativity. He has worked as a Research Scientist Intern at Adobe, Facebook Reality Labs, and Dolby Labs. Christian holds a BS in Electrical Engineering and BA in Audio Technology from Clemson University, as well as an MSc in Sound and Music Computing from the Music Technology Group at Universitat Pompeu Fabra.

Soumya Sai Vanka is a first year PhD researcher at the Centre for Digital Music, Queen Mary University of London. She is part of the AI and Music, Centre for Doctoral Training. Her research focus is mainly on exploring the idea of Music Mix similarity, Music Mix Style transfer, and Intelligent Multitrack Mixing using Self-Supervised, Semi-Supervised, and Unsupervised Learning architectures. She also writes music, produces and plays saxophone. Her educational background is a mixture of Masters in Physics and Courses in Music Production.

Gary Bromham is a part-time PhD researcher at Queen Mary University of London, researching the role that traditional studio paradigms and retro aesthetics play in intelligent music production systems (2016 -). He has several publications in this field and has contributed a chapter to the recent Routledge publication, ‘Perspectives on Music Production: Mixing Music’ (2017). He was also a research assistant on the EPSRC funded project called FAST (Fusing Audio and Semantic Technologies) where he is employed as an industry advisor (2017 - 2020). In addition to his research interests, Gary is a practising music producer, songwriter and audio engineer, with over 30 years’ experience (1989 - 2020). He has worked with artists as diverse as Bjork, Wham, Blur and U2, during a period that has witnessed several technological changes. Gary is well versed in most popular music making software and has extensive knowledge of using analog hardware, acting as a product designer and specialist for the renowned mixing desk company, Solid State Logic. He is also a frequent guest lecturer and external advisor at several universities in the UK, Norway and Sweden; speaking on songwriting, music production aesthetics and audio engineering and bringing some of his extensive knowledge and experience to both Undergraduate and Master’s degree level programs.

Marco A. Martínez Ramírez is music technology researcher at Sony in the Tokyo R&D center, where he is part of the Creative AI Lab. His research interests lie at the intersection of machine learning, digital signal processing, and intelligent music production, with a primary focus on deep learning architectures for music processing tasks. Previously, he was an audio research intern at Adobe and received his PhD from the Centre for Digital Music at Queen Mary University of London. He has a MSc in digital signal processing from the University of Manchester, UK, and a BSc in electronic engineering from La Universidad de Los Andes, Colombia. Marco also has a background in music production and mixing engineering.

Tutorial 6

Trustworthy MIR: Creating MIR applications with values

Christine Bauer, Andrés Ferraro, Emilia Gómez, Lorenzo Porcaro

Abstract

The MIR community shows an increasing interest in understanding how current technologies affect the everyday experience of people all over the world, e.g., how we listen to music, compose songs, or learn to play an instrument. As it was introduced in the FAT-MIR tutorial held at ISMIR 2019, a great discussion has aroused around the ethical, social, economic, legal, and cultural implications that the use of MIR systems have in our life.

In this tutorial, we aim at building upon and expanding the aforementioned debate, discussing the more recent results obtained by the MIR community and beyond. The goal of the tutorial is to show how values, such as fairness and diversity, can be embedded in the life cycle of MIR systems to make them trustworthy: from algorithmic design to evaluation practices and regulatory proposals. To achieve that, we will discuss examples of, among the others, popularity bias, gender bias, algorithmic bias, music styles underrepresentation, and diversity-related phenomena (e.g. filter bubbles).

This tutorial is suitable for researchers and students in MIR working in any domain, as these issues are relevant for all MIR tasks. The examples will mostly focus on music information retrieval and recommendation, but there are no prerequisites for taking this tutorial. Besides presenting recent research insights, the tutorial will integrate two hands-on sessions, where we will involve the participants in reflecting on the design of evaluation methods that take into account values for which MIR systems should be accountable.

Biography of the Presenter

Christine Bauer is an assistant professor at the Department of Information and Computing Sciences at Utrecht University, The Netherlands. Her research activities center on interactive intelligent systems. Recently, she focuses on context-aware (music) recommender systems. A core interest in her research activities are fairness in algorithmic decision-making and multi-method evaluations. Her research and teaching activities are driven by her interdisciplinary background. She holds a Doctoral degree in Social and Economic Sciences, a MSc in Business Informatics, and a Diploma degree in International Business Administration. In addition, she pursued studies in jazz saxophone. Christine holds several best paper awards and awards for her reviewing activities. Furthermore, she received the Elise Richter grant by the Austrian Science Fund. Before joining Utrecht University, she was a researcher at Johannes Kepler University Linz, WU Wien, and EC3 (Austria), and University of Cologne (Germany). In 2013 and 2015, she was a Visiting Fellow at Carnegie Mellon University (PA, USA). Christine has co-organized the workshop PERSPECTIVES 2021 at RecSys 2021 and IUadaptMe 2019 at UMAP 2019. At UMAP 2021, she gave a tutorial on Multi-Method Evaluation of Adaptive Systems. Furthermore, she was a co-chair for the Doctoral Symposium at RecSys 2021.

Andrés Ferraro (BSc/MSc in Software Engineering) is a Postdoctoral Fellow at McGill University and Mila (Quebec AI Institute), Canada. He completed his PhD at the Department of Information and Communication Technologies and Engineering of the Universitat Pompeu Fabra, Spain. His thesis uncovers multiple dimensions in which music recommender systems affect the artists and proposes alternatives to mitigate such problems. He is currently part of an interdisciplinary project, rethinking music recommender systems by considering new and alternative conceptions from the social sciences and humanities, informed by non-profit systems and critical debates over bias and discrimination. He is co-organizer of LatAm Bish Bash, a series of meetings and networking events that connect engineers, researchers, and students working on music and audio signal processing.

Emilia Gómez (BSc/MSc in Electrical Engineering, PhD in Computer Science) is Principal Investigator on Human and Machine Intelligence (HUMAINT) team at the Joint Research Center (European Commission). She is also a Guest Professor at the Music Technology Group, Universitat Pompeu Fabra, Barcelona. Her research is grounded on the Music

Information Retrieval field, where she has developed data-driven technologies to support music listening experiences. Starting from music, she studies the impact of artificial intelligence (AI) on human decision making, cognitive and socio-emotional development. Her research interests include fairness and transparency in AI, the impact of AI on jobs, and how it affects children development.

Lorenzo Porcaro (MSc Sound and Music Computing and Intelligent Interactive Systems) is a PhD candidate at the Music Technology Group, Universitat Pompeu Fabra (UPF), Spain. His research is at the intersection between Music Information Retrieval and Social Computing, and he is currently working on the assessment of the impact of music recommender systems on cultural diversity. He has collaborated in several initiatives focused on the analysis of ethical dimensions of algorithmic systems (Mechanism Design for Social Good (MD4SG); divinAI project, HUMAINT / UPF). He has also been part of national and international research projects aiming at making music more accessible through the use of technology (Musical AI, TROMPA).

Papers - Session I

INTERPRETING SONG LYRICS WITH AN AUDIO-INFORMED PRE-TRAINED LANGUAGE MODEL

Yixiao Zhang[1] **Junyan Jiang**[2,3] **Gus Xia**[2,3]
Simon Dixon[1]
[1] Centre for Digital Music, Queen Mary University of London
[2] Music X Lab, NYU Shanghai [3] MBZUAI
yixiao.zhang@qmul.ac.uk, jj2731@nyu.edu, gxia@nyu.edu, s.e.dixon@qmul.ac.uk

ABSTRACT

Lyric interpretations can help people understand songs and their lyrics quickly, and can also make it easier to manage, retrieve and discover songs efficiently from the growing mass of music archives. In this paper we propose BART-fusion, a novel model for generating lyric interpretations from lyrics and music audio that combines a large-scale pre-trained language model with an audio encoder. We employ a cross-modal attention module to incorporate the audio representation into the lyrics representation to help the pre-trained language model understand the song from an audio perspective, while preserving the language model's original generative performance. We also release the Song Interpretation Dataset, a new large-scale dataset for training and evaluating our model. Experimental results show that the additional audio information helps our model to understand words and music better, and to generate precise and fluent interpretations. An additional experiment on cross-modal music retrieval shows that interpretations generated by BART-fusion can also help people retrieve music more accurately than with the original BART. [1]

1. INTRODUCTION

Lyrics play a key role in the understanding and creation of songs, expressing emotions and delivering messages in the form of natural language [1]. Lyrics have both linguistic and musical characteristics: the field of Lyric Information Processing (LIP) can consequently be seen as a bridge between Music Information Retrieval (MIR) and Natural Language Processing (NLP), encompassing a range of new challenges such as lyric structure analysis [2], lyric semantic analysis [3], automatic lyric generation [4], and lyric understanding [5]. In this paper, we focus on the task of multimodal lyric interpretation, which requires the model to understand both the words and music of a song, and to produce a natural, concise and human-like description of its lyrics.

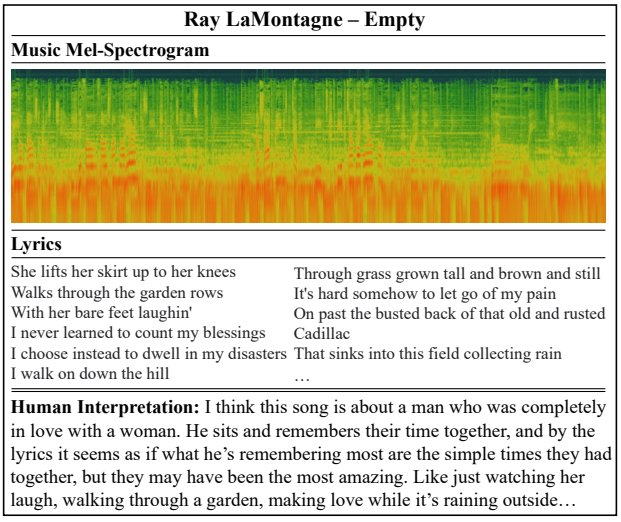


Figure 1. An example of lyrics and their interpretation in real-life, where the interpretation is written by a human. Information from the audio modality includes the representations of instruments, styles, chords, etc., which may help the model to understand the meaning of the lyrics.

In real life, human interpretations of lyrics often contain both a general summary of the lyrical theme and a detailed analysis in relation to specific lines. Figure 1 shows such an example. Considering that the human interpretation of the lyrics contains subjective elements, the lyrics interpretation task is like an extension of the lyrics summarisation task. The task requires the model to be able to (1) select an excerpt from the lyrics, as in extractive text summarisation; (2) generate explanatory text from lyrics, which is similar to abstractive text summarisation. However, previous summarisation methods for general texts [6, 7] are not necessarily applicable in the context of lyrics, because song lyrics often contain rich metaphors, poetic themes, and a high degree of rhythm [2]. Previous studies have attempted to apply extractive summarisation to song lyrics using TextRank algorithms [8] and audio-text alignment algorithms [9]. The drawback of such approaches is that the summary by itself is not enough to explain the lyrics.

[1] Open-sourced code and pretrained models: https://github.com/ldzhangyx/BART-fusion.

© Yixiao Zhang, Junyan Jiang, Gus Xia and Simon Dixon. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Yixiao Zhang, Junyan Jiang, Gus Xia and Simon Dixon, "Interpreting Song Lyrics with an Audio-Informed Pre-trained Language Model", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

To the best of our knowledge, this paper is the first study to use both extractive and abstractive methods to generate lyric interpretations.

Compared to unimodal lyric interpretation models, multimodal models can use information from the music audio domain, such as style, emotion and instrument representations, to reduce the difficulty of understanding lyrics and to improve the quality of the generated text. In recent years, Transformer-based multimodal generative pre-training models have performed well for tasks such as text understanding [10, 11] and text generation [12, 13]. Some related works have attempted to adapt existing pre-trained language models to multimodal tasks [14] or conditional generation tasks [13, 15]. The model we introduce in this paper is inspired by Yu *et al.* [14] and we choose to adapt it from a pre-trained language model (BART) [7]. Our model makes use of two modalities: the text of the song lyrics and the corresponding music audio. We add a convolutional encoder (CNN-SA) [16] to extract a representation of the audio and transfer it to the text domain by computing the similarity between the semantics of the song lyrics and the audio representation through a cross-modal attention mechanism, implemented by a multi-headed attention layer [17]. The transformed music audio representation is then fused into the semantic representation of the lyrics as an additional embedding. We discuss more details of the model in Section 2. To train and evaluate our model, we propose a new dataset, the Song Interpretation Dataset, which contains 27,834 songs with 490,000 corresponding user interpretations. This is the first large-scale open-source dataset for lyric interpretation. We describe the dataset in detail in Section 3.

We evaluate our model against the original BART model (as a baseline) on the Song Interpretation Dataset, as described in Section 4. On the lyric interpretation task, our model outperforms the baseline model according to the standard text summarisation metrics ROUGE, METEOR, and BERT-Score. Ablation experiments show that our dataset filtering techniques also improve model performance. To show the value of our model for other tasks, we also present experimental results demonstrating that it performs better than the baseline on a cross-modal retrieval task.

The main contributions of this work can be summarised as follows:

1. We present BART-fusion, the first multimodal generative model for lyric interpretation. We investigate the integration of audio representations with lyric representations and show that audio representations can improve the performance of lyric interpretation models;
2. We contribute a large-scale multimodal dataset containing paired audio, lyrics, and lyric interpretations that can be used for music understanding tasks such as lyric interpretation.

2. METHOD

In this section, we first revisit the original BART model in Section 2.1. We then discuss the approach to extract musical features from an audio spectrogram in Section 2.2. Finally, we introduce the music-text representation fusion mechanism in Section 2.3. We identify text-domain features with the subscript t and music-domain features with m.

2.1 BART Model for the Generative Task

Transformer-based pre-trained encoder-decoder language models such as BART [7], MASS [18] and T5 [19] generalize BERT [20] (due to the bidirectional encoder) and GPT [21] (with the left-to-right decoder), achieving good results on sequence-to-sequence tasks such as text summarisation and machine translation. Our model takes advantage of the text generation ability of BART, the structure of which is shown on the right side of Figure 2.

The lyric text input is firstly tokenized and embedded. We assume the lyric text sequence has L_t tokens, and the embedding dimension is d_t, resulting in an embedding $X_t \in \mathbb{R}^{L_t \times d_t}$. Following Vaswani *et al.* [22], we add an absolute positional embedding E_{pe} to get the final input features H_t^0:

$$H_t^0 = X_t + E_{\text{pe}}. \tag{1}$$

These input features are then passed to the encoder. The encoder has a stack of six layers, as illustrated in Figure 2, where a single Transformer layer is shown by a yellow box. Each Transformer layer contains a multi-head Self-Attention (SA) module and a Feed-Forward Network (FFN), each followed by a Layer Normalization (LN) module. For the i-th layer, the representation is calculated as:

$$\widetilde{H}_t^i = \text{LN}(\text{SA}(H_t^{i-1}W_Q, H_t^{i-1}W_K, H_t^{i-1}W_V)W_a + H_t^{i-1}) \tag{2}$$

$$H_t^i = \text{LN}(\text{FFN}(\widetilde{H}_t^i) + \widetilde{H}_t^i), \tag{3}$$

where $H_t^i \in \mathbb{R}^{L_t \times d_t}$, and $W_Q \in \mathbb{R}^{d_t \times d_a}$, $W_K \in \mathbb{R}^{d_t \times d_a}$ and $W_V \in \mathbb{R}^{d_t \times d_a}$ denote linear transformation matrices which map the representations to a common space. $W_a \in \mathbb{R}^{d_a \times d_t}$ linearly projects the attention value back to the desired dimensionality.

The decoder also consists of a stack of six Transformer layers, which is similar to the encoder. But the multi-head self-attention module in the decoder is masked to respect causality, and an additional multi-head encoder-decoder attention is introduced to incorporate the encoder representation.

2.2 Audio Encoder

For the multimodal lyric interpretation model, we expect the music audio modality to provide some additional semantic information, such as style, mood, instrumentation, etc., to help the model understand the lyrics better. We

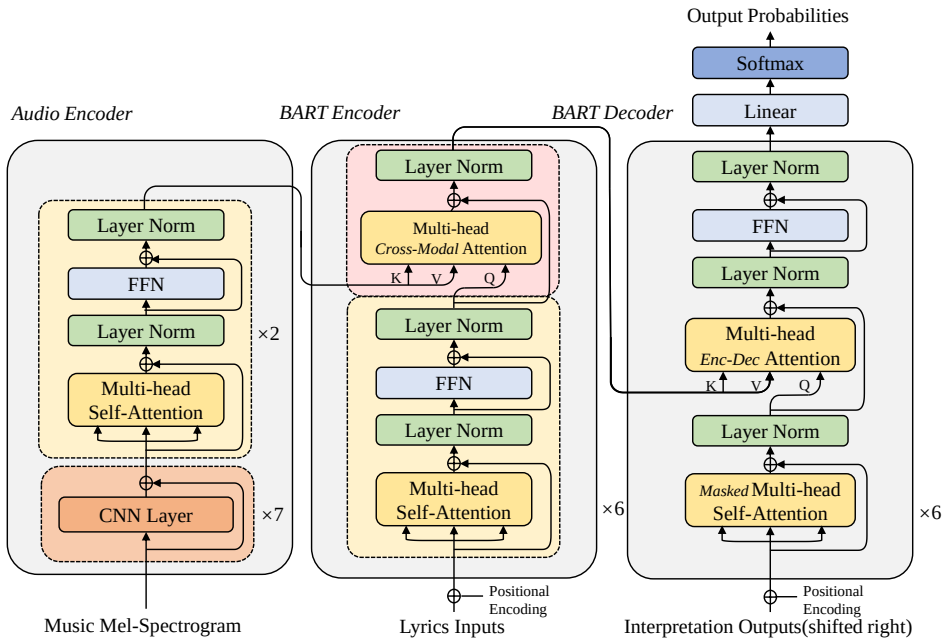


Figure 2. An overview of our proposed model. The model is divided into three parts from left to right: the audio encoder, the BART encoder, and the BART decoder. The fusion of the semantic representation of music audio and lyric text occurs in the upper part of the middle module (pink background). Only at the last two layers of the BART encoder, the music audio representation and the lyric text representation are semantically fused: the music audio representation and the lyric text representation are fed into the cross-modal attention module, and the result is added as an additional embedding to the original lyric text representation. The fused representation is fed into the BART decoder to generate an interpretation.

design an audio encoder to extract a representation following Won *et al.* [16]. The audio encoder uses a stack of CNN layers as a filter to extract local features, followed by a self-attention module to capture the global and temporal features of the audio.

The audio encoder receives an audio clip X_m and transforms it to a mel spectrogram H_m^0 as input. We compute the feature map of the i-th layer of the encoder as follows:

$$\widetilde{H}_m^i = \text{BN}(\text{CNN}_2(\text{ReLU}(\text{CNN}_1(H_m^{i-1})))) \quad (4)$$

$$H_m^i = \widetilde{H}_m^i + \text{BN}(\text{CNN}_3(H_m^{i-1})), \quad (5)$$

where BN is the Batch Normalization operation, and the CNNs are convolutional modules with different parameters. We add a residual connection to each CNN layer.

We then add two Transformer layers, identical to those in the BART encoder, to extract the final representation of music audio. Finally, we get the music audio representation $Z_m \in \mathbb{R}^{L_m \times d_m}$, where L_m and d_m denote the shape of the music audio feature map at the last CNN layer.

2.3 Representation Fusion

As shown in Figure 2, we insert a representation fusion module into the BART encoder to incorporate musical information. Inspired by Tsai *et al.* [17] and Yu *et al.* [14], we apply cross-modal attention to enable the music audio representation to be transferred to the lyric text domain. Jawahar *et al.* [23] have shown that BART encoders tend to extract semantic information in the last few layers, so we only fuse semantic representations at the final two Transformer layers.

For a specific Transformer layer i, we have the lyric text representation $H_t^i \in \mathbb{R}^{L_t \times d_t}$ and the music audio representation $Z_m \in \mathbb{R}^{L_m \times d_m}$, which is the same for each layer. We calculate the domain-adapted music audio representation with a multi-head Cross-Modal Attention (CMA) module:

$$H_{m \to t}^i = \text{CMA}(H_t^i W_Q', Z_m W_K', Z_m W_V') W_a', \quad (6)$$

where $H_{m \to t}^i \in \mathbb{R}^{L_t \times d_t}$ and the symbol $m \to t$ denotes cross-modal attention from music audio to the lyric domain. $d_{a'}$ is the dimensionality of the attention module, and similar to Eq. 3, $W_Q' \in \mathbb{R}^{d_t \times d_{a'}}$, $W_K' \in \mathbb{R}^{d_m \times d_{a'}}$ and $W_V' \in \mathbb{R}^{d_m \times d_{a'}}$ denote linear transformation matrices which map the representations to a common space. $W_a' \in \mathbb{R}^{d_{a'} \times d_t}$ linearly projects the attention value back to the lyric text dimension.

We add the cross-domain representation as an additional embedding [24] to the lyric text representation to get the final representation for the last two layers of the BART encoder:

$$H_{\text{fusion}}^i = H_t^i + H_{m \to t}^i. \quad (7)$$

3. SONG INTERPRETATION DATASET

The lack of suitable datasets has prevented deep learning models from learning to describe songs in natural lan-

guage. Related studies [25, 26] refer to some datasets, but they share two common problems: 1) the amount of data is small and 2) the datasets are not open-sourced. We propose a new dataset, the Song Interpretation Dataset, for the lyric interpretation task. [2]

The Song Interpretation Dataset combines data from two sources: (1) music and metadata from the Music4All Dataset [27], and (2) lyrics and user interpretations from SongMeanings.com [3] . We design a music metadata-based matching algorithm that aligns matching items in the two datasets with each other. In the end, we successfully match 25.47% of the tracks in the Music4All Dataset.

The dataset contains audio excerpts from 27,834 songs (30 seconds each, recorded at 44.1 kHz), the corresponding music metadata, about 490,000 user interpretations of the lyric text, and the number of votes given for each of these user interpretations. The average length of the interpretations is 97 words. Music in the dataset covers various genres, of which the top 5 are: Rock (11,626), Pop (6,071), Metal (2,516), Electronic (2,213) and Folk (1,760).

To the best of our knowledge, this is the first large-scale open-source dataset for lyric interpretation. A comparison with similar datasets is shown in Table 1.

Dataset	Music	Interpretation	Public
Choi *et al.* [25]	800	2000	×
Manco *et al.* [26]	17,354	17,354	×
Ours	**27,384**	**490,000**	✓

Table 1. A comparison of our dataset with previous music description datasets.

We observe three main issues with interpretations written by real users: (1) some interpretations are very short or very long; (2) interpretations can contain content unrelated to the lyrics themselves, and (3) some interpretations are of low quality. We therefore preprocess the dataset using two techniques:

1. We **remove overly short interpretations** with length less than 256 characters to improve data representativeness, since we find that sentences below this length are often meaningless interpretations. For interpretations longer than 2048 characters, we keep only the first 2048 characters, but ensure that the last word is complete.
2. We use a **voting-based filtering mechanism** to improve data quality. Every interpretation on SongMeanings.com has a voting result attached, indicating how much the community approves of it, so an interpretation with a higher vote is more likely to be a high-quality interpretation. We therefore create two subsets, keeping only interpretations with positive votes and interpretations with non-negative votes.

[2] The Song Interpretation Dataset is anonymously open for downloading: `https://doi.org/10.5281/zenodo.7019124`.

[3] `https://songmeanings.com/`

To enable the model to be comparable across datasets, we manually select 800 interpretations and use them as a test dataset after excluding them from the original dataset. We have specifically removed all songs that appeared in the test set from the training set to avoid data leakage issues. After the above preprocessing, the dataset has 3 different subsets with the information shown in Table 2.

Dataset Name	Train	Valid.	Test
Raw dataset	440,000	50,000	800
Dataset Full	279,283	31,032	800
Dataset w/vote ≥ 0	265,360	29,484	800
Dataset w/vote > 0	49,736	5,526	800

Table 2. A comparison of dataset sizes (in number of interpretations) with different filtering methods.

4. EXPERIMENTAL SETTINGS

4.1 Implementation Details

We pre-process the lyric text input data by truncating or padding text to 2048 tokens. All the audio signals are downsampled to 16,000 Hz sample rate and converted to short-time Fourier transform representations with a 512-point FFT and 50%-overlapping Hann window. Finally, we convert these to log mel spectrograms with 128 bins.

We use `BART-base` [7] as the pre-trained language model to construct BART-fusion, which has a 6-layer encoder and decoder. For the audio encoder (CNNSA), we use 3×3 kernels for all layers with [128, 128, 256, 256, 256, 256, 256] channels and [(2, 2), (2, 2), (2, 2), (2, 1), (2, 1), (2, 1), (2, 1)] strides. The cross-modal attention module has 1 head and 1 layer, with $d_a = 768$.

We use AdaFactor [28] as the optimizer. We set the learning rate to 6×10^{-4} and reduce it to 6×10^{-5} from the 11th epoch. For all experiments, we use a batch size of 8. We train all models for 20 epochs with early stopping of 3 epochs using the ROUGE-1 score on the validation set.

4.2 Evaluation Metrics

Since there are no existing metrics for the lyric interpretation task, we borrow several complementary metrics from similar tasks to evaluate the performance level of the model in a comprehensive manner. We use ROUGE, METEOR and BERT-Score to evaluate the generated interpretations. ROUGE is considered as the main metric for evaluation, because our task is closest to the text summarisation task.

4.2.1 ROUGE-{1, 2, L}

ROUGE [29] is a common metric for evaluating abstractive text summaries. It calculates the overlap of 1-gram phrases (R-1), 2-gram phrases (R-2) and their weighted results (R-L). In the context of lyrics, a higher ROUGE score indicates that the generated explanatory text leaves out less necessary information, which is better.

Training dataset	Method	Data size	R-1	R-2	R-L	METEOR	BERT-Score
Dataset w/random	BART	56,470	40.0	12.5	21.7	21.1	**83.7**
Dataset w/random	BART-fusion	56,470	**42.1***	**13.6***	**23.4***	**22.0***	83.3
Dataset w/voting > 0	BART	56,470	41.2$_{+}$	13.0$_{+}$	22.8$_{+}$	22.0$_{+}$	**83.6**
Dataset w/voting > 0	BART-fusion	56,470	**44.3**$^{*}_{+}$	**14.6**$^{*}_{+}$	**24.7**$^{*}_{+}$	**22.6**$^{*}_{+}$	83.3
Dataset Full	BART	316,478	44.1	14.0	24.5	22.5$_{+}$	**83.5**
Dataset Full	BART-fusion	316,478	**46.1***	**15.0***	**25.1***	**23.0***	**83.5**
Dataset w/voting ≥ 0	BART	300,712	44.8$_{+}$	14.9$_{+}$	24.7	22.7	83.9
Dataset w/voting ≥ 0	BART-fusion	300,712	**46.7**$^{*}_{+}$	**15.6**$^{*}_{+}$	**25.5**$^{*}_{+}$	**23.4***	**84.1**

Table 3. Evaluation results of BART-fusion and BART (baseline) on the Song Interpretation Dataset with different settings. $*$: BART-fusion outperforms BART with $p < 0.05$; $+$: the filtered dataset outperforms the unfiltered dataset with $p < 0.05$.

4.2.2 METEOR

METEOR [30] takes into account similar semantic information such as synonyms through WordNet and calculates the similarity score based on F-measure. It complements ROUGE by jointly measuring how semantically similar the model-generated lyric interpretation is to the reference text.

4.2.3 BERT-Score

BERT-Score [31] is mainly used to evaluate the naturalness and fluency of the generated text. We expect that incorporating the audio modality should not sacrifice the generative performance of the language model.

We use `rouge` [4], `nltk` [5] and `bert-score` [6] respectively to compute these metrics.

5. RESULTS AND ANALYSIS

5.1 Main Results

We train BART-fusion and the corresponding original BART on several different dataset settings. Results are shown in Table 3, where all values are the means of multiple independent repeated experiments. We perform a *paired Student's t-test* on the results of BART-fusion and BART, and the results on the filtered datasets and unfiltered datasets respectively, where p-value is 0.05.

Our first finding is that applying a voting-based filtering mechanism to the dataset significantly improves the performance of the models (both BART-fusion and baseline) on this task. For fairness, we take a random subset of *Dataset Full*, which we call *Dataset w/random*, to match the size of *Dataset w/voting* > 0. (We still use *Dataset Full* to compare *Dataset w/voting* ≥ 0.) The experimental results show that the performances of both BART-fusion and the baseline are significantly improved on the filtered datasets.

The experimental results also show that BART-fusion significantly outperforms the baseline, and generates more precise interpretion text for lyrics. For all dataset settings, our BART-fusion models show better performance on the ROUGE and METEOR scores.

Finally, we find that BART-fusion preserves the generative performance of the pre-trained language model while improving the generation accuracy. The performance of BART-fusion is essentially equal to that of baseline on the BERT-Score metric, which is the metric of text quality. It means that the introduction of audio modal information affects the model mainly semantically and does not affect the naturalness or fluency.

5.2 Case Study and Error Analysis

We observe that adding music modality information usually brings the benefits of accurate understanding of the theme, selecting highlighted lyric lines, and emotive sentences from the generated samples. In the case study, we select a representative example showing the generation results from different models with the same lyric input, as shown in Table 4. In the first lines, BART-fusion explains the theme of the lyrics more accurately than BART. Then, BART-fusion selects the highlighted lyric lines and gives further detailed interpretation, while the text generated by BART lacks a clear explanation of them. We find that BART-fusion talks about the mood of the song at the end, which is not present in the original BART example. [7]

We have noticed that when the lyrics are about complex topics, such as religious and philosophical topics, BART-fusion and BART sometimes fail to understand the meaning correctly and generate text that is only superficially correct, which we interpret as a lack of common sense often observed in this class of deep learning models.

5.3 Cross-Modal Retrieval Analysis

In addition, we design a cross-modal retrieval experiment to test the retrieval capability of our models. We randomly extract a sentence from the real interpretation in the test set as a query string, use this to search over a database of lyric text and musical audio on a semantic level, and return a ranked list of all possible results. We expect the correct result corresponding to the query to be as close to the top of this ranking as possible.

To accomplish this task, the model generates interpretations for all song lyrics in the database, calculates their semantic features using Sentence-BERT [32], and stores

[4] `https://github.com/pltrdy/rouge`
[5] `https://github.com/nltk/nltk`
[6] `https://github.com/Tiiiger/bert_score`

[7] We provide a demo page for more examples: `https://sites.google.com/view/bart-fusion-ismir2022`.

Lyrics: I cannot find a way to describe it / It's there inside, all I do is hide / I wish that it would just go away / What would you do, you do, if you knew? / What would you do? / / All the pain I thought I knew / All the thoughts lead back to you / Back to what was never said / Back and forth inside my head / I can't handle this confusion / I'm unable, come and take me away / / I feel like I am all alone / All by myself, I need to get around this / My words are cold, I don't want them to hurt you / If I show you, I don't think you'd understand / 'Cause no one understands / / All the pain I thought I knew / All the thoughts lead back to you / Back to what was never said / Back and forth inside my head / I can't handle this confusion / I'm unable, come and take me away / ...
Reference: I think this song is about a crush that the singer is afraid to tell the person they are in love with. She wonder if maybe the other person feels the same, but is also too afriad to tell her. Thinking "what if". Since, she won't admit it she'll never know and this not knowing ends up eating her up inside. Each day she wonders if she could be with this person who she really loves, but she just gets frustrated and eventually wishes that she never even fell in love with the person since it would make her life so much easier. It builds up so much that she just wishes someone would "come and take me away". That could mean two things. It could be death/suicide because she decides it's not worth it or she could just wish for someone else to date so she can just say she's taken and not have to worry about this crush.
BART: I think this song is about a girl who is in love with a guy, but she doesn't know how to tell him because she's afraid he'll hurt her. She's trying to convince herself that he loves her, but he doesn't understand how she feels about him. She wants him to come and take her away from all the pain she's caused and make her feel better about herself. This is what I think of when I listen to this song. I think it's about someone who is trying to tell someone they love them, but they don't want to tell them because they're afraid that they'll hurt them. I love this song, it's one of my favorites on the album.
BART-fusion: I think this song is about a girl who is in love with a guy but she is afraid to tell him because she doesn't want to ruin the friendship. In the first verse she is talking about how she can 't find a way to describe how she feels and how she wishes it would just go away. The second verse is about how the guy is telling her how he feels but she isn't sure if he really feels the same way. The chorus is her saying how she is confused about what to say and how to tell the guy. "I'm unable, come and take me away" means that she is unable to tell her feelings and she is trying to get the guy out of her head. The last verse is her telling the guy that she feels like she is going nowhere in her life, and she just wants him to take her away from all of this. That's just my take on it. I can totally relate to this song. It's a very powerful song.

Table 4. A generation example for Avril Lavigne - *Take Me Away*.

features in advance. At query time, we also use Sentence-BERT to compute the semantic feature for the query string. We return the song whose feature is most similar to the query feature by computing the cosine similarities between the query feature and features in the database.

We use Mean Reciprocal Rank (MRR) [33], a common metric for information retrieval tasks, to measure the performance of the models on this task. Formally, for a set of query strings $Q = \{q_1, q_2, ..., q_m\}$ and the corresponding database $S = \{s_1, s_2, ..., s_n\}$, the model outputs a list of songs sorted by probability for the i-th query, where the correct song is ranked k_i-th, and the model is scored as:

$$\text{score}_i = \frac{1}{k_i}, \text{where } k_i \leq n. \tag{8}$$

The final MRR is calculated by averaging the scores:

$$\text{MRR} = \frac{1}{m}\sum_{i=1}^{m}\text{score}_i = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{k_i}. \tag{9}$$

From Table 5, we find that BART-fusion outperforms the original BART on all 4 dataset settings, i.e., the interpretations generated by BART-fusion can help users find music more accurately. If we return a random ranking result, the MRR value is about 0.9%, which is much lower than the model performance.

Dataset	Method	MRR (%)
Dataset w/random	BART	25.3
Dataset w/random	BART-fusion	**27.5***
Dataset w/voting > 0	BART	26.1_+
Dataset w/voting > 0	BART-fusion	$\mathbf{28.2}^*_+$
Dataset Full	BART	26.2
Dataset Full	BART-fusion	**30.5***
Dataset w/voting ≥ 0	BART	26.4
Dataset w/voting ≥ 0	BART-fusion	$\mathbf{32.5}^*_+$

Table 5. Evaluation results for the music retrieval task. $*$: BART-fusion outperforms BART with $p < 0.05$; $+$: the filtered dataset outperforms the unfiltered dataset with $p < 0.05$.

6. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a novel model to generate interpretations from song lyrics and musical audio. Our modelis based on a pre-trained language model and generates better lyric interpretations by fusing text and music semantic representations. We also have proposed a new dataset and explored the process of dataset creation, and have investigated how different treatments of the dataset can affect the performance of the model. We have designed an additional experiment that shows that our model outperforms BART on cross-modal music retrieval tasks. Our work has a range of potential applications, such as helping people better understand English lyrics (especially for non-native English speakers) and natural language based music discovery.

The current model still has shortcomings, such as difficulty in understanding complex topics and lack of general common sense. In future work, we will try to improve the model by adding metadata and knowledge base inputs. We also plan to extend the model to describe the musical content of a song in natural language. In addition, large-scale language models may be biased, reinforce stereotypes and introduce harms, which deserves our attention in the future.

7. ACKNOWLEDGEMENT

We want to thank Mark Levy for his great contribution to this work. We also thank Yin-Jyun Luo and Liming Kuang for their enthusiastic help during the writing process of the paper. Yixiao Zhang is a research student at the UKRI Centre for Doctoral Training in Artificial Intelligence and Music, supported jointly by the China Scholarship Council, Queen Mary University of London and Apple Inc.

8. REFERENCES

[1] K. Watanabe and M. Goto, "Lyrics information processing: Analysis, generation, and applications," in *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*, 2020, pp. 6–12.

[2] M. Fell, Y. Nechaev, E. Cabrio, and F. Gandon, "Lyrics segmentation: Textual macrostructure detection using convolutions," in *COLING*, 2018, pp. 2044–2054.

[3] R. Delbouys, R. Hennequin, F. Piccoli, J. Royo-Letelier, and M. Moussallam, "Music mood detection based on audio and lyrics with deep neural net," *arXiv preprint arXiv:1809.07276*, 2018.

[4] Z. Sheng, K. Song, X. Tan, Y. Ren, W. Ye, S. Zhang, and T. Qin, "Songmass: Automatic song writing with pre-training and alignment constraint," *arXiv preprint arXiv:2012.05168*, 2020.

[5] A. Tsaptsinos, "Lyrics-based music genre classification using a hierarchical attention network," *arXiv preprint arXiv:1707.04678*, 2017.

[6] K. Al-Sabahi, Z. Zuping, and M. Nadher, "A hierarchical structured self-attentive model for extractive document summarization (HSSAS)," *IEEE Access*, vol. 6, pp. 24 205–24 212, 2018.

[7] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[8] J. Son and Y. Shin, "Music lyrics summarization method using textrank algorithm," *Journal of Korea Multimedia Society*, vol. 21, no. 1, pp. 45–50, 2018.

[9] M. Fell, E. Cabrio, F. Gandon, and A. Giboin, "Song lyrics summarization inspired by audio thumbnailing," in *RANLP*, 2019.

[10] W. Li, C. Gao, G. Niu, X. Xiao, H. Liu, J. Liu, H. Wu, and H. Wang, "UNIMO: Towards unified-modal understanding and generation via cross-modal contrastive learning," *arXiv preprint arXiv:2012.15409*, 2020.

[11] Y. Zhang, Z. Wang, D. Wang, and G. Xia, "BUTTER: A representation learning framework for bi-directional music-sentence retrieval and generation," in *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*, 2020, pp. 54–58.

[12] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. Corso, and J. Gao, "Unified vision-language pre-training for image captioning and VQA," in *AAAI*, vol. 34, no. 07, 2020, pp. 13 041–13 049.

[13] J. Chen, H. Guo, K. Yi, B. Li, and M. Elhoseiny, "VisualGPT: Data-efficient adaptation of pretrained language models for image captioning," *arXiv preprint arXiv:2102.10407*, 2021.

[14] T. Yu, W. Dai, Z. Liu, and P. Fung, "Vision guided generative pre-trained language models for multimodal abstractive summarization," *arXiv preprint arXiv:2109.02401*, 2021.

[15] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, and R. Liu, "Plug and play language models: A simple approach to controlled text generation," *arXiv preprint arXiv:1912.02164*, 2019.

[16] M. Won, S. Chun, and X. Serra, "Toward interpretable music tagging with self-attention," *arXiv preprint arXiv:1906.04972*, 2019.

[17] Y.-H. H. Tsai, S. Bai, P. P. Liang, J. Z. Kolter, L.-P. Morency, and R. Salakhutdinov, "Multimodal transformer for unaligned multimodal language sequences," in *ACL*, vol. 2019. NIH Public Access, 2019, p. 6558.

[18] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MASS: Masked sequence to sequence pre-training for language generation," *arXiv preprint arXiv:1905.02450*, 2019.

[19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[21] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[23] G. Jawahar, B. Sagot, and D. Seddah, "What does BERT learn about the structure of language?" in *ACL*, 2019.

[24] C. Li, X. Gao, Y. Li, B. Peng, X. Li, Y. Zhang, and J. Gao, "OPTIMUS: Organizing sentences via pre-trained modeling of a latent space," *arXiv preprint arXiv:2004.04092*, 2020.

[25] K. Choi, J. H. Lee, X. Hu, and J. S. Downie, "Music subject classification based on lyrics and user interpretations," *AIST*, vol. 53, no. 1, pp. 1–10, 2016.

[26] I. Manco, E. Benetos, E. Quinton, and G. Fazekas, "Muscaps: Generating captions for music audio," *CoRR*, vol. abs/2104.11984, 2021. [Online]. Available: https://arxiv.org/abs/2104.11984

[27] I. A. P. Santana, F. Pinhelli, J. Donini, L. Catharin, R. B. Mangolin, V. D. Feltrim, M. A. Domingues *et al.*, "Music4all: A new music database and its applications," in *IWSSIP*. IEEE, 2020, pp. 399–404.

[28] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," in *ICML*. PMLR, 2018, pp. 4596–4604.

[29] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, 2004, pp. 74–81.

[30] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, 2005, pp. 65–72.

[31] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, "BERTScore: Evaluating text generation with BERT," *arXiv preprint arXiv:1904.09675*, 2019.

[32] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[33] O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan, "Expected reciprocal rank for graded relevance," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, 2009, pp. 621–630.

TOWARD POSTPROCESSING-FREE NEURAL NETWORKS FOR JOINT BEAT AND DOWNBEAT ESTIMATION

Tsung-Ping Chen **Li Su**
Institute of Information Science, Academia Sinica, Taiwan
{tearfulcanon, lisu}@iis.sinica.edu.tw

ABSTRACT

Recent deep learning-based models for estimating beats and downbeats are mainly composed of three successive stages—feature extraction, sequence modeling, and post processing. While such a framework is prevalent in the scenario of sequence labeling tasks and yields promising results in beat and downbeat estimations, it also indicates a shortage of the employed neural networks, given that the post-processing usually provides a notable performance gain over the previous stage. Moreover, the assumption often made for the post-processing is not suitable for many musical pieces. In this work, we attempt to improve the performance of joint beat and downbeat estimation without incorporating the post-processing stage. By inspecting a state-of-the-art approach, we propose reformulations regarding the network architecture and the loss function. We evaluate our model on various music data and show that the proposed methods are capable of improving the baseline approach without the aid of a post-processing stage.

1. INTRODUCTION

Beat and downbeat trackings have been of long-standing interests in the communities of signal processing and music information retrieval (MIR) [1–5]. The two tasks inherently possess the class imbalance issue [6], i.e., the highly imbalanced numbers of beat (downbeat) and non-beat (non-downbeat) frames in music data, and hence remain challenging in many cases even nowadays. To tackle the beat and downbeat tracking problems, early signal processing approaches have developed a three-stage framework which consists of *feature extraction*, *sequence modeling*, and *post-processing* [7–9]. Over the past few years, great progresses on the two tasks have been made through the combination of the three-stage pipeline and deep learning models such as recurrent neural networks (RNNs) [10–12], convolutional-recurrent neural networks (CRNNs) [13], and Transformer-based networks [14]. Among the thriving research, a convolutional approach achieves the state-of-the-art performance on joint estimation of tempo, beat, and downbeat [15], in which convolutional neural networks (CNNs) and temporal convolutional networks (TCNs) [16] are placed in charge of the feature extraction and of the sequence modeling, respectively, while dynamic Bayesian networks (DBNs) [17] are used for the post-processing.

In spite of the promising results obtained by this approach, the employed networks raise three concerns. First, the CNNs for feature extraction convolve an input spectrogram with small kernels at the very beginning, and a max-pooling layer is followed immediately. Therefore, the spectro-temporal patterns might not be well-captured [18]. Second, the dilated convolutions [19] of the TCNs involve a small kernel size and an exponentially increasing dilation rate. Despite the large receptive fields obtained at higher layers, this design leads to extremely sparse samplings which join distantly-separated frames, and consequently the contextual information could be irrelevant [20]. Lastly, the assumptions made for the DBNs that the meter is unchanged or the rhythmic patterns are known [11, 21] are not always applicable to various music genres, even to popular music. For example, the Hainsworth dataset [22], which is often employed for evaluation, includes several clips of pop or rock songs with changing meter. Moreover, the DBNs involve intricate design and assumptions for representing the state and the transition throughout a piece of music, and hence introduce non-trivial works in addition to the training of the preceding neural networks.

In this paper, we cope with the joint beat and downbeat estimation task by addressing the aforementioned issues. To get rid of the post-processing DBNs while maintaining the performance, we consider to improve the network architecture and the loss function. Precisely, we propose to use scaled depthwise separable convolutions [23–25] to aggregate rich contextual information at multiple time scales. To address the class imbalance issue, the focal loss [26] and the Dice loss [27] are employed in place of the commonly used cross entropy loss. Besides, we make our model aware of the periodic structure of beat or downbeat sequences by including a label embedding network [28] during training. We evaluate the proposed architecture on various music data, including both audio and MIDI files, and demonstrate a state-of-the-art performance on the Ballroom dataset. The main contribution of this work is that we thoroughly address the architectural issues for beat and downbeat estimations. Most of the problems are shared by sequence models in general, and therefore the proposed methods could be applied to many other MIR-related tasks.


© Tsung-Ping Chen and Li Su. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Tsung-Ping Chen and Li Su, "Toward postprocessing-free neural networks for joint beat and downbeat estimation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

2. OVERVIEW OF THE STATE-OF-THE-ART

The model of [15] without the DBNs is taken as our baseline upon which we build our new model to address the architectural issues mentioned beforehand. In the following, we briefly introduce the baseline model using the three-stage framework, and present the new model thereafter. We skip the post-processing stage for we aim to expel the DBNs from the system in this work. An overview of the baseline model is illustrated at the top of Figure 1.

2.1 Feature extraction

Given an audio signal represented as a log-magnitude spectrogram of size $T \times J$, where T denotes the number of time steps and J the number of frequency bins, the CNNs of the baseline model extract high-level features using three groups of 2-D convolution and max-pooling layers. The three convolutional layers (with a kernel size of 3×3, 1×12, and 3×3, respectively) [1] capture harmonic content at different frequency scales, while the max-pooling operations (with a kernel size of 1×3) reduce the frequency dimension in three steps. In other words, the CNNs gradually transcribe the harmonic information into high-level features, and output a sequence of size $T \times D$ with D indicating the dimension of high-level features.

2.2 Sequence modeling

The high-level feature sequence from the previous stage is then passed to the TCNs for learning temporal structure. As depicted at the top of Figure 1, each TCN layer performs two 1-D dilated convolutions in parallel with a kernel of size 5 and an exponentially increasing dilation rate 2^l (resp. 2^{l+1}), where l is the layer number. The outputs of the two dilated convolutions are concatenated and reprojected to keep the feature dimensionality constant. As a result, the TCNs efficiently enlarge the receptive field within a few layers with the amount of learnable parameters increasing linearly to the number of layers. Given that the TCNs has 11 layers, the two convolutions at the last layer will expand the kernels across over 4000 (resp. 8000) time frames with each element of the two kernels being spaced 1024 (resp. 2048) frames apart. Therefore, the temporal context modelled by the TCN is more than 80 seconds (assume a frame rate of 100 fps). Afterwards, the output of the TCNs is processed by the DBNs to obtain a sequence of beat or downbeat estimates.

2.3 Potential issues

In the community of computer vision, it has been shown that stacking small convolutional kernels followed by pooling layers is effectual for extracting high-level features of an image [29–31]. Such a architecture is often adopted by audio-based, or more specifically spectrogram-based research [32–34]. Considering the nature of images, it is reasonable to aggregate information hierarchically from spatial associations. However, it is questionable that a spectrogram could be well-represented with the same architecture. The characteristic spectral information of a musical piece does not necessarily reside in the adjacent frequency bins within a small kernel. Moreover, it is also suspicious that the *compressed* frequency dimension after pooling operations is as meaningful as a downsampled image.

On the other hand, it has been observed that dilated convolutions result in the so-called *gridding artifacts* [35, 36], due to that adjacent elements in the output are calculated from completely different sets of elements in the input. Additionally, small convolutional kernels with high dilation rates relate input elements which are highly sparsely distributed and hence might lack of correlations. In the scenario of beat or downbeat estimation, stacked 1-D dilated convolutions could be preferable for they are able to encode periodic structure of the beat (downbeat) through equidistant elements of dilated kernels. However, it is not guaranteed that an inquired musical piece has a steady tempo as well as a constant rhythm. Hence, a network equipped with dilated convolutions for modeling temporal information may have limitations on expressive music.

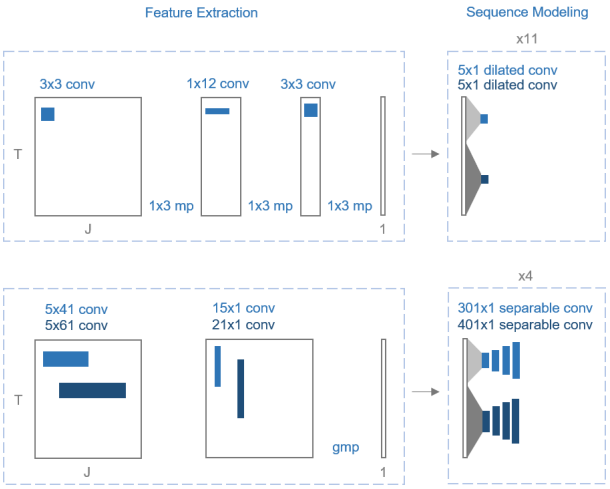


Figure 1: Architecture reformulation with respect to convolution (conv) and max pooling (mp) layers . Top: model of [15] without DBNs. Bottom: proposed architecture. The feature dimension is not shown in the figure.

3. APPROACH

3.1 Task formulation

We treat the joint beat and downbeat estimation task as a sequence labeling problem [37, 38]. Given an input spectrogram $\mathbf{X} \in \mathbb{R}^{T \times J}$, the task involves the assignment of a categorical label $\in \{\texttt{downbeat}, \texttt{xbeat}, \texttt{neither}\}$ [2] to each time step $t \in \mathrm{T}$. We employ a model architecture which mainly consists of a feature extraction network (f_{ex}), a sequence modeling network (f_{seq}), and an estima-

[1] A kernel size $M \times N$ specifies a convolution window across M time frames and N frequency bins.

[2] The label `xbeat` refers to a beat which is not a downbeat.

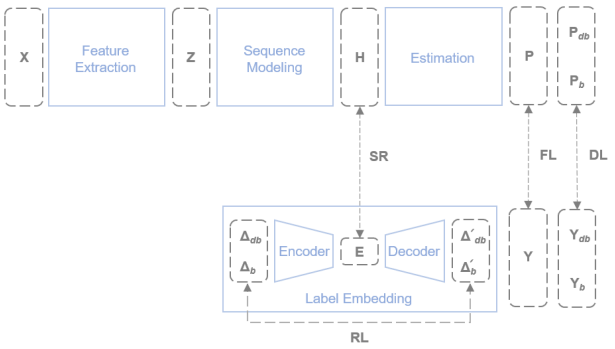


Figure 2: Proposed architecture. Dashed borders indicate data sequences, and solid ones denote neural networks. The losses includes focal loss (FL), Dice loss (DL), reconstruction loss (RL), and structural regularization (SR).

tion network (f_{est}) for the task:

$$\begin{aligned} \mathbf{P} &= f_{est}(\mathbf{H}), \\ \mathbf{H} &= f_{seq}(\mathbf{Z}), \\ \mathbf{Z} &= f_{ex}(\mathbf{X}), \end{aligned} \tag{1}$$

where $\mathbf{Z}$, $\mathbf{H} \in \mathbb{R}^{T \times D}$ are higher-level feature sequences, and $\mathbf{P} \in \mathbb{R}^{T \times 3}$ is a sequence of model estimates indicating the probabilities of the three categorical labels across all time steps. The overall model architecture is schematically shown in Figure 2.

3.2 Reformulation of feature extraction

We propose to use kernels of larger sizes for the 2-D convolution layers and remove the intermediate pooling layers in f_{ex}. As shown at the bottom of Figure 1, the feature extraction network consist of two layers, denoted as $f_{ex}^{(1)}$ and $f_{ex}^{(2)}$. The first layer captures harmonic information distributed across the frequency bins of an input spectrogram, while the second layer aggregates the harmonic information along the temporal dimension. Within each layer l, we employ two parallel convolutions with kernel sizes of $w_{ex,1}^{(l)}$ and $w_{ex,2}^{(l)}$. For these convolution operations, we keep the frequency dimension of the input spectrogram uncompressed, and thus the output shape will be $T \times J \times D$. Finally, we employ a global max pooling layer (GMP) [39] to reduce the frequency dimension (J) of the output, yielding the feature sequence $\mathbf{Z} \in \mathbb{R}^{T \times D}$. Let $f_c(\cdot, d, w)$ denote a standard convolution parameterized by the number of filters d and the kernel size w. The feature extraction network can be formulated as follows:

$$\begin{aligned} \mathbf{Z} = f_{ex}(\mathbf{X}) &= \underset{j \in J}{\mathrm{GMP}}(f_{ex}^{(2)}(f_{ex}^{(1)}(\mathbf{X}))), \\ f_{ex}^{(l)}(\cdot) &= [f_c(\cdot, D, w_{ex,1}^{(l)}), f_c(\cdot, D, w_{ex,2}^{(l)})]\mathbf{W}_{ex}^{(l)}, \end{aligned} \tag{2}$$

where $\mathbf{W}_{ex}^{(l)} \in \mathbb{R}^{2D \times D}$ is a learnable parameter matrix at the l-th layer, and $[\cdot, \cdot]$ indicates a concatenation operation.

3.3 Reformulation of sequence modeling

To maintain a large receptive field while get rid of the sparse sampling issue introduced by dilated convolutions, we use depthwise separable convolutions [23] (hereafter separable convolutions) instead. Separable convolution, denoted as $f_{sc}(\cdot, d, w)$, factorizes a standard convolution into a depthwise convolution and a pointwise convolution. This factorization allows a network to expand the receptive field using a large kernel size without drastically increasing the number of parameters. Similarly to the TCNs, we use two separable convolutions with kernel sizes of $w_{seq,1}$ and $w_{seq,2}$, as shown in Figure 1. Each sequence modeling layer, denoted as $f_{seq}(\cdot)$, is thus formulated as below:

$$f_{seq}(\cdot) = f_{sc}(\cdot, D, w_{seq,1}) + f_{sc}(\cdot, D, w_{seq,2}). \tag{3}$$

Considering that rhythmic patterns are tempo-dependent, that is, the inter-beat or the inter-downbeat interval will vary according to the current tempo, it is crucial for a network to learn tempo-invariant, or *scale-invariant* patterns [40]. To achieve scale invariance, we simply introduce a set of dilation rates, or *scale factors* $\mathbf{s} = \{r_s\}_{s=1}^{S}$ to each separable convolution. The *scaled* separable convolution, accordingly, will operate S times on the same input with the kernel being expanded by the given set of dilation rates:

$$\underset{\mathbf{s}}{f_{sc}}(\cdot, d, w) = (f_{sc}(\cdot, d, w, r_1), \ldots, f_{sc}(\cdot, d, w, r_S)), \tag{4}$$

where $(\cdot, \ldots, \cdot)$ denotes a stacking operation. Therefore, the scaled separable convolution is capable of capturing rhythmic patterns at different tempi simultaneously. Note that the dilation employed here differs from that of the original TCNs in two aspects. First, we employ the dilation for modeling tempo variance rather than for expanding the receptive field. Second, the parameters of a scaled separable convolution are shared among the S operations.

By introducing the scale factors, we add a new dimension to the output of the scaled separable convolution, whose shape thus becomes $T \times D \times S$. We summarize the scale information by applying a gating mechanism, termed *scale summarization* (SS), which is similar to the *Squeeze-and-Excitation* operation of the SENet [41]. Let $\mathbf{U} \in \mathbb{R}^{T \times D \times S}$ denote the output of a scaled separable convolution. We use global average pooling (GAP) along both the time and the feature dimensions to calculate scalewise statistics $\mathbf{z} \in \mathbb{R}^{S}$. Subsequently, a gating function $\mathbf{g} \in \mathbb{R}^{S}$ is generated by using a projection layer with learnable parameters $\mathbf{W} \in \mathbb{R}^{S \times S}$ and a softmax activation. Lastly, we apply the gate function to $\mathbf{U}$ for obtaining the summarized output $\mathbf{U}' \in \mathbb{R}^{T \times D}$:

$$\begin{aligned} \mathbf{U}' = \mathrm{SS}(\mathbf{U}) &= \sum_{s=1}^{S} \mathbf{U}\mathbf{g}, \\ \mathbf{g} = \underset{s \in S}{\mathrm{softmax}}(\mathbf{W}\mathbf{z}), \quad \mathbf{z} &= \underset{t \in T, d \in D}{\mathrm{GAP}}(\mathbf{U}). \end{aligned} \tag{5}$$

In conclusion, the L-layer sequence modeling network can be formally expressed as follows:

$$\begin{aligned} \mathbf{H}^{(l)} &= f_{seq}^{(l)}(\mathbf{H}^{(l-1)}), \\ f_{seq}^{(l)}(\cdot) &= \mathrm{SS}(\underset{\mathbf{s}}{f_{sc}}(\cdot, D, w_{seq,1}) + \underset{\mathbf{s}}{f_{sc}}(\cdot, D, w_{seq,2})), \end{aligned} \tag{6}$$

with boundaries $\mathbf{H}^{(0)} = \mathbf{Z}$ and $\mathbf{H}^{(L)} = \mathbf{H}$.

3.4 Estimation

We employ a convolutional layer with a kernel size of w_{est} followed by a softmax activation function to estimate the probabilities of being `downbeat` (db), `xbeat` (xb), and `neither` (n) at each time step:

$$\begin{aligned}[\mathbf{P}_{db}, \mathbf{P}_{xb}, \mathbf{P}_{n}] = \mathbf{P} &= f_{est}(\mathbf{H}) \\ &= \underset{d}{\text{softmax}}(f_c(\mathbf{H}, d=3, w_{est})),\end{aligned} \tag{7}$$

where $\mathbf{P}_{db}, \mathbf{P}_{xb}, \mathbf{P}_{n} \in \mathbb{R}^T$ indicate the estimated probabilities of the three categorical labels across T time steps. During inference, we generate a downbeat sequence ($\mathbf{I}_{db}$) and a beat sequence ($\mathbf{I}_b$) from $\mathbf{P}_{db}$ and $\mathbf{P}_{xb}$ by finding the local maximums (LM):

$$\begin{aligned}\mathbf{I}_{db} &= \text{LM}(\mathbf{P}_{db}), \\ \mathbf{I}_b &= \text{LM}(\mathbf{P}_b) = \text{LM}(\mathbf{P}_{db} + \mathbf{P}_{xb}).\end{aligned} \tag{8}$$

3.5 Loss function

Cross entropy is often used as the loss function in the beat and the downbeat estimation tasks. Since the cross entropy loss treats all samples equally, the network optimization will be dominated by vast amount of easy samples, i.e., time frames not belonging to the beat and the downbeat. Similar cases can be found in computer vision-related tasks such as object detection, where exists an extreme foreground-background disproportion. The focal loss (FL) [26] is therefore proposed to weigh more on hard or easily mis-classified samples by introducing a modulating term to the cross entropy loss:

$$\text{FL} = -\frac{1}{N}\sum_{i=1}^{N} m_i \times y_i \times \log(p_i), \tag{9}$$

where p_i is the estimated probability of sample i being classified as the ground truth y_i, and $m_i = (1-p_i)^{\gamma_i}$ is the modulating term with γ_i being a controllable factor. It is noteworthy that the FL, as same as the cross entropy, is calculated on individual samples and hence unable to reflect spatial or temporal relations between samples. On the other hand, the Dice loss (DL) [27], which has been applied to semantic segmentation of images, measures the overlap between the estimations of N samples and the ground truths, and thus can delineate the regions of interest, e.g., beat positions in a given musical sequence:

$$\text{DL} = 1 - \frac{2\sum_{i=1}^{N} p_i \times y_i}{\sum_{i=1}^{N} p_i^2 + \sum_{i=1}^{N} y_i^2}. \tag{10}$$

We propose to use a hybrid of the FL and the DL in place of the cross entropy loss, for they address the class imbalance issue from two complementary aspects: the FL accentuates *individual* hard samples whereas the DL underlines *collective* similarity of the minority samples. In practice, we compute the FL on $\mathbf{P}$ while the DL on both $\mathbf{P}_{db}$ and $\mathbf{P}_b$. By combining the two losses, we urge our network to focus on the minority classes, i.e., beats and downbeats, both at the frame level and at the sequence level.

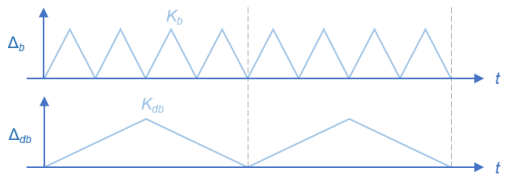


Figure 3: Representations of beat (top) and downbeat (bottom) for label embedding (assume a meter of `4/4`). The horizontal axes are the time dimensions while the vertical axes indicate the phase classes. K_b and K_{db} are the numbers of classes for beat and downbeat, respectively.

3.6 Label embedding

Another concern in regard to the loss function is that the periodic structure of a beat or downbeat sequence is not considered. A recent work deals with the periodicity by estimating beat *phase* instead of beat presence at each frame [42]. Accordingly, the beat estimation task is reformulated as a sequence labeling problem in which the beat phase is represented as a discrete sawtooth wave with period equal to the interbeat interval. This reformulation, however, enlarges the estimation space as the beat phase is categorized into K classes with K being the phase resolution.

Alternatively, we leverage the label embedding approach [28] for learning the periodic structure. As illustrated in Figure 3, a sequence of ground-truth annotations is represented as a discrete triangular wave $\mathbf{\Delta}_{\{db,b\}} \in \mathbb{R}^{T \times K_{\{db,b\}}}$ whose troughs locate at the beat or downbeat frames. We then employ a vanilla autoencoder to encode the represented annotations in an unsupervised manner. The encoder (f_{enc}) and the decoder (f_{dec}) used are both a simple 1-D convolution layer with a kernel size of w_e, i.e., $f_{enc}(\cdot) = f_{dec}(\cdot) = f_c(\cdot, D, w_e)$. The label embedding network is formulated as follows:

$$\begin{aligned}[\mathbf{V}'_{db}, \mathbf{V}'_{b}] &= f_{dec}(\mathbf{E}), \\ \mathbf{E} &= f_{enc}([\mathbf{V}_{db}, \mathbf{V}_{b}]), \\ \mathbf{V}_{db} &= \mathbf{\Delta}_{db}\mathbf{W}_{db}, \mathbf{V}_{b} = \mathbf{\Delta}_{b}\mathbf{W}_{b},\end{aligned} \tag{11}$$

where $\mathbf{W}_{\{db,b\}} \in \mathbb{R}^{K_{\{db,b\}} \times D}$ are learnable phase embedding matrices, $\mathbf{V}_{\{db,b\}} \in \mathbb{R}^{T \times D}$ are sequences of embeddings, $\mathbf{E} \in \mathbb{R}^{T \times D}$ is a sequence of joint embeddings, and $\mathbf{V}'_{\{db,b\}} \in \mathbb{R}^{T \times D}$ are the reconstructions of $\mathbf{V}_{\{db,b\}}$. We compute the reconstruction loss (RL) as follows:

$$\begin{aligned}\text{RL} &= \text{FL}(\mathbf{\Delta}_{db}, \mathbf{\Delta}'_{db}) + \text{FL}(\mathbf{\Delta}_{b}, \mathbf{\Delta}'_{b}), \\ \mathbf{\Delta}'_{db} &= \underset{k \in K}{\text{softmax}}(\mathbf{V}'_{db}\mathbf{W}_{db}^{\top}), \\ \mathbf{\Delta}'_{b} &= \underset{k \in K}{\text{softmax}}(\mathbf{V}'_{b}\mathbf{W}_{b}^{\top}).\end{aligned} \tag{12}$$

As illustrated in Figure 2, the joint embedding $\mathbf{E}$ is taken as a structural regularization (SR) of $\mathbf{H}$ (the output of the sequence modeling network) by minimizing the mean squared error (MSE):

$$\text{SR} = \text{MSE}(\mathbf{E}, \mathbf{H}). \tag{13}$$

Hence, the total loss $\mathscr{L} = \text{FL} + \text{DL} + \text{RL} + \text{SR}$. We train the whole networks in an end-to-end fashion. As the

contextual information of the annotations is encoded in $\mathbf{E}$, adding this *topology-aware* regularization term (SR) to the loss computation enables the network to explicitly learn the structural characteristics [43]. The source code of the proposed model is available online. [3]

4. EVALUATION

4.1 Data preparation

Four datasets are employed in the evaluation, including the Ballroom [44], the Hainsworth [22], the GTZAN [45], and the ASAP [46]. Each audio file (with a sampling rate of 44,100 Hz) in the datasets is transformed into a log-magnitude spectrogram with a total of 81 frequency bins from 30 Hz to 17,000 Hz, by using the Python package *librosa* [47]. Specifically, the short-time Fourier transform (STFT) with a window size of 92.9 ms (4,096 samples) and a hop size of 23.2 ms (1,024 samples) is applied, and the output spectrogram is mapped onto the Mel scale. The per-bin first-order difference is calculated for each spectrogram as an additional feature (only the positive differences are retained) [16]. In consequence, each audio file is represented as $\mathbf{X} \in \mathbb{R}^{T\times 81\times 2}$ with T being the time steps of the corresponding spectrogram. Besides, the MIDI data of the ASAP are also used for evaluation. Following [48], we represent each MIDI file by four features: the *pitch profile* $\in \mathbb{R}^{T\times 88}$ (i.e., pianoroll), the *onset profile* $\in \mathbb{R}^{T\times 88}$, the *spectral flux* $\in \mathbb{R}^{T}$, and the *inter-onset interval* $\in \mathbb{R}^{T}$. We modify our feature extraction network accordingly for the MIDI representation, and keep the other parts of the architecture unchanged. Specifically, the pitch profile and the onset profile are concatenated together and fed into the feature extraction network. The output of the feature extraction network is then concatenated with the other two features and taken as the input of the succeeding layer.

On the other hand, the beat and downbeat annotations are represented as binary sequences, $\mathbf{Y}_b, \mathbf{Y}_{db} \in \{0,1\}^T$, where a time step $t \in T$ has a value of 1 if it belongs to the beat or the downbeat, and 0 otherwise. To treat joint beat and downbeat estimation as a sequence labeling problem, we further generate $\mathbf{Y} \in \{0,1\}^{T\times 3}$ based on $\mathbf{Y}_{db}$ and $\mathbf{Y}_b$, which is a sequence of one-hot vectors indicating the categorical label of each time step. The $\mathbf{Y}$ is used to compute the FL (with $\mathbf{P}$) while the $\mathbf{Y}_b$ and the $\mathbf{Y}_{db}$ are for the DL (with $\mathbf{P}_b$ and $\mathbf{P}_{db}$, respectively), as depicted in Figure 2.

4.2 Experiment

We evaluate the proposed architecture on the four datasets and report the beat and the downbeat estimation performances using the standard F1 measure with a tolerance window of ±70 ms. The hyperparameters of the architecture are set as in Table 1. The model of [15] without DBNs (i.e., the top one in Figure 1) is employed as our baseline model. The numbers of learnable parameters are around 110K and 63K for the proposed model and the baseline,

Input	
Number of frequency bins	$J = 81$ for Audio $J = 88$ for MIDI
Feature extraction	
Kernel size	$w^{(1)}_{ex,1} = 5 \times 41$
Kernel size	$w^{(1)}_{ex,2} = 5 \times 61$
Kernel size	$w^{(2)}_{ex,1} = 15 \times 1$
Kernel size	$w^{(2)}_{ex,2} = 21 \times 1$
Feature dimension	$D = 20$
Sequence modeling	
Kernel size	$w_{seq,1} = 301$
Kernel size	$w_{seq,2} = 401$
Scale factor	$\mathbf{s} = \{1, 2, 3, 4\}$
Number of scales	$S = 4$
Number of layers	$L = 4$
Estimation	
Kernel size	$w_{est} = 7$
Label embedding	
Number of beat phase classes	$K_b = 150$
Number of downbeat phase classes	$K_{db} = 500$
Kernel size	$w_e = 7$

Table 1: Hyperparameters of the proposed architecture.

respectively. Evidently, we can employ separable convolutions to enlarge the receptive field without a catastrophic increase of model capacity. To investigate the effectiveness of the proposed methods, we further perform an ablation study by removing one of the reformulations: feature extraction (`abl_ex`), sequence modeling (`abl_seq`), loss function (`abl_loss`), and label embedding (`abl_lab`).

In line with [15], the Ballroom and the Hainsworth are both split for 8-fold cross-validations. [4] Note that the two datasets are used separately rather than merged as a single cross-validation set. The GTZAN is divided into 10 parts according to the built-in genre labels, with which a 10-fold cross-validation is performed to inspect the performance variance with respect to genre. [5] For the ASAP, we first create a subset by selecting audio recordings which have a paired MIDI file (519 pairs in total). Afterwards, a test set is built from the subset with musical pieces by the four composers: Glinka, Mozart, Schubert, and Rachmaninoff; the remaining data of the subset are used for training. The paired MIDI files are used as the audio counterparts to examine the impact of input modality on the performance.

All the training data are augmented in two ways: 1) by shifting the pitch of an audio signal *before* the STFT is applied, and 2) by changing the hop size of the STFT window. [6] For the ASAP, we only apply the pitch augmentation as its total duration is significantly longer than the other three datasets. By the pitch augmentation, we can easily increase the data amount without extra annotation

[3] `https://github.com/Tsung-Ping/Joint-beat-and-downbeat-estimation`

[4] The details of the data splitting can be found at `https://github.com/superbock/ISMIR2020`.

[5] `jazz.00003`, `jazz.00009`, `jazz.00010`, `jazz.00014`, `jazz.00018`, `jazz.00020` are excluded for they have no downbeat annotations; `reggae.00086` is damaged and also excluded.

[6] We use pitch shifts ∈ {-5 ∼ +6} (semitones) and hop sizes ∈ {18.9, 20.3, 21.8, 23.2, 26.1, 29.0, 31.9} (ms) for the data augmentation.

Task	Model	Ballroom	Hainsworth	GTZAN										ASAP	
				blues	classical	country	disco	hiphop	jazz	metal	pop	reggae	rock	Audio	MIDI
Beat	baseline	96.60	84.18	83.20	51.62	**96.65**	**97.76**	94.61	68.40	82.46	93.84	92.76	89.90	**71.43**	72.45
	proposed	**96.81**	**86.28**	**86.63**	**63.23**	94.82	97.63	**95.58**	**79.80**	**86.48**	**94.81**	**93.05**	**93.01**	71.40	**75.70**
Downbeat	baseline	92.06	66.18	59.25	14.70	79.70	82.71	72.86	33.47	64.64	**84.67**	51.06	76.49	49.46	57.34
	proposed	**94.21**	**69.11**	**60.98**	**30.17**	**83.91**	**86.28**	**79.10**	**50.66**	**67.21**	83.86	**56.87**	**76.55**	**63.92**	**67.43**

Table 2: Evaluation results in terms of F1 score (%). For the Ballroom and the Hainsworth, the scores are the averaged performances over a 8-fold cross-validation. For the GTZAN, the score on each genre is obtained in a 10-fold cross-validation manner. For the ASAP, the performances on the audio data and on the MIDI data are separately shown.

Task	Model	Ballroom	Hainsworth
Beat	abl_ex	96.08 (-0.73)	85.49 (-0.79)
	abl_seq	96.98 (+0.17)	85.96 (-0.32)
	abl_loss	95.27 (-1.54)	86.52 (+0.24)
	abl_lab	96.54 (-0.27)	86.83 (+0.55)
Downbeat	abl_ex	92.16 (-2.05)	62.27 (-6.84)
	abl_seq	94.16 (-0.05)	69.73 (+0.62)
	abl_loss	91.99 (-2.22)	66.79 (-2.32)
	abl_lab	93.61 (-0.60)	69.58 (+0.47)

Table 3: Ablation study on the Ballroom and the Hainsworth. Performance in terms of F1 score (%) and relative improvement against the proposed model (in parentheses) are provided.

cost; moreover, the model can explicitly learn rhythmic features independent of absolute pitch. By the hop-size augmentation, we can obtain more training data of different tempi with a slight adjustment of the annotations.

4.3 Result

The evaluation results are summarized in Table 2, showing the performance on each dataset in terms of beat estimation and downbeat estimation. When evaluated on the Ballroom and the Hainsworth, the proposed model outperforms the baseline in both beat and downbeat estimations. Moreover, it is surprising that the evaluation results on the Ballroom are even better than [15] (beat: 96.20; downbeat: 91.6) and [14] (beat: 96.20; downbeat: 93.7) considering that we didn't use a combination of datasets for training and DBNs for post-processing; in addition, our model is smaller in capacity than the best model of [14] which has around 4.7M learnable parameters. On the other hand, the results on the Hainsworth indicate that more efforts should be put in dealing with music data of diverse styles. For instance, the choruses collected in the Hainsworth are extremely flexible in tempo at the beginning and particularly at the end of each phrase, and therefore a phrase segmentation technique might be incorporated in the tasks.

For the GTZAN, the proposed model demonstrates its superiority over the baseline in almost all the cases. The average beat estimation performances are 85.12% and 88.50% for the baseline and the proposed model, respectively, while the average downbeat estimation performances are 61.96% and 67.56%. According to the evaluation results, the rhythmic characteristics of `classical` and `jazz` are quite distinct from the other genres. Nevertheless, our model still surpasses the baseline by around 15.5 and 17.2 percentage points in estimating the downbeats of `classical` and `jazz`, respectively. These improvements are notable and indicate the capability of our model in both tasks, especially in downbeat estimation.

For the audio data of the ASAP, the proposed model performs comparably to the baseline on the beat estimation while remarkably outperforms the baseline on the downbeat estimation. For the MIDI data of the ASAP, our model achieves greater results both on estimating the beats and the downbeats. It is worth noting that in all cases, the estimation result is better on the MIDI data than on the audio counterpart. Given that automatic music transcription (AMT) [49–51] is an active research topic in the field of MIR, it could be promising to incorporate AMT techniques into the beat and the downbeat estimations.

Finally, as shown in Table 3, the ablation study indicates that our reformulations of the feature extraction and the loss function have notable positive effect especially on estimating downbeats, while the improvements by the other components are inconsistent over the tasks and datasets. We also observed that for the Hainsworth, the model without the label embedding (`abl_lab`) performs slightly better. This might result from the limited expressiveness of $\mathbf{H}$ due to the regularization on it. A more extensive study is required to justify the proposed reformulations.

5. CONCLUSION

We have addressed the joint beat and downbeat estimation task based on a state-of-the-art approach. By inspecting the potential issues in this approach, we proposed several reformulations to further the performance of deep neural networks. We experimentally showed that the proposed architecture is capable of outperforming the state-of-the-art approach without the aid of a post-processing network. In the scenario of deep learning-based beat and downbeat estimations, as well as in many sequence labeling frameworks, it is common to involve a post-processing stage in addition to the deep neural networks since the outputs by the networks are usually coarse when a simple thresholding method is applied. While involving a post-processing stage often leads to an improvement over the preceding deep learning models, it hinders the formulation of end-to-end training and indicates a necessity to reconsider the employed neural networks. Hopefully, we are able to build a model tailored to the task of interest with a deeper look at the network architecture from the perspective of data.

6. REFERENCES

[1] M. Goto and Y. Muraoka, "A beat tracking system for acoustic signals of music," in *Proceedings of the 2nd ACM International Conference on Multimedia*, 1994, pp. 365–372.

[2] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.

[3] L. M. Smith, "Beat critic: Beat tracking octave error identification by metrical profile analysis," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, 2010, pp. 99–104.

[4] H. Grohganz, M. Clausen, and M. Müller, "Estimating musical time information from performed MIDI files," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 35–40.

[5] S. Durand, J. P. Bello, B. David, and G. Richard, "Feature adapted convolutional neural networks for downbeat tracking," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 296–300.

[6] R. Anand, K. G. Mehrotra, C. K. Mohan, and S. Ranka, "An improved algorithm for neural network classification of imbalanced training sets," *IEEE Transactions on Neural Networks*, vol. 4, no. 6, pp. 962–969, 1993.

[7] M. Goto and Y. Muraoka, "An audio-based real-time beat tracking system and its applications," in *Proceedings of the International Computer Music Conference (ICMC)*, 1998.

[8] M. E. P. Davies and M. D. Plumbley, "A spectral difference approach to downbeat extraction in musical audio," in *Proceedings of the 14th European Signal Processing Conference (EUSIPCO)*, 2006, pp. 1–4.

[9] D. P. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[10] F. Krebs, S. Böck, M. Dorfer, and G. Widmer, "Downbeat tracking using beat synchronous features with recurrent neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 129–135.

[11] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 255–261.

[12] C. Chiu, A. W. Su, and Y. Yang, "Drum-aware ensemble architecture for improved joint musical beat and downbeat tracking," *IEEE Signal Processing Letters*, vol. 28, pp. 1100–1104, 2021.

[13] M. Fuentes, B. McFee, H. C. Crayencour, S. Essid, and J. P. Bello, "A music structure informed downbeat tracking system using skip-chain conditional random fields and deep learning," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 481–485.

[14] Y.-N. Hung, J.-C. Wang, X. Song, W.-T. Lu, and M. Won, "Modeling beat and downbeat estimations with a time-frequency Transformer," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 401–405.

[15] S. Böck and M. E. P. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 574–582.

[16] M. E. P. Davies and S. Böck, "Temporal convolutional networks for musical audio beat tracking," in *Proceedings of the 27th European Signal Processing Conference (EUSIPCO)*, 2019, pp. 1–5.

[17] F. Krebs, S. Böck, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 72–78.

[18] J. Pons, O. Slizovskaia, R. Gong, E. Gómez, and X. Serra, "Timbre analysis of music audio signals with convolutional neural networks," in *Proceedings of the 25th European Signal Processing Conference (EUSIPCO)*, 2017, pp. 2744–2748.

[19] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, 2016.

[20] P. Wang, P. Chen, Y. Yuan, D. Liu, Z. Huang, X. Hou, and G. W. Cottrell, "Understanding convolution for semantic segmentation," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 1451–1460.

[21] F. Krebs, S. Böck, and G. Widmer, "Rhythmic pattern modeling for beat and downbeat tracking in musical audio," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 227–232.

[22] S. Hainsworth and M. D. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP Journal on Applied Signal Processing*, vol. 15, pp. 2385–2395, 2004.

[23] L. Sifre, "Rigid-motion scattering for image classification," Ph.D. thesis, École Polytechnique, 2014.

[24] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1800–1807.

[25] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *ArXiv e-prints*, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[26] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2999–3007.

[27] F. Milletari, N. Navab, and S. Ahmadi, "V-Net: Fully convolutional neural networks for volumetric medical image segmentation," in *Proceedings of the 4th International Conference on 3D Vision (3DV)*, 2016, pp. 565–571.

[28] Z. Akata, F. Perronnin, Z. Harchaoui, and C. Schmid, "Label-embedding for image classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 7, pp. 1425–1438, 2016.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[31] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[32] F. Korzeniowski and G. Widmer, "A fully convolutional deep auditory model for musical chord recognition," in *Proceedings of the 26th IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2016, pp. 1–6.

[33] K. Choi, G. Fazekas, and M. B. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 805–811.

[34] S. Böck, M. E. P. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 486–493.

[35] F. Yu, V. Koltun, and T. A. Funkhouser, "Dilated residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 636–644.

[36] R. Hamaguchi, A. Fujita, K. Nemoto, T. Imaizumi, and S. Hikosaka, "Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018, pp. 1442–1450.

[37] X. Ma and E. H. Hovy, "End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

[38] A. Akbik, D. Blythe, and R. Vollgraf, "Contextual string embeddings for sequence labeling," in *Proceedings of the 27th International Conference on Computational Linguistics (COLING)*, 2018, pp. 1638–1649.

[39] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free? - weakly-supervised learning with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 685–694.

[40] B. D. Giorgi, M. Mauch, and M. Levy, "Downbeat tracking with tempo invariant convolutional neural networks," in *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 216–222.

[41] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7132–7141.

[42] T. Oyama, R. Ishizuka, and K. Yoshii, "Phase-aware joint beat and downbeat estimation based on periodicity of metrical structure," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021, pp. 493–499.

[43] A. Mosinska, P. Márquez-Neila, M. Kozinski, and P. Fua, "Beyond the pixel-wise loss for topology-aware delineation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3136–3145.

[44] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 14, no. 5, pp. 1832–1844, 2006.

[45] G. Tzanetakis and P. R. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[46] F. Foscarin, A. McLeod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: a dataset of aligned scores and performances for piano transcription," in *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 534–541.

[47] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in Python," in *Proceedings of the 14th python in science conference (SCIPY)*, vol. 8, 2015.

[48] Y. Chuang and L. Su, "Beat and downbeat tracking of symbolic music data using deep recurrent neural networks," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2020, pp. 346–352.

[49] A. Ycart, A. McLeod, E. Benetos, and K. Yoshii, "Blending acoustic and language model predictions for automatic music transcription," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 454–461.

[50] Y. Wu, B. Chen, and L. Su, "Multi-instrument automatic music transcription with self-attention-based instance segmentation," *IEEE ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 28, pp. 2796–2809, 2020.

[51] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. H. Engel, "Sequence-to-sequence piano transcription with transformers," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021, pp. 246–253.

MUSIC TRANSLATION: GENERATING PIANO ARRANGEMENTS IN DIFFERENT PLAYING LEVELS

Matan Gover
Simply
matan@hellosimply.com

Oded Zewi
Simply
oded@hellosimply.com

ABSTRACT

We present a novel task of "playing level conversion": generating a music arrangement in a target difficulty level, given another arrangement of the same musical piece in a different level. For this task, we create a parallel dataset of piano arrangements in two strictly well-defined playing levels, annotated at individual phrase resolution, taken from the song catalog of a piano learning app. In a series of experiments, we train models that successfully modify the playing level while preserving the musical 'essence'. We further show, via an ablation study, the contributions of specific data representation and augmentation techniques to the model's performance.

In order to evaluate the performance of our models, we conduct a human evaluation study with expert musicians. The evaluation shows that our best model creates arrangements that are almost as good as ground truth examples. Additionally, we propose MuTE, an automated evaluation metric for music translation tasks, and show that it correlates with human ratings. Demos are available online. [1]

1. INTRODUCTION

In this paper we tackle the task of generating piano arrangements for specific playing difficulty levels, conditioned on piano arrangements of the same music in a different playing level. The purpose driving this work is to significantly accelerate our rate of content creation for our piano learning app, Simply Piano. [2] In Simply Piano, we have a library of songs for beginner piano learners to practice. Our library contains arrangements in various playing levels, to match the skills acquired by our learners over their piano journey. Arrangements are prepared by expert musicians based on a pre-defined set of piano pedagogy guidelines. These guidelines are strict and are designed to maintain a uniform playing method and skill level in order to help learners familiarize themselves with the piano in a systematic way. Our aim is to be able to automatically generate multiple arrangements, spanning our range of playing levels, from a single human-generated arrangement at a given level.

Our contributions are two-fold. First, we introduce the task of playing level conversion. We share our dataset preparation method and discuss various experiments regarding choice of music representation and data augmentation methods. Second, we develop and share an automated evaluation metric that can be used for music translation or generation tasks where a reference is available. This automated evaluation metric provides a fast, low-effort estimation of human ratings.

2. RELATED WORK

The symbolic music generation field has advanced considerably in recent years. Specifically, the community has been fast in adopting and adapting capable sequence-to-sequence models, such as Transformer [1], which were originally developed for natural language processing tasks such as machine translation.

Some works focus on unconditional music generation, that is, generating music from scratch [2–4]. Many recent ones are based on Transformer and its variants [2, 4–10]. Other works tackle music generation conditioned on specific inputs: emotions [11], structure [12], theme [13], or musical attributes such as note density [14]. In [2, 15], an accompaniment is generated conditioned on a melody or chord progression.

While most papers work in the performance domain (training on MIDI performances), some work in the score domain as we do. Some use custom tokenized score representations [16, 17] while others operate on text-based notation formats like ABC [18, 19].

Works that are most closely related to this paper are those concerning symbolic music style transfer: generating a musical piece in a target style given the same piece in another style. Musical 'style' can refer to various attributes [20]: symbolic abstractions (score), expressive timing and dynamics (performance), and acoustic details (sound). Of relevance to us is only the first of the three.

Due to the lack of datasets with parallel examples in different domains, most works use unsupervised methods for learning style features. In [9], a Transformer with an encoder bottleneck is used to learn a global style representation. This style representation is then combined with melody representations and fed into a decoder to generate the same melody in a different style. [21] also learns a style

[1] https://www.matangover.com/music-translation/
[2] https://www.hellosimply.com/simply-piano

© M. Gover and O. Zewi. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. Gover and O. Zewi, "Music Translation: Generating Piano Arrangements in Different Playing Levels", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

representation in an unsupervised manner using an autoencoder bottleneck together with a style classifier. [22] uses inductive biases in the encoder to disentangle chord and texture factors. In [23, 24], CycleGAN is used to transfer music between genres. Difficulty level classifiers such as [25–27] could also be used for generating weakly-labeled non-parallel training data for level translation.

In [28], style translation is performed with supervised learning using synthetic parallel data. To our knowledge, our work is the first to perform supervised domain transfer for symbolic music with real-world parallel data.

3. DATA PREPARATION AND REPRESENTATION

Our proprietary dataset of piano arrangements is taken from the song library of Simply Piano. For each song in the library, expert musicians have created arrangements in up to three levels: Essentials (easy), Intermediate, and Pre-Advanced (more difficult but still aimed at learners). Strict arrangement guidelines have been developed by our musicians to create an approachable and engaging learning path for users. We use a pedagogy system based on hand positions, where the aim is to initially minimize the player's need to shift their hands, and then gradually introduce new hand positions and musical concepts as users progress.

Figure 1 illustrates the different difficulty levels. In this paper, we focus on two levels only: Essentials and Intermediate. Specifically, we tackle the task of translating from Intermediate to Essentials. The main difference between the two levels are in hand positions, rhythmic complexity, and harmonic complexity (amount of simultaneous notes). In Essentials, we only allow a small number of positions and keep position shifts to a minimum. We keep the number of chords small and emphasize the melody. We also limit the range of allowed pitches and rhythms: in Essentials we generally do not use tied notes or sixteenth notes. In both Essentials and Intermediate, we do not use tuplets and multiple indepdendent voices on the same staff.

When approaching the task of song level translation, initial experiments showed that translating entire songs at once is a difficult task: models we trained did not learn a meaningful mapping between levels. The reason is probably that song structure can vary greatly between levels in our dataset. That is, some levels of the same song omit certain phrases, while other levels include extra phrases, or change the phrase order. It seems that for full-song mapping, a larger (or cleaner) dataset is needed.

For this reason, we focus our work on translating individual musical phrases. Fortunately, we could make use of existing annotations for this purpose. Each of our arrangements is divided by musicians into phrases, based on the song structure. For example, a song could have the following phrases: Intro, Verse 1, Verse 2, Chorus, Verse 1, Ending. Phrase names stay consistent between different levels of the same song, allowing for minor variations such as 'Phrase 1' $\Leftrightarrow$ 'Phrase 1a'.

We use the phrase boundary annotations to derive a dataset of parallel phrases from our library of arrangements. This is illustrated in Figure 2. We start from two

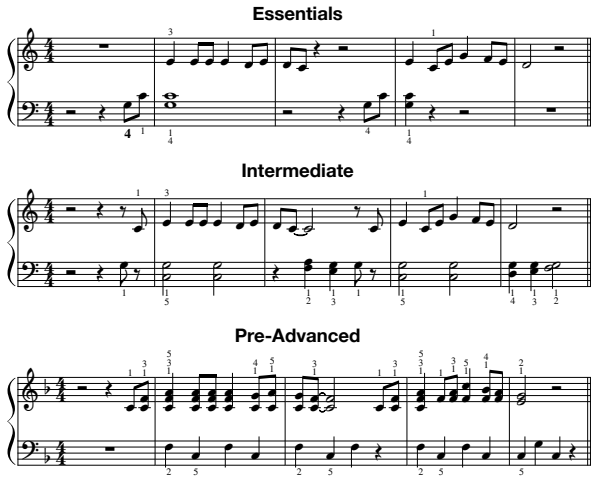


Figure 1. Comparison of three difficulty levels. From bottom to top: *Pre-Advanced* is fully harmonized with rhythmic bass in the left hand and chords in the right hand. *Intermediate* is transposed to the easier key of C major and contains a simplified accompaniment. The right hand chords are omitted, and the melody is split differently between the hands to accomodate for less hand shifts. In *Essentials*, the accompaniment is reduced to a minimum, and a tied note in the melody is removed to simplify the rhythm.

parallel arrangements of the same song in the source and target levels. We discard arrangements where the target and source have different time signatures. We then transpose the source arrangement to the same key as the target arrangement (see details below).

Following that, we match phrases from the source and target levels using heuristics that take into account phrase order, names, and durations. These heuristics were crucial for obtaining a dataset of sufficient size. To account for added or removed phrases, we compute a diff between the source and target phrase names. Phrases with exact name (and order) matches are then considered to be parallel if their duration difference is 2 measures or less. For phrases with no exact name matches, we diff the phrase durations instead, and consider phrases as parallel if their durations match and their names are sufficiently similar.

In some songs, different levels were written in different keys (e.g., Essentials in C major and Intermediate in D major), to fit the arrangement to the desired playing level. For our task, it was important that source and target phrases maintain the same key, so that the model could learn a consistent mapping. Since we didn't have key annotations for the dataset, we implemented a heuristic method for transposition estimation.

We initially tried existing methods for key estimation [29] for each of the phrases, however these weren't accurate enough. Since we only need to estimate the transposition (and not the actual key), we came up with a dedicated heuristic: convert each of the arrangements (source s and target t) into a "pitch-class piano-roll", a boolean matrix (P_s and P_t) with time and pitch-class dimensions, in which 1 signifies the given pitch-class is active at the given time.

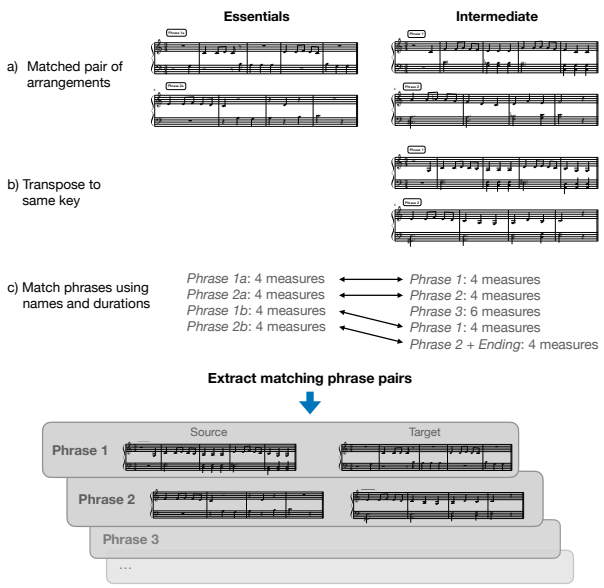


Figure 2. Data preparation: we start with two arrangements of the same song in different levels, transpose them to the same key, and extract matching phrase pairs.

We then compute the pitch-class overlap ($\sum P_s \cap P_t$) between the piano-rolls for each possible transposition (out of 12 possibilities) and take the one that gives the maximum overlap. We use this value to transpose the source level's phrases to match the target level's key.

While the transposition might slightly affect the source phrases' difficulty, the target's difficulty is kept intact. We found that including these examples improves the generated results, since it allows us to increase the dataset size considerably.

3.1 Music Representations

Sequence-to-sequence models such as Transformer operate on a stream of tokens. In order to use these models we must represent music as a sequence, even if it is polyphonic or consists of multiple tracks. Since symbolic music is multi-dimensional in nature, various ways to turn music into a token sequence have been proposed. MidiTok [30] gives a good summary of various representations.

It is crucial to choose a representation that fits the given task. We experimented with three representations for piano music: MIDI-like, Notes, and Notes+Hands. The MIDI-like representation uses note on and note off tokens, along with time shift tokens to signify the passing of time [2, 31]. MIDI-like representations are commonly used, perhaps because it is easy to derive them directly from MIDI files. For our use case, the MIDI-like representation has two limitations: it forces the model to meticulously track active notes in order to later output corresponding note off tokens (potentially leading to syntax errors), and it does not encode the metrical structure of music, potentially leading to compound alignment errors if a single time-shift is wrong.

To counter these problems we employ the Notes representation, which uses three tokens for each note: offset, pitch, and duration. The offset token signifies the note's time offset from the beginning of the measure. A 'bar' token is output at the beginning of each measure. This is similar to REMI [6] (but without velocity tokens), in that it (partially) encodes the metrical structure of music.

Since our output is sheet music, we must output two separate staves for the right and left hands. Since the MIDI-like and Notes representations do not encode track information, we use a heuristic by which all notes on or above middle C are assigned to the right hand, and any note lower than middle C is given to the left hand. Chords (notes with identical onset and offset times) are always grouped to one hand. This heuristic generally matches our dataset's specific characteristics, but is quite coarse and leads to hand assignment errors.

To solve this, we test an additional representation, Notes+Hands, which is identical to the Notes representation but adds a 'hand' token to each note. As stated in [15], such a representation emphasizes the harmonic ('vertical') aspect of music. In the case of piano music, sorting notes by time (and only then by track) emphasizes the overall coherence of the two piano hands over the melodic coherence of each hand separately.

3.2 Data Augmentation

Since our dataset is small, even small models quickly overfit it, limiting our ability to scale model size, consequently limiting the amount of information the model can learn. Previous work has shown that data augmentation can turn an overfitting problem into an underfitting problem, allowing us to iteratively increase model size [32]. We followed this protocol by gradually adding data augmentation methods and increasing model size to improve the final results.

To match our translation task, we needed augmentations that meaningfully alter both source and target phrases, without corrupting the relationship between them. We implemented the following augmentation methods: adding empty measures at random locations; randomly cutting the beginning or end of phrases; removing some measures randomly; and rhythm augmentation (doubling the duration of each note) in some measures.

We purposefully do not include transposition in the list of augmentations, even though it is commonly used in other works. As described above, the arrangement style in our dataset is not transposition-invariant: many features depend on absolute pitch such as hand positions, fingering, key signatures, range limits and hand allocation.

We release our data augmentation code, [3] built on top of the muspy library [33]. We believe that these augmentation methods could be useful for other symbolic music generation tasks, especially when working with small datasets.

4. MODEL

We use a classic Transformer model [1], specifically the BART [34] encoder-decoder implementation from the hug-

[3] https://github.com/matangover/muspy-augment

gingface `transformers` library [35]. The specifics of this implementation (compared to classic Transformer) are that it uses learned position embeddings (we saw slightly better results with these compared to sinusoidal embeddings) and GeLU rather than ReLU as the activation function (this did not seem to make a difference in our experiments). As in the original Transformer, we use a shared weight matrix for the encoder, decoder, and output embedding layers.

We experimented with various model configurations: model dimension, number of layers, number of attention heads. We found that larger models ($d_{\text{model}} > 64$) quickly overfit our training dataset and do not achieve good performance on the validation dataset. The optimal model dimension was found to be 32–64 (with the feed-forward layer dimension always set to $4 \cdot d_{\text{model}}$ as in the original Transformer). The optimal number of layers was 3 to 5, depending on the amount of data augmentation. As discussed in Section 3.2, use of more data augmentation enables us to increase model size without overfitting the data.

We trained using the Adam optimizer [36] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, a learning rate of 0.003 and 1,000 warmup steps.

Our dataset contains a total of 5,543 phrase pairs taken from 1,191 songs. We split the data into train (5,241 phrases), validation (244 phrases), and test (58 phrases) splits. We split phrases by song (never including phrases from the same song in different splits) to avoid data leakage between splits. The validation set was used to stop model training after validation loss stopped decreasing. The test set was used for final evaluation (see Section 7).

During prediction, the model outputs a probability distribution for the next output token given all the input tokens and the previous output tokens. However, if at each decoding step we simply pick the most likely output (greedy decoding), we might end up with a non-optimal sequence [37]. We experimented with two decoding methods: beam search and sampling. We found that sampling produces more diverse results (due to its random nature) but beam search produces overall superior results for our task.

5. INTERFACE FOR INTERACTIVE USE

While our model can be used unattended to create arrangements in the target level, in practice we found that it is beneficial to keep musicians "in the loop" by allowing them to use the model interactively rather than in a 'fire and forget' fashion. We created an interface in which a musician can load a source arrangement, translate it phrase-by-phrase to the target level, and review the results side by side.

Furthermore, the auto-regressive nature of the model enables interesting use cases: a musician could manually modify some notes and ask the model to re-generate the subsequent music accordingly. Additionally, if we use random sampling decoding (see Section 4) we could generate multiple alternatives for each measure using different random seeds and offer the musician a choice. We could also offer knobs that control sampling parameters such as temperature (for controlling the output diversity vs. quality trade-off) or top-k/top-p thresholds (ways of eliminating unlikely outputs to increase the probability of choosing more likely predictions).

In this way, our model becomes part of the creative process, rather than replacing it. It becomes a tool that assists musicians in their job.

6. EVALUATION METRIC

Evaluation is often difficult for music generation due to the absence of pre-defined criteria and because output quality is subjective [38]. Standard practices are reporting perplexity (the likelihood of the ground truth test data given the trained model) and conducting human evaluation studies [2, 4, 9, 15, 31]. Some works also calculate scores based on distributions of certain musical features, either comparing generated data to ground truth distributions [9, 15, 39] or using self-similarity in the generated data itself [3].

In machine translation (of text), evaluation metrics such as the popular BLEU metric [40] have been developed to measure the correspondence between generated translations and ground truth references. BLEU has been shown to correlate with human ratings. We are not aware of a similar metric for music generation tasks. To this end, we propose MuTE (Music Translation Evaluation), an automated evaluation metric for symbolic music translation or generation tasks where a reference is available.

MuTE is a score between 0 and 1. Like BLEU and similar metrics, it is designed to reflect the correspondence between a machine-generated example and a human-generated reference. BLEU measures word n-gram precision (the portion of correct predictions out of all predictions) and deliberately omits recall because of the desire to allow for multiple reference translations of the same phrase. In our case, we only have a single reference for each phrase, hence we can easily use both precision and recall, and combine them using the harmonic mean to give the commonly used F1 score [41].

MuTE works by treating the model as a multi-label classifier that predicts at every time step what pitches are active. To compute the MuTE score, we convert the reference and target music to piano-rolls (where the time unit is set to a certain small metrical unit – in our case, a sixteenth note), and calculate the F1 score over pitches. We do not compute a global score for the entire piece, since that would mean disregarding note order and timing. Instead, we treat each time-slice of the piano-rolls as an individual sample, compute an F1 score for each time-slice, and average those scores. For time-slices where both the reference and target are silent (and hence precision and recall calculation would result in division by zero), we set the score to 1.

We also need to account for cases in which the target differs from the reference in its duration – if, for example, the model 'skips' some measures of the input. We design a procedure that is intended to match the way a human would judge such cases, by detecting skipped or added measures and adjusting the comparison accordingly.

For this, we precede the scoring step with a measure-level alignment procedure. We perform the alignment us-

ing dynamic time warping [42]. The feature vectors for the alignment are each measure's pitch-class piano-roll (see Section 3). We found that using pitch-class piano-rolls gives a more robust alignment compared to regular piano-rolls. The distance between each two feature vectors is computed using the Hamming distance (the proportion of elements that disagree between the two vectors). We use the computed alignment path to align the reference and target's (regular) piano-rolls. The aligned piano-rolls are used to compute the F1 score, while masking out the misaligned segments to assign them a score of 0, thus penalizing the model for any alignment errors.

One additional element of MuTE that pertains to our use case is the desire to reflect the separation between the two hands: we would like to penalize wrong hand allocation of notes. For this reason, we use 'track-specific' MuTE: we calculate a separate MuTE score for each hand and average them to get the final score. We found the mean-of-hands metric to correlate better with human ratings than the vanilla MuTE score calculated over both tracks combined.

We note that the measure-based alignment method is specifically suitable for our use case because we use a measure-based representation for our models, and we noticed that models sometimes omit or repeat some measures. For other use cases, the alignment could be computed over individual time steps or beats, or skipped altogether.

Figure 3 illustrates the calculation of the MuTE score. In Section 7 below, we show that the MuTE score correlates with human ratings. While simple and effective, the MuTE score has caveats. First, it does not differentiate between sustained and repeated notes. For example, a half note is considered identical to two consecutive quarter notes of the same pitch. Second, it does not allow for minor variations that might be acceptable to a human rater, such as octave changes and rhythmic variations. Third, the hand-specific version does not account for notes that are missing in one hand but present in the other hand. We plan to address these shortcomings in a future version.

We release a Python library for computing the MuTE score [4] and we welcome researchers to adopt it or adapt it for their use cases as needed.

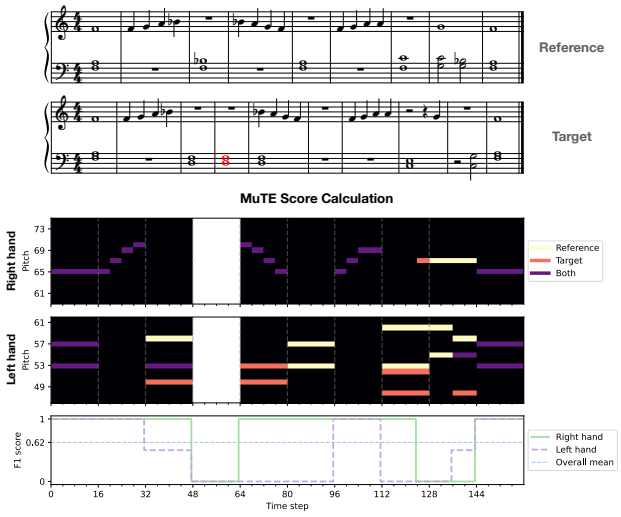


Figure 3. Illustration of the MuTE score calculation. The target score has an added measure (marked in red) with regard to the reference. The misaligned region is masked (in white) on the piano-rolls. The per-hand sample-wise pitch F1 score is calculated from the aligned piano-rolls, and scores are averaged to get the final MuTE score.

Figure 4. Scores for each model version, as rated by human experts. Our best model (Augment) achieves ratings almost as high as human generated examples (Human).

7. RESULTS

We report the results of our experiments on converting Intermediate level arrangements to Essentials, and compare our models' outputs to matching reference arrangements created by expert musicians. For evaluation we conduct a human evaluation study, calculate MuTE scores (see Section 6) and report the correlation between human ratings and MuTE scores. Furthermore, we run several ablation studies to study the effect of our data augmentation techniques and the choice of music representation.

7.1 Human Evaluation Study

For this study we selected 11 songs from the test set (none of these songs' phrases appeared at training). From each song we extracted a single phrase and translated it from Intermediate level to Essentials, using 4 variants of the model that differ in representation and augmentation. We also extracted the matching phrases from the ground truth Essentials arrangement for comparison.

The first 3 model variants differ only in the music representation they use: MIDI-like, Notes, and Notes+Hands (see details in Section 3.1). The fourth model variant uses the Notes+Hands representation and adds data augmentation (see Section 3.2) with randomly varying parameters to about half of the training examples.

This resulted in 55 examples, which were then rated by 6 expert musicians who are familiar with the Essentials level guidelines. The examples were randomly ordered and all identifiers of model version and ground truth were removed. Musicians were told all 5 examples are different model versions and were not aware of the details of the ver-

[4] https://github.com/matangover/mute

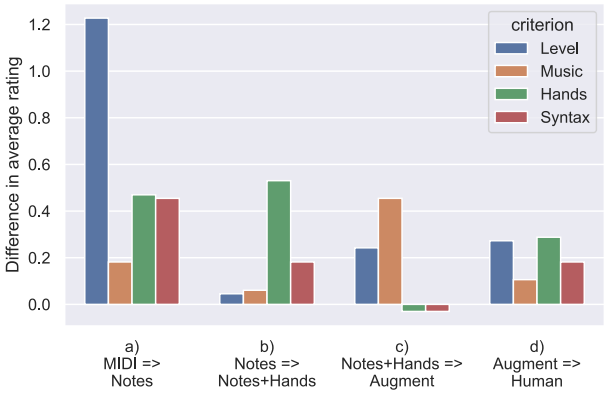


Figure 5. Difference in average rating, per criterion, between model variants. a) Moving from MIDI to Notes improved across all criteria, mostly Level. b) Adding hand information improved hand assignment. c) Augmentation improved preserving musical content. d) No particular criterion stands out in the difference from our best model to the human-generated ground truth.

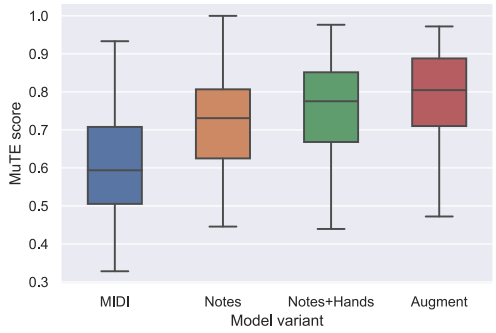


Figure 6. Evaluation of our model variants on the test set using the MuTE score.

sions or of the presence of human-generated ground truth. Ratings between 1–5 were collected in 5 criteria:

1. Meeting Essentials level guidelines (Level)
2. Preserving musical content (Music)
3. Correctly assigning hands according to allowed hand positions (Hands)
4. Avoiding syntactic style errors such as crossover voices (Syntax)
5. Maintaining structure: avoiding missing or extra bars (Alignment)

The maximum possible total score is 25. Figure 4 shows the distributions of scores for each model variant and for the ground truth. These results show that the changes between model variants improved overall output quality. The best model (Augment) achieves ratings almost as high as the ground truth (Human).

In Figure 5 we show the gains in each criterion from the changes in each model variant. Some differences are easily explainable, for example, adding hand information improved the correct assignment of hands. Interestingly,

Figure 7. A scatter-plot showing the relationship between human ratings and MuTE scores. The black line shows a linear regression model fit and the shaded area shows the 95% confidence interval for the regression estimate.

the difference between our best model to the ground truth cannot be attributed to any specific criterion. We left out the Alignment criterion from the figure, as ratings under 5 were very rare. The few ratings below 5 were given almost only to the MIDI model variant, which is more prone to alignment errors due to the lack of *bar* tokens.

7.2 Automated evaluation

We calculate MuTE scores (see Section 6) for the entire test set (58 phrases from 12 songs). The results are shown in Figure 6, and confirm the human evaluation results. Additionally, for the 55 examples rated by human musicians, we compare the MuTE scores to the human ratings (the MuTE score of the ground truth compared to itself is always 1). As shown in Figure 7, we find that MuTE scores are correlated with human ratings with a Pearson correlation coefficient of 0.56 ($p = 4 \cdot 10^{-29}$). While the correlation is not perfect, it shows that MuTE can be used as an estimator for human ratings. Furthermore, MuTE correlates better with the total human rating than with any of the individual criteria's ratings: the correlation coefficients with the indvidiual criteria are 0.27 (Alignment), 0.37 (Hands), 0.47 (Level), 0.37 (Music), and 0.29 (Syntax). The similarity of figures 6 and 4 further shows that MuTE scores can be used to rank models with similar results to human rankings, while being cheaper and faster to compute.

8. CONCLUSION

We presented the task of playing level conversion: converting piano arrangements from one difficulty level to another. We ran several experiments focused on simplifying arrangements to an easier level, and showed the importance of data representation and augmentation. Our best model creates arrangements that achieve human ratings almost as high as reference arrangements composed by expert musicians. We designed the MuTE evaluation metric and showed that it correlates with human ratings. For the benefit of the community, we share our data augmentation and evaluation code.

9. ACKNOWLEDGEMENTS

Thanks to the musicians at Simply for creating the arrangements used to train our models, and for helping perform the evaluation. Thanks to Eran Aharonson for valuable discussions and feedback.

10. REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[2] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer: Generating music with long-term structure," in *International Conference on Learning Representations*, 2018.

[3] G. Mittal, J. Engel, C. Hawthorne, and I. Simon, "Symbolic music generation with diffusion models," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[4] J. Liu, Y. Dong, Z. Cheng, X. Zhang, X. Li, F. Yu, and M. Sun, "Symphony generation with permutation invariant language model," *arXiv preprint arXiv:2205.05448*, 2022.

[5] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[6] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020.

[7] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound Word Transformer: Learning to compose full-song music over dynamic directed hypergraphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

[8] A. Muhamed, L. Li, X. Shi, S. Yaddanapudi, W. Chi, D. Jackson, R. Suresh, Z. Lipton, and A. J. Smola, "Transformer-GAN: symbolic music generation using a learned loss," in *4th Workshop on Machine Learning for Creativity and Design at NeurIPS*, 2020.

[9] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, "Encoding musical style with Transformer autoencoders," in *International Conference on Machine Learning*, 2020, pp. 1899–1908.

[10] C. Payne, "MuseNet," *OpenAI Blog*, 2019. [Online]. Available: https://openai.com/blog/musenet

[11] S. Sulun, M. E. P. Davies, and P. Viana, "Symbolic music generation conditioned on continuous-valued emotions," *IEEE Access*, vol. 10, 2022.

[12] K. Chen, W. Zhang, S. Dubnov, G. Xia, and W. Li, "The effect of explicit structure encoding of deep neural networks for symbolic music generation," in *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*, 2019.

[13] Y.-J. Shih, S.-L. Wu, F. Zalkow, M. Muller, and Y.-H. Yang, "Theme Transformer: Symbolic music generation with theme-conditioned transformer," *IEEE Transactions on Multimedia*, 2022.

[14] J. Ens and P. Pasquier, "MMM: Exploring conditional multi-track music generation with the Transformer," *arXiv preprint arXiv:2008.06048*, 2020.

[15] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "PopMAG: Pop music accompaniment generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020.

[16] M. Suzuki, "Score Transformer: Transcribing quantized MIDI into comprehensive musical score," in *Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[17] ——, "Piano fingering estimation and completion with Transformers," in *Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[18] C. Geerlings and A. Meroño-Peñuela, "Interacting with GPT-2 to generate controlled and believable musical sequences in ABC notation," in *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*, 2020, pp. 49–53.

[19] B. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," in *1st Conference on Computer Simulation of Musical Creativity*, 2016.

[20] S. Dai, Z. Zhang, and G. G. Xia, "Music style transfer: A position paper," in *Proceeding of the International Workshop on Musical Metacreation (MUME)*, 2018.

[21] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[22] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," in *Proceedings of the 21nd International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[23] G. Brunner, Y. Wang, R. Wattenhofer, and S. Zhao, "Symbolic music genre transfer with CycleGAN," in *IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2018, pp. 786–793.

[24] G. Brunner, M. Moayeri, O. Richter, R. Wattenhofer, and C. Zhang, "Neural symbolic music genre transfer insights," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2019, pp. 437–445.

[25] S.-C. Chiu and M.-S. Chen, "A study on difficulty level recognition of piano sheet music," in *2012 IEEE International Symposium on Multimedia*, 2012, pp. 17–23.

[26] Y. S. Ghatas, M. B. Fayek, and M. M. Hadhoud, "Generic symbolic music labeling pipeline," *IEEE Access*, vol. 10, pp. 76 233–76 242, 2022.

[27] P. Ramoneda, N. C. Tamer, V. Eremenko, X. Serra, and M. Miron, "Score difficulty analysis for piano performance education based on fingering," in *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 201–205.

[28] O. Cífka, U. Şimşekli, and G. Richard, "Supervised symbolic music style translation using synthetic data," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[29] D. Temperley, "What's key for key? The Krumhansl-Schmuckler key-finding algorithm reconsidered," *Music Perception*, vol. 17, no. 1, pp. 65–100, 1999.

[30] N. Fradet, J.-P. Briot, F. Chhel, A. E. F. Seghrouchni, and N. Gutowski, "MidiTok: A Python package for MIDI file tokenization," in *Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[31] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, 2018.

[32] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Interspeech*, 2019.

[33] H.-W. Dong, K. Chen, J. McAuley, and T. Berg-Kirkpatrick, "MusPy: A toolkit for symbolic music generation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[34] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7871–7880.

[35] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38–45.

[36] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, 2015.

[37] D. Ippolito, R. Kriz, J. Sedoc, M. Kustikova, and C. Callison-Burch, "Comparison of diverse decoding methods from conditional language models," in *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*, 2019, pp. 3752–3762.

[38] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4773–4784, 2020.

[39] S. Wu and Y. Yang, "The Jazz Transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures," in *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 142–149.

[40] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "BLEU: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[41] C. J. Van Rijsbergen, "Foundation of evaluation," *Journal of documentation*, vol. 30, no. 4, pp. 365–373, 1974.

[42] S. Dixon and G. Widmer, "MATCH: A music alignment tool chest," in *Proceedings of the 6th International Society for Music Information Retrieval Conference (ISMIR)*, 2005, pp. 492–497.

SCALING POLYPHONIC TRANSCRIPTION WITH MIXTURES OF MONOPHONIC TRANSCRIPTIONS

Ian Simon **Josh Gardner*** **Curtis Hawthorne**
Ethan Manilow* **Jesse Engel**
Google Research, Brain Team
{iansimon, fjord, jesseengel}@google.com
jpgard@cs.washington.edu, ethanm@u.northwestern.edu

ABSTRACT

Automatic Music Transcription (AMT), in particular the problem of automatically extracting notes from audio, has seen much recent progress via the training of neural network models on musical audio recordings paired with aligned ground-truth note labels. However, progress is currently limited by the difficulty of obtaining such note labels for natural audio recordings at scale. In this paper, we take advantage of the fact that for monophonic music, the transcription problem is much easier and largely solved via modern pitch-tracking methods. Specifically, we show that we are able to combine recordings of real monophonic music (and their transcriptions) into artificial and musically-incoherent mixtures, greatly increasing the scale of labeled training data. By pretraining on these mixtures, we can use a larger neural network model and significantly improve upon the state of the art in multi-instrument polyphonic transcription. We demonstrate this improvement across a variety of datasets and in a "zero-shot" setting where the model has not been trained on any data from the evaluation domain.

1. INTRODUCTION

The variety of sounds that can appear in a musical recording is effectively infinite. Numerous instruments, articulation styles, dynamics, recording conditions, synthesizer parameters, processing effects, and more can all interact to produce the enormous space of sounds that can be heard in recorded music. This diversity of sound is a challenge for Automatic Music Transcription (AMT), the task of extracting a symbolic representation from a musical recording. In this paper we focus on the specific task of automatically extracting a symbolic representation consisting of musical *notes*. For each note we would like to infer its pitch, absolute timing of its onset and offset in seconds, and the instrument that played the note. We do not attempt to infer a musical *score* with time signatures, key signatures, etc., merely a sequence of notes with absolute timestamps that can be represented as MIDI [1].

Most recent progress in AMT has been driven by neural networks trained on musical recordings paired with their note transcriptions. However, this progress is currently bottlenecked by the limited number of existing ground truth transcriptions, which is in turn bottlenecked by the difficulty of creating note transcriptions that are precisely aligned with audio. Creating ground truth transcriptions manually is possible only for trained musicians, and is a notoriously tedious process. Furthermore, trained musicians rarely annotate the timing of note events to the precision needed for training AMT systems.

Besides being limited in number, existing transcribed recordings are also limited in *character*, as only a few methods are able to produce such aligned data efficiently:

Capture a musical performance using an instrument equipped with sensors, e.g. a Disklavier [2] or guitar equipped with hexaphonic pickup [3]. This yields highly accurate ground-truth transcriptions but is difficult to scale across many instruments due to the difficulty of needing to equip each instrument with specialized sensors. In addition, data created in this way is likely to be limited to a small number of recording environments, as it is easier to invite multiple performers to record in a single location than to create many sensor-equipped instruments, distribute them to performers, and ask the performers to share recordings with captured transcriptions.

Synthesize a sequence of notes using a software synthesizer [4, 5]. While this technique produces high-quality audio-label alignment, the space of sounds that can be generated by a synthesizer only partially covers the space of instrument sounds an AMT system is expected to transcribe. In particular, for instruments such as violin and saxophone where the sound of each note and transition is mediated by a large number of control parameters (typically "provided" by a human performer via body positioning), we do not yet have good methods for generating music that matches the realism and diversity of human performances.

Align an audio recording and its corresponding symbolic score using dynamic time warping [6]. This technique

* Work done as a Google Brain Student Researcher.

© I. Simon, J. Gardner, C. Hawthorne, E. Manilow, and J. Engel. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** I. Simon, J. Gardner, C. Hawthorne, E. Manilow, and J. Engel, "Scaling Polyphonic Transcription with Mixtures of Monophonic Transcriptions", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

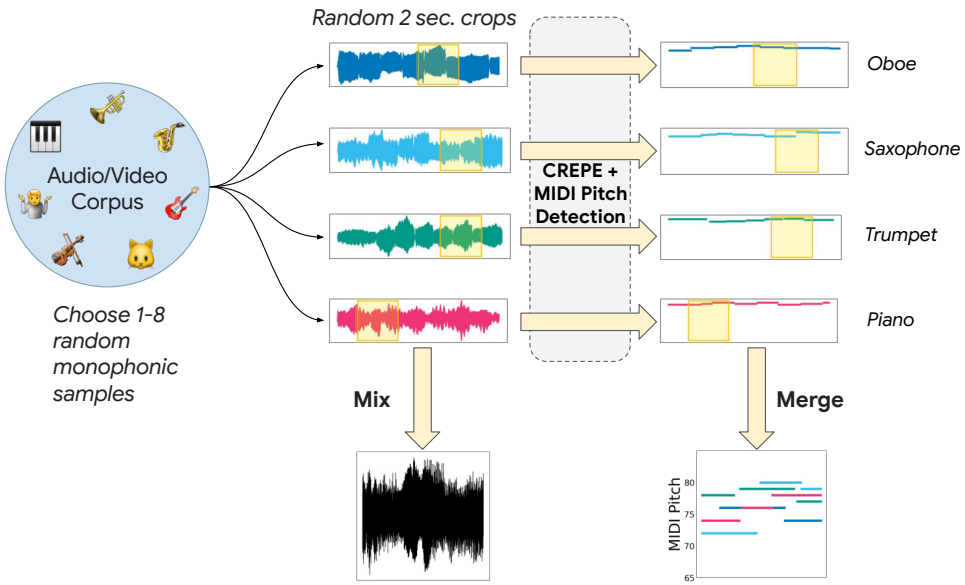


Figure 1. Overview of our dataset generation pipeline. Monophonic recordings are gathered from an audio/video corpus. Each training example selects 1-8 20 second clips, and then takes random ∼2 second crops from each clip for MIDI pitch detection. The resulting audio clips are mixed and MIDI clips are merged to form a polyphonic mixture for training.

has the advantage of being applicable to pre-existing audio recordings where a score is available. However, the resulting labels often end up poorly aligned due to dynamic time warping errors or when the recording does not exactly match the provided score.

In this paper, we demonstrate a new technique for creating training data for AMT models: transcribing in-the-wild monophonic (i.e. one note playing at a time) recordings and mixing the audio and transcriptions together as training data for a neural network. Monophonic recordings are much easier to transcribe due to the existence of highly accurate pitch trackers such as CREPE [7]. This technique does not suffer from any of the limitations described above; it can be applied to any monophonic audio clip, and the audio and note labels are accurately aligned.

By generating a large number of polyphonic mixes of transcribed monophonic recordings, we are able to greatly increase both the *quantity* and *diversity* of data used to train multi-instrument polyphonic transcription models. An important part of scaling our transcription data is that we mix recordings without regard to whether or not they "go together"; the mixtures are rhythmically and harmonically incoherent. Nonetheless, using these recordings allows us to train larger and better models that surpass the existing state of the art in multi-instrument AMT.

2. RELATED WORK

2.1 Automatic Music Transcription

The field of automatic music transcription (AMT) has a rich history in MIR, but has been advancing rapidly in recent years as AMT models begin to take advantage of deep learning models and large-scale datasets. For example, the Onsets & Frames model by Hawthorne et al. [2] uses a deep convolutional and recurrent neural network to jointly predict note onsets and frames (subsequently treated as an adversarial task by Kim & Bello [8]); Kelz et al. [9] use a convolutional network to model ADSR envelopes in piano transcription; Manilow et al. [10] perform simultaneous transcription and source separation with a multiheaded network; Cheuk et al. [11] use semi-supervised learning to improve transcription on low-resource datasets and later explore the effect of adding a spectrogram reconstruction loss [12], and recently Maman & Bermano [13] demonstrate the use of synthetic data and iterative alignment to enable training on weakly-aligned scores.

The unsupervised pretraining of large neural sequence architectures has been critical to scaling performance in natural language processing (NLP) [14], computer vision [15], and automatic speech recognition (ASR) [16], but the dynamics of pretraining are only beginning to be empirically well-understood [17–19]. Recently some AMT systems have adopted the same sequence architectures used in NLP (e.g. MT3 [20]), but using *pretraining* in combination with these architectures for AMT is a relatively unexplored area. While in this paper we only examine the effects of large scale pretraining on MT3, the data strategies presented are orthogonal to these improvements in model architectures and can ideally work in concert to create better overall AMT systems.

2.2 Dataset Mixing and Labeling Heuristics

Heuristically-labeled and randomly-mixed data have been used to improve large deep learning models in other tasks and domains. For example, data augmentation strategies that combine existing labeled examples are common in computer vision [21–23]; such strategies have been shown to reduce memorization of corrupt labels [21],

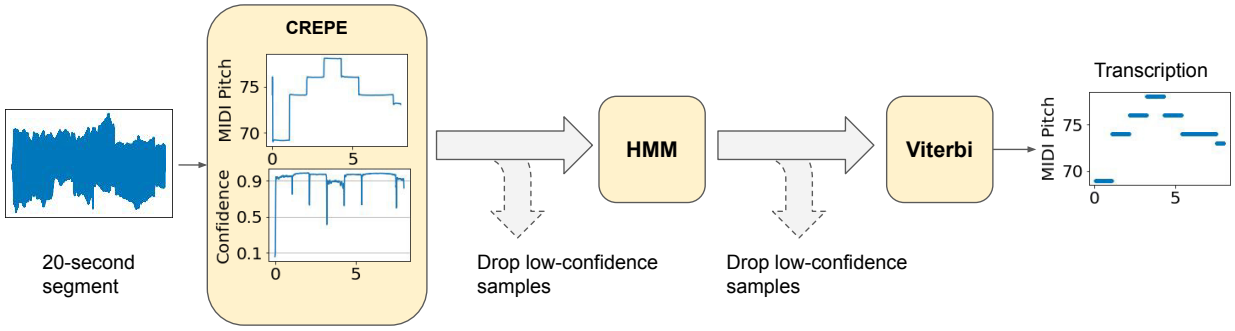


Figure 2. Overview of pitch tracking pipeline. HMM parameters are described in Section 3.3.

improve prediction and robustness on out-of-distribution data [21, 24], stabilize training of generative models [21], and reduce supervised models' overconfidence on samples outside the training distribution [25]. In the domain of AMT, Callender et al. [26] use data mixing to recombine random crops of drum audio samples and find that it improves a drum transcription model.

Heuristic techniques have also been used for labeling data and training MIR models. For example, Salamon et al. [27] use an analysis-synthesis framework for f_0 annotation where the model predictions are synthesized and mixed back into the input audio, and show that this produces results that are statistically indistinguishable from models trained on the manually-annotated mixes. Maman and Bermano [13] use dynamic time warping to generate music transcription labels from synthesized audio. In ASR, Fonseca et al. [28] show that unsupervised representation learning (via contrastive learning) can rival state-of-the-art models without expensive human-labeled datasets for audio scene separation. Furthermore, training with incoherently mixed single-instrument recordings, as we do here, is a widespread technique used for training music source separation systems [5, 29–32]. However, we are unaware of any systems that use heuristically-labeled *and* randomly-mixed data to scale AMT systems.

3. MODEL AND TRAINING PROCESS

3.1 Model Architecture

For all experiments, we use an encoder-decoder Transformer [33] architecture. Our model and training process is almost identical to that of Gardner et al. [20], with two main exceptions. First, while we use the T5.1.1 [1] [14] "Small" architecture used in [20] for some experiments, for others we use the larger T5.1.1 "Base" model.

Second, we pretrain our model on a large dataset consisting of mixes of automatically-transcribed monophonic recordings drawn from an Internet-scale pool of videos; this is the main contribution of the current work. In our experiments, we vary the number of pretraining and fine-tuning steps for the model as shown in Table 1. We share our code publicly at `https://github.com/magenta/mt3`, along with a checkpoint pretrained on our dataset of monophonic mixes.

[1] `https://github.com/google-research/text-to-text-transfer-transformer/blob/main/released_checkpoints.md#t511`

3.2 Data Representation

We use the exact input and output representation of Gardner et al. [20]: log-Mel spectrograms as input and a MIDI-like output vocabulary containing time, pitch, note on/off, instrument, "tie" section token, and drum tokens. This output vocabulary is capable of representing arbitrary multi-instrument polyphonic MIDI. As in Gardner et al. [20], we ignore note *velocities* as they are not present in most ground-truth transcriptions.

Again following Gardner et al., we split the input audio (and target labels at training time) into segments of 2.048 seconds (256 spectrogram frames at a hop size of 8 ms) [34]. At inference time, we transcribe each segment independently and concatenate their transcriptions. We also use the "tie" representation of Gardner et al. which requires the model to declare all notes that are active at the beginning of each segment. At inference time, if the model fails to declare a note that was active in the previous segment, we turn off the note.

3.3 Monophonic Detection and Transcription

We process a dataset of music recordings obtained in the wild to (a) detect clips that are likely to be monophonic, and (b) transcribe those clips into sequences of notes. While our heuristic process is far from the ideal approach to monophonic transcription, its main benefit is that it is simple and easy to apply to an Internet-scale data corpus.

Our process of creating a set of transcription labels from unlabeled audio data is illustrated in Figure 2, and proceeds as follows. First, we split each recording into non-overlapping 20-second segments. Then, we use the CREPE [7] pitch tracker to extract f_0 and confidence values from each segment at a frame rate of 100 Hz. If any of the four 5-second sub-segments has fewer than 20% of its confidence values above 0.95, we discard the entire segment. We do not run Viterbi smoothing on the f_0 values.

We then model the sequence of f_0 and confidence values with a hidden Markov model, where the hidden state

Model Size	# Pretrain Steps	# Finetune Steps	MAESTRO	Cerberus4	GuitarSet	MusicNet	Slakh	URMP
small (MT3)	0	1M	.82	.76	.78	.34	.55	.50
base (MT3)	0	1M	.85	.78	.78	.26	.61	.51
small (ours)	1M	100k	.84	.81	**.82**	.35	.59	.73
base (ours)	500k	100k	**.87**	**.83**	**.82**	**.38**	**.66**	**.78**
base (ours) vs. small (MT3)		($\Delta\%$ Rel.)	+6.0%	+9.2%	+5.1%	+12%	+20%	+56%

Table 1. Onset+Offset+Program F1 scores on different datasets, taking into account pitch, instrument, onset time, and offset time. Numbers in the first row are taken directly from Gardner et al. [20] and represent the previous state of the art. Numbers in the second row are from using the existing MT3 training procedure and data with a larger capacity model. The next two lines show the effect of our pretraining procedure. Pretraining on monophonic mixes yields some benefit even for the "Small" model, but does even better when model size is increased to "Base". Using a "Base" model without pretraining does considerably worse than with pretraining.

Model	MAESTRO	Cerberus4	GuitarSet	MusicNet	Slakh2100	URMP
			Frame F1			
small (MT3)	.86	.87	.89	.68	.79	.83
base (ours)	**.90**	**.91**	**.91**	**.73**	**.85**	**.92**
			Onset F1			
small (MT3)	.95	.92	.90	.50	.76	.77
base (ours)	**.97**	**.95**	**.91**	**.56**	**.83**	**.90**
			Onset+Offset+Program F1			
small (MT3)	.82	.76	.78	.34	.55	.50
base (ours)	**.87**	**.83**	**.82**	**.38**	**.66**	**.78**

Table 2. Transcription improvement over MT3 for Frame, Onset, and Onset+Offset+Program F1.

is a MIDI pitch value (0-127) or rest. (Note that we do not model onsets separately.) The transition distribution models the probability of a state change, set to expect two state changes per second or probability 0.04 of changing at any frame. The f_0 observations are modeled as a Gaussian distribution centered at the corresponding MIDI note frequency, with a standard deviation of 0.2 semitones. We use a mixture of 3 Gaussians to model octave errors, with probability 0.025 of an octave error occurring in either direction, respectively. We model $P(\mathrm{r}|\mathrm{c})$ independently of f_0, where r is the rest state and c is the CREPE confidence value. We treat c^v as the probability a frame is not a rest, where we empirically determine $v = 7.5$.

We use the forward algorithm to compute $P(f_0, \mathrm{c})$ and discard the audio segment if the log-likelihood per frame is less than 0.3. This is useful for filtering out vocal recordings or other monophonic audio that is not well modeled by a sequence of discrete notes using the equal-tempered Western scale. Then, we use the Viterbi algorithm to infer the maximum-likelihood sequence of notes.

The above process and parameters were determined empirically on a small amount of unlabeled audio from our in-the-wild dataset. We believe that this process could be replaced by a more rigorous monophonic transcription model such as the one in Wu et al. [35]; however, we find that our process provides sufficiently strong results in practice to benefit AMT training. We plan to release our source code publicly, including our monophonic detection and transcription code. While we are unable to share our dataset, we believe that our results can likely be replicated on any large corpus of in-the-wild musical audio.

4. EXPERIMENTS

Our experiments measure how our pretraining method affects the accuracy of a Transformer-based transcription model. This section describes our experimental process, from data gathering to model training and evaluation.

4.1 Gathering Monophonic Recordings

We downloaded 10M audio recordings from an online audio/video sharing site, filtering based on metadata in an attempt to restrict ourselves to solo non-vocal musical performances on pitched instruments (i.e. not drums). We then split each recording into non-overlapping 20-second chunks and apply the monophonic detection and transcription process described in Section 3.3. Only about 1% of the chunks are detected as monophonic; we believe this is because most of the solo recordings are of performances on polyphonic instruments such as piano or guitar. Still, we choose not to exclude these instruments entirely as they are occasionally performed monophonically.

We also attempt to use the associated metadata to identify the instrument in each recording, which we use as instrument labels in the training examples. Table 3 shows the number of training clips identified as each instrument; we use the instrument vocabulary of Gardner et al. [20], originating in Manilow et al. [5]. In early experiments,

Instrument	# Clips	Instrument	# Clips
Violin	327,914	Oboe	3,234
Flute/Piccolo	96,507	Bass Guitar	2,264
Tenor Sax	71,737	Electric Piano	2,059
Acoustic Guitar	59,231	Tuba	1,526
Electric Guitar	58,037	Harp	1,233
Clarinet	53,513	Bassoon	1,114
Trumpet	52,695	Xylophone	897
Cello	23,856	Double Bass	751
Viola	22,027	Baritone Sax	317
Sopr./Alto Sax	18,784	Synth	288
Piano	18,626	Voice	123
Trombone	14,969	Organ	99
French Horn	7,135		

Table 3. Number of labeled segments for each instrument in our dataset after filtering as shown in Figure 2.

we found that these metadata-based instrument labels only led to a minor improvement in the model's performance, so even in the case where metadata is unavailable we still expect our overall approach to work.

After the entire data gathering process, we are left with ~5,000 hours of automatically-transcribed monophonic music recordings. This is 3 times more than all of our fine-tuning and evaluation datasets combined, 20 times more if we exclude synthetic audio, and over 100 times more if we exclude synthetic audio and piano. Furthermore, by mixing together random segments of monophonic audio at training time, we increase the effective dataset size so substantially that the model is unlikely to ever see the same combination of source recordings twice during training.

4.2 Pretraining

We pretrain an encoder-decoder Transformer [33] model as described in Section 3.1 on mixes of up to 8 monophonic recordings and their corresponding note labels. The process that generates each training example is as follows:

1. Choose the number of tracks k for the mix uniformly at random from 1-8.
2. Take the next k 20-second clips.
3. From each clip, choose a 2.048-second segment uniformly at random.
4. Mix the audio from the k clips together by summation, then peak normalize.
5. Combine the note labels from the k transcriptions into a single stream. If each transcription has already been serialized, we can perform a merge to ensure the note labels end up in the correct temporal order.

The much larger training datasets generated by our heuristic labeling enables us to scale the model size that we use, allowing us to potentially leverage the scaling benefits of large Transformer models [17]. To that end, we test two different sizes of model from T5 [14]: the T5.1.1 "Small" model used in MT3, and the relatively larger "Base" model. When using the smaller "Small" T5.1.1 model, we train for 1M steps. While pretraining the "Small" model continues to improve up to 1M steps, we found that 500k pretraining steps works best for the "Base" model; this is in line with prior research which has suggested that larger pretrained models are more susceptible to diverging or overfitting [17].

Our first experiment is modeled after the one proposed in Gardner et al. [20], where we train on a combination of multiple datasets and then evaluate on a held-out portion of each dataset. Our goal with this experiment is to measure the effect of pretraining on a collection of standard transcription datasets, using standard transcription metrics. Following Gardner et al., we evaluate on 6 datasets:

MAESTRO [2]: a dataset of 1276 classical piano performances with MIDI captured via Disklavier.

Slakh [5]: a dataset of 2100 synthesized songs from the Lakh MIDI Dataset [36]. We train on 10 random subsets of tracks from each song, and evaluate on full mixes.

Cerberus4 [10]: a subset of the instruments in Slakh, where mixes consist of exactly 4 tracks: guitar, piano, bass, and drums.

GuitarSet [3]: a dataset of 360 guitar recordings, transcribed using a hexaphonic pickup.

MusicNet [6]: a dataset of 330 classical music recordings, with MIDI files aligned by dynamic time warping.

URMP [37]: a dataset of 44 classical music recordings with instruments independently recorded and transcribed.

We measure transcription performance using the F1 metric over notes (Onset+Offset+Program), where in order for two notes to "match" they must have the same pitch and instrument and be close enough in onset time and offset time. We use the default tolerances in the `mir_eval` [38] library: onset times must be within 50 ms to match, and offset times must be within the larger of 50 ms or 20% of the note's true duration.

Our results are shown in Table 1. When using a "Small" model, pretraining provides a small increase in F1 score across all datasets. A further advantage of pretraining is that it lets us scale up to a "Base" model, which provides a more significant boost. Note that using a "Base" model without pretraining does not provide the same performance improvement, likely because the existing datasets are simply too small for the model to take advantage of its increased capacity. Table 2 compares our best model to MT3 across all F1 metrics and shows that pretraining on monophonic mixtures leads to substantial improvements in the state-of-the-art. It's worth noting that MusicNet is now known to have many misalignements, that are likely the cause of the lower scores [13].

Model Size	# Pretrain Steps	# Finetune Steps	MAESTRO	Cerberus4	GuitarSet	MusicNet	Slakh	URMP
small (ours)	1M	0	.04	.00	.13	.03	.00	.19
base (ours)	500k	0	.03	.00	.12	.03	.00	.22
small (MT3)	0	525k	.28	.07	.19	.14	.02	.17
base (ours)	500k	1000	**.39**	**.11**	**.29**	**.17**	**.06**	**.33**

Table 4. Onset+Offset+Program F1 scores (considering pitch, instrument, and onset/offset times) using the LODO methodology, where each dataset in turn is held out from training and used only for evaluation. The first two rows use no finetuning, the third row is taken directly from Gardner et al. [20], and the fourth row uses a model pretrained on monophonic mixes and finetuned for only 1000 steps. LODO scores are lower for all datasets, but pretraining provides a considerable boost. This evaluation is somewhat unfair to Slakh as it contains instruments that do not appear in the other datasets.

Model	MAESTRO	Cerberus4	GuitarSet	MusicNet	Slakh2100	URMP
			Frame F1			
small (MT3)	.60	.55	.58	.53	.55	.76
base (ours)	**.65**	**.57**	**.69**	**.62**	**.64**	**.82**
			Onset F1			
small (MT3)	.28	.21	**.78**	.18	.14	.23
base (ours)	**.83**	**.54**	.71	**.48**	**.45**	**.54**
			Onset+Offset+Program F1			
small (MT3)	.28	.07	.19	.14	.02	.17
base (ours)	**.39**	**.11**	**.29**	**.17**	**.06**	**.33**

Table 5. Zero-shot transcription improvement over MT3 for Frame, Onset, and Onset+Offset+Program F1.

4.3 Zero-Shot Experiments

While standard in AMT, the methodology of the previous experiment is unsatisfactory in that for most real applications, we want to be able to automatically transcribe recordings from musical domains that were not present in our training data; splitting a homogeneously-constructed dataset into "training" and "test" partitions is not sufficient. This "out-of-domain" or "zero-shot" transfer is considered an extremely challenging task in AMT [13, 20]. We simulate the zero-shot condition with a leave-one-dataset-out (LODO) methodology, similar to the experiment described in the appendix of Gardner et al. [20].

For each of the five evaluation datasets or "folds" (since Slakh and Cerberus overlap we combine them into a single fold), we train a model on four of the folds and test on the other fold. This ensures that the model has not seen any data from the test domain during training, a much more difficult task as the model must be robust to a wider range of instrument timbres and recording conditions. Under LODO evaluation, Gardner et al. report an Onset+Offset+Program F1 score of less than 0.3 for all datasets, and less than 0.2 for all non-MAESTRO datasets.

Our results for the LODO evaluation are shown in Tables 4 and 5. Although our LODO F1 scores are lower than in the supervised case due to the difficulty of the task, pretraining on monophonic mixes increases the F1 score for all datasets. This is unsurprising, as our pretraining data exposes the model to a much wider range of instrument sounds and recording conditions. Possibly more surprising is that the pretrained model does very poorly without further finetuning on existing datasets (first two rows of Table 4); we have no satisfying explanation for this phenomenon and leave it to future work.

5. CONCLUSION

We have shown that we can take advantage of the relative ease of monophonic music transcription to improve upon the state of the art in multi-instrumental polyphonic music transcription by obtaining a large number of monophonic recordings, heuristically transcribing them, and mixing several of them together at a time as pretraining examples for a neural network model. This yields improvements across many datasets, in both standard and zero-shot multi-task settings. Notably, this pretraining boosts performance of downstream transcription models despite the fact that the pretraining audio is musically "incoherent", consisting of randomly-mixed monophonic audio tracks without regard to key, tempo, style, composition, or instrumentation.

Our work provides the first evidence that Internet-scale pretraining can be used to improve Transformer-based AMT models, and we do so with a simple set of heuristics that is straightforward to implement and fast to execute. While the benefits of large-scale pretraining for large Transformer models are well-known, the benefits of pretraining have yet to arrive to the AMT community, despite the wide availability of unlabeled audio data relative to the amount of audio with transcription-quality labels. We hope that the methods presented here, along with the accompanying open-source checkpoints and code, enable further advances in using large-scale audio data for scaling AMT models beyond the scope of existing supervised datasets.

6. REFERENCES

[1] MIDI Manufacturers Association and others, "The complete MIDI 1.0 detailed specification," *Los Angeles, CA, The MIDI Manufacturers Association*, 1996.

[2] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *ICLR*, 2019.

[3] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, "GuitarSet: A dataset for guitar transcription." in *ISMIR*, 2018.

[4] M. Cartwright and J. P. Bello, "Increasing drum transcription vocabulary using data synthesis," in *DAFX*, 2018.

[5] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *WASPAA*, 2019.

[6] J. Thickstun, Z. Harchaoui, and S. M. Kakade, "Learning features of music from scratch," in *ICLR*, 2017.

[7] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," in *ICASSP*, 2018.

[8] J. W. Kim and J. P. Bello, "Adversarial learning for improved onsets and frames music transcription," *arXiv preprint arXiv:1906.08512*, 2019.

[9] R. Kelz, S. Böck, and G. Widmer, "Deep polyphonic ADSR piano note transcription," in *ICASSP*, 2019.

[10] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," in *ICASSP*, 2020.

[11] K. W. Cheuk, D. Herremans, and L. Su, "ReconVAT: A semi-supervised automatic music transcription framework for low-resource real-world data," in *ACM Multimedia*, 2021.

[12] K. W. Cheuk, Y.-J. Luo, E. Benetos, and D. Herremans, "The effect of spectrogram reconstruction on automatic music transcription: An alternative approach to improve transcription accuracy," in *ICPR*, 2021.

[13] B. Maman and A. H. Bermano, "Unaligned supervision for automatic music transcription in the wild," *arXiv preprint arXiv:2204.13668*, 2022.

[14] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text Transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[15] J. Beal, H.-Y. Wu, D. H. Park, A. Zhai, and D. Kislyuk, "Billion-scale pretraining with vision transformers for multi-task visual representations," in *WACV*, 2022, pp. 564–573.

[16] O. Hrinchuk, M. Popova, and B. Ginsburg, "Correction of automatic speech recognition with Transformer sequence-to-sequence model," in *ICASSP*, 2020.

[17] D. Hernandez, J. Kaplan, T. Henighan, and S. McCandlish, "Scaling laws for transfer," *arXiv preprint arXiv:2102.01293*, 2021.

[18] Y. Tay, M. Dehghani, J. Rao, W. Fedus, S. Abnar, H. W. Chung, S. Narang, D. Yogatama, A. Vaswani, and D. Metzler, "Scale efficiently: Insights from pre-training and fine-tuning Transformers," *arXiv preprint arXiv:2109.10686*, 2021.

[19] K. Lu, A. Grover, P. Abbeel, and I. Mordatch, "Pretrained Transformers as universal computation engines," *arXiv preprint arXiv:2103.05247*, 2021.

[20] J. P. Gardner, I. Simon, E. Manilow, C. Hawthorne, and J. Engel, "MT3: Multi-task multitrack music transcription," in *ICLR*, 2022.

[21] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[22] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *ICCV*, 2019.

[23] J.-H. Kim, W. Choo, and H. O. Song, "Puzzle mix: Exploiting saliency and local statistics for optimal mixup," in *ICML*, 2020.

[24] L. Zhang, Z. Deng, K. Kawaguchi, A. Ghorbani, and J. Zou, "How does mixup help with robustness and generalization?" in *ICLR*, 2020.

[25] S. Thulasidasan, G. Chennupati, J. A. Bilmes, T. Bhattacharya, and S. Michalak, "On mixup training: Improved calibration and predictive uncertainty for deep neural networks," in *NeurIPS*, 2019.

[26] L. Callender, C. Hawthorne, and J. Engel, "Improving perceptual quality of drum transcription with the Expanded Groove MIDI Dataset," *arXiv preprint arXiv:2004.00188*, 2020.

[27] J. Salamon, R. M. Bittner, J. Bonada, J. J. Bosch, E. Gómez Gutiérrez, and J. P. Bello, "An analysis/synthesis framework for automatic f_0 annotation of multitrack datasets," in *ISMIR*, 2017.

[28] E. Fonseca, A. Jansen, D. P. W. Ellis, S. Wisdom, M. Tagliasacchi, J. R. Hershey, M. Plakal, S. Hershey, R. C. Moore, and X. Serra, "Self-supervised learning from automatically separated sound scenes," in *WASPAA*, 2021.

[29] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *ICASSP*, 2017.

[30] L. Prétet, R. Hennequin, J. Royo-Letelier, and A. Vaglio, "Singing voice separation: A study on training data," in *ICASSP*, 2019.

[31] X. Song, Q. Kong, X. Du, and Y. Wang, "CatNet: Music source separation system with mix-audio augmentation," *arXiv preprint arXiv:2102.09966*, 2021.

[32] Q. Kong, Y. Cao, H. Liu, K. Choi, and Y. Wang, "Decoupling magnitude and phase estimation with deep ResUNet for music source separation," in *ISMIR*, 2021.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[34] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, "Sequence-to-sequence piano transcription with Transformers," in *ISMIR*, 2021.

[35] Y. Wu, E. Manilow, Y. Deng, R. Swavely, K. Kastner, T. Cooijmans, A. Courville, C.-Z. A. Huang, and J. Engel, "MIDI-DDSP: Detailed control of musical performance via hierarchical modeling," in *ICLR*, 2022.

[36] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-MIDI alignment and matching*. Columbia University, 2016.

[37] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, 2018.

[38] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "`mir_eval`: A transparent implementation of common MIR metrics," in *ISMIR*, 2014.

ATTENTION-BASED AUDIO EMBEDDINGS FOR QUERY-BY-EXAMPLE

Anup Singh[1,2] **Kris Demuynck**[1] **Vipul Arora**[2]

[1] IDLab, Department of Electronics and Information Systems, imec - Ghent University, Belgium
[2] Department of Electrical Engineering, Indian Institute of Technology Kanpur, India

{anup.singh, kris.demuynck}@ugent.be, vipular@iitk.ac.in

ABSTRACT

An ideal audio retrieval system efficiently and robustly recognizes a short query snippet from an extensive database. However, the performance of well-known audio fingerprinting systems falls short at high signal distortion levels. This paper presents an audio retrieval system that generates noise and reverberation robust audio fingerprints using the contrastive learning framework. Using these fingerprints, the method performs a comprehensive search to identify the query audio and precisely estimate its timestamp in the reference audio. Our framework involves training a CNN to maximize the similarity between pairs of embeddings extracted from clean audio and its corresponding distorted and time-shifted version. We employ a channel-wise spectral-temporal attention mechanism to better discriminate the audio by giving more weight to the salient spectral-temporal patches in the signal. Experimental results indicate that our system is efficient in computation and memory usage while being more accurate, particularly at higher distortion levels, than competing state-of-the-art systems and scalable to a larger database.

1. INTRODUCTION

Audio fingerprinting is the principal component of an audio identification task. Finding perceptually similar audio in a massive audio corpus is computationally and memory expensive. Audio fingerprinting is a technique that derives a content-based audio summary and links it with similar audio fragments in the database. It allows for an efficient and quick search against other audio fragments. There are several possibilities for fingerprinting applications on digital devices, such as smartphones and TVs, that are becoming ubiquitous. Music identification on mobile devices, based on query-by-example, is a common use case in which a user hears a song in a public area and wants additional information about it [1, 2]. The second-screen service is another interesting fingerprinting application gaining attention [3]. It provides meta information of the broadcast content a user is watching/listening to on their devices. In addition, the broadcast sponsors also get benefits from informing viewers and listeners of their products and services. With the ease of music sharing across digital platforms, there is an increasing need to identify copyrighted content, a task which can also be accomplished by audio fingerprinting.

An audio fingerprinting system faces certain challenges for reliable audio identification in a real-world context. First, it should identify the query audio snippets corrupted with distortions such as background noise and reverberation. Secondly, the system must recognize audio using a few seconds of the audio snippet, which is crucial for fingerprinting systems embedded in digital devices to provide users with an interactive experience. Lastly, the system should generate fingerprints and search in a database in a computationally and memory-efficient manner.

In the past several decades, many audio fingerprinting systems have been developed for audio retrieval tasks. The fingerprinting method proposed by Wang *et al.* [4] (Shazam) is widely used. It captures a set of salient peaks in the audio spectrogram, assuming they remain unaffected under audio distortions. Further, these peaks are transformed into hashes to enable a fast search. Philips fingerprinting system [5] is another widely known approach that generates binary fingerprints based on energy changes across spectral-temporal space. However, this approach is computationally intensive. Another approach, named Waveprint, has been introduced in [6]. Waveprint computes the binary fingerprints using top-k wavelets and subsequently processes them using the Min-Hash technique to obtain the compact representations. This system deploys a fast indexing algorithm known as LSH (locality-sensitive hashing) for an efficient audio search. The system proposed in [7] utilizes pseudo-sinusoidal components to derive fingerprints robust against noise and reverberation. These systems, however, rely on handcrafted features that make it difficult to accurately identify the query audio when it is distorted with severe background noise and reverberation. Moreover, these systems require long (>5s) audio queries to deliver accurate results.

Deep learning, particularly unsupervised learning, has recently been introduced in the audio fingerprinting domain. [8] proposed SAMAF which utilizes a sequence-to-sequence autoencoder consisting of LSTM layers. Google's music recognizer system [9] trains a CNN based on the triplet loss function to generate compact audio fin-

© A. Singh, K. Demuynck, and V. Arora. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** A. Singh, K. Demuynck, and V. Arora, "Attention-Based Audio embeddings for Query-by-Example", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

gerprints. Another approach proposed in [10] trains a similar encoder as [9] using the contrastive learning framework and performs a maximum inner product search within a minibatch during training.

Typically, the audio is rich in content and contains irrelevant and redundant information that needs to be suppressed or eliminated to generate discriminative audio embeddings. This fact gives rise to the question of how to enable a CNN to capture the salient information in the signal and suppress the irrelevant ones. In this work, we seek to address this problem using the attention mechanism, inspired by their success in the audio domain [11, 12]. To this end, we propose to augment the resnet-like architecture with a channel-wise spectral-temporal attention mechanism. The temporal attention mechanism [13] has been widely employed in the recurrent neural networks to reweigh the recurrent output at different time indices and combine them to produce a meaningful feature vector at a particular time index. However, the spatial attention mechanism, i.e. attention on feature dimensions, has not been investigated earlier in the context of robust audio representations. Therefore, with the proposed approach, we aim to learn a multidimensional attention mask applied to the CNN features that assign more weight to the salient spatial patches and vice versa. As a result, we expect that the CNN will generate robust audio embeddings. In addition, our work focuses on performing a comprehensive search for our system to be applicable in audio synchronization tasks.

2. PROPOSED METHOD

Our work aims to generate embeddings for each audio segment of length L, extracted with a hop size of H from an audio track.

Our method builds on simCLR [14], a simple contrastive framework for learning visual representations. It maximizes the similarity between latent representations corresponding to an image under different augmented views using the contrastive loss function.

We employed this framework in the audio domain by mapping pairs of spectrograms corresponding to clean audio and the distorted one closer to one other in the latent space. In our work, the clean audio was distorted with signal distortions such as background noise and reverberation. Furthermore, a time offset was added to the distorted audio.

2.1 Audio Preprocessing

We randomly select a segment x_{clean} of length L from an audio track and resample it to a sampling rate of F_s for training the neural network model. Note that randomly chosen segments may be mostly silent and may thus not contribute to the training of the model. Moreover, such segments introduce errors in the retrieval process. Therefore, the segments with energy levels lower than a threshold t were discarded.

2.2 Data Augmentation pipeline

We generate a positive sample x_{pos} corresponding to each x_{clean} by stochastically applying a sequence of augmentations to x_{clean}. The following augmentations are considered:

- **Time offset**: A temporal shift of up to 40% of H is added to deal with the potential temporal inconsistencies in the real world situation.
- **Noise mixing:** A randomly selected background noise is added in the range of 0-25dB SNR.
- **Reverberation:** The audio segment is filtered with a randomly selected room impulse response to simulate the room acoustic environments.
- **SpecAugment:** Two randomly chosen time and frequency maskings are applied in the spectrogram. The width of the mask is set to 0.1% of their respective axis dimension. Moreover, this augmentation serves as a regularizer to prevent overfitting.

Layer	Input size	Output size
Encoder:		
CNN layer	1×64×96	32×64×96
ResBlock1	32×64×96	32×64×96
ResBlock2	32×64×96	64×32×48
...		
ResBlock6	512×4×6	1024×2×3
Flatten		6144
Projection Head:	$d * i$	$d * o$
Conv1D + ELU	128×48	128×32
Conv1D	128×32	128×1

Table 1. The proposed model employs a resnet-like architecture as the encoder and a projection head that maps the encoder output to a low-dimensional latent space. The projection head consists of 2 linear layers, each split into d branches with input size i and output size o. The model takes a log Mel spectrogram as the input.

2.3 Architecture

2.3.1 CNN encoder with Spatial-Temporal Attention

Residual networks [15] are characterized by skip connections that circumvent some intermediate layers and merge their input and output. The main advantage of such architectures is to prevent the vanishing gradient problem, which hinders training deep network architectures. Therefore, we propose a resnet-like architecture enhanced by the spatial-temporal attention mechanism that enables the CNN to learn discriminative audio representations.

Our proposed architecture contains a front-end that consists of a CNN layer and a resnet block with a kernel stride of 1x1 to retain the full temporal information since we want the discriminative embeddings to also be able to distinguish between fragments that have a similar timbre but different temporal evolution, for example adjacent audio

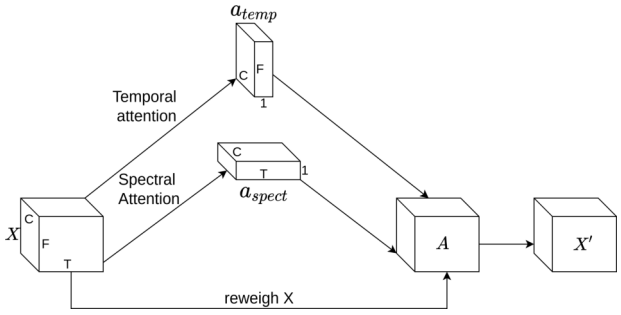


Figure 1. The illustration of spectral-temporal attention mechanism on input feature.

segments. Furthermore, the back-end consists of subsequent resnet blocks containing 2 convolutional layers. The spatial dimensions of each block are downsampled using a stride of 2x2, while the depth of each block is designed to be double of the preceding layer. Note that every convolution layer uses kernels of size 3x3, and a ReLU activation and batch normalization follows each convolution layer. The overall structure of the architecture is presented in Table 1.

CNNs have demonstrated their ability to extract complex features from low-level features, such as the log Mel spectrogram. However, the CNN features are translation-invariant, which implies that the spatial regions are treated equally in the feature map, which may not be useful in the audio context. Therefore, using the attention mechanism, we aim to enhance the interesting spatial patches by assigning more weight to time indices and frequency bands containing salient information and vice versa.

In order to apply the spectral-temporal attention mechanism, a channel-wise mask is computed and applied to the CNN features X of dimension $C \times F \times T$. We use two different attention weights, denoted by $a^{temp} \in \mathbb{R}^{C\times F\times 1}$ for temporal attention and $a^{spect} \in \mathbb{R}^{C\times 1\times T}$ for spectral attention that are modeled as:

$$\boldsymbol{a^{temp}} = \mathrm{softmax}(\boldsymbol{X^T W_{temp}}), \tag{1}$$

$$\boldsymbol{a^{spect}} = \mathrm{softmax}(\boldsymbol{X^T W_{spect}}), \tag{2}$$

where $\boldsymbol{W_{temp}}$ and $\boldsymbol{W_{spect}}$ are learnable weights. Furthermore, the spectral-temporal attention mask $\boldsymbol{A} \in \mathbb{R}^{C\times F\times T}$ is computed using the outer product between attention weights:

$$\boldsymbol{A} = \boldsymbol{a^{spect}} \otimes \boldsymbol{a^{temp}} \times S \tag{3}$$

where, S is a scaling factor to rescale the attention mask to enable easy gradient flow during model training. Finally, the CNN feature map is reweighted using the mask $\boldsymbol{A}$ as:

$$\boldsymbol{X'} = \boldsymbol{A} * \boldsymbol{X} \tag{4}$$

2.3.2 Projection Head

The final output of the CNN encoder is projected into a lower-dimensional latent space via a projection head. The fully connected layers in the projection head result in a large number of model weights. Therefore, as used in previous studies [9, 16], we utilize the split head to split the encoder output into d branches. Each branch is then passed through the corresponding linear layers and their final outputs are finally concatenated to generate a d-dimensional embedding, followed by L^2 normalization.

2.4 Contrastive Learning framework

We employ a contrastive learning approach [14] for training the model to map the similar audio samples closer together and pull away different audio samples. As discussed in Section 2.2, pairs of clean segments and their corresponding distorted versions are generated for training the model. These pairs are equivalent to the anchor and positive sample pairs as denoted in the simCLR framework. For each sample in a mini-batch of size N (N/2 pairs), there are N-2 negative samples since we also allow a positive sample to be an anchor sample in the batch. We train the proposed model using these pairs that are mapped into the latent space where the contrastive loss is formulated. We use the normalized temperature-scaled cross-entropy loss as devised in [14]:

$$\mathcal{L}_{\mathrm{i,j}} = -\log \frac{\exp(\mathrm{sim}(\mathrm{e_i}, \mathrm{e_j})/\tau)}{\sum_{\mathrm{k}=1}^{\mathrm{N}-1} 1_{[\mathrm{k}\neq\mathrm{i}]} \exp(\mathrm{sim}(\mathrm{e_i}, \mathrm{e_k})/\tau)}, \tag{5}$$

where, τ is a temperature hyperparameter that allows learning from hard negatives. The cosine similarity is used to calculate pairwise similarity.

2.5 Database-creation and Indexing

The trained model is used as a fingerprinter to extract fingerprints from the audio tracks. The fingerprint for a given audio track is generated as follows: First, we extract segments of length L with a hop size of H. These segments are pre-processed as mentioned in Section 2.1. Then, the audio segments are transformed into log Mel spectrograms and fed to the model to generate encoded embeddings. We term these embeddings as subfingerprints, which form a fingerprint of the entire audio track when stacked in the time–ordering sequence. Finally, we extract the subfingerprints of every audio segment in the database and link them with their corresponding timestamps and the audio identifier to create a reference database.

Finding the nearest neighbors for a sample in a d-dimensional space is not a trivial task (when *d>50*) [6]. Also, the brute-force search strategy in a massive database is computationally expensive. Therefore, we employ the Locality Sensitive Hashing (LSH) [17] to index the subfingerprints database. It allows for efficient retrieval by only comparing a fraction of the dataset to retrieve the exact match. Moreover, it adds the noise-robustness property to the retrieval process.

2.6 Retrieval process

This section describes the process of finding the closest matching audio snippet from an indexed database, given a

query audio snippet. For a given query audio, we follow the same procedure to generate subfingerprints as mentioned in the previous subsection. Suppose the query $Q = (q_m)_{m=1}^{M}$ is composed of M subfingerprints. We use LSH to retrieve the nearest match for each subfingerprint q_m, denoted as I_m (the index). To identify the closest matching audio track, we retain counts of the audio identifiers corresponding to the retrieved matches. The query audio is finally identified with the audio identifier with the maximum counts.

We also locate the precise timestamp of the query audio in the identified audio track. To accomplish this, we first eliminate the retrieved matches that do not belong to the identified audio track. Next, we generate the set of candidate sequences S_i of length M with their starting indices as $I_i = I_m - m$. Finally, we select candidate sequence S_i, which have at least 50% intersecting retrieved indices. The timestamp corresponding to I_i is chosen as the matching timestamp of query audio in the retrieved audio track.

3. EXPERIMENTAL SETUP

3.1 Dataset

- **Free Music Archival (FMA)** [18]: is an open, large-scale, and easily accessible dataset commonly used for various music information retrieval tasks. We used the *fma_large* and *fma_medium* versions of the dataset for model training and testing, respectively. There are 106k and 25k 30s audio clips in the *fma_large* and *fma_medium*, respectively. Note that we excluded the shared clips between both datasets for training the model.
- **Noises:** We used a range of real-world background noises for training and testing. We extracted the noise signals from database [1], which includes a wide range of sounds from the MUSAN corpus [19], for training the model. For testing the system, we used the ETSI database [2], from which we chose seven distinct background noises: Babble, Living Room, Cafeteria, Car, Workplace, Traffic, and Train Station.
- **Room Impulse Responses (RIRs):** We used real RIRs [1] corresponding to different acoustic environments, ranging from a small room to a large hall, for training the model. We selected six RIRs from Aachen Impulse Response Database [20] with t60 values ranging from 0.1s to 0.8s for testing the system.

3.2 Implementation details

We compared our approach with a baseline system [10] that generates fingerprints using an encoder similar to Now-Playing's [9] architecture and does a comprehensive search. To the best of our knowledge, it is the only method that employs a neural network model to generate robust audio fingerprints and performs better than conventional approaches. We chose the log Mel spectrograms as input to our model and the baseline method. Table 2 lists the experiments configurations used for developing our audio fingerprinting system.

Parameter	Value
Sampling rate (F_s)	16 kHz
Audio segment length (L)	960 ms
Energy threshold (t)	0 dB
log Mel spectrogram dimensions (F $\times$ T)	64 $\times$ 96
Subfingerprint hop length (H)	100 ms
Subfingerprint dimensions (d)	128
Batch size (N)	512
Scaling factor (S)	100
LSH configuration:	
Tables	50
Hash bits	18
Number of probes	200

Table 2. Experiments configurations

We trained the models with the Adam [21] optimizer for 150 epochs using the cyclic learning rate. The initial learning rate was 5e-4 and reached a maximum of 5e-2 in 40 epochs. We also tweaked the temperature hyperparameter in the [0.01-0.1] range and found no significant benefits. The model was trained on a single NVIDIA Tesla V100 GPU for about 40 hours.

We built a reference database of $\sim$7.3M fingerprints using the *fma_medium* dataset. We used LSH implementation [3] to index the generated audio subfingerprints. Furthermore, the retrieval performance was equivalent to the brute force search after fine-tuning the LSH, with less than a 0.1% drop in retrieval accuracy at various distortion levels.

3.3 Evaluation metric

We used the following metric for system evaluation at audio/segment level retrieval:

$$accuracy = \frac{n\ hits\ @\ top\text{-}1}{n\ hits\ @\ top\text{-}1 + n\ miss\ @\ top\text{-}1} \times 100 \quad (6)$$

Note that a match is declared correct for the segment level search if the located timestamp of the query in the correct retrieved audio is within $\pm$ 50 ms.

3.4 Search query

We used the *fma_medium* dataset to generate 10,000 search queries, which were randomly extracted from different audio tracks. To assess system performance at different distortion levels, we distorted queries with noise, reverberation, and a combination of both. To generate a noisy reverberant audio query, we first convolved both the audio and the noise signal with the RIR corresponding to a t60 level

[1] https://www.openslr.org/28/
[2] https://docbox.etsi.org/
[3] https://github.com/FALCONN-LIB/FALCONN

of 0.5s and then added both signals at SNR levels ranging from 0dB to 25dB. In addition, we created queries of lengths: 1s, 2s 3s, and 5s to test our system efficiency for varying lengths.

4. RESULTS AND DISCUSSION

In the following, we present the performance of our system under different distortion conditions. We compare our system with the baseline system as mentioned in the previous section and the Audfprint system.

4.1 VS. Baseline system

- **Noise:** Table 3 presents the performance of the systems in noisy conditions. It can be seen that our system performs better than the baseline system by a reasonable margin, particularly at the 0dB and 5dB SNR levels. Moreover, our system can precisely locate the timestamp with reasonable accuracy, given enough query length (>2s) in very high noise conditions too. The performance gap narrows with the increase in SNR level and becomes smaller from 20dB onwards. We noted that the performance gap between systems was less than 2% at SNR levels of 20dB and more, irrespective of the query lengths.

- **Reverb:** As can be seen in Table 4, the retrieval accuracy of the systems drops with an increase in t60 levels, i.e., high reverberant environments. Furthermore, we discovered that the reverberation causes embeddings correspondings to adjacent audio segments to be very similar, which is most likely the reason that the correct match was not found at the top-1 rank in many cases, which resulted in low retrieval performance of the system. Nevertheless, the presented results indicate that our system performs effectively even in high reverberation environments for short query snippets. The baseline system is quite effective in a low reverberation environment with more than 80% retrieval accuracy. However, its performance falls short at higher reverberations, even with long query snippets.

- **Noise and Reverb:** The systems were tested in a more challenging situation by considering a noisy and reverberant environment. As shown in Table 5, the performance of both systems degrades with the addition of reverberation compared to their performance under noisy conditions. Due to added reverberation, the retrieval accuracy of our system drops by around 15% at 0dB SNR, but it improves on the less noisy conditions. On the contrary, the baseline system performs poorly, particularly at 0dB SNR. Furthermore, the accuracy gap remains over 15% compared to its performance in noisy conditions, even at higher SNR levels. It indicates that our system is more resilient against noisy and reverberant environments than the baseline system.

Method	Query length (s)	0dB	5dB	10dB	15dB
Ours	0.96	**76.8**	**84.8**	**87.4**	**88.5**
Baseline		62.7	79.4	85.5	87.6
Ours	2	**82.7**	**90.8**	**93.2**	**94.1**
Baseline		71.1	86.5	90.4	92.0
Ours	3	**83.9**	**92.9**	**94.6**	**95.5**
Baseline		76.8	88.5	91.4	93.8
Ours	5	**85.8**	**93.9**	**94.7**	**96.8**
Baseline		79.8	89.4	91.5	94.2

Table 3. Top-1 hit rate (%) performance in the segment-level search for varying query lengths in noisy conditions.

Method	Query length (s)	0.2s	0.4s	0.5s	0.7s	0.8s
Ours	0.96	**85.1**	**84.1**	**78.8**	**83.3**	**74.4**
Baseline		78.4	75.5	67.5	75.1	62.2
Ours	2	**89.1**	**87.3**	**80.6**	**85.4**	**75.0**
Baseline		85.5	81.3	72.5	78.6	64.9
Ours	3	**90.1**	**87.9**	**81.0**	**86.8**	**76.3**
Baseline		87.2	83.3	73.9	80.8	66.9
Ours	5	**91.2**	**89.6**	**82.6**	**88.4**	**77.4**
Baseline		87.8	84.3	75.1	81.9	67.6

Table 4. Top-1 hit rate (%) performance in the segment-level search for varying query lengths in reverberant conditions.

The above-stated results show that our system performs reasonably well with short query snippets in different distortion environments. Furthermore, it indicates that our system does not require long queries to achieve reasonable performance at higher distortion levels. However, the performance of the systems improves with the increase in the query length using our proposed simple yet effective sequence search strategy. Therefore, the system can be fed with longer queries to obtain more reliable results.

Attention mechanism effect: Based on system performance in a noisy reverberant environment, we examine the effectiveness of adding an attention mechanism to the model. Table 6 shows that the attention mechanism improves retrieval accuracy, particularly at low SNR levels, with small benefits at higher SNR levels. Furthermore, we noted similar results hold true in other distortion conditions, i.e., noisy and reverberant environments. These results support that the spatial-temporal attention mechanism enhances the CNN to generate robust audio embeddings.

Embedding dimensions: We examined the effect of the audio embedding dimension on the system performance using the 0.96s queries. We observe that shrinking the dimensions from 128 to 64 has little impact on the system, especially in the reverberant environment. However, in the noisy reverberant and noisy environments, retrieval

Method	Query length(s)	0dB	5dB	10dB	15dB
Ours	0.96	**60.3**	**76.6**	**81.3**	**82.8**
Baseline		27.3	58.7	70.7	73.9
Ours	2	**66.4**	**83.5**	**86.9**	**88.0**
Baseline		39.0	69.6	76.5	78.7
Ours	3	**67.9**	**85.1**	**88.2**	**89.3**
Baseline		47.1	75.2	80.2	81.4
Ours	5	**69.5**	**87.1**	**90.5**	**91.9**
Baseline		54.7	77.3	81.8	82.8

Table 5. Top-1 hit rate (%) performance in the segment-level search for varying query lengths in noisy reverberant conditions.

	0dB	5dB	10dB	15dB	20dB
Attention	60.3	76.6	81.3	82.8	84.6
No attention	52.4	68.1	75.7	79.9	82.3

Table 6. The effect of added attention mechanism on Top1-hit rate performance of the system in noisy reverberant environments for 0.96s long queries.

accuracy declines by 5.6% and 3.8% at 0dB SNR, respectively, whereas accuracy losses are minimal at other SNR levels. Furthermore, increasing the number of dimensions from 128 to 256 does not provide significant improvements. This investigation allows reducing dimensions further without a significant performance drop to resolve the space constraints problem.

Computational and Memory load: We further investigated the efficiency of our system based on its computational and memory requirements. The final size of the sub-fingerprints database is roughly 1.25GB, with 128 32-bit floating numbers representing each audio subfingerprint. We use the *Intel Xeon Platinum 8268* CPU to do an in-memory search due to the small database size. We report that the LSH takes about 0.01s to retrieve top-5 matches for a query subfingerprint. Moreover, our system takes about 0.25s to process a 3s long query and locate its timestamp in the identified reference audio. It is also worth noting that the retrieval process can be sped up by employing a parallel search method. Furthermore, there is scope for investigating the computation load reduction by storing subfingerprints in low-bit floating numbers without a significant drop in the retrieval accuracy.

4.2 VS. Audfprint

We also compare our system with the Audfprint [4] based on the Shazam method. We notice that Audfprint performs poorly in the segment level search, particularly at high distortion levels. Therefore, we only compare its performance with ours for the audio identification task. Moreover, its performance deteriorates with short query lengths; hence we present the retrieval results with 5s audio query snippets in Table 7.

It can be seen that our system delivers excellent retrieval accuracy, with over 95% accuracy under high distortion conditions. On the contrary, the Audfprint method performance degrades severely at high distortion levels, which indicates that our system also outperforms the conventional approach for the audio identification task. It should be noted that the Audfprint system uses a hash table to index the database, resulting in reduced fingerprint database size. Furthermore, the size of the database generated by Audfprint is around 400 MB, which is roughly three times less than ours.

Distortion	Method	0dB	5dB	10dB	15dB
Noise	Ours	**95.0**	**98.7**	**98.9**	**99.2**
	Audfprint	72.1	82.7	89.4	91.2
Noise+ Reverb	Ours	**84.3**	**96.8**	**98.5**	**98.9**
	Audfprint	64.8	79.4	87.2	92.3
		0.2s	0.4s	0.5s	0.7s	0.8s
Reverb	Ours	**99.2**	**99.5**	**98.9**	**99.6**	**98.7**
	Audfprint	96.1	94.6	81.8	89.6	40.2

Table 7. Top-1 hit rate (%) performance in the audio-level search in different distortion conditions.

5. CONCLUSION

This paper presents an audio fingerprinting system robust against high noise and reverberation conditions. Our work focuses on generating robust audio embeddings by employing a contrastive learning framework. Moreover, we propose to enhance CNN with the channel-wise spectral-temporal attention mechanism to reweigh the CNN features. This enables CNN to assign more weight to salient patches in the CNN features, resulting in discriminative audio embeddings. Furthermore, our system performs a comprehensive search to precisely estimate the timestamp of the query in the identified reference audio using a simple sequence search strategy, which makes our system applicable to audio synchronization tasks. Our system performs well compared to the baseline [10] and Audfprint [4] methods. Also, our system is computationally and memory-efficient due to the compact embeddings that make our system deployable on an extensive database. The future direction of this work is to obtain discrete audio embeddings to speed up the audio retrieval process.

6. ACKNOWLEGMENT

This work has been supported by the research grant(PB/EE/2021128-B) from Prasar Bharati. We would also like to extend our gratitude to IITK Paramsanganak for their GPU computing resources.

[4] https://github.com/dpwe/audfprint

7. REFERENCES

[1] "Shazam music recognition service," http://www.shazam.com/.

[2] "Soundhound," http://www.soundhound.com/.

[3] C. Howson, E. Gautier, P. Gilberton, A. Laurent, and Y. Legallais, "Second screen tv synchronization," in *2011 IEEE International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*. IEEE, 2011, pp. 361–365.

[4] A. Wang, "The shazam music recognition service," *Communications of the ACM*, vol. 49, no. 8, pp. 44–48, 2006.

[5] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system." in *Ismir*, vol. 2002, 2002, pp. 107–115.

[6] S. Baluja and M. Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern recognition*, vol. 41, no. 11, pp. 3467–3480, 2008.

[7] T. Shibuya, M. Abe, and M. Nishiguchi, "Audio fingerprinting robust against reverberation and noise based on quantification of sinusoidality," in *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2013, pp. 1–6.

[8] A. Báez-Suárez, N. Shah, J. A. Nolazco-Flores, S.-H. S. Huang, O. Gnawali, and W. Shi, "Samaf: Sequence-to-sequence autoencoder model for audio fingerprinting," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 2, pp. 1–23, 2020.

[9] B. Gfeller, B. Aguera-Arcas, D. Roblek, J. D. Lyon, J. J. Odell, K. Kilgour, M. Ritter, M. Sharifi, M. Velimirović, R. Guo, and S. Kumar, "Now playing: Continuous low-power music recognition," in *NIPS 2017 Workshop: Machine Learning on the Phone*, 2017.

[10] S. Chang, D. Lee, J. Park, H. Lim, K. Lee, K. Ko, and Y. Han, "Neural audio fingerprint for high-specific audio retrieval based on contrastive learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3025–3029.

[11] E. Cakır, G. Parascandolo, T. Heittola, H. Huttunen, and T. Virtanen, "Convolutional recurrent neural networks for polyphonic sound event detection," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 6, pp. 1291–1303, 2017.

[12] H. Wang, Y. Zou, D. Chong, and W. Wang, "Environmental Sound Classification with Parallel Temporal-Spectral Attention," in *Proc. Interspeech 2020*, 2020, pp. 821–825.

[13] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Sep. 2015, pp. 1412–1421.

[14] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020, pp. 1597–1607.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3270–3278.

[17] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 253–262.

[18] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "Fma: A dataset for music analysis," *arXiv preprint arXiv:1612.01840*, 2016.

[19] D. Snyder, G. Chen, and D. Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[20] M. Jeub, M. Schäfer, and P. Vary, "A binaural room impulse response database for the evaluation of dereverberation algorithms," in *Proceedings of International Conference on Digital Signal Processing (DSP)*, IEEE, IET, EURASIP. Santorini, Greece: IEEE, Jul. 2009, pp. 1–4.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

SIATEC-C: COMPUTATIONALLY EFFICIENT REPEATED PATTERN DISCOVERY IN POLYPHONIC MUSIC

Otso Björklund
University of Helsinki

ABSTRACT

The use of point-set representations of music enable repeated pattern discovery to be performed on polyphonic music. The discovery of patterns containing polyphony is also enabled by the use of point-set representations. The SIA and SIATEC algorithms discover repeated patterns in point-sets by computing maximal translatable patterns and their translational equivalence classes. While the algorithms are relatively efficient, their application to larger pieces of music is not viable due to quadratic space complexity. This paper introcudes a novel algorithm, SIATEC-C, for repeated pattern discovery in point-set representations of music. The algorithm discovers repeated patterns and finds all of their occurrences, while running with subquadratic space complexity. The algorithm can also provide significant running time improvements over the comparable SIATEC algorithm. The computational performance of the algorithm is compared with SIATEC. The accuracy of the algorithm is also evaluated on the JKU-PDD data set.

1. INTRODUCTION

This paper presents a novel algorithm for computationally efficient repeated pattern discovery in symbolically represented polyphonic music. The main contribution of the algorithm is efficient production of candidate patterns for a post-processing phase where the patterns can be refined, and musically unimportant patterns can be filtered out.

The goal of repeated *pattern discovery* is to find musically important patterns and their occurrences in a piece of music (*intra-opus*) or in a corpus (*inter-opus*). In repeated pattern discovery no query is given by the user unlike in *pattern matching*, where the goal is to find occurrences of a query pattern [1]. Decomposing a piece of music into its constituent elements is a fundamental part of music analysis [2]. Repeated pattern discovery can thus be applied to multiple problems in computational music analysis, such as motivic analysis [3, 4], tune classification [5], segmentation [6], and style analysis [7]. A compressed representation of a piece formed by its repeated patterns can even be considered an analysis of the piece [8, 9].

© O. Björklund. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** O. Björklund, "SIATEC-C: Computationally efficient repeated pattern discovery in polyphonic music", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

Repeated pattern discovery is a challenging problem due to certain characteristics of music. Repetition is prevalent in music causing algorithms that focus just on finding repetitions to output large numbers of patterns even for short pieces of music [4, 10]. Often most of the repeated patterns discovered by an algorithm do not correspond to patterns that have musical significance [11]. Instead of relying on repetition alone for discovering musically significant pattern, methods for identifying which patterns are musically important are needed.

The SIATEC-C algorithm presented in this paper improves upon the running time and space complexity of previous point-set algorithms by avoiding the computation of patterns with large temporal gaps and small patterns that are already included in a larger pattern.

2. RELATED WORK

String representations of music have been employed for both pattern matching and discovery in monophonic music. Monophonic music can be represented as a string for pattern matching [12] or a set of strings for pattern discovery [13]. String representations have been often used for mining *closed* patterns, that is, patterns that cannot be extended without reducing the number of occurrences [4, 10]. Monophonic music can also be represented as a pitch signal which enables the use of signal processing methods. Wavelet analysis is used by [14] for repeated pattern discovery in monophonic music.

Polyphonic music can be split into monophonic voices in order to use monophonic pattern discovery methods (e.g., in [13, 15]. Voice separation can be challenging if the voice information is not included, and using monophonic pattern discovery on separate voices cannot be directly used to discover polyphonic patterns or patterns that move from one voice to another, such as *call-and-response* patterns in jazz.

Point-set representations of music enable pattern matching [16] and discovery in polyphonic music [11] with patterns that can be polyphonic. Time-shifts and transpositions of a pattern can be performed by translating the points of the pattern by a vector. The SIA-family of algorithms (see [17] for an overview) discover repeated patterns in music by computing *maximal translatable patterns* (MTP) and finding their occurrences by computing the *translational equivalence classes* (TEC) for the patterns ([11]). The definitions of MTPs and TECs are covered in section 3.

The SIA algorithm by [11] computes all MTPs in a k-dimensional point-set of n points in $O(kn^2 \log n)$ worst-case time and $O(kn^2)$ space, and the SIATEC algorithm computes the TECs of all MTPs in $O(kn^3)$ worst-case time and $O(kn^2)$ space. The worst-case space complexity of computing all MTPs has been improved upon by [18] to $O(kn)$. Another variant of SIA is the SIAR algorithm by [19], which aims to improve the computational performance of MTP computation by restricting the number of difference vectors by a sliding window.

A musical pattern may be *time-warped* and *time-scaled*. [20] present an algorithm that can discover transposed and time-warped repeating patterns in a two-dimensional point-set of n points in $O(n^2 \log n)$ time. The algorithm by [21] can discover transposed and time-scaled repeated patterns in a two-dimensional point-set in $O(n^4 \log n)$ time.

The number of patterns discovered by computing MTPs can often be very large even for small point-sets [11]. Various approaches based on the use of *compression ratio*, *compactness*, and other heuristics have been developed for selecting the musically most important patterns from the set of discovered MTPs. The COSIATEC and SIATEC-Compress algorithms [9, 22] compute a compressed representation of a point-set. The algorithm by [23] similarly aims to compute a representation formed by musically important patterns. [24] further develops the idea of computing a highly compressed representation of a point-set by recursively splitting the input point-set and refining the TECs by removing redundant translation vectors.

MTPs can contain large temporal gaps, such that, the MTP may consist of a musically important pattern and additional points called *isolated members* [19, 25]. [19] proposes a solution to the problem of isolated membership by using an additional processing stage called *compactness trawling* to split MTPs into smaller patterns without isolated members. Post-processing MTPs with compactness trawling in SIA has been found to improve precision and recall [26] over plain SIA. Compactness trawling combined with the symbolic fingerprinting method presented in [27] have been employed in the SIARCT-CFP algorithm to discover inexact occurrences of repeated patterns with high precision and recall [28].

3. BACKGROUND AND DEFINITIONS

In a point-set representation of music, note events are represented as points and a piece of music is represented as a point-set $D \subset \mathbb{R}^k$, where k is the dimensionality of the points. Often $k = 2$, where the first dimension represents the onset time of a note event, and the second dimension represents the pitch of the note event. The rest of this paper assumes that the points are two-dimensional, and $p.x$ denotes the onset time and $p.y$ the pitch of a point p. The number of points in a point-set D is denoted by $|D|$ or n. By using only the onset time of the note events, patterns can be matched based on the *inter-onset-intervals* (IOI) of the adjacent notes. This allows for variations in the durations of the notes. The IOI between consecutive note events p_1 and p_2 is defined as the difference $p_2.x - p_1.x$.

A point-set can be sorted by using a *lexicographical ordering* [11]. A two-dimensional point p_1 is considered to be less than a point p_2, if and only if, $p_1.x < p_2.x$ or $p_1.x = p_2.x$ and $p_1.y < p_2.y$. Lexicographical ordering can be extended to points of any dimensionality. A lexicographically sorted point-set is denoted D_s.

A pattern $P \subset \mathbb{R}^k$ is also a point-set, and P is said to occur in D if $P \subseteq D$. Shifting a pattern P in time and transposing it can be expressed as translation by a vector t, where $t.x$ is the time shift and $t.y$ is the transposition. Translating a pattern P by a vector t is denoted $P + t$ [11]. The points in patterns are assumed to be in ascending lexicographical order in this paper.

The maximal translatable pattern, ***MTP***, of a vector t in a point-set D is defined by [11] as

$$MTP(t, D) = \{d \mid d \in D \wedge d + t \in D\}. \quad (1)$$

The MTP of t in D is formed by the set of all points ind D that can be translated by t so that the translated points are also included in D. Negating the translation t produces the same MTP, that is, $MTP(t, D) = MTP(-t, D)$ [11], therefore only MTPs for translations that are greater than the zero vector are considered. All difference vectors between points in a point-set referred to in this paper are also assumed to be greater than the zero vector. All MTPs in a point-set can be computed with the SIA algorithm [11].

There are at most $n(n-1)/2$ MTPs in a point-set with n points. This occurs when differences between any pair of points are distinct. Conversely, there are at least $n - 1$ MTPs in a point-set. The minimum number of MTPs occurs in a point-set where all points are placed on a line with a constant distance between adjacent points. These extreme cases are unlikely to occur in point-sets representing music, however, they are useful for analysis of algorithms that compute MTPs. A point-set with a minimum number of MTPs is denoted by D_{min} and a point-set with a maximum number of MTPs is denoted by D_{max}.

A pattern P may have multiple occurrences in a point-set D. The set of all occurrences of P in D is represented by the *translational equivalence class* (TEC) [11] of the pattern. Two patterns A and B are considered translationally equivalent if, and only if, there exists a translation t such that $A = B + t$. Translational equivalence of A and B is denoted by $A \equiv_{\mathrm{T}} B$. The translational equivalence of patterns can be compared using their *vectorized representations* [11]:

$$VEC(P) = \langle p_2 - p_1, p_3 - p_2, \ldots, p_l - p_{l-1} \rangle, \quad (2)$$

where p_i denotes the ith point of a lexicographically sorted pattern P of length l. Two patterns are translationally equivalent if and only if their vectorized representations are equal [11]. The TEC of a pattern P in a point-set is the set of all subsets that are translationally equivalent to P:

$$TEC(P, D) = \{Q \mid Q \equiv_{\mathrm{T}} P \wedge Q \subseteq D\}. \quad (3)$$

The SIATEC algorithm can be used to compute the TECs of all MTPs in a point-set [11]. The TEC of a pattern P

can be represented by storing one occurrence of the pattern and the translation vectors required to produce all other occurrences of P [11].

4. THE SIATEC-C ALGORITHM

The SIATEC-C algorithm computes TECs of patterns in a point-set D, such that, the IOI between adjacent points in a pattern is at most a given threshold δ. SIATEC-C also avoids producing small patterns when the points covered by the pattern are already covered by a larger discovered pattern. SIATEC-C thus avoids producing patterns with isolated members while reducing the running time and memory footprint. This approach aims at similar results as compactness trawling in [19]. Cutting patterns at large IOI gaps has also been suggested in [29].

The outline of SIATEC-C is described in algorithm 1. The pseudocode aims at a concise presentation of the algorithm. A more thorough pseudocode description and a reference implementation of the algorithm that covers all details is also made available [1]. The algorithm takes as its inputs a point-set D and the IOI thershold δ. The first components of the points are assumed to represent to onset times of note events. The algorithm outputs the discovered patterns and all of their occurrences represented as TECs.

The algorithm begins by sorting the input point-set in ascending lexicographical order to produce the point-set D_s. The variables T and W are used for tracking a sliding window for each point in D_s in the onset dimension. The sliding windows are used in computing MTPs in order to restrict the number of MTPs that need to be kept in memory simultaneously. The array T keeps track of the index from where to continue computation on each iteration, and the array W keeps track of the upper bounds of the windows. The indexing in the pseudocode starts at 1 and array access is denoted by brackets. For T the value at index i stores the index in D_s for where the next sliding window starts. The value at index i of W stores the upper bound of the sliding window for the ith point of D_s. The indices in T are initialized to the range from 1 to n (line 4) and on line 5 the upper bounds are initialized for each point $p \in D_s$ to be $p.x + \delta$. The values in the array C keep track of the size of the largest pattern occurrence that covers the corresponding point in D_s.

The *difference index* structure I is computed by the COMPUTEDIFFINDEX function. Difference vectors between all pairs of points, p_i and p_j, for which the IOI between the points does not exceed the theshold δ, are computed. The differences along with the index-pairs $\langle i, j \rangle$ are stored in the intermediate array I'. The array is sorted in ascending lexicographical order by the difference vectors and indices. The sorted array is partitioned by the difference vectors, so that an array of entries of the form $\langle v, [\langle s_1, t_1 \rangle, \ldots, \langle s_i, t_i \rangle] \rangle$ is created. Each entry contains a difference vector v and the corresponding *source* and *target* indices. The source indices s_i are the indices of points in D_s that can be translated by v within D_s, and the corresponding target indices are of the points that are produced by translating the point at the source index by v. The index structure I is sorted in ascending order of difference vectors and all source and target indices for an entry are also in ascending order.

[1] https://github.com/otsob/siatec-c-code

Algorithm 1 SIATEC-C Algorithm

```
1:  function SIATEC-C(D, δ)
2:      D_s ← SORT_Lex(D)
3:      n ← |D_s|
4:      T ← [1, 2, ..., n]
5:      W ← INITWINDOWBOUNDS(D_s, δ)
6:      C ← [0, 0, ... 0] of n zeros
7:      I ← COMPUTEDIFFINDEX(D_s, δ)
8:      while T[1] ≤ n do
9:          M ← COMPUTEMTPS(D_s, T, W)
10:         M' ← CUTANDSORT(M, δ)
11:         for P ∈ M' do
12:             if IMPROVESCOVER(P, C) then
13:                 FINDTRANSLATORS(P, I, D_s, C)
14:                 OUTPUTTEC
```

In the main loop of the SIATEC-C algorithm (lines 8–14 of 1), MTPs are computed for translation vectors within the sliding windows by the COMPUTEMTPS function. The MTPs are computed by first computing all translations between pairs of points where the target point is within the sliding window of the source point. The indices of the source points are stored in pairs with the translations. The array thus produced is sorted in ascending lexicographical order and partitioned by the translation vectors. The function is otherwise equal to the SIA algorithm [11], except that the difference vectors are limited by the sliding windows defined by the arrays T and W, and the indices of the MTP and its translated occurrence are also stored. The sliding windows are used to avoid keeping all $O(n^2)$ differences in memory at the same time. On each iteration the indices in T are updated to the point just outside the current window and then the sliding window upper bounds in W are incremented by δ.

The produced MTPs can have gaps in them that exceed the threshold δ. Thus the MTPs are cut on line 10 to produce the set of patterns M', where the IOI between no adjacent patterns points exceeds δ. The patterns are also sorted in descending order of size to ensure that larger patterns are handled first. The function IMPROVESCOVER checks if the pattern, or its translated version, is larger than any of the patterns that cover the same points. A pattern is considered to improve the cover only if it improves the cover value of at least one point. This step reduces the number of small and duplicate patterns that would be otherwise output by the algorithm. Small patterns may be output by the algorithm even if a larger pattern covering the same points is discovered. This occurs in the case that the small pattern is found on an earlier iteration of the main loop (lines 8–14).

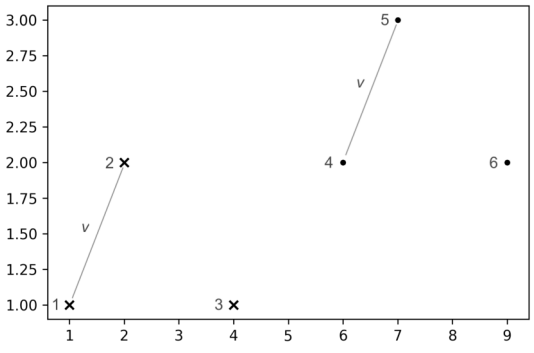
4.1 Finding translators

Finding the translators of a pattern P is achieved by traversing the index-pairs stored in I using the vectors of the vectorized representation $VEC(P)$. This is equivalent to finding translationally equivalent prefixes of P and extending the prefixes until they are equal in length to P.

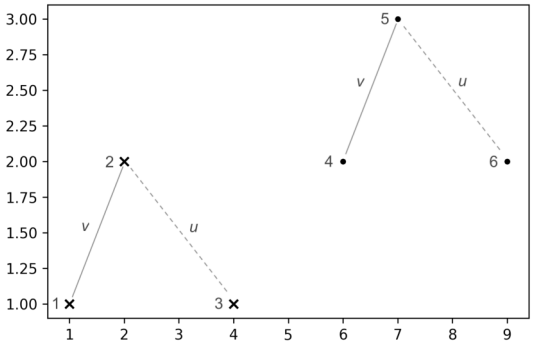
Algorithm 2 SIATEC-C: Find translators and update cover

```
1:  function FindTranslators(P, I, D_s, C)
2:      V ← VEC(P)
3:      v ← V[1]
4:      A ← { t | ⟨s, t⟩ ∈ FindIndices(v, I) }
5:      for i ∈ [2, ..., |V|] do
6:          v ← V[i]
7:          A′ ← FindIndices(v, I)
8:          A ← { t | ⟨s, t⟩ ∈ A′ ∧ s ∈ A }
9:      l ← P[|P|]
10:     τ ← { D_s[i] − l | i ∈ A }
11:     C ← UpdateCover(P, A, C, I)
12:     return τ
```

Figure 1 illustrates the process of finding the translators of a pattern P, $VEC(P) = [v, u]$ with a very minimal point-set example. The crosses and points form two three-point patterns that are translationally equivalent. First binary search is used to find the index pairs for v from I, returning the index pairs $[\langle 1, 2\rangle, \langle 4, 5\rangle]$. The second elements of these pairs are the indices of points that can be translated with u to continue translationally equivalent prefixes of P. On the next iteration the index-pairs associated with u are retrieved producing the index-pairs $[\langle 2, 3\rangle, \langle 5, 6\rangle]$. The target indices 2 and 5 of the vector v are matched with the source indices of u to find that the translationally equivalent prefixes can be extended with the points at indices 3 and 6 to find the last points of translationally equivalent occurrences of P. The translators can be computed simply as the difference between the last points of the found occurrenc es and the last point of P.



(a) Step 1: Prefix of length 2



(b) Step 2: Prefix of length 3

Figure 1: FindTranslators example

4.2 Time and space complexity

The following theorems present the worst case time and space complexity of SIATEC-C.

Theorem 4.1. *Let D be a 2-dimensional point-set with n points. Let m be the largest number of points in any span of length δ in the onset dimension and let h be the number of points in the largest MTP in D. Then the worst case time complexity of* SIATEC-C *is $O(hn^2 \log nm)$.*

Proof. Computing the difference index I requires computing $O(nm)$ difference vectors, sorting them, and partitioning. Thus ComputeDiffIndex runs in $O(nm \log nm)$ time.

The number of iterations the main loop on lines 8–14 of algorithm 1 executes is approximately $\frac{n}{\delta} = O(n)$. Computing the MTPs requires also computing $O(nm)$ difference vectors, sorting and partitioning them into MTPs, and then sorting the MTPs by size, thus running in $O(nm \log nm)$ time. However, the total amount of computation required to compute MTPs in the loop performs the same number of difference vector computations and comparisons as computing all MTPs and sorting them by size, thus the total amount of work needed for MTP computation during the execution of the algorithm is $O(n^2 \log n)$ just as in SIA [11].

For a pattern P, the size of its vectorized represenation is $|P| - 1$. The FindTranslators function is thus run on $O(n^2)$ difference vectors in total. For each difference vector, the loop finds the index pairs from I in $O(\log nm)$ time using binary search and computes the intersection of A with the source indices in A'. The number of index pairs that can be found for a difference vector v in I is equal to the size of the largest MTP in D, denoted by h. Thus computing the intersection of sorted arrays is linear in h, resulting in time complexity of $O(h \log nm)$ for a single difference vector in P. Overall, finding all translators for all produced patterns has a worst case time complexity of $O(hn^2 \log nm)$.

The overall worst case time complexity of the algorithm is thus dominated by computing the translators, resulting in a worst case time complexity of $O(hn^2 \log nm)$.

□

Theorem 4.2. *Let D be a 2-dimensional point-set with n points, and let m be the largest number of points in any span of length δ in the onset dimension. Then the worst case space complexity of* SIATEC-C *is* $O(nm)$.

Proof. Computing the difference index I requires storing $O(nm)$ difference vectors and corresponding index pairs, and after partitioning the number of difference vectors and index pairs does not increase. Therefore I takes $O(nm)$ space.

On each iteration of the main loop on lines 8–14 of algorithm 1 the MTP computation requires keeping $O(nm)$ difference vectors in memory.

In the FINDTRANSLATORS function the number of index pairs contained in A or A' is at most the equal to the size of the largest MTP in D. Therefore the space complexity of FINDTRANSLATORS is $O(n)$.

The space complexity of SIATEC-C is dominated by I and the MTP computation, therefore the worst case space complexity is $O(nm)$. □

5. RESULTS

This section contains the computational performance results of the SIATEC-C algorithm and its evaluation on the JKU-PDD dataset.

5.1 Computational Performance

The running time and memory usage of the SIA, SIAR ($r = 1$), SIATEC, and SIATEC-C algorithms was measured on two types of point-sets: D_{min} and random patterns. The point-set sizes ranged from 1000 to 10000 in increments of 1000. The D_{min} dataset was chosen as it produces the worst-case running time of SIATEC-C and can thus provide an estimate of the upper bound of running time and memory usage. Random patterns were used instead of concatenating short pieces of music together to control the sizes of MTPs in the benchmark data.

The algorithms were implemented using the Rust [2] programming language. The measurements were performed on a machine running Ubuntu 20.04 with an Intel i7-processor and 16GB of memory. The IOI threshold parameter δ of SIATEC-C was set to a 50.0 for the artificial point-sets, for music point-sets $\delta = 4$ was used throughout, corresponding to one measure in $\frac{4}{4}$ time.

The measurements for running times are plotted in figure 2 and the maximum heap usage measurements are plotted in figure 3. Log-scale is used on the y-axis as the measurements vary greatly in the range of values. The most significant running time improvements SIATEC-C can provide compared to SIATEC can be seen in the plot for the running time on the random pattern point-sets. Even with the largest point-sets, the running time of SIATEC-C is $22.2s$, while the running time of SIATEC exceeds $2500s$. On random pattern point-sets SIAR is the fastest algorithm.

In the case of the D_{min} point-sets the running time of SIATEC and SIATEC-C behaves relatively similarly. With SIAR the D_{min} produces worst-case performance leading the performance of SIAR to be comparable to that of SIA. This is explained by the worst-case time complexity of SIAR, which on a k-dimensional point-set of size n is $O(kn^3)$ [30].

The memory usage was measured using the Heaptrack software [3] that only measures heap memory. With the largest random patterns point-set SIATEC-C uses only $26.08MB$ and with the largest D_{min} point-set $78.20MB$.

On the random patterns point-sets SIAR runs with the smallest memory footprint. However, the D_{min} point-sets illustrates the quadratic space complexity of SIAR [30], with the memory footprint of SIAR exceeding that of SIATEC-C. Thus replacing SIA with SIAR in SIATEC will not guarantee a smaller memory footprint than can be obtained with SIATEC-C.

SIAR can be a very performant algorithm on many point-sets, however, its performance varies greatly depending on the size of the largest MTPs in the input point-set. In order to investigate the impact of the worst-case time and space complexities between SIATEC-C and SIAR, both algorithms were run on a point-set representation [4] of Beethoven's 9th symphony ($n = 107,355$). SIATEC-C ($\delta = 4$) ran in approximately 28 minutes with peak heap usage of $1.97GB$ while SIAR ($r = 1$) ran in approximately 1 hour 7 minutes with peak heap usage $5.64GB$. While SIAR can be the most performant algorithm on small point-sets, due to its worst-case time and space complexity there is no guarantee that it will be the most performant on large point-sets.

5.2 Evalution on JKU-PDD

The accuracy of the SIATEC-C algorithm was evaluated on the JKU-PDD data set [31]. A version of SIATEC-C without any post-processing was evaluated to investigate whether it is capable of achieving establishment precision and recall comparable to other point-set algorithms that have been shown to benefit from post-processing, e.g., compactness trawling [26, 28].

The COSIATEC and SIATECCompress compression algorithms [22] produce a compressed representation of the input point-set by selecting TECs produced by SIATEC. The algorithms COSIATEC-C and SIATEC-CCompress are otherwise equal to COSIATEC and SIATECCompress except they use SIATEC-C instead of SIATEC for producing TECs.

Table 1 displays the mean values of the MIREX metrics over the monophonic and polyphonic corpus of JKU-PDD. Compared to SIATEC and SIAR, SIATEC-C produces fewer patterns and achieves slighlty improved establishment precision and recall.

6. DISCUSSION AND CONCLUSION

In this paper we have presented a novel algorithm SIATEC-C for repeated pattern discovery in symbolic polyphonic

[2] https://www.rust-lang.org

[3] https://github.com/KDE/heaptrack

[4] Converted from `https://musescore.com/openscore/scores/5733014`.

Algorithm	Corpus	N_{points}	$N_{patterns}$	N_{gt}	P_{est}	R_{est}	$F1_{est}$	P_{3L}	R_{3L}	$F1_{3L}$	$P_{occ}(c=0.75)$	$R_{occ}(c=0.75)$	$F1_{occ}(c=0.75)$	$P_{occ}(c=0.5)$	$R_{occ}(c=0.5)$	$F1_{occ}(c=0.5)$
SIATEC	monophonic	677.2	30014.8	6.2	0.128	0.679	0.208	0.072	0.613	0.125	0.681	0.569	0.617	0.459	0.561	0.503
SIATEC-C ($\delta = 4$)	monophonic	677.2	970.0	6.2	**0.189**	**0.890**	**0.308**	**0.131**	**0.852**	**0.227**	**0.842**	**0.854**	**0.844**	**0.558**	**0.824**	**0.649**
SIAR ($r = 1$)	monophonic	677.2	5365.0	6.2	0.148	0.679	0.236	0.091	0.505	0.149	0.685	0.422	0.509	0.496	0.391	0.424
COSIATEC	monophonic	677.2	15.2	6.2	0.136	0.234	0.169	0.085	0.199	0.117	0.165	0.165	0.165	0.256	0.192	0.219
SIATECCompress	monophonic	677.2	10.6	6.2	0.124	0.116	0.114	0.068	0.091	0.075	0.000	0.000	0.000	0.000	0.000	0.000
COSIATEC-C	monophonic	677.2	28.8	6.2	0.090	0.214	0.124	0.088	0.217	0.122	0.000	0.000	0.000	0.120	0.038	0.058
SIATEC-CCompress	monophonic	677.2	21.4	6.2	0.087	0.148	0.109	0.068	0.130	0.088	0.200	0.110	0.142	0.200	0.110	0.142
SIATEC	polyphonic	1289.0	59081.8	5.4	0.105	0.690	0.178	0.066	0.595	0.117	0.677	0.543	0.593	0.499	0.530	0.501
SIATEC-C ($\delta = 4$)	polyphonic	1289.0	977.6	5.4	**0.131**	**0.775**	**0.217**	**0.097**	**0.675**	**0.164**	**0.868**	**0.708**	**0.759**	0.570	**0.645**	**0.577**
SIAR ($r = 1$)	polyphonic	1289.0	12721.4	5.4	0.116	0.635	0.195	0.089	0.483	0.147	0.683	0.476	0.544	**0.588**	0.419	0.477
COSIATEC	polyphonic	1289.0	19.6	5.4	0.091	0.196	0.122	0.056	0.172	0.083	0.157	0.157	0.157	0.290	0.224	0.253
SIATECCompress	polyphonic	1289.0	15.8	5.4	0.103	0.121	0.108	0.059	0.092	0.069	0.000	0.000	0.000	0.000	0.000	0.000
COSIATEC-C	polyphonic	1289.0	41.2	5.4	0.070	0.161	0.095	0.058	0.143	0.081	0.000	0.000	0.000	0.050	0.019	0.027
SIATEC-CCompress	polyphonic	1289.0	24.6	5.4	0.093	0.194	0.122	0.077	0.171	0.102	0.170	0.170	0.170	0.296	0.206	0.226

Table 1: Mean MIREX metrics on JKU-PDD (highest metric values in bold)

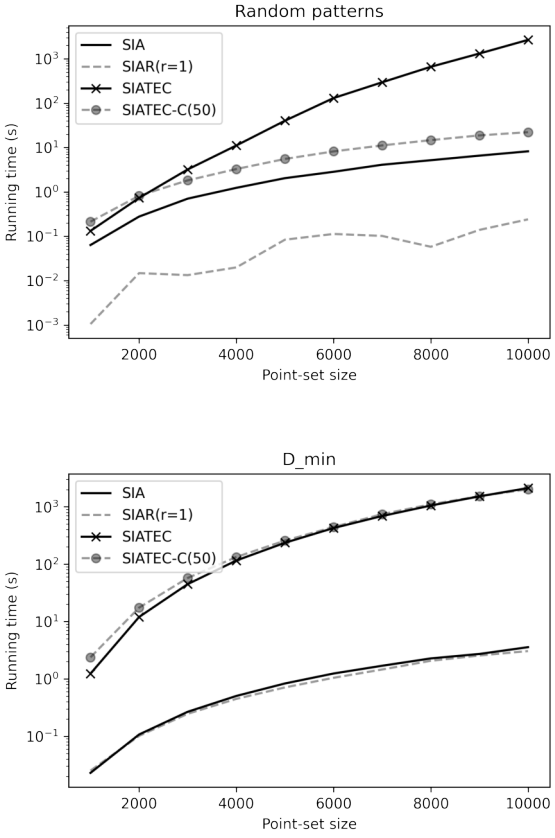


Figure 2: Running times

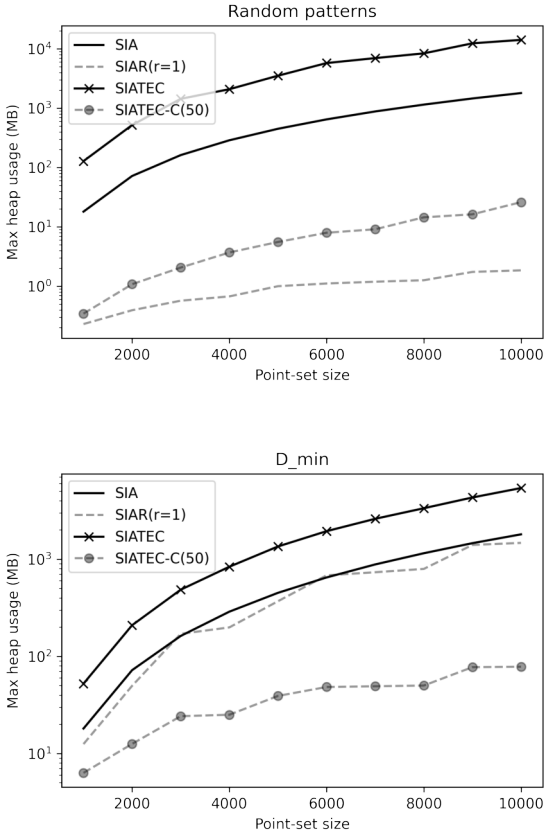


Figure 3: Maximum heap usages

music. The algorithm is based on previous research on repeated pattern discovery in polyphonic music using point-set representations of music [11].

The SIATEC-C algorithm can provide significant running time improvements over SIATEC in discovering patterns and their occurrences when the input consists of patterns that vary in size. In terms of worst-case performance, SIATEC-C does not provide improvements over SIATEC in running time. The most significant improvement SIATEC-C can provide in terms of computational efficiency is its small memory footprint, in which SIATEC-C can outperform SIAR. By keeping the memory usage small, repeated pattern discovery based on point-set representations can be applied to much longer pieces of music than previously.

The simple heuristic of cutting patterns at large IOI gaps in SIATEC-C was found to perform at least as well as the MTP-TEC computation performed by SIATEC in terms of precision and recall. Using SIATEC-C as the TEC algorithm for the compression algorithms COSIATEC and SIATECCompress did not improve their precision or recall. A different approach to filtering and refining the patterns produced by SIATEC-C is thus needed.

The version of SIATEC-C presented in this paper uses the size of patterns as a means of prefiltering. The cover array approach can also be used with other measures that can be computed for a point-set pattern, such as compactness [11, 22]. Evaluating the musical importance of a pattern is a challenging problem. As SIATEC-C also finds all occurrences of the patterns it discovers, the algorithm can be extended with various pattern filtering methods.

7. ACKNOWLEDGEMENTS

This paper has benefited from discussions with Kjell Lemström and Antti Laaksonen, and from email correspondence with Tom Collins. We also thank the anonymous reviewers for their insightful comments on the paper. This work was supported by the Eemil Aaltonen Foundation grant (grant number 220014K).

8. REFERENCES

[1] B. Janssen, W. B. de Haas, A. Volk, and P. van Kranenburg, "Finding repeated patterns in music: State of knowledge, challenges, perspectives," in *Sound, Music, and Motion. CMMR 2013. Lecture Notes in Computer Science, vol 8905*, M. Aramaki, O. Derrien, R. Kronland-Martinet, and S. Ystad, Eds. Springer, Cham, 2013, pp. 277–297.

[2] I. D. Bent and A. Pople, "Analysis," in *Grove Music Online*. Oxford University Press, 2001, accessed: 5 May 2022. [Online]. Available: https://www.oxfordmusiconline.com/grovemusic/view/10.1093/gmo/9781561592630.001.0001/omo-9781561592630-e-0000041862

[3] O. Lartillot and P. Toiviainen, "Motivic matching strategies for automated pattern extraction," *Musicae Scientiae, Discussion Forum 4A*, pp. 281–314, 2007.

[4] O. Lartillot, "Automated motivic analysis: An exhaustive approach based on closed and cyclic pattern mining in multidimensional parametric spaces," in *Computational Music Analysis*, D. Meredith, Ed. Springer International Publishing, 2016, pp. 273–302.

[5] D. Meredith, "Using point-set compression to classify folk songs," in *The Fourth International Workshop on Folk Music Analysis (FMA 2014)*, Istanbul, Turkey, 2014.

[6] P. Allegraud, L. Bigo, L. Feisthauser, M. Giraud, R. Grould, E. Leguy, and F. Levé, "Learning sonata form structure on mozart's string quartets," *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, pp. 82–96, 2019.

[7] T. Collins, A. Arzt, H. Frostel, and G. Widmer, "Using geometric symbolic fingerprinting to discover distinctive patterns in polyphonic music corpora," in *Computational Music Analysis*, D. Meredith, Ed. Springer International Publishing, 2016, pp. 445–474.

[8] C. Louboutin and D. Meredith, "Using general-purpose compression algorithms for music analysis," *Journal of New Music Research*, vol. 45, no. 1, pp. 1–16, 2016.

[9] D. Meredith, "Analysis by compression: Automatic generation of compact geometric encodings of musical objects," in *The Music Encoding Conference*, Mainz, Germany, 2013.

[10] D. Conklin, "Mining contour sequences for significant closed patterns," *Journal of Mathematics and Music*, vol. 15, no. 2, pp. 112–124, 2021. [Online]. Available: https://doi.org/10.1080/17459737.2021.1903591

[11] D. Meredith, K. Lemström, and G. A. Wiggins, "Algorithms for discovering repeated patterns in multidimensional representations of polyphonic music," *Journal of New Music Research*, vol. 31, no. 4, pp. 321–345, 2002.

[12] K. Lemström, *String Matching Techniques for Music Retrieval*. PhD thesis, University of Helsinki, 2000.

[13] D. Conklin and C. Anagnostopoulou, "Representation and discovery of multiple viewpoint patterns," in *Proceedings of the International Computer Music Conference (ICMC 2001)*, La Habana, Cuba, 2001.

[14] G. Velarde, D. Meredith, and T. Weyde, "A wavelet-based approach to pattern discovery in melodies," in *Computational Music Analysis*, D. Meredith, Ed. Springer International Publishing, 2016, pp. 303–333.

[15] D. Humphreys, K. Sidorov, A. Jones, and D. Marshall, "An investigation of music analysis by the application of grammar-based compressors," *Journal of New Music Research*, vol. 50, no. 4, pp. 312–341, 2021. [Online]. Available: https://doi.org/10.1080/09298215.2021.1978505

[16] E. Ukkonen, K. Lemström, and V. Mäkinen, "Geometric algorithms for transposition invariant content-based music retrieval," in *Proceedings of the 4th International Conference on Music Information Retrieval (ISMIR 2003)*, Baltimore, Maryland, USA, 2003.

[17] D. Meredith, "Analysing music with point-set compression algorithms," in *Computational Music Analysis*, D. Meredith, Ed. Springer International Publishing, 2016, pp. 335–366.

[18] A. Laaksonen and K. Lemström, "On the memory usage of the SIA algorithm family for symbolic music pattern discovery," in *Eighth International Conference on Mathematics and Computation in Music*, 2022, in press.

[19] T. Collins, *Improved methods for pattern discovery in music, with applications in automated stylistic composition*. PhD thesis, The Open University, 2011.

[20] A. Laaksonen and K. Lemström, "Transposition and time-warp invariant algorithm for detecting repeated patterns in polyphonic music," in *6th International Conference on Digital Libraries for Musicology*, 2019.

[21] A. Laaksonen, K. Lemström, and O. Björklund, "Transposition and time-scaling invariant algorithm for detecting repeated patterns in polyphonic music," in *Eighth International Conference on Mathematics and Computation in Music*, 2022, in press.

[22] D. Meredith, "COSIATEC and SIATECCompress: Pattern discovery by geometric compression," in *MIREX 2013. Competition on Discovery of Repeated Themes and Sections*, Curitiba, Brazil, 2013.

[23] J. C. Forth, *Cognitively-motivated geometric methods of pattern discovery and models of similarity in music*. PhD thesis, Department of Computing, Goldsmiths, University of London, 2012.

[24] D. Meredith, "RecurSIA-RRT: Recursive translatable point-set pattern discovery with removal of redundant translators," in *Machine Learning and Knowledge Discovery in Databases*, P. Cellier and K. Driessens, Eds. Cham: Springer International Publishing, 2019, pp. 485–493.

[25] T. Collins and D. Meredith, "Maximal translational equivalence classes of musical patterns in point-set representations," in *Mathematics and Computation in Music: 4th International Conference, MCM 2013, Proceedings. Lecture Notes in Computer Science, Vol. 7937*. Springer, Berlin, 2013, pp. 88–99.

[26] T. Collins, J. Thurlow, R. Laney, A. Willis, and P. H. Garthwaite, "A comparative evaluation of algorithms for discovering translational patterns in baroque keyboard works," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010)*, Utrecht, Netherlands, 2010.

[27] A. Arzt, S. Böck, and G. Widmer, "Fast identification of piece and score position via symbolic fingerprinting," in *Proceedings of the 13th International Conference on Music Information Retrieval (ISMIR 2012)*, Porto, Portugal, 2012.

[28] T. Collins, A. Arzt, S. Flossmann, and G. Widmer, "SIARCT-CFP: Improving precision and the discovery of inexact musical patterns in point-set representations," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, Curitiba, Brazil, 2013.

[29] G. A. Wiggins, "Models of musical similarity," in *Musicae Scientiae, Discussion Forum 4A*, 2007, pp. 315–338.

[30] O. Björklund, *Improving the running time of repeated pattern discovery in multidimensional representations of music*. Master's Thesis, University of Helsinki, 2018. [Online]. Available: https://ethesis.helsinki.fi/repository/handle/123456789/21114

[31] T. Collins, "Mirex 2013 competition: Discovery of repeated themes and sections," 2013, accessed: 12 April 2022. [Online]. Available: https://www.music-ir.org/mirex/wiki/2013:Discovery_of_Repeated_Themes_\%26_Sections

TAILED U-NET: MULTI-SCALE MUSIC REPRESENTATION LEARNING

Marcel A. Vélez Vásquez **John Ashley Burgoyne**
Music Cognition Group · Institute for Logic, Language, and Computation · University of Amsterdam
marcel.velezv@gmail.com j.a.burgoyne@uva.nl

ABSTRACT

Self-supervised learning has steadily been gaining traction in recent years. In music information retrieval (MIR), one promising recent application of self-supervised learning is the CLMR framework (contrastive learning of musical representations). CLMR has shown good performance, achieving results on par with state-of-the-art end-to-end classification models, but it is strictly an encoding framework. It suffers the characteristic limitation of any encoder that it cannot explicitly combine multi-timescale information, whereas a characteristic feature of human audio perception is that we tend to perceive all frequencies simultaneously. To this end, we propose a generalization of CLMR that learns to extract and explicitly combine representations across different frequency resolutions, which we coin the tailed U-Net (TUNe). TUNe architectures combine multi-timescale information during a decoding phase, similar to U-Net architectures used in computer vision and source separation, but have a tail added to reduce sample-level information to a smaller pre-defined number of representation dimensions. The size of the decoding phase is a hyperparameter, and in the case of a zero-layer decoding phase, TUNe reduces to CLMR. The best TUNe architectures, however, require less training time to match CLMR performance, have superior transfer learning performance, and are competitive with state-of-the-art models even at dramatically reduced dimensionalities.

1. INTRODUCTION

Representation learning is a fast-moving sub-field of machine learning that seeks to distill information encoded in different types of input signals into less noisy abstract representations that are suitable for various downstream tasks. Such representations, which are learned without explicit supervision, have been successfully applied to a broad variety of musical and non-musical task domains [1], including audio tagging [2, 3] and speech recognition [4]. After self-supervised training, the learned representations are then evaluated by *probing*, a term originating from natural language processing [5–7]. In MIR, probing is also known as shallow network training or transfer learning [2, 8–13]. When probed, these self-supervised learning methods perform comparably to end-to-end trained models [3, 14, 15].

One of the more common learning architectures used for MIR tasks is the convolutional neural network (CNN) [16–19]. CNNs typically consist of either only an encoder path or an encoder and decoder path. One distinctive CNN with both an encoder *and* a decoder path model is the U-Net, consisting of variants of the encoder and decoder paths called contractive and expansive paths. Our work focuses specifically on adapting the U-Net architecture [20] to representation learning. To the best of our knowledge, there is little to no published research in computer vision, MIR, or signal processing that has considered the potential of U-Nets for representation learning.

U-Nets originated from the field of biomedical image segmentation, where they were introduced with the goal of being more data-efficient and less time-consuming to train for segmentation tasks, while also being able to perform well with relatively few data points and in the presence of class imbalance [20, 21]. U-Net architectures have shown top performance in segmentation, winning the ISBI cell tracking contest by a large margin in 2015 [20]. One of the arguments for how well U-Net architectures perform is the way the contractive and expansive paths allow the network to incorporate features across multiple resolutions.

U-Nets have also been shown to perform well within the audio domain. Their application to source separation in the time-frequency domain yielded state-of-the-art results [22], after which the U-Net established itself for source separation in the raw audio domain as well [23]. In raw audio and speech generation, U-Nets perform on par with Wavenet [24], with fewer parameters and faster inference [25]. We are also motivated to explore U-Net-like architectures for representation learning because of their intuitive relation to the *slow feature hypothesis* [26], which states that much of the meaningful information contained in signals changes gradually, over larger timescales. Without requiring a formal transformation to the frequency domain, U-Net architectures can extract these slow features alongside fine-grained high-frequency information by encoding and combining features across multiple timescales [25].

Given the scarcity of publicly available labelled data in MIR, we focus further on U-Nets for *self-supervised* learning. Specifically, we adapt the contrastive semi-supervised learning method from CLMR [3], since this has shown great promise on label training efficiency and generalisability of learned representations to out-of-domain MIR datasets. Specifically, we generalise the SampleCNN [27] en-

© M.A. Vélez Vásquez and J.A. Burgoyne. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M.A. Vélez Vásquez and J.A. Burgoyne, "Tailed U-Net: Multi-Scale Music Representation Learning", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

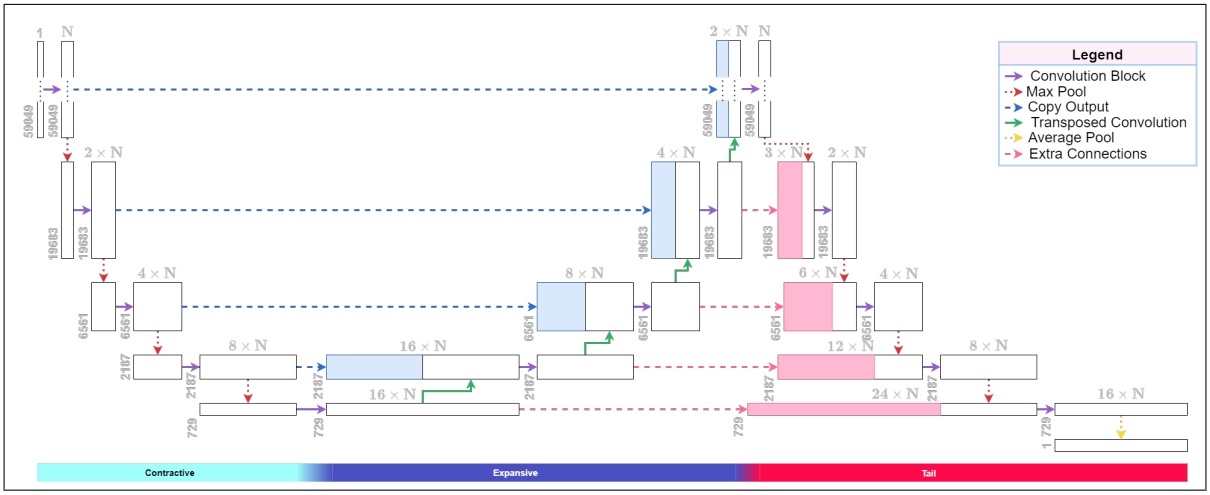


Figure 1: The TUNe architecture extracts features containing information obtained at multiple timescales. To this end, it consists of three main components: (1) the *contractive path* iteratively extracts features at a given timescales and reduces the signal resolution; (2) the *expansive path* upsamples the extracted features at lower resolutions and concatenates them into higher-resolution features than those obtained from the contractive path; and finally, (3) the *tail* combines the features extracted at multiple timescales and reduces their spatial resolution, ultimately yielding a single, low-dimensional, multi-timescale representation for an input signal.

coder architecture used in CLMR with a new architecture we dub the *tailed U-Net* (TUNe). TUNe architectures are like U-Net architectures, but with an additional contractive path – the tail – extending the original architecture by a mapping from a representation at the input resolution to a reduced latent representation size (see Figure 1). Intuitively, the 'U' shape of the network extracts a representation at the original temporal resolution, encoding a combination of slow feature patterns with higher-frequency components from the input signal, whereas the tail learns temporally reduced patterns from this enriched signal.

Our main contribution is this novel angle for the use of U-Nets for representation learning on signal data. Furthermore, we investigate a number of architectural setups with different model sizes to assess the performance and parameter efficiency of TUNe networks. In order to evaluate the learned representations, we transfer them to a range of benchmark MIR tasks, showing competitive performance with a drastically shortened training regime, parameter count, and representation dimensionality.

2. RELATED WORK

State-of-the art algorithms for audio-based MIR tasks (e.g., chord recognition, key detection, and music audio tagging) are generally built on one of three input forms: (1) time-frequency representations of the audio signal, (2) raw time-domain audio, or (3) a combination of raw audio and time-frequency representations [28]. Among the top-performing raw audio input architectures are musicCNN [2], JukeBox [29], and SampleCNN [27]. SampleCNN was introduced for raw audio classification and later adapted for CLMR's contrastive learning setting. We use CLMR's version of SampleCNN as a reference point for both performance and number of parameters. Moreover, the TUNe convolution blocks introduced in Section 3 are based on the filter–stride–max pooling operation used in SampleCNN, which downshifts the effective frequency range modelled by the convolution kernels applied in subsequent layers. Note that because of this use of pooling operations, much high-frequency information is discarded in the extraction of the encoding from the input. Instead, our proposed approach explicitly combines these subsampled lower-frequency features with high-frequency features obtained at the original temporal resolution.

For a range of benchmark MIR datasets, CLMR is currently the best-performing self-supervised model that does not require industry-scale hardware to train [14]. It is a self-supervised learning framework, and as such, it requires no labelled data for training. Specifically, it is a contrastive learning framework: given two similar inputs to the network (e.g., two segments from the same song), the loss function is designed to ensure that the learned representations of these samples should lie closer to each other than the representations of samples from two different songs. CLMR also includes music-specific data augmentations to ensure a robust representation learning framework for MIR, for example, transposing the pitch of a segments up or down and still instructing the network to predict a closer distance to the original fragment than to the other audio segments in the batch. Because mapping the representations with a projector head instead of directly using the network representation results in a better representations [30], the output layer is replaced with an identity function and a projector head is added to the models trained with CLMR.

3. TAILED U-NET ARCHITECTURES

The architecture for representation learning we explore in this work is based on the traditional U-Net [20]. The U-

Net's success on segmentation tasks has been argued to come from its explicit extraction of features at multiple resolutions, which allows it to respond to fine-grained features as well as long-range dependencies in the input. In representation learning for audio, we are interested in constructing encodings which similarly carry both high- and low-frequency information. Because U-Nets were originally designed for image segmentation, however, variants of the architecture traditionally produce an output at the spatial resolution of the input: $\mathbb{R}^{I \times C}$ where I is the input length and C the number of classes given by the task. Representation learning models, on the other hand, aim to decouple the dimensionality of the representation from the temporal resolution of the original input signal, to accommodate usage in a wide variety of downstream tasks. They typically output a single representation: $\mathbb{R}^{R}$ with R the size of the representation. To address this mismatch, we propose adding a tail module to the U-Net. The tail module serves (1) to combine enriched sequence length representation obtained at multiple resolutions in the contractive and expansive path, and (2) to reduce the resolution of these features to a single compact multi-timescale representation. See Figure 1 for an overview of our architecture.

In addition to the tail module, we introduce another improvement on the TUNe architectures: TUNe+. TUNe+ networks have additional connections between the expansive and tail path (the pink arrows and blocks in Figure 1). This explicit transfer of information at lower resolutions should allow for better multi-timescale feature information flow, further improving the obtained representations.

3.1 Contractive Path

The contractive path of the network is built from a block consisting of a convolution, a batch normalisation layer, and a ReLU activation function, following CLMR's encoding architecture [3]. This block is repeated N_{con} times (default 4). In Figure 1, this block is indicated with purple arrows. The output of this block is saved to combine with the corresponding expansive block later. After each block, max pooling is applied to extract the most prominent patterns at a given resolution, represented by the red arrows in Figure 1. The output of the max pooling is then passed on either to the next layer in the contractive path or, after the N_{con}-th block, to the beginning of the expansive path. Every layer in the encoder path doubles the amount of channels. The convolution operation is performed with a kernel size of 3 and a stride of 1, and the max pooling operation with a kernel size of 3 and stride of 3. Furthermore, the convolution operation is applied on a padded input, such that input and output to the convolution are of equal sequence length.

3.2 Expansive Path

Whereas each layer of the contractive path reduces the signal resolution, the expansive path of the network is made up of blocks that are paired with blocks from the contractive path, each one gradually *increasing* the signal resolution again. The expansive path block, which is repeated N_{exp} times (default 4), consists of a strided transposed convolution, represented by the green arrows in Figure 1, with a kernel size of 3, stride of 3, and 0 padding, to keep the dimensionality the same as the output of the contractive block at the same depth. The output of each strided transposed convolution is concatenated with the output of its pair from the contractive path (see the blue arrows leading into the blue-and-white blocks in Figure 1). Next, a convolution followed by batch normalisation and ReLU activation is used to combine the multi-scale features in the concatenated outputs. The upsampling by strided transposed convolution enables the network to combine lower frequency features with the higher frequency information obtained at higher resolutions in the contractive path. The input–output channel ratio for the strided transposed convolution is 2:1, and for the convolution block is also 2:1, because of the concatenation of the encoder block and strided transposed convolution output.

3.3 Tail

The blocks in the tail of a TUNe model are identical to the blocks of the contractive path, each of the N_{tai} blocks (default 4) doubling the number of channels until the representation size is reached. In a TUNe+ model, the tail blocks differ from the contractive path blocks because TUNe+ tail blocks are paired with blocks from the expansive path. As illustrated by the pink arrows in Figure 1, each TUNe+ tail block combines the output of the previous tail block with the output of the expansive block at the same resolution. The input–output channel ratio in this case is 3:2, because the input gets ⅔ of the input channels from the expansive path and ⅓ from the previous tail block. Depending on the number of downsampling steps in the tail module, any remaining temporal dimensions O are projected to a single representation using average pooling, mapping from $\mathbb{R}^{O \times R}$ to $\mathbb{R}^{R}$ with R the representation size. This operation is indicated by the yellow arrow in Figure 1.

The TUNe network architecture allows for flexible network adjustments. When removing all tail layers the resulting TUNe is equivalent to the original U-Net. If all expansive layers are removed there is no way to distinguish contractive and tail layers, and adding six contractive or tail layers is equivalent to CLMR. Since the output does not have to end with the same dimensionality as the input, one can add and remove contractive, expansive, or tail layers without needing to add or remove layers along the other paths. We leverage this compositionality to conduct experiments with the TUNe architecture that explore representations built up over a differing number of timescales, in order to explore the impact of the structure of each respective module on the capacity of the resulting representation to perform in downstream tasks.

4. EXPERIMENTS

To explore the validity and performance of our proposed setup, we trained multiple TUNe architectures with varying path lengths for 1000 epochs on the MagnaTagATune

Variant	N_{con}	N_{exp}	N_{tai}	Filters	Parameters (M)	MTT_{AUC}	MTT_{AP}
Vanilla TUNe	4	4	4	34	2.4	87.7	33.0
TUNe Contractive+1	5	5	4	18	2.3	88.3	33.9
TUNe Contractive+2	6	6	4	9	2.1	88.6	34.6
TUNe Contractive+3	7	7	4	4	1.7	88.0	33.5
TUNe Expansive-1	4	3	3	34	2.3	87.6	33.0
TUNe Expansive-2	4	2	2	35	2.4	87.7	33.1
TUNe Expansive-3	4	1	1	38	2.3	87.6	33.0
TUNe Tail+1	4	4	5	28	2.3	88.2	34.4
TUNe Tail+2	4	4	6	19	2.3	88.7	35.2
TUNe Tail+3	4	4	7	15	2.3	89.1	36.5
TUNe Tail+4	4	4	8	13	2.3	89.2	36.6
TUNe Tail+5	4	4	9	11	2.1	89.2	36.5
TUNe CLMR-tail	4	4	10*	10*	2.5	89.4	36.7
TUNe+	4	4	9	11	2.2	89.2	36.6
Vanilla TUNe Small	4	4	4	11	0.4	86.8	31.9
TUNe+ Large	4	4	9	34	7.4	89.4	37.1
TUNe+ Smaller Rep	4	4	9	11	1.4	89.2	36.1
musicnn 10 000 [2]	-	-	-	–	11.8**	90.7	38.4
TUNe Tail+5 10 000	**4**	**4**	**9**	**11**	**2.1**	**89.5** (**89.6**)	**37.0** (**36.7**)
TUNe+ 10 000	**4**	**4**	**9**	**11**	**2.2**	**89.3** (**89.8**)	**37.1** (**37.1**)
CLMR 10 000 [3]	10	0	0	–	2.4	88.7 (89.3)	35.6 (36.0)

Table 1: TUNe variant performance on the MagnaTagATune (MTT) tag prediction task with number of parameters and number of initial filters. The performance is measured after 1000 epochs (except where noted otherwise) with the area under the receiver operating characteristic curve MTT_{AUC}, and the average precision, MTT_{AP}. The table is divided into six sections: the Vanilla TUNe model with no layers added; the results of adding contractive layers; the results of removing expansive layers; the results of adding tail layers; the results parameter-efficiency experiments; and the results of the best TUNe models and the CLMR baseline after training for 10 000 epochs. In addition to the shallow probe, we trained a probe with an extra linear layer for the longer trained models and report this score in parentheses. TUNe Tail+5 and TUNe+ at 10 000 epochs are the best-performing models overall, exceeding CLMR's performance at 10 000 epochs and performing only slightly worse than state-of-the-art end-to-end–trained musicnn. Multiple other variants match CLMR performance even though only trained for 1 000 epochs.

* For the CLMR-tail experiment, the number of filters corresponds to the initial number of filters used for the contractive and expansive path. For the tail, we used the same architecture as CLMR's SampleCNN and report the number of blocks.

** Number of parameters is taken from the last reported number of parameters, musicnn [31].

audio dataset. As baseline, we compare to CLMR [3]. We ensure a fair comparison by varying the number of channels in each TUNe architecture to obtain a total parameter count comparable to the CLMR baseline. Next, we trained five variants: (1) a version where the tail was fixed to the published pre-trained CLMR model, in order to test whether the contractive and expansive paths (the 'U') add information; (2) a TUNe+ network, still restricted to have no more parameters than CLMR, to test whether the extra connections between the expansive path and the tail allow for better feature information flow; (3) a filter-restricted model, to test whether the number of filters can be a bottleneck for the deeper models; (4) a large TUNe+ network with an unrestricted number of parameters; and (5) a model with a smaller representation dimension. Finally, we evaluated the out-of-domain dataset generalisability of our two best models on three different datasets for the same probing task. [1]

[1] Pretrained models and source code for all experiments are available to download at `https://github.com/Marcel-Velez/TUNe`

4.1 Hyperparameters and Preprocessing

The hyperparameters we used for pre-training were the same as [3], following their setup with data augmentations. We used an Adam optimiser [32], and He initialisation [33] for all convolutional layers. As input, we sampled audio files at 22 050 Hz for 59 049 samples. We also used the same architecture for the projector head as [3]. This output is then used for the contrastive learning objective. For every experiment, we used a batch size of 96. In order to be able to train with such a batch size, we trained every model data-parallel (DP) on two Titan RTXs, except for Vanilla TUNe and TUNe+ Large. To train these variants with a batch size of 96, we used three Titan RTXs.

4.2 Exploring the TUNe Architecture

Because the number of permutations of how many contractive layers are added, expansive layers are subtracted, tail layers are added grows exponentially, we chose to investigate the influence of each of these paths separately. Every model was trained on the audio of the MagnaTagA-

Tune (MTT) dataset [34]. This dataset consists of 25 863 music clips of 29 seconds of audio from 5223 songs. Each of these clips has one or more tags, making it a multi-class classification task dataset. We use the same train-validation-test split as is common within MIR [35–37]. After training, each model was probed on the MTT tagging task using a single linear-layer probe and evaluated using two metrics. The first metric is the receiver operating characteristic curve (MTT_{AUC}), which is popular but can be positively biased for imbalanced datasets [38], and the average precision (MTT_{AP}). We report results for each model in Table 1.

The only constraint for these variations was to have fewer parameters than CLMR, so as to exclude the model being bigger being a possible reason for performing better. The TUNe architecture follows a fixed input channel–output channel ratio per layer, and thus in order to keep the number of parameters smaller than CLMR, we only had to change the number of output channels of the first block. The remainder of the network changes in proportion. A complete overview of the number of parameters and initial output filters can be found in Table 1.

4.2.1 Contractive Path Depth

When varying the contractive path depth, we also added an equal number of expansive layers, in order to keep the upper resolution of the U and the tail the same; the tail remained unchanged so that the final representation size remained unchanged. Starting from the vanilla default of 4 contractive layers, adding layers increased performance up until $N_{\text{con}} = 6$, suggesting that the extra contractive and expansive layers – a deeper U – do allow for better integration of feature information for the representation dimensionality. Contractive+3 ($N_{\text{con}} = 7$) drops slightly in performance, which initially seems contradictory to the Contractive+1 and +2 results. We believe this can be attributed to the potentially exponential parameter growth of adding layers to the contractive and thus also expansive paths: to prevent the exponential growth and remain within our constraint on the maximum number of parameters, adding layers entails reducing the number of filters. TUNe Contractive+3, for example, has only 4 initial filters, as compared to 34 in Vanilla TUNe, and this may no longer be enough to achieve good performance.

To test this hypothesis, we ran Vanilla TUNe with the number of filters from TUNe+ (labelled Vanilla TUNe Small in Table 1), and conversely ran TUNe+ with the number of filters from Vanilla TUNe (labelled TUNe+ Large in Table 1). Vanilla TUNe performance dropped and TUNe+ Large performance increased under these conditions, suggesting that the number filters could indeed be the bottleneck for Contractive+3.

4.2.2 Expansive Path Height

In order to analyze the effect of expansive path height, we applied a similar procedure. For every expansive layer we removed, we also removed a tail layer to keep the final representation dimensionality unchanged. Because this modification reduced dimensionality, we were able to *add* initial filters in order to come closer to the dimensionality of CLMR. Nonetheless, removing expansive layers seems to have little influence: removing up to three layers of the expansive path and tail leaves performance essentially unchanged. Put differently, the extra initial filters seem to be able to compensate for reduced integration of the higher frequency timescale features due to a shallower U.

To see whether the U shape in fact adds information and increases performance, we trained a TUNe model with a tail path identical to pretrained CLMR. If the U shape does not add information, such a model should perform equally well or worse than baseline CLMR; if it performs better, then there is evidence that the contractive and expansive paths are contributing signal enrichment important for representation learning. Indeed, TUNe with a CLMR tail performs better with a single layer probe than baseline CLMR can even after 10 times as many training epochs and a multi-layer probe. It seems that the contractive–expansive path pair is a powerful performance enhancer for time-domain music representation learning.

4.2.3 Tail Length

The model performance after adding 1 to 5 layers to the tail path shows a steady increase per layer added until Tail+4, after which performance seems to plateau. In general, we should expect longer tails to improve performance, because the model average pools the remaining sequence length at the end of the tail path. With fewer tail layers, we average over a longer sequence, and when averaging, detailed information is replaced with an aggregation, thereby losing possible important information. The parameter constraint could again be responsible for the eventual plateau, as we see that the TUNe+ Large model from before also performs better than either Tail+4 or Tail+5. But it is a plateau, not a decrease: Tail+5 performs equivalently to Tail+4, and it does so with fewer parameters and potentially less loss of precision from averaging than Tail+4. We choose Tail+5 as our best model from this block.

4.2.4 Extra Connections

The model with extra connections, TUNe+, performs comparably to the Tail+5 model, with only .1 higher MTT_{AP}. It still shows similar MTT_{AUC} scores as CLMR 10 000 epochs trained and outperforms CLMR MTT_{AP}-wise. To further explore the influence of the added connections we chose TUNe+ to be one of the two variants used for the probing experiments (Section 4.3).

4.2.5 Smaller Representations

All of these variants showed reasonable performance for the same representation size, with only slightly varying numbers of parameters. To test the parameter efficiency of TUNe architectures, we trained another variant of the TUNe+ with the same number of initial filters, but reducing the representation size from 512 to 256. The results were impressive. Even with 44% fewer parameters and a representation size half that of the other TUNe architectures we tested, TUNe+ Smaller Rep still performs equally

well as CLMR after 10 000 epochs. Compared to the best TUNe architectures, it achieves comparable MTT_{AUC} and only marginally worse MTT_{AP}.

4.2.6 Longer Training

In order to be conservative with computing resources, we first trained all of the aforementioned models for 1000 epochs only and probed with only a single linear layer. As a final comparison, we trained the two best models, TUNe Tail+5 and TUNe+, an additional 9 000 epochs. Because of the random data augmentations and the size of MTT, training for more epochs results in the models 'hearing' an increasing amount of natural variation in the audio, which in turn improves performance. We evaluated these models using the same probing tasks, once with a single linear layer and once with a two-layer probe to see how much introducing nonlinearity could increase performance. In Table 1, we report these additional multi-layer performance figures in brackets. The MTT_{AUC} score does increase with the multilayer probe, but the MTT_{AP}, on the contrary, decreases. Overall, however, these two 10 000-epoch models are the best performing models from our entire series of experiments, achieving slightly lower performance than the musically motivated end-to-end trained musicnn [2] and outperforming CLMR 10 000 epoch results.

4.3 Probing Tasks

In order to compare the TUNe performance to state-of-the-art models, we evaluate TUNe with probing: training a shallow model on downstream tasks [10]. We use the same datasets as [3] for the CLMR training.

When probing a model, we evaluate the pre-trained model representations on a different dataset than the model is pre-trained on. This evaluation on a different dataset is done by training a probe, often a single linear layer or a multi-layer perceptron with one hidden layer. This probe takes the output representation of the main model as input and outputs the desired classes or values for the task in question. Probing is often used to test certain representation characteristics, in our case, to see how well the learned representations from other types of datasets generalise to music tagging.

We ran the probing experiment to see how well our network could generalise representations when trained on three different datasets: the medium Free Music Archive dataset [39], the fault-filtered GTZAN dataset [40,41] containing 930 songs, and the McGill Billboard dataset [42] containing 712 songs. Next, we probed the trained models on the MTT dataset, of which the results are displayed in Table 2. For this probing experiment, we used a learning rate of 3e^{-4}, weight decay of 10^{-6}, and an early stopping mechanism. Early stopping occurred if the probe's validation score did not improve for five epochs.

TUNe architectures allow for excellent out-of-dataset representation generalisability. Even when both TUNe Tail+5 and TUNe+ were trained on the McGill Billboard dataset, which is more than 33 times smaller than MTT, the models still only perform 4% worse on AUC than the models pre-trained on MTT. When they are trained on the GTZAN dataset, which is about the same size as McGill Billboard, both variants outperform CLMR regardless of (non-MTT) training set. With pre-training on the MTT-sized FMA dataset, the TUNe models perform almost as well as they did in the original MTT-only experiments.

Probing Variant	Training Data	MTT_{AUC}	MTT_{AP}
TUNe+	FMA	89.1 (89.4)	36.2 (36.1)
TUNe Tail +5	FMA	88.9 (89.2)	35.3 (35.9)
CLMR	FMA	86.2 (86.6)	30.6 (31.2)
TUNe+	GTZAN	87.2 (87.9)	32.6 (33.9)
TUNe Tail +5	GTZAN	86.9 (87.5)	32.6 (33.0)
CLMR	GTZAN	81.9 (85.4)	26.2 (29.5)
TUNe Tail +5	Billboard	84.7 (85.8)	28.6 (29.9)
TUNe+	Billboard	84.5 (85.9)	28.7 (30.5)
CLMR	Billboard	82.7 (84.2)	26.9 (27.8)

Table 2: TUNe Tail +5, TUNe+ and CLMR out-of-domain probing experiments. The table shows the probing performance on MagnaTagATune of each of the three models, trained on the Free Music Archive medium (FMA), fault-filtered GTZAN, and the McGill Billboard dataset. In addition to the shallow probe, we trained a probe with an extra linear layer and report this score in parentheses. CLMR results are taken from [3].

5. CONCLUSION

In this paper, we introduced TUNe network architectures, a generalisation of a recent representation learning framework called CLMR. TUNe brings the strengths of U-Nets to representation learning. TUNe networks comprise three sections, called the contractive, expansive, and tail paths, which can be flexibly lengthened or shortened. We performed several experiments exploring the contribution of each of these paths and compared them against CLMR. We evaluated TUNe's performance in three ways. First, we trained and evaluated variants of our model with CLMR on MagnaTagATune (MTT), outperforming CLMR marginally after training for a fraction of CLMR's time. Second, we evaluated the best two models with out-of-domain probing tasks. Both TUNe architectures improved upon the already competitive CLMR performance and showed that TUNe architectures allow for an even better generalisation of music representations. In the supplemental material, we include the results of further experiments showing how TUNe architectures can also achieve competitive performance on other downstream tasks, even at small model sizes. TUNe sets a new standard for parameter efficiency and the ability of modern self-supervised networks to extract salient features, and we hope that it will encourage MIR researchers to use self-supervised music representations more widely.

6. REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE*

transactions on pattern analysis and machine intelligence, vol. 35, no. 8, pp. 1798–1828, 2013.

[2] J. Pons and X. Serra, "musicnn: Pre-trained convolutional neural networks for music audio tagging," 2019.

[3] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, 2021.

[4] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 449–12 460, 2020.

[5] D. Hupkes, S. Veldhoen, and W. Zuidema, "Visualisation and 'diagnostic classifiers' reveal how recurrent and recursive neural networks process hierarchical structure," *Journal of Artificial Intelligence Research*, vol. 61, pp. 907–926, 2018.

[6] A. Conneau, G. Kruszewski, G. Lample, L. Barrault, and M. Baroni, "What you can cram into a single vector: Probing sentence embeddings for linguistic properties," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, vol. 1, 2018, pp. 2126–2136. [Online]. Available: https://aclanthology.org/P18-1198

[7] J. Hewitt and C. D. Manning, "A structural probe for finding syntax in word representations," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, 2019, pp. 4129–4138.

[8] A. Van Den Oord, S. Dieleman, and B. Schrauwen, "Transfer learning by supervised pre-training for audio-based music classification," in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, 2014.

[9] P. Hamel, M. E. Davies, K. Yoshii, and M. Goto, "Transfer learning in MIR: Sharing learned latent representations for music audio classification and similarity," in *Proceedings of the 14th International Society for Music Information Retrieval Conference*, 2013.

[10] K. Choi, G. Fazekas, M. B. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017.

[11] J. Lee, J. Park, and J. Nam, "Representation learning of music using artist, album, and track information," Paper presented at the Machine Learning for Music Discovery Workshop, International Conference on Machine Learning, 2019.

[12] Q. Huang, A. Jansen, L. Zhang, D. P. Ellis, R. A. Saurous, and J. Anderson, "Large-scale weakly-supervised content embeddings for music recommendation and tagging," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 8364–8368.

[13] J. Kim, J. Urbano, C. Liem, and A. Hanjalic, "One deep music representation to rule them all? A comparative analysis of different representation learning strategies," *Neural Computing and Applications*, vol. 32, no. 4, pp. 1067–1093, 2020.

[14] R. Castellon, C. Donahue, and P. Liang, "Codified audio language modeling learns useful representations for music information retrieval," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, 2021.

[15] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," arXiv:2111.06377, 2021.

[16] B. Zhang, J. Leitner, and S. Thornton, "Audio recognition using mel spectrograms and convolution neural networks," Online report, Noiselab, University of California, 2019.

[17] J. S. Gómez, J. Abeßer, and E. Cano, "Jazz solo instrument classification with convolutional neural networks, source separation, and transfer learning," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018, pp. 577–584.

[18] M. A. Rohit, A. Bhattacharjee, and P. Rao, "Four-way classification of tabla strokes with models adapted from automatic drum transcription," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, 2021.

[19] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[21] S. Minaee, Y. Y. Boykov, F. Porikli, A. J. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[22] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep U-Net convolutional networks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017, pp. 23–27.

[23] D. Stoller, S. Ewert, and S. Dixon, "Wave-U-Net: A multi-scale neural network for end-to-end audio source separation," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[24] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," Paper presented at the Speech Synthesis Workshop, International Speech Communication Association, 2016.

[25] D. Stoller, M. Tian, S. Ewert, and S. Dixon, "Seq-U-Net: A one-dimensional causal U-Net for efficient sequence modelling," in *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, 2020, pp. 2893–2900.

[26] L. Wiskott and T. J. Sejnowski, "Slow feature analysis: Unsupervised learning of invariances," *Neural Computation*, vol. 14, no. 4, pp. 715–770, 2002.

[27] J. Lee, J. Park, K. L. Kim, and J. Nam, "SampleCNN: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, no. 1, p. 150, 2018.

[28] M. Lederle and B. Wilhelm, "Combining high-level features of raw audio waves and mel-spectrograms for audio tagging," Submission to the Detection and Classification of Acoustic Scenes and Events challenge, 2018.

[29] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," arXiv:2005.00341, 2020.

[30] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning*, 2020, pp. 1597–1607.

[31] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[32] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2014.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[34] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *Proceedings of the 10th International Society for Music Information Retrieval Conference*, 2009, pp. 387–392.

[35] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018.

[36] S. Dieleman and B. Schrauwen, "Multiscale approaches to music audio feature learning," in *Proceedings of the 14th International Society for Music Information Retrieval Conference*, 2013, pp. 116–121.

[37] M. Tagliasacchi, B. Gfeller, F. de Chaumont Quitry, and D. Roblek, "Pre-training audio representations with self-supervision," *IEEE Signal Processing Letters*, vol. 27, pp. 600–604, 2020.

[38] J. Davis and M. Goadrich, "The relationship between precision–recall and ROC curves," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 233–240.

[39] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017.

[40] B. L. Sturm, "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use," arXiv:1306.1461, 2013.

[41] C. Kereliuk, B. L. Sturm, and J. Larsen, "Deep learning and music adversaries," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2059–2071, 2015.

[42] J. A. Burgoyne, J. Wild, and I. Fujinaga, "An expert ground truth set for audio chord recognition and music analysis," in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, 2011, pp. 633–638.

[43] J. Spijkervet, "Contrastive Learning of Musical Representations," Master's thesis, University of Amsterdam, 2021.

[44] F. Weninger, F. Eyben, and B. Schuller, "On-line continuous-time music mood regression with deep recurrent neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 5412–5416.

[45] E. Koh and S. Dubnov, "Comparison and analysis of deep audio embeddings for music emotion recognition," arXiv:2104.06517, 2021.

[46] Pioneer, "Rekordbox v3.2.2," 2015, available online at http://www.cp.jku.at/datasets/giantsteps/.

[47] J. Jiang, G. G. Xia, and D. B. Carlton, "Crowd annotation for audio key estimation," 2019.

[48] F. Medhat, D. Chesmore, and J. Robinson, "Masked conditional neural networks for audio classification," in *Proceedings of the International Conference on Artificial Neural Networks*, 2017, pp. 349–358.

[49] F. Korzeniowski and G. Widmer, "End-to-end musical key estimation using a convolutional neural network," in *Proceedings of the 25th European Signal Processing Conference*, 2017, pp. 966–970.

[50] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[51] P. Knees, Á. Faraldo Pérez, H. Boyer, R. Vogl, S. Böck, F. Hörschläger, and M. Le Goff, "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*, 2015.

[52] M. Soleymani, M. N. Caro, E. M. Schmidt, C.-Y. Sha, and Y.-H. Yang, "1000 songs for emotional analysis of music," in *Proceedings of the 2nd ACM International Workshop on Crowdsourcing for Multimedia*, 2013, pp. 1–6.

DDSP-BASED SINGING VOCODERS: A NEW SUBTRACTIVE-BASED SYNTHESIZER AND A COMPREHENSIVE EVALUATION

Da-Yi Wu[1*] **Wen-Yi Hsiao**[2*] **Fu-Rong Yang**[3*] **Oscar Friedman**[4]
Warren Jackson[5] **Scott Bruzenak**[4] **Yi-Wen Liu**[3] **Yi-Hsuan Yang**[1,2]

[1] Academia Sinica, [2] Taiwan AI Labs, [3] National Tsing Hua Univ., [4] 470 Music Group, [5] PARC
{ericwudayi2, s101062219, fjbcrs34}@gmail.com

ABSTRACT

A vocoder is a conditional audio generation model that converts acoustic features such as mel-spectrograms into waveforms. Taking inspiration from Differentiable Digital Signal Processing (DDSP), we propose a new vocoder named SawSing for singing voices. SawSing synthesizes the harmonic part of singing voices by filtering a sawtooth source signal with a linear time-variant finite impulse response filter whose coefficients are estimated from the input mel-spectrogram by a neural network. As this approach enforces phase continuity, SawSing can generate singing voices without the phase-discontinuity glitch of many existing vocoders. Moreover, the source-filter assumption provides an inductive bias that allows SawSing to be trained on a small amount of data. Our evaluation shows that SawSing converges much faster and outperforms state-of-the-art generative adversarial network- and diffusion-based vocoders in a resource-limited scenario with only 3 training recordings and a 3-hour training time.*

1. INTRODUCTION

Singing voice synthesis (SVS) aims to generate human-like singing voices from musical scores with lyrics [1–9]. State-of-the-art (SOTA) voice synthesis techniques involve two stages: acoustic feature modeling from musical scores and audio sample reconstruction via a so-called "vocoder." A *neural vocoder* takes an acoustic feature such as mel-spectrogram as input and outputs a waveform using deep learning networks [10–22]. However, phase discontinuities within partials often occur due to the difficulty of reconstructing realistic phase information from a mel-spectrogram. This may lead to a short-duration broadband transient perceived as "glitch" or "voice tremor," which is more audible during long utterances commonly found in singing [18], as exemplified in Figure 1.

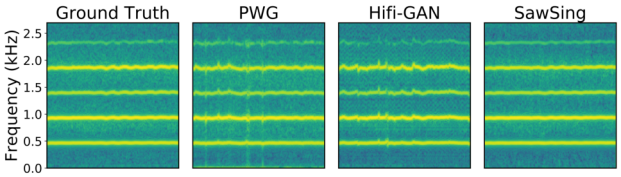


Figure 1: The magnitude spectrograms of a long utterance of an original recording ('ground truth') and those reconstructed by two widely-used neural vocoders, Parallel WaveGAN (PWG) [14] and HiFi-GAN [15], and the proposed SawSing. Each vocoder is trained on 3 hours of recordings from a female singer until convergence. We see glitches in the results of PWG and HiFi-GAN.

Differentiable Digital Signal Processing (DDSP) [23] introduces a new paradigm for neural audio synthesis. It incorporates classical digital signal processing (DSP) synthesizers and effects as differentiable functions within a neural network (NN), and combines the expressiveness of an NN with the interpretability of classical DSP. The use of phase-continuous oscillators is a potential solution to the phase problem from which regular neural vocoders suffer, and the strong inductive bias of this approach may obviate the need of large training data. Furthermore, DDSP has already succeeded in achieving sound synthesis of, and timbre transfer between, monophonic instruments [23–30]. These motivate us to explore whether the DDSP approach can be applied to build a singing vocoder.

This paper proposes SawSing, a DDSP-based singing vocoder which reconstructs a monophonic singing voice from a mel-spectrogram. The architecture of SawSing similarly consists of an NN and classical DSP components; unlike DDSP, its DSP portion is a subtractive harmonic synthesizer which filters a sawtooth waveform containing all possible harmonic partials, plus a subtractive noise synthesizer which filters uniform noise. The sawtooth signal enforces phase continuity *within* partials, thereby avoiding the glitches. Moreover, the partials of a sawtooth signal are guaranteed to be in phase, so it also enforces the phase coherence *between* partials, intrinsic to human voices. The function of the NN, on the other hand, is to infer from the mel-spectrogram the fundamental frequency (f0) of the sawtooth signal and the filter coefficients of the harmonic and noise synthesizers for each time frame.

*Equal contribution. Preliminary work was done while Wu was a remote intern working with Friedman at 470 Music Group, LLC.

© Wu *et al.*. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Wu *et al.*, "DDSP-based Singing Vocoders: A New Subtractive-based Synthesizer and A Comprehensive Evaluation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

In our experiments, we use data from two singers (each three hours) of the MPop600 Mandarin singing corpus [31]. We compare the performance of SawSing with the neural source-filter (NSF) model [12], two existing DDSP-based synthesizers, the original additive-based DDSP [23] and the differentiable wavetable synthesizer [27], and a few famous neural vocoders, i.e., two generative adversarial network (GAN)-based models [5, 14] and a diffusion-based model [21]. We consider both a regular scenario where the vocoders are trained for days using the 3-hour dataset, and a resource-limited scenario with constraints on training data and training time. Our experiments show that SawSing converges much faster and outperforms the other vocoders in the resource-limited scenario.

The main contribution of the paper is two-fold. First, we show that despite differences between instrumental sounds and singing voices [32], the classic idea of subtractive synthesis [33, 34] can be applied to singing voices using the DDSP approach.[1] Second, we provide empirical evidences showing that DDSP-based vocoders can compare favorably with sophisticated, SOTA neural vocoders. Furthermore, since DDSP-based vocoders are lightweight and training-efficient, they have the potential to be used in creative and real-time scenarios of singing expression with limited training data of a target singing voice [36, 37].

We open source our code at `https://github.com/YatingMusic/ddsp-singing-vocoders/`. For audio examples, visit our demo webpage `https://ddspvocoder.github.io/ismir-demo/`.

2. BACKGROUND

Neural vocoders often aim to reconstruct a waveform $\mathbf{y} \in \mathbb{R}^{1\times T}$ from an input mel-spectrogram $\mathbf{X} \in \mathbb{R}^{M\times N}$:

$$\mathbf{y} = f_{\text{vocoder}}(\mathbf{X}), \tag{1}$$

where M, N, T denote respectively the number of mel filter banks, spectral frames, and time-domain samples. The conversion from $\mathbf{X}$ to $\mathbf{y}$ can be done, for example, by upsampling $\mathbf{X}$ multiple times through transposed *convolutions* until the length of the output sequence matches the temporal resolution of the raw waveform [13, 15]. As usual reconstruction loss functions such as mean-square errors cannot reflect the perceptual quality of the reconstruction, GAN-based approaches [13–15] learn discriminators to better guide the learning process of the generator (i.e., f_{vocoder}). Newer diffusion-based approaches [20, 21] avoid the use of discriminators and learn to convert white Gaussian noises $\mathbf{z} \in \mathbb{R}^{1\times T}$ (i.e., of the same length as $\mathbf{y}$) into structured waveform $\mathbf{y}$ through a denoising-like Markov chain, using $\mathbf{X}$ as a condition. The mapping process between $\mathbf{X}$ and $\mathbf{y}$ of such neural vocoders appears to be a black box that is hard to interpret. However, given sufficient training data (e.g., recordings amounting to 24 hours [15, 20, 21, 38] or 80 hours [18]) and training time (e.g., days), SOTA neural vocoders can reconstruct the waveforms with high fidelity.

The majority of neural vocoders, however, have been originally developed for speech. When the rate of utterances is fast, as is common in speech, the glitches resulting from the phase discontinuities within partials may be perceptually masked by the natural transients of the voice. However, during singing, where long utterances are common, these discontinuities are more audible.

To improve the performance of GAN-based vocoders for singing voices, the idea of incorporating the f0 information has been explored recently. PeriodNet [16] uses f0 to create sine excitation as input to Parallel WaveGAN (PWG) [14] to model the periodic part of human voices. Guo *et al.* [17] further filter such an f0-driven excitation signal with a linear time-variant finite impulse response (LTV-FIR) filter whose coefficients are estimated from the input mel-spectrogram, and use the resulting "harmonic signal" as input to PWG and MelGAN [13]. SingGAN [18] uses more complicated "adaptive feature learning" layers to incorporate the f0. These models were shown to outperform older GAN-based vocoders such as PWG and MelGAN in listening tests, but no evaluations against the newest GAN-based vocoder HiFi-GAN [15] and diffusion-based vocoders were reported. Moreover, their evaluation did not consider resource-limited scenarios.

We propose in this paper a radically different approach that uses traditional DSP synthesizers (instead of upsampling convolutions) as the backbone for f_{vocoder}. While the ideas in DDSP have flourished and been applied to synthesizing not only instrumental sounds [23–27], but also audio effects [39–43], their application to singing synthesis remains under-explored. The only exception, to our knowledge, is the preliminary work presented by Alonso and Erkut [44], which employed exactly the same additive synthesizer as the original DDSP paper [23]. However, they did not compare the performance of their vocoder with any other vocoders. Our work extends theirs by using a subtractive harmonic synthesizer instead, with comprehensive performance evaluations against SOTA neural vocoders such as HiFi-GAN and FastDiff [21].

We note that, while a DDSP-based vocoder may solve the glitch problems by inducing continuous phase hypothesis using a harmonic synthesizer, this hypothesis may constrain the model learning ability. Experiments reported in this paper are needed to study its performance.

Publicly-available training corpora for singing tend to be much smaller than those for speech [31, 45] (often $\leq$10 hours). Therefore, besides tackling the glitch problem, our premise is that SawSing can learn faster than prevalent neural vocoders without a large training corpus, due to its strong inductive bias. Moreover, the success of SawSing may pave the way for the exploration of other advanced DSP components for singing synthesis in the future.

3. ORIGINAL DDSP-ADD SYNTHESIZER

The idea of DDSP is to use DSP synthesizers to synthesize the target audio, with the parameters of the synthesizers Φ

[1] We note that the use of a sawtooth waveform in DDSP-based models has been attempted for speech synthesis [35] and instrumental synthesizer sound matching [26], but its application to singing vocoder is new.

inferred from the mel-spectrogram with an NN. Namely,

$$\mathbf{y} = f_{\text{DSP}}(\Phi)\,, \quad \Phi = f_{\text{NN}}(\mathbf{X})\,. \tag{2}$$

The original DDSP model [23], referred to as **DDSP-Add** below, adopts the *harmonic-plus-noise* model for synthesis [46] and decomposes a monophonic sound into a periodic (harmonic) component $\mathbf{y}_{\text{h}}$ and a stochastic (noise) component $\mathbf{y}_{\text{n}}$, i.e., $\mathbf{y} = \mathbf{y}_{\text{h}} + \mathbf{y}_{\text{n}}$, and reconstructs them separately with an *additive* harmonic oscillator (thus the name "-Add") and a *subtractive* noise synthesizer. [2] The former computes $\mathbf{y}_{\text{h}}$ as a weighted sum of K sinusoids corresponding to the f0 and its integer multiples up to the Nyquist frequency (for anti-aliasing), for $t \in [1, T]$:

$$\mathbf{y}_{\text{h}}^{\text{DDSP-Add}}(t) = A(t) \sum_{k=1}^{K} c_k(t)\, \sin(\phi_k(t))\,, \tag{3}$$

where $A(t)$ is the global amplitude corresponding to the time step t, $c_k(t)$ is the amplitude of the k-th harmonic satisfying $\sum_{k=1}^{K} c_k(t) = 1, c_k(t) \geq 0$, and the instantaneous phase $\phi_k(t)$ is computed by integrating the instantaneous frequency $kf_0(t)$, i.e., $\phi_k(t) = 2\pi \sum_{\tau=0}^{t} kf_0(\tau) + \phi_{0,k}$, with $\phi_{0,k}$ initial phase, set to zero. The parameters A, c_k, f_0 are estimated by f_{NN} for each frame $i \in [1, N]$ and then upsampled to the time-domain with linear interpolation. On the other hand, $\mathbf{y}_{\text{n}}$ is obtained by convolving a uniform noise signal ζ ranging from -1 to 1 (with the same length as a frame) with an LTV-FIR filter $\psi_{\text{n}}(i) \in \mathbb{R}^{L_{\text{n}}}$ estimated per frame:

$$\bar{\mathbf{y}}_{\text{n}}(i) = \zeta * \psi_{\text{n}}(i)\,. \tag{4}$$

The final $\mathbf{y}_{\text{n}}$ is obtained by overlap-adding sequence of segments $\bar{\mathbf{y}}_{\text{n}}(i)$ for the frames $i = 1 \ldots N$. Jointly, the parameters $\Phi := \{A(i), \{c_k(i)\}_{k=1}^{K}, f_0(i), \psi_{\text{n}}(i)\}_{i=1}^{N}$ are estimated from the mel-spectrogram $\mathbf{X}$ per frame by f_{NN}, which is a small network with few parameters.

Engel *et al.* [23] showed that DDSP-Add can synthesize realistic violin sounds with only 13 minutes of expressive solo violin performances as training data. Alonso and Erkut [44] employed DDSP-Add for singing synthesis, but with limited performance evaluation.

4. PROPOSED SAWSING VOCODER

Under the same harmonic-plus-noise signal model [46], SawSing modifies the the harmonic synthesizer of DDSP-Add [23] with two ideas. First, given the f0 estimated from $\mathbf{X}$, SawSing approximates $\mathbf{y}_{\text{h}}$ by a sawtooth signal, which contains an equal number of even and odd harmonics with decaying magnitudes, dropping the coefficients A and c_k:

$$\widetilde{\mathbf{y}_{\text{h}}}^{\text{SawSing}}(t) = \sum_{k=1}^{K} \frac{1}{k} \sin(\phi_k(t))\,. \tag{5}$$

[2] The terms "additive" and "subtractive" are used to describe how a signal is synthesized. An additive synthesizer generates sounds by combining multiple sources such as oscillators or wavetables, while a subtractive synthesizer creates sounds by using filters to shape a source signal, typically with rich harmonics, such as a square or sawtooth wave [46].

Second, $\widetilde{\mathbf{y}_{\text{h}}}$ is treated as the "excitation signal" and shaped into the desirable $\mathbf{y}_{\text{h}}$ by means of an LTV-FIR filter $\psi_{\text{h}}(i) \in \mathbb{R}^{L_{\text{h}}}$ (that is different from $\psi_{\text{n}}(i)$). To apply the filter, we extract the segment of $\widetilde{\mathbf{y}_{\text{h}}}$ corresponding to the same frame i and multiply its short-time Fourier Transform (STFT) element-wise with the STFT of $\psi_{\text{h}}(i)$ in the frequency domain, before converting it back to the time domain with the inverse STFT and overlap-adding. SawSing uses the same subtractive noise synthesizer as DDSP-Add. Therefore, the parameters to be estimated from $\mathbf{X}$ by f_{NN} are $\Phi^{\text{SawSing}} := \{f_0(i), \psi_{\text{h}}(i), \psi_{\text{n}}(i)\}_{i=1}^{N}$.

We observe that to compute $\mathbf{y}_{\text{h}}$, DDSP-Add learns NN to attenuate each of the k source harmonics *individually* (i.e., with c_k), while SawSing entails a *source-filter* model [12], using the f0-constrained sawtooth signal in Eqn. (5) as the excitation source and a time-varying filter $\psi_{\text{h}}(i)$ decided by the NN for spectral filtering. The filter coefficients correspond to formants produced by the vocal folds and do not correlate with f0.

Besides differences in the harmonic synthesizer, SawSing also uses a different loss function from DDSP-Add. For monophonic instrumental sounds, Engel *et al.* [23] showed it effective to use the multi-resolution STFT (MSSTFT) loss as the reconstruction loss for training. This loss considers the difference between the magnitude spectrograms of the target and synthesized audio, denoted as $\mathbf{S}_j$ and $\widehat{\mathbf{S}_j}$ below, for J different resolutions.

$$l_{\text{MSSTFT}} = \sum_{j=1}^{J} \|\mathbf{S}_j - \widehat{\mathbf{S}_j}\|_1 + \|\log(\mathbf{S}_j) - \log(\widehat{\mathbf{S}_j})\|_1\,. \tag{6}$$

For singing voices, however, we found that MSSTFT loss alone cannot train adequately. We introduce an additional f0-related loss term to facilitate learning:

$$l_{f_0} = \|\log(f_0) - \log(\widehat{f_0})\|_1\,, \tag{7}$$

where the target f0 (f_0) and the estimated one ($\widehat{f_0}$) are both extracted by the WORLD vocoder [47]. Thus, our Sawsing loss function becomes $l_{\text{total}} = l_{\text{MSSTFT}} + l_{f_0}$. Moreover, we found that training is unstable unless the gradients between f_{DSP} and the head of f_{NN} for f0 prediction are detached.

4.1 Implementation Details

First, we resampled the audio recordings to 24 kHz and quantized them to 16 bits. Next we cropped the recordings into 2-second excerpts (i.e., $T = 48$k) and extracted 80-band mel-spectrograms from each ($M = 80$), with a Hann window of 1024 samples for STFT and a hop size of 240 samples (i.e., 10ms). Accordingly, we set $N = 200$.

We used filter length $L_{\text{h}} = 256$ for the harmonic synthesizer for SawSing, and filter length $L_{\text{n}} = 80$ for the subtractive noise synthesizers. We used at most $K = 150$ sinusoids for SawSing. To avoid sound clipping, we applied a global scaling factor of 0.4 to the sawtooth signal in Eqn. (5) to ensure that the range of the summed sinusoids always lies in $[-1, 1]$.

We chose a lite version of the Conformer architecture for f_{NN} [48], for its well-demonstrated effectiveness in

capturing both local and global information in a sequence of acoustic features in speech tasks. It consists of a pre-net (shallow 1D convolution with ReLU activation and group normalization), a self-attention stack (3 layers), a convolution stack (2 layers) with post layer normalization, and a final linear layer whose output dimension is equal to the number of synthesis coefficients. We used the Adam optimizer with 0.002 learning rate.

While the original DDSP-Add paper [23] uses $J = 6$ for MSSTFT, we found setting $J = 4$ to be sufficient in our implementation. Specifically, we used four different FFT sizes (128, 256, 512, 1024) with 75% overlapping among adjacent frames. While it is possible to introduce a scaling factor to control the balance between l_{MSSTFT} and l_{f_0}, we found doing so does not markedly improve the result.

5. EXPERIMENTAL SETUP

5.1 Baselines

Our evaluation considers in total six baselines. The first three explicitly employ f0, while the last three do not.

First, we adopted two existing DDSP-based vocoders, the original additive-based DDSP (**DDSP-Add**) [23, 44] and the differentiable wavetable synthesizer (DWTS) [27]. **DWTS** replaces the fixed sinusoids in the additive harmonic synthesizer of DDSP-Add by K' learnable (rather than pre-defined) one-cycle waveforms ("the wavetables") $\mathbf{w}_k \in \mathbb{R}^B, k \in [1, K']$, to gain flexibility to model a wider variety of sounds (but only tested on instrumental sounds in [27]). Mathematically, $\mathbf{y}_{\text{h}}^{\text{DWTS}}(t) = A(t) \sum_{k=1}^{K'} c_k(t)\sigma(\mathbf{w}_k, \phi_\pi(t))$, where σ is an indexing function that returns a sample of $\mathbf{w}_k$ according to the instantaneous modulo phase $\phi_\pi(t)$ computed from $f_0(t)$. For fair comparison, we used the same Conformer-like architecture for the f_{NN} for DDSP-Add, DWTS, and SawSing, and the same noise synthesizer. Moreover, while the original DDSP-Add used only l_{MSSTFT}, thus we used $l_{\text{total}} = l_{\text{MSSTFT}} + l_{f_0}$ for all three in our implementation. Like SawSing, we set $K = 150$ for DDSP-Add. DWTS only needs small K' as the wavetables are learnable; we set $K' = 20$, with wavetable length being $B = 512$.

We also employed the neural source-filter (**NSF**) waveform model [12], which was proposed before the notion "DDSP" was coined [23]. Unlike SawSing, NSF uses unweighted sinusoids (i.e., $\sum_{k=1}^{K} \sin(\phi_k(t))$) as the source signal, and uses stacked dilated-convolution blocks instead of a simple LTV-FIR filter. We adapted the open-source code from the original authors to implement NSF, as well as the following three baselines.

For GAN-based neural vocoders, we used **PWG** [14] and **HiFi-GAN** [15], both of which were developed for speech, not singing.[3] PWG is a non-autoregressive version of WaveNet [10] that learns to transform a random noise into target audio with 30 layers of dilated residual convolution blocks, conditioning on the mel-spectrogram. For HiFi-GAN, we used the most powerful "V1" configuration [15], which converts a mel-spectrogram into a waveform directly via 12 residual blocks. It uses a sophisticated multi-receptive field fusion module in the generator, and multiple multi-scale and multi-period discriminators [15].

An increasing number of diffusion-based vocoders have been proposed in the past two years for speech [19–22]. We adopt as a baseline the **FastDiff** model [21], which has been shown to beat HiFi-GAN V1 [15] and diffusion-based models WaveGrad [19] and DiffWave [20] in the mean-opinion-score (MOS) of vocoded speech in listening tests. However, while a noise schedule predictor has been devised to reduce the sampling steps of the denoising Markov chain, the inference time of FastDiff is still around 10 times slower than HiFi-GAN, according to [21].

None of these baselines have been trained on MPop600, which is a relatively new dataset. Therefore, we trained all these models from scratch with the MPop600 data.

5.2 Dataset & Scenarios

Our data is from MPop600 [31], a set of accompaniment-free Mandarin singing recordings with manual annotation of word-level audio-lyrics alignment. Each recording covers the first verse and first chorus of a song. We used the data from a female singer (named `f1`) and a male singer (`m1`); each has 150 recordings. For each singer, we reserved 3 recordings (totalling 3.4–3.6 minutes in length) as the *test* set for subjective evaluation, 24 or 21 recordings (around 27–28 minutes) as the *validation* set for objective evaluation, and used the rest (around 3 hours) as the *training* set. We trained vocoders for `m1` and `f1` independently.

To study the training efficiency of different approaches, we considered the following two scenarios. We used the same validation and test sets for both scenarios.

(a) *Regular* [3h data, well-trained]: we used the full training data to train the vocoders for each singer for up to 2.5 days (i.e., when the training loss of most vocoders converged), and picked the epoch that reaches the lowest validation loss for each vocoder independently. We note that the amount of training time in this "regular" scenario is smaller than those seen in speech vocoders [38], posing challenges for all the considered models.

(b) *Resource-limited* [3min data, 3h training]: in a rather extreme case, we randomly picked 3 recordings from the training set (that collectively cover every phoneme at least once) per singer (3.2–3.4 minutes) for training, using always the epoch at 3-hour training time.

For fair comparison, we train the vocoders of different approaches using a dedicated NVIDIA GeForce RTX 3090 GPU each, fixing the batch size to 16 excerpts.

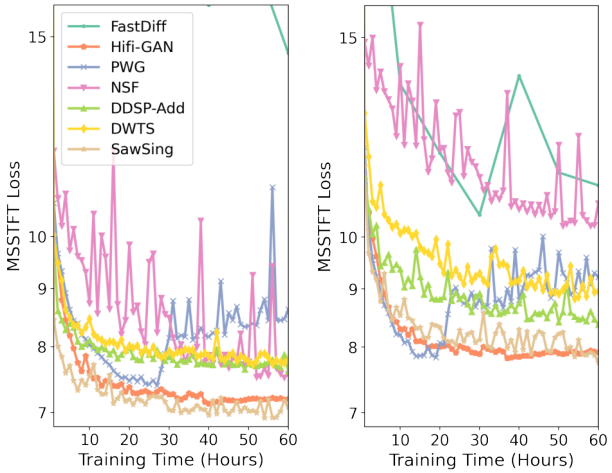
6. OBJECTIVE EVALUATION

For objective evaluation, we reported the MSSTFT and the mean absolute error (MAE) in f0, as well as the Fréchet audio distance (FAD) [49] between the validation data and the reconstructed ones by the vocoders. FAD measures the

[3] While we did not consider SingGAN [18] in the evaluation for lack of open source code, we should have included PeriodNet [16] for it seems easy to implement. Unfortunately we are aware of PeriodNet too late.

Model	Para-meters	RTF	MSSTFT ↓ Female (a)	MSSTFT ↓ Female (b)	MSSTFT ↓ Male (a)	MSSTFT ↓ Male (b)	MAE-f0 (cent) ↓ Female (a)	MAE-f0 (cent) ↓ Female (b)	MAE-f0 (cent) ↓ Male (a)	MAE-f0 (cent) ↓ Male (b)	FAD ↓ Female (a)	FAD ↓ Female (b)	FAD ↓ Male (a)	FAD ↓ Male (b)
FastDiff [21]	15.3M	0.017	14.5	17.9	11.1	16.9	31	110	48	131	2.29	7.40	3.53	10.0
HiFi-GAN [15]	13.9M	0.004	7.13	16.7	7.82	18.9	34	247	34	433	0.59	3.50	0.51	10.5
PWG [14]	1.5M	0.007	7.39	13.0	7.83	14.8	35	129	29	126	0.36	6.15	2.56	6.29
NSF [12]	1.2M	0.006	7.51	10.9	10.2	13.4	37	**50**	30	82	0.49	3.73	2.08	4.83
DDSP-Add [44]	0.5M	0.003	7.61	9.29	8.37	12.1	**28**	70	**24**	**80**	0.56	0.92	1.06	2.09
DWTS [27]	0.5M	0.019	7.72	9.75	8.83	13.0	**28**	127	**24**	662	0.60	2.98	0.36	8.58
SawSing	0.5M	0.003	**6.93**	**8.79**	**7.76**	**11.7**	32	76	30	**80**	**0.12**	**0.38**	**0.22**	**0.59**

Table 1: Objective evaluation results of three existing neural vocoders (the first three), three existing DDSP-based vocoders (middle) and the proposed SawSing vocoder, trained on either a female or a male singer, in either (a) *regular* scenario [3h data, well-trained] or (b) *resource-limited* scenario [3min data, 3h training]. RTF stands for real-time factor (the inference time in seconds for a one-second excerpt). In each column, we highlight the best result in bold, the second best underlined.



(i) female (ii) male

Figure 2: The MSSTFT loss on the validation set of different vocoders in the 3-hour data & well-trained scenario.

similarity of the real data distribution and generated data distribution in an embedding space computed by a pre-trained VGGish-based audio classifier, and may as such better reflect the perceptual quality of the generated audio.

Figure 2 shows the validation MSSTFT loss as a function of the training time in the regular scenario for the two singers. In Figure 2(i), SawSing converges faster than the other models and reaches the lowest loss (i.e., 6.93), followed by HiFi-GAN and PWG. While DDSP-add and DWTS converge similarly fast as SawSing, they reach at a slightly higher loss (around 7.50). In Figure 2(ii), Sawing, HiFi-GAN and PWG perform comparably in the first 20 hrs. For both singers, PWG overfits when the training time gets too long. Moreover, FastDiff converges the most slowly, followed by NSF. Even with 60-hour training time, the MSSTFT of FastDiff remains to be high (e.g., 14.5 for the female singer), suggesting that our training data might not be big enough for this diffusion-based model.[4]

Table 1 shows the scores in all the three metrics on the validation set for both scenarios, using the epoch (a) at the lowest validation loss or (b) at 3h training. Despite having few trainable parameters, SawSing performs the best in MSSTFT and FAD across both scenarios and both singers, demonstrating its effectiveness as a singing vocoder. For scenario (b), DDSP-Add obtains the second-lowest MSSTFT and FAD across the two singers.

For MAE-f0, SawSing attains scores comparable to the best baseline models. The average MAE-f0 of SawSing is less than a semitone (100 cents). Future work can use a specialized module (e.g., [50]) for the f0 prediction part in f_{NN} of SawSing to further improve the MAE-f0.

Table 1 also shows that the performance gap between scenarios (a) and (b) in all the three metrics tend to be greater for the diffusion- and GAN-based vocodoers than for NSF and the DDSP-based vocoders. Besides, among the evaluated models, the performance gap between (a) and (b) is the smallest in the result of SawSing. In the resource-limited scenario (b), the FAD of HiFi-GAN reaches only 3.50 and 10.5 for the female and male singers, respectively, while the FAD of SawSing can be lower than 1.0. This demonstrates that a strong inductive bias like those employed in NSF and the DDSP-based vocoders is helpful in scenarios with limited training data and training time.

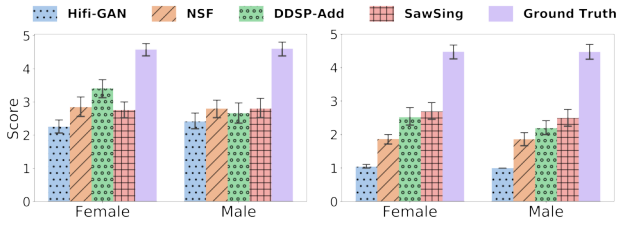
Table 1 also displays the real-time factor (RTF) of the models when being tested on a single NVIDIA 3090 GPU. We see that SawSing and DDSP-Add have the lowest RTF (i.e., run the fastest), followed by HiFi-GAN.

According to Table 1, HiFi-GAN performs the best on average among the first three vocoders across scenarios and singers. NSF and the DDSP-based vocoders obtain comparable scores, but DWTS is notably slower. Hence, we pick HiFi-GAN, NSF, DDSP-Add and SawSing to be further evaluated in the user study below.

7. SUBJECTIVE EVALUATION

We conducted an online study to evaluate the performance of the 4 selected models. We had 2 sets of questionnaires, one for the female and the other for the male singer. For each singer, we prepared 8 clips from the 3 *testing* recordings (i.e., totally unseen at training/validation time), each clip corresponding to the singing of a *full sentence*. We

[4] In the original paper [21], FastDiff was trained on 24 hours of speech data from a female speaker [38], using 4 NVIDIA V100 GPUs. We tried DiffWave [20] but it converged similarly slow on our data.



(a) 3h data, well-trained (b) 3min data, 3h training

Figure 3: MOS with 95% confidence intervals for subjective evaluation of vocoders trained in the two scenarios.

let the vocoders trained in scenario (a) to reconstruct the waveforms from the mel-spectrograms of 4 of the clips, and those of (b) for the other 4 clips. A human subject was requested to use a headset to listen to 5 versions of each of the clip, namely the original 'ground truth' recording and the reconstructed ones by the 4 selected models, with the ordering of the 5 versions randomized, the ordering of the 8 clips randomized, and not knowing the scenario being considered per clip. The loudness of the audio files were all normalized to –12dB LUFS beforehand using `pyloudnorm` [51]. After listening, the subject gave an opinion score from 1 (poor) to 5 (good) in a 5-point Likert scale to rate the audio quality for each audio file.

Figure 3 shows the MOS from 23 anonymized participants for the female and 18 participants for the male singer. In scenario (a), we see that the MOS of the vocoders, including the SOTA HiFi-GAN, mostly reaches 2–3 only, suggesting that training a vocoder on 3-hour data is already challenging. As HiFi-GAN involves a complicated GAN training and much more parameters, its MOS turns out to be significantly lower than those of NSF and the DDSP-based vocoders (p-value$<$0.05 in paired t-test). Interestingly, while there is no statistical difference among the MOS of NSF, DDSP-Add and SawSing for the male singer in scenario (a), DDSP-Add unexpectedly outperforms both NSF and SawSing by a large margin, with statistically significant difference (p-value$<$0.05).

Listening to the result of SawSing reveals that its output contains an audible electronic noise, or "buzzing" artifact, notably when singers emphasize the airflow with breathy sounds and for unvoiced consonants such as `/s/` and `/t/`. DDSP-Add is free of such an artifact. As shown in Figure 4, such artifact appears to due to *redundant* harmonics generated by SawSing that "connect" the harmonics of two adjacent phonemes at its harmonic signal x_{h} for breathing and unvoiced consonants. This may be due to the limited capacity of the LTI-FIR filter of SawSing in distinguishing between the nuances of voiced (V) and unvoiced (NV) components during sound transients, modeling a transient even as a harmonic signal. Unfortunately, it seems that this artifact cannot be reflected in any training loss functions (and objective metrics) we considered, so the network fails to take it into account while updating the parameters. Furthermore, human ears are sensitive to such an artifact, contributing to the lower MOS of SawSing compared to DDSP-Add, despite that SawSing might perform better in

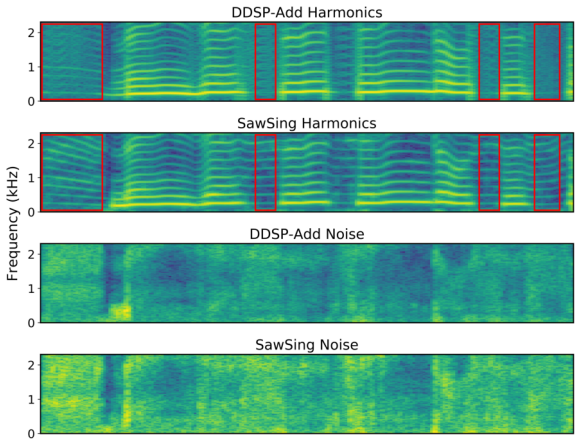


Figure 4: The spectrograms of the harmonic signal x_{h} and noise signal x_{n} generated by DDSP-Add and SawSing for the same clip. The red rectangles highlight the moments the buzzing artifact of SawSing emerges.

other phonemes and long utterances.

Figure 3 also shows that the DDSP-based vocoders do outperform HiFi-GAN greatly in the resource-limited scenario (b) with only 3 training recordings, nicely validating the training efficiency of the DDSP-based vocoders. While the MOS of either DDSP-Add or SawSing is above 2; that of HiFi-GAN is only around 1, i.e., its generation is barely audible. Moreover, SawSing outperforms DDSP-Add in this scenario for both singers, with significant MOS difference for the male singer (p-value$<$0.05), though not for the female singer. This shows that, despite the buzzing artifact, the training efficiency of SawSing can give it an edge over other vocoders in resource-limited applications.

Inspired by [5], we implement a postprocessing method that uses Parselmouth [52] to get V/NV flags and sets the harmonic synthesizer amplitudes to zero for the NV portions. This removes much of the artifact (see the demo page). We share the code on our GitHub repo. Future work can incorporate the V/NV flags at the training phase.

8. CONCLUSION

In this paper, we have presented SawSing, a new DDSP-based vocoder that synthesizes an audio via the summation of a harmonic component obtained from filtered sawtooth waves and a stochastic component modeled by filtered noise. Moreover, we presented objective and subjective evaluations complementing the lack of experiments in the recent work of Alonso and Erkut [44], demonstrating for the first time that both SawSing and DDSP-Add [23, 44] compare favorably with SOTA general-purpose neural vocoders such as HiFi-GAN [15] and FastDiff [21] for singing voices in a regular-resource scenario, and has a great performance margin in a resource-limited scenario.

Future work can improve the harmonic filter of SawSing to resolve the artifact, and use lighter-weight non-causal convolutions [53] for f_{NN} for real-time applications. We can also implement SawSing as a VST audio plugin for usages in creative workflows and music production [54].

9. ACKNOWLEDGEMENT

We are grateful to Rongjie Huang and Yi Ren for sharing with us the code of FastDiff, and Shuhei Imai for helping proofread the paper. We also thank the anonymous reviewers for their constructive feedbacks. Our research is funded by grants NSTC 109-2628-E-001-002-MY2 and NSTC 109-2221-E-007-094-MY3 from the National Science and Technology Council of Taiwan.

10. REFERENCES

[1] P. R. Cook, "Singing voice synthesis: History, current work, and future directions," *Computer Music Journal*, vol. 20, no. 3, pp. 38–46, 1996.

[2] J. Lee, H.-S. Choi, C.-B. Jeon, J. Koo, and K. Lee, "Adversarially trained end-to-end Korean singing voice synthesis system," in *INTERSPEECH*, 2019.

[3] M. Blaauw and J. Bonada, "Sequence-to-sequence singing synthesis using the feed-forward Transformer," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2020, pp. 7229–7233.

[4] Y. Ren, X. Tan, T. Qin, J. Luan, Z. Zhao, and T.-Y. Liu, "DeepSinger: Singing voice synthesis with data mined from the web," in *ACM Int. Conf. Knowledge Discovery & Data Mining*, 2020, pp. 1979–1989.

[5] J. Chen, X. Tan, J. Luan, T. Qin, and T.-Y. Liu, "HiFiSinger: Towards high-fidelity neural singing voice synthesis," *arXiv preprint arXiv:2009.01776*, 2020.

[6] Y. Hono *et al.*, "Sinsy: A deep neural network-based singing voice synthesis system," *IEEE Trans. Audio, Speech and Lang. Proc.*, vol. 29, pp. 2803–2815, 2021.

[7] Y.-P. Cho, F.-R. Yang, Y.-C. Chang, C.-T. Cheng, X.-H. Wang, and Y.-W. Liu, "A survey on recent deep learning-driven singing voice synthesis systems," in *IEEE Int. Conf. Artificial Intelligence and Virtual Reality*, 2021.

[8] C.-F. Liao, J.-Y. Liu, and Y.-H. Yang, "KaraSinger: Score-free singing voice synthesis with VQ-VAE using Mel-spectrograms," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2022.

[9] J. Liu, C. Li, Y. Ren, F. Chen, P. Liu, and Z. Zhao, "DiffSinger: Singing voice synthesis via shallow diffusion mechanism," in *AAAI Conference on Artificial Intelligence*, 2022.

[10] A. v. d. Oord *et al.*, "WaveNet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[11] N. Kalchbrenner *et al.*, "Efficient neural audio synthesis," in *Int. Conf. Machine Learning*, 2018.

[12] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter waveform models for statistical parametric speech synthesis," *IEEE/ACM Trans. Audio Speech Lang. Process.*, vol. 28, pp. 402–415, 2020.

[13] K. Kumar *et al.*, "MelGAN: Generative adversarial networks for conditional waveform synthesis," *arXiv preprint arXiv:1910.06711*, 2019.

[14] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2020, pp. 6199–6203.

[15] J. Kong, J. Kim, and J. Bae, "Hifi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Information Processing Systems*, 2020.

[16] Y. Hono *et al.*, "PeriodNet: A non-autoregressive waveform generation model with a structure separating periodic and aperiodic components," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2021.

[17] H. Guo, Z. Zhou, F. Meng, and K. Liu, "Improving adversarial waveform generation based singing voice conversion with harmonic signals," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2022.

[18] F. Chen, R. Huang, C. Cui, Y. Ren, J. Liu, and Z. Zhao, "SingGAN: Generative adversarial network for high-fidelity singing voice generation," in *ACM Multimedia*, 2022.

[19] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "WaveGrad: Estimating gradients for waveform generation," in *Int. Conf. Learning Representations*, 2021.

[20] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "DiffWave: A versatile diffusion model for audio synthesis," in *Int. Conf. Learning Representations*, 2021.

[21] R. Huang, M. W. Y. Lam, J. Wang, D. Su, D. Yu, Y. Ren, and Z. Zhao, "FastDiff: A fast conditional diffusion model for high-quality speech synthesis," in *Int. Joint Conf. Artificial Intelligence*, 2022.

[22] M. W. Y. Lam, J. Wang, D. Su, and D. Yu, "BDDM: Bilateral denoising diffusion models for fast and high-quality speech synthesis," in *Int. Conf. Learning Representations*, 2022.

[23] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Int. Conf. Learning Representations*, 2021.

[24] A. Bitton, P. Esling, and T. Harada, "Neural granular sound synthesis," in *Int. Computer Music Conf.*, 2021.

[25] B. Hayes, C. Saitis, and G. Fazekas, "Neural waveshaping synthesis," in *Int. Soc. Music Information Retrieval Conf.*, 2021, pp. 254–261.

[26] N. Masuda and D. Saito, "Synthesizer sound matching with differentiable dsp," *Proc. International Society for Music Information Retrieval*, 2021.

[27] S. Shan, L. Hantrakul, J. Chen, M. Avent, and D. Trevelyan, "Differentiable wavetable synthesis," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2022, pp. 4598–4602.

[28] M. Carney, C. Li, E. Toh, P. Yu, and J. Engel, "Tone Transfer: In-browser interactive neural audio synthesis," in *Workshop on Human-AI Co-Creation with Generative Models*, 2021.

[29] F. Ganis, E. F. Knudesn, S. V. K. Lyster, R. Otterbein, D. Südholt, and C. Erkut, "Real-time timbre transfer and sound synthesis using DDSP," in *Sound and Music Computing Conf.*, 2021.

[30] S. Nercessian, "End-to-end zero-shot voice conversion using a DDSP vocoder," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2021, pp. 1–5.

[31] C.-C. Chu, F.-R. Yang, Y.-J. Lee, Y.-W. Liu, and S.-H. Wu, "MPop600: A Mandarin popular song database with aligned audio, lyrics, and musical scores for singing voice synthesis," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conf.*, 2020, pp. 1647–1652.

[32] J. Sundberg, *The Science of the Singing Voice*. Northern Illinois University Press, 1989.

[33] J. Lane, D. Hoory, E. Martinez, and P. Wang, "Modeling analog synthesis with DSPs," *Computer Music Journal*, vol. 21, no. 4, pp. 32–41, 1997.

[34] A. Huovilainen and V. Välimäki, "New approaches to digital subtractive synthesis," in *Inr. Computer Music Conf.*, 2005.

[35] Z. Liu, K.-T. Chen, and K. Yu, "Neural homomorphic vocoder," in *INTERSPEECH*, 2020.

[36] G. Greshler, T. Shaham, and T. Michaeli, "Catch-a-waveform: Learning to generate audio from a single short example," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[37] S. Nercessian, "Zero-shot singing voice conversion," in *Int. Soc. Music Information Retrieval Conf.*, 2020, pp. 70–76.

[38] K. Ito, "The LJ speech dataset," 2017.

[39] B. Kuznetsov, J. D. Parker, and F. Esqueda, "Differentiable IIR filters for machine learning applications," in *Int. Conf. Digital Audio Effects*, 2020.

[40] M. A. M. Ramírez, O. Wang, P. Smaragdis, and N. J. Bryan, "Differentiable signal processing with black-box audio effects," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2021, pp. 66–70.

[41] C. J. Steinmetz, J. Pons, S. Pascual, and J. Serrà, "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2021, pp. 71–75.

[42] S. Nercessian, A. Sarroff, and K. J. Werner, "Lightweight and interpretable neural modeling of an audio distortion effect using hyperconditioned differentiable biquads," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2021, pp. 890–894.

[43] B.-Y. Chen, W.-H. Hsu, W.-H. Liao, R. M. A. Martínez, Y. Mitsufuji, and Y.-H. Yang, "Automatic DJ transitions with differentiable audio effects and generative adversarial networks," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2022, pp. 466–470.

[44] J. Alonso and C. Erkut, "Latent space explorations of singing voice synthesis using DDSP," *arXiv preprint arXiv:2103.07197*, 2021.

[45] Y. Wang, X. Wang, P. Zhu, J. Wu, H. Li, H. Xue, Y. Zhang, L. Xie, and M. Bi, "Opencpop: A high-quality open source Chinese popular song corpus for singing voice synthesis," *arXiv preprint arXiv:2201.07429*, 2022.

[46] X. Serra and J. Smith, "Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition," *Computer Music Journal*, vol. 14, no. 4, pp. 12–24, 1990.

[47] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Transactions on Information and Systems*, vol. E99.D, no. 7, pp. 1877–1884, 2016.

[48] A. Gulati *et al.*, "Conformer: Convolution-augmented Transformer for speech recognition," in *INTERSPEECH*, 2020.

[49] K. Kilgour *et al.*, "Fréchet Audio Distance: A metric for evaluating music enhancement algorithms," *arXiv preprint arXiv: 1812.08466*, 2019.

[50] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "CREPE: A convolutional representation for pitch estimation," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2018, pp. 161–165.

[51] C. J. Steinmetz and J. D. Reiss, "pyloudnorm: A simple yet flexible loudness meter in Python," in *Proc. AES Convention*, 2021.

[52] Y. Jadoul, B. Thompson, and B. de Boer, "Introducing Parselmouth: A Python interface to Praat," *Journal of Phonetics*, vol. 71, pp. 1–15, 2018.

[53] A. Caillon and P. Esling, "Streamable neural audio synthesis with non-causal convolutions," *arXiv preprint arXiv:2204.07064*, 2022.

[54] E. Deruty, M. Grachten, S. Lattner, J. Nistal, and C. Aouameur, "On the development and practice of AI technology for contemporary popular music production," *Transactions of the International Society for Music Information Retrieval*, vol. 5, no. 1, pp. 35–49, 2022.

EQUIVARIANT SELF-SUPERVISION FOR MUSICAL TEMPO ESTIMATION

Elio Quinton
Universal Music Group

ABSTRACT

Self-supervised methods have emerged as a promising avenue for representation learning in the recent years since they alleviate the need for labeled datasets, which are scarce and expensive to acquire. Contrastive methods are a popular choice for self-supervision in the audio domain, and typically provide a learning signal by forcing the model to be invariant to some transformations of the input. These methods, however, require measures such as negative sampling or some form of regularisation to be taken to prevent the model from collapsing on trivial solutions. In this work, instead of invariance, we propose to use equivariance as a self-supervision signal to learn audio tempo representations from unlabelled data. We derive a simple loss function that prevents the network from collapsing on a trivial solution during training, without requiring any form of regularisation or negative sampling. Our experiments show that it is possible to learn meaningful representations for tempo estimation by solely relying on equivariant self-supervision, achieving performance comparable with supervised methods on several benchmarks. As an added benefit, our method only requires moderate compute resources and therefore remains accessible to a wide research community.

1. INTRODUCTION

Manually annotated data is scarce and expensive to acquire, becoming a bottleneck to the continued progress of supervised learning methods in recent years. This is particularly true in the music domain, where a large proportion of content available is under copyright. Unlabelled data is comparatively abundant and inexpensive. Self-Supervised Learning (SSL) methods, which do not require labeled training data, have shown very promising development by rendering larger unlabelled datasets usable for training and yielding performance comparable or surpassing supervised benchmarks on Natural Language Processing (NLP) [1, 2], computer vision tasks [3], speech processing [4] and music tasks [5]. Self-supervised learning is still only nascent in musical audio, in comparison with other domains. To date it has only been considered with target downstream tasks that are a limited subset of the Music Information Retrieval (MIR) field, e.g. auto-tagging, genre classification or cover song detection [5–7]. With this contribution, we propose to extend the application of a SSL framework to the rhythmic properties of music by applying it to the tempo estimation task, which state of the art is still within the realm of supervised methods [8–11].

Self-supervision is typically realised by creating a pretext task providing a training signal from which it is expected that the model can learn useful representations. The general intuition underpinning this family of approaches is that accurately performing on the pretext task requires the model to learn meaningful representations of the domain at hand. For domains where data is made of discrete tokens, such as text, pretext tasks like masked language modelling or next sentence prediction as have proven to be effective and brought a step change in NLP [2]. Taking inspiration from it, comparable approaches can be devised in the symbolic music domain, which data is also in the form of discrete tokens [12]. In domains where data is dense and continuous, such as images or audio, one option is to apply similar token-based methods to discrete representations of the dense input [13, 14]. Alternatively, Siamese network frameworks [15], where two models process two views of a given input, are a popular choice for handling continuous data directly. In particular, contrastive methods where the pretext task consists in discriminating whether two inputs (or set thereof) should yield a similar, or dissimilar representation have been shown to be effective in computer vision [3] and in the audio domain [16–19], including musical audio [5–7].

In the contrastive learning framework, for every training sample, two views are generated by applying random data augmentations. The training objective then constrains the model to produce representations that are *invariant* to the augmentations applied to the training data and yet discriminative between different samples. The choice of augmentations is a critical design choice that impacts the quality and properties of representation learnt [20]. For example, in [5] the model is trained to be invariant to pitch-shifting. While this is appropriate for fine-tuning on an auto-tagging task, it would not be suitable for downstream tasks such as chord or key estimation. Generally speaking, the invariance constraints explicitly specifies what the representations should *not* be sensitive to. Conversely, *equivariant* constraints can be applied to enforce preferences on what features or concepts may be desirable to capture in the learnt representations [21, 22]. This is particularly at-

© E. Quinton. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** E. Quinton, "Equivariant self-supervision for musical tempo estimation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

tractive in scenarios where it is not clear how invariance constraints can yield a suitable representation, such as F0 estimation [23], or for guaranteeing that the representations capture specific semantically meaningful dimensions such as harmony or rhythm, which may be beneficial for applications such as music search and discovery.

In this work, instead of *invariance*, we propose to use *equivariance* as a self-supervision constraint to learn audio representations that specifically capture musical tempo, from unlabelled data. We propose a Siamese network framework where we produce two views of each training sample by applying a time stretching transformation with random factor and impose an equivariance constraint between the representations. Although the tempo of the training samples is unknown, as a result of the time-stretching transformation, the tempo of each sample is modified in two different, but known, ways. From there we derive a loss function that exploits this information and enforces the equivariant constraint. Because we wish the representations to capture rhythmic properties of music only, we also add audio augmentations to promote robustness against potentially confounding attributes of the audio signal. As is customary with self-supervised pre-training [3, 5], we evaluate the quality of the representations learnt with our method by freezing the model and fine tuning a linear layer on a downstream tempo estimation task.

Our key contributions are the following. 1- To the best of our knowledge, our work is the first to rely solely on an equivariance-based objective to learn musical audio representations. 2- We derive a simple loss function that formally prevents the network from collapsing on a trivial constant solution during training, without requiring any form of regularisation or negative sampling. 3- Our experiments show that it is possible to learn meaningful representations for tempo estimation by solely relying on equivariant self-supervision, and to achieve performance comparable with supervised methods on several benchmarks. 4- Our experiments demonstrate out-of-domain transferability of the representations learnt across multiple datasets. 5- As an added benefit, our method only requires moderate compute resources and therefore remains accessible to a wide research community. In order to further reproducibility we make code and pre-trained models available [1] .

2. BACKGROUND

2.1 Tempo Estimation

In many musical traditions, tempo is one of the fundamental characteristics of music. Tempo estimation was also one of the first tasks to have been explored in the field of MIR [24–28]. Historically, tempo estimation was performed by first extracting an onset detection function [29,30] and then perform tempo estimation via the computation of inter-onset intervals or some form of periodicity function from the onset detection curve [24, 26, 28, 31, 32].

With the introduction of deep learning, like most other MIR tasks, tempo estimation methods have gradually moved towards end-to-end learning where the deep neural network would jointly perform the feature extraction (e.g. onset detection curve) and tempo estimation [8–10]. This evolution brought a boost in performance so that the tempo estimation state of the art benchmark have been based on supervised deep learning for some years [8–10]. In line with this observation, we also adopt a deep neural network model described in detail in Section 3.1.

2.2 Siamese networks and trivial solutions

Most of the methods that have been shown to be successful at self-supervised representation learning for domains where data is dense and continuous, such as image and audio [3, 5, 33–35], are variations of the Siamese networks architecture [15]. Another trait these methods have in common is that they aim to maximise the similarity between the representations obtained from different transformations of a training sample. This problem, however, admits trivial solutions. For example, if the model produces a constant output irrespective of input, the representations of the two views would always be identical. Providing a mitigation strategy to prevent the model to collapse to a trivial solution is a major challenge in this family of methods.

A variety of strategies have been proposed to guard against the trivial solution collapse issue in the literature. SimCLR relies on negative sampling, where for each training sample all other samples in the mini-batch are considered as negatives [3]. This class of methods tend to benefit from large batch sizes. The Barlow twins approach prevents collapse by introducing an additional redundancy term to the training loss [36]. The authors also note that the method require smaller batch size than SimCLR. Deep cluster [37] and SwAV [33] prevents collapse by incorporating a clustering mechanism so that image features are not compared directly. Departing from the contrastive approach, BYOL relies only on positive pairs and prevents collapse by introducing asymmetry in the training scheme by updating the model parameters using only one transformed view of the input while the other view is used as a target [34]. In a similar line of work, the SimSiam authors note that the 'stop-gradient' operation introduces some asymmetry that is critical in preventing the collapse to trivial solutions [38]. Also relying on asymmetry, MoCo mitigates the trivial solution collapse by including momentum encoder [39].

The methods described above introduce trivial solution collapse mitigation strategies of varying degree of complexity that are found to work in practice even though their formulation still formally accepts trivial constant solutions. In Section 3.2 we propose a simple framework that formally does not admit a trivial constant solution, does not require negative samples [3, 5], large batches [3, 5], asymmetry [34, 35, 38, 39], non-differentiable operators [33] or stop gradients [38].

2.3 Equivariant objective

A representation $q(w)$ is invariant if it remains unchanged for a certain transformation k of the input w:

[1] https://github.com/Quint-e/equivariant-self-supervision-tempo

$$q(k \cdot w) = q(w) \quad (1)$$

In the case of equivariance, the representation reflects the transformation applied to the input:

$$q(k \cdot w) = k \cdot q(w) \quad (2)$$

Although the bulk of self-supervised methods developed so far rely on the notion of invariance to transformations of the training samples, we argue here that equivariance can provide a useful learning signal for forcing representations to capture meaningful properties of the input data. Recent works in the computer vision domain show that employing an equivariant objective in addition to more common invariant objective is beneficial [21, 22]. In this scenario, the equivariance is enforced against transformations such as rotation or scale. Similarly, equivariance is starting to be explored as an additional training objective in the video domain [40].

In the musical audio domain, SPICE relies on a composite training objective with an equivariance constraint to learn pitch representations from unlabelled data at its core [23]. SPICE generates two views of each training sample by applying pitch-shifting and aims to learn representations that are equivariant to the pitch of musical audio. SPICE requires strong regularisation to train effectively, which is achieved by adding an extra decoder network (discarded at inference time) and corresponding audio input reconstruction term to the loss.

In contrast with previous works, the method we propose here solely relies on a simple equivariance loss term. It does not include an invariance-based objective [21, 22] or any extra regularisation loss term [23].

3. METHODS

3.1 Model

Our model pipeline is a close adaptation of the Temporal Convolutional Network (TCN) architecture architecture introduced first in [41] for beat tracking and later extended to joint beat tracking and tempo estimation in [42]. From the audio downmixed to mono, we compute a magnitude spectrogram with a window and FFT size of 2048 samples and a hop size of 441 samples (i.e. 100 frames per second for audio sampled at 44100Hz). The FFT magnitude spectrogram is then mapped to a Mel Spectrogram with 81 bins ranging from 30 to 17,000Hz, and finally logarithmic compression is applied.

The Log Mel Spectrogram is then fed to a neural network constituted of two main building blocks and which architecture is depicted in Figure 1. The input is first fed through a batch normalisation layer followed by 3 convolutional blocks. It is then processed through 8 TCN layers with 16 filters each and geometrically increasing dilation rates from 2^0 to 2^7. Because we focus on learning tempo representations in this work, we drop the beat tracking branch and adapt the tempo branch architecture so that our network outputs a 16-d vector representation $\mathbf{h}$. All other design parameters are identical to [42]. We chose this TCN architecture because it is specifically designed to model temporal characteristics of music and has been shown to provide very competitive performance despite using very little parameters (33k).

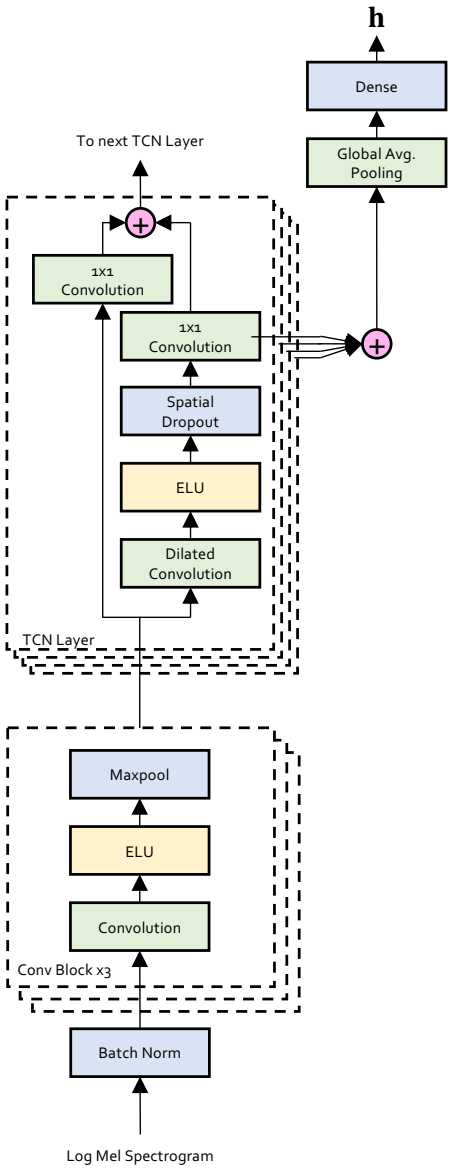


Figure 1. Temporal convolutional neural network (TCN) architecture. The Log Mel spectrogram is first processed through 3 convolution blocks and then through 8 TCN layers with 16 filters and geometrically increasing dilation rates from 2^0 to 2^7. The network outputs a 16-d vector embedding $\mathbf{h}$.

3.2 Training strategy

The objective of the training strategy described below is to learn representations that capture tempo information without having access to tempo annotations at training time. Starting from the observation that the time-stretching modifies the tempo of a music piece, we propose to use equivariance to time-stretching as a self-supervision objective to

learn tempo representations.

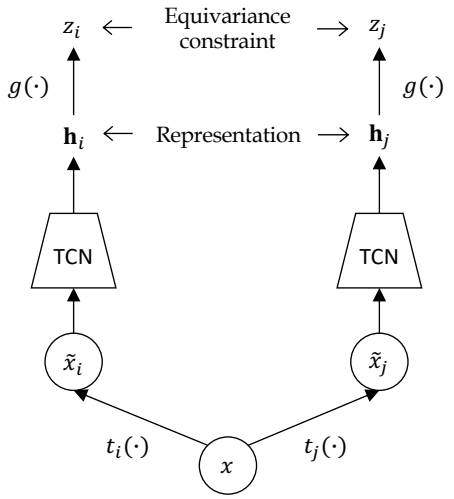


Figure 2. Equivariant self-supervision framework. Two distinct time-stretching transformations (t_i and t_j) are applied to a training sample x to obtain to correlated views ($\tilde{x}_i$ and $\tilde{x}_j$). The TCN network and projection head $g(\cdot)$ are trained to produce a pseudo-tempo scalar z that is equivariant to the time stretching transformation of the input. The projection head is discarded after training.

For a recording taken from the training set, let x be an excerpt of length l_x selected at random in the recording, with unknown tempo $y \in \mathbb{R}$. For each sample x, two views $(\tilde{x}_i, y_i)$ and $(\tilde{x}_j, y_j)$ are produced by applying time stretching transformations noted t_i and t_j respectively using Sox [2]. For each view, the time stretching rate α is drawn uniformly at random from $[1-r, 1+r]$ where r is a hyper-parameter to the training procedure. As a result, we obtain:

$$\tilde{x}_i = t_i(x), \qquad y_i = \alpha_i \cdot y \tag{3}$$

$$\tilde{x}_j = t_j(x), \qquad y_j = \alpha_j \cdot y \tag{4}$$

where the right hand side of eq. (3) and eq. (4) materialises the transformation of the tempo of each view.

In order to allow efficient batch processing at training time, we force the augmented views $\tilde{x}_i$ and $\tilde{x}_j$ to all have the same length l_x by cropping if they are longer and padding with zeros if shorter. In all our experiments we set l_x to be 600,000 audio samples (13.6s at 44.1 kHz).

Because the true tempo y is unknown, we propose to use a pseudo-tempo representation $z \in \mathbb{R}$ as the output of model at training time. Let $f(\cdot)$ be the transformation applied by the TCN and $g(\cdot)$ be a linear projection head, so that $g(f(x)) = z$. The objective is then to constrain the TCN and projection head stack to be equivariant to the time-stretching transformation of the input, so that:

$$g\left(f\left(t(x)\right)\right) = \alpha \cdot z \tag{5}$$

[2] http://sox.sourceforge.net

Since the two views are derived from the same training sample, it is trivial to show that the equivariance formulation expressed in eq. (5) yields:

$$\alpha_i \cdot z_j = \alpha_j \cdot z_i \tag{6}$$

In other words, the equivariance objective is met if eq. (6) is true. Based on this, we can derive the following training loss that is minimised when the equivariance objective is met:

$$\mathcal{L} = \left| \frac{z_i}{z_j} - \frac{\alpha_i}{\alpha_j} \right| \tag{7}$$

Note how this formulation does not allow the model to collapse on a trivial constant solution to minimise the loss. Producing a constant z value for any input does not yield a minimal loss because α values are drawn at random for every training sample, which means that the ratio $\frac{\alpha_i}{\alpha_j}$ varies for every pair of training sample views.

Other formulations of the loss function that are minimised when eq. (6) is true can be derived, but may allow trivial solutions. For example loss functions such as $\mathcal{L}' = |\alpha_i \cdot z_j - \alpha_j \cdot z_i|$ or $\mathcal{L}'' = \left| z_i - \frac{\alpha_i \cdot z_j}{\alpha_j} \right|$ admit a trivial optimal solution for $z_i = z_j = 0$.

3.3 Training parameters

In all our experiments we use a batch size of 16 samples, the Adam optimiser [43] with initial learning rate of 0.001. We pre-train the model for 20 epochs and fine-tune for 100 epochs.

With the aim to promote robustness against non tempo-related attributes of audio signals, we add the following optional random audio augmentations during pre-training: gain, polarity inversion, gaussian noise and SpecAugment frequency masking [44]. Note that our loss function remains unchanged whether or not we apply these augmentations.

3.4 Datasets

For self-supervised training we use the MagnaTagaTune dataset (MTT) [45]. It contains around 25k audio tracks but no tempo annotations.

For fine-tuning and evaluation we use datasets commonly used in the tempo estimation litterature that do contain tempo annotations. Namely we use the following datasets: GTZAN [46], Hainsworth [47] , Giantsteps [48, 49] and ACM Mirum [50].

3.5 Evaluation

After pre-training, we wish to evaluate the representations learnt by the TCN. The projection head $g(\cdot)$ is discarded and all the network weights frozen. We then attach a linear classification head with 300 units and softmax layer to represent the range [0,300] BPM, apply a smoothing window to the ground truth label similar to [42] and fine-tune using the cross-entropy loss.

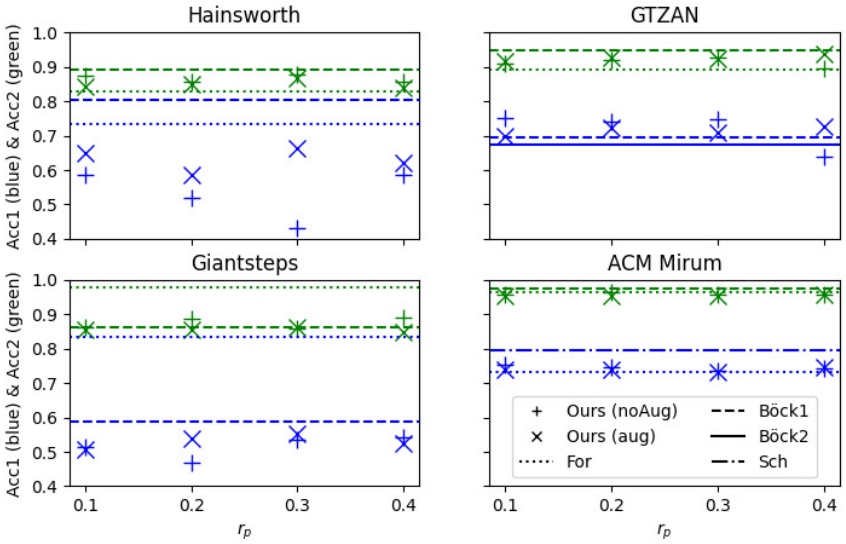


Figure 3. Performance metrics for our proposed method, compared against supervised benchmarks. We report both Accuracy 1 (in blue) and Accuracy 2 (in green). We report results for 8 pre-training conditions, which are combinations of using augmentations ("aug") or not ("noAug") and of $r_p \in \{0.1, 0.2, 0.3, 0.4\}$. In all cases the model is fine-tuned with a time-stretching of strength $r_f = 0.2$. Horizontal lines represent the supervised benchmarks "Böck1" [8], "Böck2" [42], "For" [51], "Sch" [9]. In the interest of legibility, for every dataset and every metric we only display the highest and lowest performing baselines.

In order to evaluate generalisation capability of our model, we perform cross-dataset evaluation. This means that the metrics we report are always computed on a dataset that has never been seen by the model during the pre-training or fine-tuning stages.

In order for our results to be comparable with existing literature, we report tempo performance metrics *Accuracy 1* and *Accuracy 2* scores with a $\pm 4\%$ tolerance [28]. *Accuracy 1* measures the accuracy of the model's prediction of the exact ground truth tempo, while *Accuracy 2* also allows for "octave errors" with factor $\{2, 3, \frac{1}{2}, \frac{1}{3}\}$. We leave further evaluation considerations for future work [52].

4. EXPERIMENTS

4.1 Robustness against trivial solutions

In a preliminary experiment, we ran the pre-training with alternative losses $\mathcal{L}'$ and $\mathcal{L}''$, as defined in Section 3.2. It systematically collapsed to a trivial solution $z \approx 0$. Conversely, we use the loss function $\mathcal{L}$ described in Eq. 7 in all the experiments reported in this paper and did not observe collapse, which confirms its robustness against trivial constant solutions.

4.2 Influence of pre-training augmentation parameters

Figure 3 shows the performance metrics on evaluation datasets for supervised baselines and our method after pre-training on the MTT dataset and cross-dataset fine-tuning and evaluation. We report results for 8 pre-training conditions, which are combinations of using audio augmentations ("aug") or not ("noAug"), and of the strength of time-stretching during pre-training $r_p \in \{0.1, 0.2, 0.3, 0.4\}$. In all cases the model is fine-tuned with a time-stretching of strength $r_f = 0.2$ (see section 4.3).

It appears that adding augmentations yields no notable performance benefit except on Acc1 on the Hainsworth dataset. We hypothesise this is because the TCN architecture already has a strong inductive bias towards temporal and rhythmic structures, which makes it robust against potentially confounding attributes of the audio signal.

The choice and strength of input transformations has been shown to have an impact on the performance of contrastive learning [20]. Intuitively, one expects that very gentle augmentations may not allow to learn robust representations, on the other hand too strong an augmentation may lose semantic content and therefore not allow to learn at all. In our experiments, we observe that the performance tends to dip slightly at the extreme ends of the range of values for r_p. However, it is interesting to note that we do not observe a dramatic degradation of performance even though the transformations applied then (e.g. $r_p = 0.4$) are musically extreme.

Overall these results suggest that the pre-training phase is fairly robust to the choice of time-stretching parameters and to the absence of audio augmentation.

4.3 Influence of fine-tuning augmentation parameters

At the fine-tuning stage, tempo estimation is formulated as a multiclass classification problem, where each class corresponds to a tempo value. Because it is likely that the datasets used for fine-tuning may not contain training sam-

Method	r_f	Hainsworth		GTZAN		Giantsteps		ACM Mirum	
		Acc1	Acc2	Acc1	Acc2	Acc1	Acc2	Acc1	Acc2
Ours	0.0	0.604	*0.838*	*0.691*	0.887	0.512	0.809	0.704	0.943
Ours	0.1	0.586	0.824	*0.719*	0.881	0.456	0.791	*0.757*	*0.965*
Ours	0.2	0.518	*0.856*	*0.741*	*0.919*	0.470	*0.886*	*0.747*	*0.965*
Ours	0.3	0.550	*0.829*	**0.785**	*0.921*	0.438	0.846	0.700	0.958
Ours	0.4	0.541	*0.829*	*0.778*	*0.926*	0.472	*0.884*	0.724	0.952
Schreiber [9]	-	0.770	0.842	0.694	0.926	0.730	0.893	**0.795**	0.974
Foroughmand [51]	-	0.734	0.829	0.697	0.891	**0.836**	**0.979**	0.733	0.965
Böck 1 [8]	-	**0.806**	**0.892**	0.697	**0.950**	0.589	0.864	0.741	**0.976**
Böck 2 [42]	-	-	-	0.673	0.938	0.764	0.958	0.749	0.974

Table 1. Influence of time stretching parameters during Fine-tuning, and comparison to supervised baselines. In all cases, our model is pre-trained on MTT with $r_p = 0.2$ and no audio augmentations. Highest performance for each metric and dataset shown in bold. Metrics where our model outperforms at least one of the baselines in italics.

ples for each tempo in the full BPM range considered, we also apply a time-stretching augmentation to increase the range of tempi seen during fine tuning.

Table 1 summarises the results using a model pre-trained with $r_p = 0.2$ and no invariant augmentations, for a range of fine-tuning time-streching augmentation strength r_f. As expected, it appears that applying some time-stretching ($r_f > 0$) generally improves performance over not applying any ($r_f = 0$). We also note that the optimal value varies from one dataset to the next. For example results seem to be optimal for relatively large augmentation strength on the GTZAN dataset while they would be optimal for smaller values on ACM Mirum.

4.4 Comparison to supervised benchmarks

Figure 3 shows supervised benchmarks as horizontal lines. Our proposed method's Accuracy 2 performance is comparable with supervised benchmarks on all datasets. Accuracy 1 performance is more inconsistent. It lags behind supervised methods on Hainsworth and Giantsteps datasets, while it is comparable with supervised benchmarks on ACM Mirum. Notably, our method outperforms all supervised baselines in Accuracy 1 on GTZAN. Similar conclusions can be drawn from the results shown in Table 1, under different fine-tuning configurations.

Note that we evaluate our models on datasets that have never been seen during pre-training or fine-tuning and still observe performance generally competitive with supervised benchmarks. Also taking into account that we only fine-tune a linear layer, this result indicates a promising degree of robustness of the representation learnt during pre-training against domain shift.

5. CLOSING WORDS

In this work, we introduced an approach to use equivariance as a self-supervision signal to learn audio tempo representations from unlabelled data. We derive a simple loss function that prevents the network from collapsing on a trivial solution during training, without requiring any form of regularisation or negative sampling. Our experiments show not only that it is possible to learn meaningful representations for tempo estimation by solely relying on equivariant self-supervision, but also demonstrate that we can achieve performance comparable with supervised methods on most benchmarks. We also show that the representations learnt exhibit promising robustness against pre-training and fine-tuning hyper-parameters as well as against domain shift. As an added benefit, our method only requires moderate compute resources by making use of a small model and not requiring large batch sizes, therefore keeping it accessible to a wide research community.

We believe these results open a number of interesting avenues for future work. This paper is focused on tempo estimation but there is potential to extend our investigation to other MIR tasks revolving around rhythmic properties such as beat tracking, metrical structure estimation or rhythm pattern identification. In addition, because no annotated data is required for pre-training, there is potential for investigating applications to low resource musical genres or genres underserved by traditional and current supervised methods. Since our loss function is very simple, it could also be added as an extra objective in SSL frameworks for learning general music representations, at a moderate cost of increased complexity. Last but not least, while invariance typically used in contrastive learning is naturally connected with classification problems, equivariant self-supervision is well suited to tasks that can be formulated as regression problems. We therefore believe there is potential to explore many more applications of equivariant SSL in the music domain and beyond.

6. REFERENCES

[1] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *arXiv preprint arXiv:1910.10683*, 2019.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,"

arXiv:1810.04805, Oct. 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[3] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," *arXiv:2002.05709*, Feb. 2020. [Online]. Available: http://arxiv.org/abs/2002.05709

[4] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[5] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," *arXiv preprint arXiv:2103.09410*, 2021.

[6] D. Yao, Z. Zhao, S. Zhang, J. Zhu, Y. Zhu, R. Zhang, and X. He, "Contrastive Learning with Positive-Negative Frame Mask for Music Representation," *arXiv preprint arXiv:2203.09129*, 2022.

[7] H. Zhao, C. Zhang, B. Zhu, Z. Ma, and K. Zhang, "S3T: Self-Supervised Pre-training with Swin Transformer for Music Classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 606–610.

[8] S. Böck, F. Krebs, and G. Widmer, "Accurate tempo estimation based on recurrent neural networks and resonating comb filters," in *International Society for Music Information Retrieval (ISMIR) Conference*, 2015.

[9] H. Schreiber and M. Müller, "A Single-Step Approach to Musical Tempo Estimation Using a Convolutional Neural Network." in *International Society for Mu- sic Information Retrieval (ISMIR) Conference*, 2018, pp. 98–105.

[10] S. Böck and M. E. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 12–16.

[11] X. Sun, Q. He, Y. Gao, and W. Li, "Musical Tempo Estimation Using a Multi-scale Network," *arXiv preprint arXiv:2109.01607*, 2021.

[12] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[13] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-supervised learning of discrete speech representations," *arXiv preprint arXiv:1910.05453*, 2019.

[14] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[15] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. IEEE, 2006, pp. 1735–1742.

[16] D. Jiang, W. Li, M. Cao, W. Zou, and X. Li, "Speech simclr: Combining contrastive and reconstruction objective for self-supervised speech representation learning," *arXiv preprint arXiv:2010.13991*, 2020.

[17] L. Wang, P. Luc, Y. Wu, A. Recasens, L. Smaira, A. Brock, A. Jaegle, J.-B. Alayrac, S. Dieleman, and J. Carreira, "Towards learning universal audio representations," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[18] P. Manocha, Z. Jin, R. Zhang, and A. Finkelstein, "CDPAM: Contrastive learning for perceptual audio similarity," *arXiv preprint arXiv:2102.05109*, 2021.

[19] S. Srivastava, Y. Wang, A. Tjandra, A. Kumar, C. Liu, K. Singh, and Y. Saraf, "Conformer-Based Self-Supervised Learning for Non-Speech Audio Tasks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[20] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, "What makes for good views for contrastive learning?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 6827–6839, 2020.

[21] R. Dangovski, L. Jing, C. Loh, S. Han, A. Srivastava, B. Cheung, P. Agrawal, and M. Soljačić, "Equivariant Contrastive Learning," *arXiv preprint arXiv:2111.00899*, 2021.

[22] Y. Wang, J. Zhang, M. Kan, S. Shan, and X. Chen, "Self-supervised equivariant attention mechanism for weakly supervised semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 12 275–12 284.

[23] B. Gfeller, C. Frank, D. Roblek, M. Sharifi, M. Tagliasacchi, and M. Velimirović, "SPICE: Self-supervised pitch estimation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1118–1128, 2020.

[24] E. D. Scheirer, "Tempo and beat analysis of acoustic musical signals," *The Journal of the Acoustical Society of America*, vol. 103, p. 588, 1998.

[25] A. T. Cemgil, B. Kappen, P. Desain, and H. Honing, "On tempo tracking: Tempogram Representation and Kalman filtering," *Journal of New Music Research*, vol. 29, no. 4, pp. 259–273, 2000.

[26] S. Dixon, "Automatic extraction of tempo and beat from expressive performances," *Journal of New Music Research*, vol. 30, no. 1, pp. 39–58, 2001.

[27] M. Alonso, B. David, and G. Richard, "A study of tempo tracking algorithms from polyphonic music signals," in *Proceedings of the 4th. COST 276 Workshop*, 2003.

[28] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.

[29] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A tutorial on onset detection in music signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 13, no. 5, pp. 1035–1047, 2005.

[30] S. Böck and G. Widmer, "Maximum filter vibrato suppression for onset detection," in *Proc. of the 16th Int. Conf. on Digital Audio Effects (DAFx)*, 2013.

[31] M. E. Davies and M. D. Plumbley, "Causal Tempo Tracking of Audio." in *International Society for Music Information Retrieval (ISMIR) Conference*, 2004.

[32] G. Peeters, "Time variable tempo detection and beat marking," in *Proceedings of the ICMC*, 2005.

[33] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9912–9924, 2020.

[34] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, and M. Gheshlaghi Azar, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 271–21 284, 2020.

[35] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Byol for audio: Self-supervised learning for general-purpose audio representation," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[36] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, "Barlow twins: Self-supervised learning via redundancy reduction," *arXiv preprint arXiv:2103.03230*, 2021.

[37] M. Caron, P. Bojanowski, A. Joulin, and M. Douze, "Deep clustering for unsupervised learning of visual features," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 132–149.

[38] X. Chen and K. He, "Exploring Simple Siamese Representation Learning," *arXiv preprint arXiv:2011.10566*, 2020.

[39] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[40] S. Jenni and H. Jin, "Time-equivariant contrastive video representation learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9970–9980.

[41] E. P. MatthewDavies and S. Böck, "Temporal convolutional networks for musical audio beat tracking," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[42] S. Böck, M. E. Davies, and P. Knees, "Multi-task learning of tempo and beat: learning one to improve the other," in *20th International Society for Music Information Retrieval Conference (ISMIR 2019)*, 2019.

[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[44] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[45] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of Algorithms Using Games: The Case of Music Tagging." in *ISMIR*, 2009, pp. 387–392.

[46] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[47] S. W. Hainsworth and M. D. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP Journal on Applied Signal Processing*, vol. 2004, pp. 2385–2395, 2004.

[48] P. Knees, A. Faraldo Perez, H. Boyer, R. Vogl, S. Böck, F. Horschlager, and M. Le Goff, "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

[49] H. Schreiber and M. Müller, "A Crowdsourced Experiment for Tempo Estimation of Electronic Dance Music." in *ISMIR*, 2018, pp. 409–415.

[50] G. Peeters and J. Flocon-Cholet, "Perceptual tempo estimation using GMM-regression," in *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, 2012, pp. 45–50.

[51] H. Foroughmand and G. Peeters, "Deep-rhythm for tempo estimation and rhythm pattern recognition," in *International Society for Music Information Retrieval (ISMIR)*, 2019.

[52] H. Schreiber, J. Urbano, and M. Müller, "Music Tempo Estimation: Are We Done Yet?" *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.

HOW MUSIC FEATURES AND MUSICAL DATA REPRESENTATIONS AFFECT OBJECTIVE EVALUATION OF MUSIC COMPOSITION: A REVIEW OF THE CSMT DATA CHALLENGE 2020

Yuqiang Li[1] **Shengchen Li**[1] **George Fazekas**[2]

[1]Xi'an Jiaotong-Liverpool University
[2]Queen Mary University of London
{yuqiang.li19@student., shengchen.li@}xjtlu.edu.cn,
g.fazekas@qmul.ac.uk

ABSTRACT

Tools and methodologies for distinguishing computer-generated melodies from human-composed melodies have a broad range of applications from detecting copyright infringement through the evaluation of generative music systems to facilitating transparent and explainable AI. This paper reviews a data challenge on distinguishing computer-generated melodies from human-composed melodies held in association with the Conference on Sound and Music Technology (CSMT) in 2020. An investigation of the submitted systems and the results are presented first. Besides the structure of the proposed models, the paper investigates two important factors that were identified as contributors to good model performance: the specific music features and the music representation used. Through an analysis of the submissions, important melody-related music features have been identified. Encoding or representation of the music in the context of neural network modes are found noticeably impacting system performance through an experiment where the top-ranked system was re-implemented with different input representations for comparison purposes. Besides demonstrating the feasibility of developing an objective music composition evaluation system, the investigation presented in this paper also reveals some important limitations of current music composition systems opening opportunities for future work in the community.

1. INTRODUCTION

With the rapid development of AI, automatic music composition systems are considered to approach the quality of human-composed melodies in their output under certain conditions. The Conference on Sound and Music Technology (CSMT) organised a data challenge in 2020 to investigate how to tell apart human-composed melodies from computer-generated melodies, where participants were required to develop an algorithm (or a system) that can distinguish computer-generated melodies from human-composed ones. The submitted algorithms can potentially be used to prevent computer systems from being accused of music copyright infringement, and to objectively measure the performance of different algorithmic composition systems.

The primary task in this data challenge was to develop a system to identify human-composed melodies in a dataset where they are mixed with computer-generated melodies. The generated melodies come from several state-of-the-art music generation frameworks [1], including Variational Auto-Encoder (VAE) [2], Transformer [3] and Generative Adversarial Network (GAN) [4]. A training set with only generated melodies was released first, followed by an evaluation set with human-composed melodies mixed in. Two datasets were used for training the music generation systems separately: Bach Chorales [1] in Music21 [5] and HookTheory [2]. The generated and composed melodies thus followed two styles associated with Bach and more generally the pop genre.

The rankings of the participants were determined by the AUC score in ROC tests, based on the task of identifying human-composed melodies. 7 teams participated in the data challenge with a total of 14 submitted systems, covering various types of algorithms. For instance, rule-based system [6], LSTM (Long short-term memory) [7–9], AE (auto-encoder) [10] and more conventional SVM (support vector machine) [11]. The top-ranked submission by Li *et. al* [7] obtained an AUC score of 0.88, which demonstrated the feasibility of using a computer to identify the music source, at least in a specific context of the two styles by distinguishing human compositions from generated melodies.

The results of the data challenge are further investigated in three ways: style, pitch feature and music representation. The sub-rankings regarding the two music styles are also compared, though little difference was observed. However, the system performance shows significant differences in terms of different pitch features in melodies and music representation methods. Moreover, the paper also presents a revised version of the top-ranked system with

© Y. Li, S. Li, and G. Fazekas. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Y. Li, S. Li, and G. Fazekas, "How Music Features and Musical Data Representations Affect Objective Evaluation of Music Composition: A Review of the CSMT Data Challenge 2020", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] https://web.mit.edu/music21/doc/moduleReference/moduleCorpusChorales.html
[2] https://www.hooktheory.com/

different input representations to discuss how music representation affects the identification of melodies.

The identification of computer-generated melodies could be used to ensure that no copyright is claimed for fully-automated computer-generated melodies. [12]. Moreover, measuring similarity between human and computer-generated melodies could become a component of Generative Adversarial Networks (GAN) and similar frameworks for automatic and assistive composition, as well as reduce the subjective bias in the evaluation of music generation systems.

The remainder of the paper is organised as follows: we first review the CSMT 2020 data challenge including submissions and results. The performance differences between submissions regarding different musical features are then investigated. Two experiments on the impacts of music representations are presented followed by a brief conclusion.

2. DATA CHALLENGE

2.1 Task Definition and Organisation

The proposed task for the data challenge was to distinguish human-composed melodies from computer-generated melodies, all in the form of 8-bar single-track MIDI files. A training dataset was released first, containing only computer-generated melodies. The evaluation dataset, with equal numbers of human-composed melodies and generated melodies, was released without the associated labels one month before the final submission deadline. The final results were ranked using the AUC score in ROC test, which identifies human-composed melodies from computer-generated ones.

Time	Event
July 15th, 2020	Release of the development dataset
August 15th, 2020	Release of the evaluation dataset
September 15th, 2020	Deadline of submission (prediction, model and report)
October 20th, 2020	Submission review finished
November 4th, 2020	Result announcement

Table 1: Timeline of the CSMT 2020 Data Challenge

2.2 Dataset

A detailed specification of the dataset, especially the training dataset, can be found in [1]. The components of evaluation set are shown in Table 2 to provide a clearer context of the results presented in this paper. Notice that according to [1], 95% of the human-composed melodies in the evaluation dataset are randomly sampled from those that were used to train the three types of generative models.

2.3 Baseline System

An Auto-Encoder (AE) neural network was used as the baseline system of the challenge, whose source code is available on the official website. Figure 1 plots the model architecture. The model uses a bottleneck structure with the cross entropy loss.

	MTrans	MVAE	MNet	Human	Total
Bach	600	200	200	1000	2000
Pop	600	200	200	1000	2000
Total	1200	400	400	2000	4000

Table 2: The evaluation dataset used in the challenge. "MTrans", "MVAE", "MNet" are short for Music Transformer [13], MusicVAE [2] and MidiNet [4], respectively. Music styles "Bach" and "Pop" are listed separately.

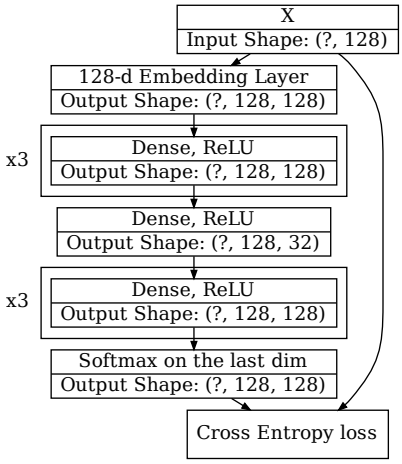


Figure 1: Baseline Auto-Encoder Network Architecture

The baseline model was trained with only real melodies from the Nottingham Folk Tunes dataset [3], consisting of 1,200 American and British songs. The chord information provided in the dataset was not used. The baseline system characterises features of human-composed melodies with an auto-encoder, where computer-generated melodies will result in a high reconstruction loss.

2.4 Overview of the Submitted Systems

For each submitted system, participants were allowed to provide multiple predictions based on different settings or parameters of a model. The challenge received 14 sets of predictions submitted by 7 participants. To simplify references to the submitted systems, aliases are introduced in Table 3 to represent the 8 models. In the table, the presented AUC score is the highest among all predictions of each submitted system.

Around half of the submitted systems outperformed the baseline, these are listed in Table 3. Most participants adopted Deep Neural Networks (DNN), but conventional ML algorithms and rule-based methods were also observed.

There are three main types of methodologies among the submissions: outlier detection, binary classification and heuristics. The outlier detection systems [10] and [14] only learn the feature distribution of the samples of one class

[3] https://github.com/jukedeck/nottingham-dataset

and considers the other class as outliers. Binary classifiers [7–9, 11] use both positive and negative samples for training. The rule-based system [6] uses a set of heuristic rules to test the input melody.

The outlier detection method adopts the idea of Task 2 from the DCASE data challenge 2020. With this configuration, an auto-encoder system is trained in an unsupervised manner for either computer-generated melodies or human compositions. The samples from the outlier class are expected to result in a high reconstruction loss. Besides the baseline system, Ding & Ma [10] and Yu *et al.* [14] used this idea and were ranked 4th and 8th respectively.

The best-performing system [7], Guo *et al.* [8], Xia *et al.* [9], and Wang *et al.* [11] adopted the idea of binary classification. External data, such as Wikifonia, Nottingham and Midi Man [4] . was used to train the classifiers with both positive and negative samples.

The only rule-based system [6] has ranked 2nd as a team, and 3rd as a system, outperforming quite a few ML and DNN systems. In summary, this system takes an 8-bar melody as input and finds the most possible key centers for each bar. The unusual key changes are then counted for every 2 consecutive bars. The key center of a bar is determined by finding out a natural major or harmonic minor scale among all possible scales such that this scale contains most of the notes in this bar. Unusual key changes are defined as any modulations to a distant key, which means the tonic of the new key is 3 fifths or more apart from the original tonic. The success of the system points out that there is still a long way to go for computer music generation systems in learning particular music theories.

3. IMPACT OF STYLE

The evaluation dataset contains equal number of Bach-style and Pop-style melodies. The model performances in each of these two subsets are listed as AUC scores in the column *Bach* and *Pop* of Table 3. Compared with the overall AUC scores, there is no strong difference shown in the model performances given the different style subsets. An exception is that the top-ranked and the second-ranked systems by Li *et. al* are clearly better at evaluating Bach-style melodies than Pop-style ones with differences in the AUC score more than 0.1.

4. IMPACT OF PITCH FEATURES

4.1 Pitch Features

Pitch features are commonly used for the evaluation of generative music systems [15], specifically on melodies. Three pitch features of melodies are proposed for investigating how pitch features impact the performance of the submitted systems: Interval Mean (IM), Interval Standard Deviation (ISD) and Major-scale-rate Standard Deviation (MSD).

Suppose $\mathbf{X}$ is a melody denoted by a MIDI pitch sequence without duration information $[p_1, p_2, \ldots, p_n], 0 \leq p \leq 127$, the first-order forward difference of the pitch sequence $\mathbf{X}$ can be represented as $\Delta\mathbf{X} = [p_2 - p_1, p_3 - p_2, \ldots, p_n - p_{n-1}]$, each of which is an signed integer.

IM and ISD are defined as

$$\mathrm{IM}(\mathbf{X}) := \mathbb{E}[\Delta\mathbf{X}] \tag{1}$$

$$\mathrm{ISD}(\mathbf{X}) := \mathrm{Std}\left(\Delta\mathbf{X}\right) \tag{2}$$

where "Std" is the standard deviation.

Major Scale Rate $\mathrm{MSR}(\mathbf{X})$ is defined as a function that maps a melody $\mathbf{X}$ to a 12-dimensional vector $[c_0, c_1, \ldots, c_{11}]$, where c_i is the number of notes in $\mathbf{X}$ that belong to the natural major scale with pitch class i as the tonic (namely, C major, C♯ major, ..., B major). The vector is normalised by dividing all components by their sum so that $\sum_i \mathrm{MSR}(\mathbf{X})_i = 1$. Major-scale-rate Standard Deviation (MSD) is obtained by

$$\mathrm{MSD}(\mathbf{X}) := \mathrm{Std}\left(\mathrm{MSR}(\mathbf{X})\right). \tag{3}$$

IM approximates the overall tendency of a pitch sequence. For example, if $\mathrm{IM}(\mathbf{X}) > 0$, the melody $\mathbf{X}$ must end at a pitch higher than the starting pitch [5] . ISD measures the unevenness of a melody. A melody with a higher ISD usually sounds more stochastic and chaotic. MSD measures the extent to which a melody is concentrated on a specific (natural major) scale.

4.2 Impact of Pitch Features on Model Performance

For each proposed feature, the following procedure is used to compare the performance of the submitted models given melodies with different IM, ISD and MSD.

For each pitch feature, the feature values are calculated for all melodies in the evaluation dataset. According to feature values, melodies are grouped into 30 bins that equally divide the whole range of feature values. In each bin, the mean absolute error (MAE) between the predicted score and the label of the melody (0 for generated and 1 for human-composed) was used as the measurement of system performance.

4.2.1 Interval Mean

Figure 2a compares how the MAEs were distributed on different IM ranges for different submissions. A dashed KDE plot was also used to show IM distribution for all melodies in the dataset. The majority of the dataset has a IM value in the range of $[-0.75, 0.75]$.

Among all the models, as plotted in the Figure 2a, the two LSTM-based models submitted by Li *et al.* (the blue and the orange curves) produced comparatively lower errors than the others. Xia (the pink curve) submitted another LSTM-based model, with a similar performance to

[4] https://www.reddit.com/r/datasets/comments/3akhxy/the_largest_midi_collection_on_the_internet/

[5] Numerically, IM equals the starting pitch subtracted from the ending pitch.

Rank	Submission Name	Alias	Model	Dataset	Flags	Bach	Pop	All
1	You_Li_NYU_Zhuowen_Lin_GATech_uni [7]	Li-uni	LSTM	Midi Man (1,000), CSMT (6,000) [1]	DCA	0.94	0.83	0.88
2	You_Li_NYU_Zhuowen_Lin_GATech_bi [7]	Li-bi	Bi-LSTM			0.89	0.71	0.80
3	Yang_Deng_Netease [6]	Deng	Statistics	(None)	S..	0.77	0.76	0.76
4	Mingshuo_Ding_PKU_Yinghao_Ma_CMU [10]	Ding	ALBERT AE	Fake only, CSMT (6,000)	DRA	0.70	0.67	0.68
5	Jingyue_Guo_PATech [8]	Guo	Bi-LSTM	Wikifonia (6,675), CSMT (6,000)	DC.	0.61	0.65	0.63
-	Baseline	Baseline	MLP-AE	Real only, Nottingham (1,034)	DR.	0.58	0.63	0.61
6	Yiting_Xia [9]	Xia	LSTM	Real* (5,742), CSMT (6,000)	DCA	0.57	0.64	0.60
7	Jiaxing_Yu_ZJU [14]	Yu	CNN-AE	Fake only, MusicVAE (4,000)	DR.	0.62	0.48	0.56
8	Yuxiang_Wang_BIT [11]	Wang	SVM	Nottingham (1,034), CSMT (6,000)	MC.	0.55	0.47	0.51

*: From multiple data sources.
Flag 1: **D**: DNN, **M**: Traditional ML, **S**: Statistical. Flag 2: **C**: Classifier-based, **R**: Reconstruction-based. Flag 3: **A**: Dataset augmentation was performed.

Table 3: A list of all submissions with AUC scores for the Bach subset, Pop subset and the entire set presented.

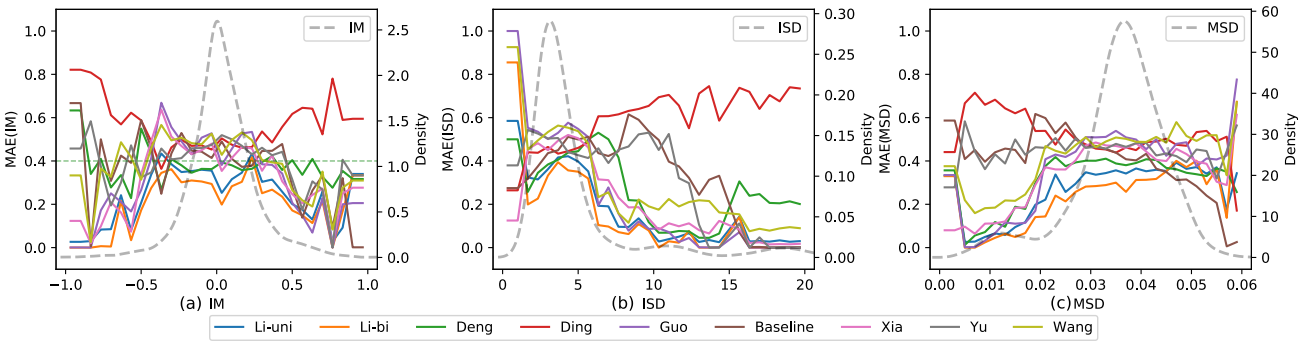


Figure 2: Model MAE on different features. The feature KDE is plotted in a dashed line style to characterise the distribution of music features.

the previous two when IM > 0.5, but higher errors when IM < 0.5. Deng's system (the green curve) makes prediction by counting the bars that modulate to a distant key. As a visual aid, a horizontal dashed green line was plotted to show that the MAE of Deng's model stays around 40%. Especially for IM in $[-0.5, 0.5]$, Deng's model surpassed a few DNN models with low errors only second to Li's models. This model also displayed a stable and consistent performance even with the melodies that have extreme IM values.

4.2.2 *Interval Standard Deviation*

ISD roughly measures the evenness of the melody. Melodies with low ISD are extremely stable and repetitive thus harder to distinguish. Melodies with high ISD tend to be stochastic and more random, which are relatively simpler to distinguish.

Figure 2b illustrates the MAE distributions of different models in different ISD bins. The leftmost highlighted area indicates that most DNN-based models (LSTM, CNN and AE) have poor performance when ISD is within $(0, 2.5]$. However, on the right hand side of Figure 2b, when it comes to melodies with ISD in $[15, 20]$ (which is extremely high), the system performance improves. This evidence shows that most models have successfully learned to recognise some melodies as the outliers if they are stochastic and contain many jumping intervals.

For melodies with higher ISD (ISD > 5), where neighbour pitches are less likely to be fully covered by the convolutional kernel, Yu's CNN-based model (grey curve) produced noticeably more errors than the rest of the models. Ding's BERT-based model (red curve) has an even higher error when ISD is high, indicating that the model is possibly over-fitted with poor generalisation.

4.2.3 *Major-scale-rate Standard Deviation*

A melody with a higher MSD tends to be more diatonic and less chromatic. Figure 2c reveals the performance of submissions regarding different MSD ranges. With lower MSD value (MSD < 0.02), all the LSTM models [7–9] retained relatively lower MAEs than the others, hinting some potential relationship between recognising atonal melodies and the characteristics of LSTM network itself.

Deng's proposed rule-based system, designated to inspecting the tonal properties of a melody also showed a low-error curve, only behind the top-ranked LSTM systems. The stability and strong generalisation property of Deng's system enabled its survival of the rare test cases. For instance, around MSD $= 0.06$, (very high, the melody being too diatonic), some ML or DNN models suddenly collapsed with high errors while Deng's system only had a slight increment in error, with the absolute value remaining less than 0.4.

IM, ISD and MSD can effectively reveal the melody characteristics and hence can be used to explain the model performance using different aspects. The distributions of these features can guide the training process of a music generation system, or as a series of a-priori weights to refine the predictions in a discriminative model according to how common the melody is.

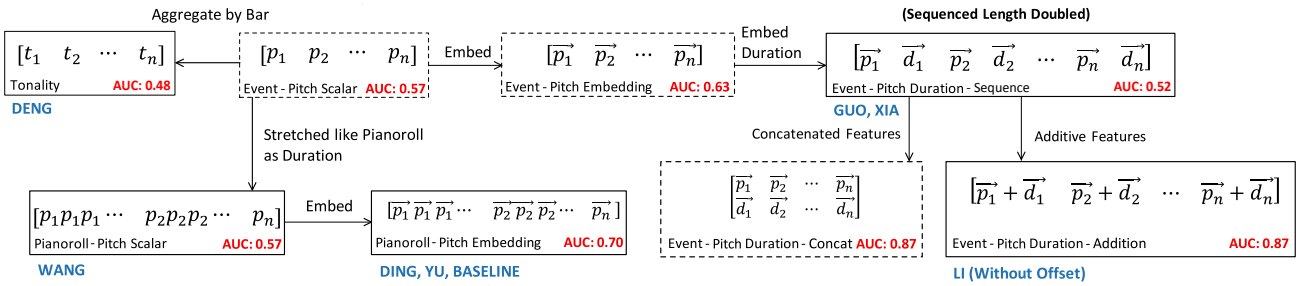


Figure 3: Representation relationship diagram. The edge label denotes the transformation from a representation to another. The aliases of the participants who used a certain type of representation are listed on the left beneath the box, uppercase bold. The AUC score in the bottom right corner refers to the results of the experiment mentioned in Section 5.1.

5. IMPACT OF MUSIC REPRESENTATION

From the submissions, the differences in music representation are focused on two aspects: the input representation and the metric unit of time. Various music representations can be distinguished mainly by what features of notes are selected and how they are organised in structural data. The metric unit of time refers to the minimal time unit to denote the music data. In this study, the top-ranked system by Li *et al.* [7], an LSTM-based binary classifier, has been slightly revised to adopt different types of music representations in order to investigate the impact of music representation on the same task. All re-implemented models are trained 50 epochs using the same training dataset used by Li, (as listed in Table 3) and are evaluated on the evaluation dataset. ROC AUC is used as the evaluation strategy.

5.1 Melody Representations

Three aspects of a melody representation will be discussed in section 5.1: the evenness along time steps, the subsets of features regarding notes and the method of feature fusion.

Evenness of time steps (ETS) The term *Pianoroll* is used if the duration is implicitly represented in the form of pianoroll. One step in these always stands for a constant length of time. Otherwise, the term *event* will be used.

Subset of note features (SNF) Features regarding notes include pitch, duration, velocity [6], *etc*. Particularly in this challenge, only "*Pitch*" and "*Duration*" are used.

Feature fusion method (FFM) Feature fusion is usually achieved in the following three ways if more than one note features is used:

1. Use separate events in consecutive time steps (sequentially), *e.g.*, $\left[\vec{p_1}, \vec{d_1}, \vec{p_2}, \vec{d_2}, \cdots\right]$
2. Concatenate the embedded vectors of different features of the same note into a higher-dimensional vector, *e.g.*, $\left[\vec{p_1} \| \vec{d_1}, \vec{p_2} \| \vec{d_2}, \cdots\right]$
3. Add the embedded vectors of different features of the same note, *e.g.*, $\left[\vec{p_1} + \vec{d_1}, \vec{p_2} + \vec{d_2}, \cdots\right]$

where $\vec{p_i}$ and $\vec{d_i}$ are the embedded vectors (in the same dimension) of the pitch and duration event token for the i-th note in the melody. The representation methods will be named in a format of *ETS-SNF-FFM* as shown in Figure 3.

The only exception of representation is *Tonality* representation, as used by Deng *et. al*. Deng's system represents the input data via the tonality distributions of all bars in a melody [6]. The tonality distribution, denoted by the set of all possible key candidates, summarises the tonal characteristics at a bar level rather than a single note. This leads to the representation name *Tonality* representation.

All the melody representations used by the submitted models can be categorised into 5 types according to the three aspects, plus the extra *tonality*. Besides these that have been used, another three representations were added to better illustrate the relationship among different representations, namely *Event-Pitch-Scalar*, *Event-Pitch-Embedding* and *Event-PitchDuration-Concat*. The former two contain only pitch information, differentiated by whether embedding is used. The *Event-PitchDuration-Concat* representation concatenates pitch vectors and duration vectors instead of element-wise adding. In Li's model [7], the embedding dimension is chosen to be 128.

Figure 3 reveals how a melody representation is related to another through a certain transformation. For instance, starting from the most simple pitch-only *Event-Pitch-Scalar* representation, grouping the pitches by bar gives the *Tonality* representation. Stretching the sequence according to the note position and duration will yield a pianoroll. Using embedding will produce *Event-Pitch-Embedding* and *Pianoroll-Pitch-Embedding*.

5.2 Metric Unit of Time in Music Representation

The metric unit of time in a representation refers to the minimal time unit that the positions and durations of all notes in a melody are always the multiples of. This concept is also known as midi resolution. Quantisation is the procedure that re-samples a midi file to a specific resolution. 3 submissions chose a specific metric unit of time for pre-processing all the melodies before the model input. The results suggest that the unit of metric should be carefully determined according to the specific model. Generally, a larger metric unit will result in fewer rhythmic patterns and time steps, especially for pianoroll representations. This reduces the difficulty in modelling rhythmic and temporal features at the cost of ignoring the nuances.

[6] This paper will not discuss the velocity feature since it was not involved in the data challenge.

As some officially provided melodies were generated without clear beats due to the flaws of some generation systems, the choice of metric unit can be vital in this challenge.

12th of a quarter note is a commonly used metric unit of time, where durations are multiples of 8th notes and 8th triplets, as default used by the Python package `music21` when opening midi files [5]. 12th was adopted by Li [7] and Xia [9]. 24th was adopted by Yu [14] and Guo [8], as used in the Python package `pypianoroll` [16]. Ding's [10] and Wang's [11] submissions used only 4th, a much wider step, which resulted in much lower AUC scores. However, Deng's system [6] did not use the note duration. These facts and the rankings suggest that the choice of larger metric unit could potentially increase the task difficulty of this challenge due to the loss of rhythmic details.

5.3 Effects of Melody Representations on Model Performance

Another experiment was conducted to support the assumption that melody representation impacts the model performance of this task. The top-ranked system by Li, an LSTM-based binary classifier, was re-implemented so that all the 8 representations aforementioned can be used as the model input. For all the re-implemented and re-trained models, the dataset specification remained unchanged as listed in Table 3. The resulting AUC scores are presented in Table 4 and also at the bottom right corner in the boxes of Figure 3. The quantisation options are selected based on the best system performance. For example, the *Pianoroll-Pitch-Scalar* and the *Pianoroll-Pitch-Embedding* representations must use a 3-time larger quantisation grid compared to others so that the sequence length is decreased from around 400 (8 bars, 4 beats and 12 subdivisions per beat) to 100 (4 subdivisions per beat), which allowed the model to converge. The model failed to converge for representations *Event-PitchDuration-Sequence* and *Tonality*. The former failed probably because the doubled sequence length exceeded the network's capacity. The later, despite being short enough, seemed not to help the model learn advanced tonal features purely from the bar tonality distributions. By Deng *et. al*'s report [6], the bar tonality scores, before being used as features, are adjusted according to the confidence of each bar having a clear tonal center, which may be difficult to be learnt by Li's model.

#	Representation	AUC
1	Event-Pitch-Scalar	0.57
2	**Tonality** [6]	0.48
3	**Pianoroll-Pitch-Scalar** [11]	0.57
4	Event-Pitch-Embedding	0.63
5	**Pianoroll-Pitch-Embedding** [10, 14]	0.70
6	**Event-PitchDuration-Sequence** [8, 9]	0.52
7	**Event-PitchDuration-Addition** [7]	0.87
8	Event-PitchDuration-Concatenation	0.87

Table 4: Different input representations tested on the same system. Bold ones came from the submissions.

The results are basically consistent to the assumption that melody representations will impact the model performance in this data challenge. Notice that *Tonality* and *Event-PitchDuration-Sequential* used on the top-performing model did not result in effective classifiers with the AUC-ROC score being only 0.48 and 0.52, which was possibly due to the limited information represented by the tonality vectors and the over-long sequential representation respectively. Figure 3 combined with the results also to some extent displayed the advantages and drawbacks of different melody representations applied to the same model. This provides a reference to researchers when choosing input representations for their music data.

5.4 Effects of Metric Unit on Model Performance

Another set of experiments were conducted to investigate the influence of metric unit on the model performance. The system is modified to accommodate different unit metrics used with the original representation methods of **Event-PitchDuration-Addition**. The attempted metric units are 8th, 12th, 16th and 24th of a quarter note.

#	Metric Unit	Vocab Size	AUC
1	8th	89(p) + 8(d)	0.83
2	12th	89(p) + 15(d)	0.87
3	16th	89(p) + 16(d)	0.88
4	24th	89(p) + 94(d)	0.86

Table 5: Model performances using different metric units of time. The framed line refers to the original representation used by Li's submitted systems.

The results from Table 5 indicates the influence of metric unit of time on the model performance. When the metric unit is small, more rhythmic nuances are preserved and the corresponding vocabulary size (the number of tokens needed to donate all possible duration [or positional] information in the *training* dataset, in Li's manner [7]) will usually be larger. The accordingly changing AUC-ROC scores suggest the necessity of carefully choosing the metric unit of time for a specific music representation.

6. CONCLUSION

The CSMT 2020 Data Challenge evaluated objective systems that identify generated melodies from human compositions. With no limitation on the methodologies and external dataset, participants designed and submitted a wide range of systems utilising DNNs, heuristic methods, and traditional ML models. According to the challenge results, this paper investigated the influence of music style, pitch features and music representations on the model performance. Minor influence of the two music styles (*Bach* and *Pop*) are observed from the results. However, noticeable impacts are observed regarding pitch features and input representation of melodies. This paper suggests that the pitch features could be effectively used to diagnose the model performance for both music generation systems and objective evaluation systems. Besides, melody representations are also important factors that should be carefully selected in related tasks.

7. ACKNOWLEDGEMENT

The authors acknowledge the contribution from all members in the organisation committee of the CSMT Data Challenge 2020. This work is partially supported by Beijing Zhongwen (Xiamen) Law Firm.

8. REFERENCES

[1] S. Li, Y. Jing, and G. Fazekas, "A Novel Dataset for the Identification of Computer Generated Melodies in the CSMT Challenge," in *Proceedings of the 8th Conference on Sound and Music Technology.* Springer Singapore, Apr. 2021, pp. 177–186.

[2] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, Jul. 2018, pp. 4364–4373.

[3] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*. New York, NY, USA: Association for Computing Machinery, Oct. 2020, pp. 1180–1188.

[4] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A Convolutional Generative Adversarial Network for Symbolic-domain Music Generation," *arXiv:1703.10847 [cs]*, Jul. 2017.

[5] M. Cuthbert and C. Ariza, "Music21: A Toolkit for Computer-Aided Musicology and Symbolic Music Data." in *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, Jan. 2010, p. 642.

[6] Y. Deng, Z. Xu, L. Zhou, H. Liu, and A. Huang, "Research on AI Composition Recognition Based on Music Rules," *arXiv:2010.07805 [cs]*, Oct. 2020.

[7] Y. Li and Z. Lin, "Melody Classifier with Stacked-LSTM," *arXiv:2010.08123 [cs, eess]*, Nov. 2020.

[8] J. Guo, A. Liu, and J. Xiao, "Melody Classification based on Performance Event Vector and BRNN," *arXiv:2010.07562 [cs, eess]*, Oct. 2020.

[9] Y. Xia, Y. Jiang, and T. Ye, "Music Classification in MIDI Format based on LSTM Mdel," *arXiv:2010.07739 [cs, eess]*, Oct. 2020.

[10] M. Ding and Y. Ma, "A Transformer Based Pitch Sequence Autoencoder with MIDI Augmentation," *arXiv:2010.07758 [cs, eess]*, Feb. 2021.

[11] Y. Wang, Y. Liu, and L. Zhang, "An AI-Generated Melody Detection System Based on Support Vector Machine," *unpublished*, 2020.

[12] B. L. T. Sturm, M. Iglesias, O. Ben-Tal, M. Miron, and E. Gómez, "Artificial Intelligence and Music: Open Questions of Copyright Law and Engineering Praxis," *Arts*, vol. 8, no. 3, p. 115, Sep. 2019.

[13] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[14] J. Yu, J. Wang, and Y. Zhao, "Identification of AI-Generated Melodies," *unpublished*, 2020.

[15] L.-C. Yang and A. Lerch, "On the Evaluation of Generative Models in Music," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4773–4784, May 2020.

[16] H.-W. Dong, W.-Y. Hsiao, and Y.-H. Yang, "Pypianoroll: Open Source Python Package for Handling Multitrack Pianoroll," in *Pypianoroll: Open Source Python Package for Handling Multitrack Pianoroll.*, 2018.

YM2413-MDB: A MULTI-INSTRUMENTAL FM VIDEO GAME MUSIC DATASET WITH EMOTION ANNOTATIONS

Eunjin Choi[1] **Yoonjin Chung**[2] **Seolhee Lee**[1]
JongIk Jeon[3] **Taegyun Kwon**[1] **Juhan Nam**[1]
[1] Graduate School of Culture Technology, KAIST, South Korea
[2] Graduate School of AI, KAIST, South Korea
[3] Department of Industrial Design, KAIST, South Korea
{jech, yoonjin.chung, seolhee_lee, matji, ilcobo2, juhan.nam}@kaist.ac.kr

ABSTRACT

Existing multi-instrumental datasets tend to be biased toward pop and classical music. In addition, they generally lack high-level annotations such as emotion tags. In this paper, we propose YM2413-MDB, an 80s FM video game music dataset with multi-label emotion annotations. It includes 669 audio and MIDI files of music from Sega and MSX PC games in the 80s using YM2413, a programmable sound generator based on FM. The collected game music is arranged with a subset of 15 monophonic instruments and one drum instrument. They were converted from binary commands of the YM2413 sound chip. Each song was labeled with 19 emotion tags by two annotators and validated by three verifiers to obtain refined tags. We provide the baseline models and results for emotion recognition and emotion-conditioned symbolic music generation using YM2413-MDB.

1. INTRODUCTION

Music generation has been regarded as an attractive research topic for decades. Due to the revival of neural networks, the trend of music generation research has moved from rule-based to data-driven music approaches [1, 2]. In the data-driven approaches, the dataset determines music styles of the generated output and thus it is desired to have a wide assortment of datasets that cover diverse music genres and styles. However, the majority of music datasets are limited to a single instrument, particularly the piano [3–7]. Multi-instrumental datasets mainly cover popular music [8, 9] or classical music [10]. In this regard, we introduce a new dataset from 80s FM video game music originally played for YM2413, a programmable sound generator based on FM sound synthesis. We name it the YM2413 Music Database or YM2413-MDB.

© Eunjin Choi, Yoonjin Chung, Seolhee Lee, JongIk Jeon, Taegyun Kwon, Juhan Nam. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Eunjin Choi, Yoonjin Chung, Seolhee Lee, JongIk Jeon, Taegyun Kwon, Juhan Nam, "YM2413-MDB: A Multi-Instrumental FM Video Game Music Dataset with Emotion Annotations", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

YM2413-MDB provides not only a unique flavor of retro game music but also addresses two important issues in music generation research: the fixed number of instruments and the lack of high-level annotations. The most widely used multi-instrumental music dataset is the Lakh MIDI dataset (LMD). While it is the only large-scale MIDI dataset so far, the musical quality of MIDI files is not consistent within the dataset because it is gathered from public sources. Therefore, most studies using LMD generated music with a limited number of instruments for pop music [11–18]. Other datasets, such as JSB Chorale [10] and NES-MDB [19], have only four fixed instruments. Furthermore, the majority of these datasets lack high-level annotations such as genre or mood, which can be used for conditional music generation. Although songs in LMD are matched to entries in the Million Song Dataset (MSD), which has high-level tag annotations, the labels are rather noisy. There exist two symbolic emotion music datasets manually labeled [5, 20]. However, they include only piano music. YM2413-MDB tackles the two issues with the following characteristics.

- **Multi-instrumental 80s FM Game Music**: The dataset consists of 80's Sega game music played by YM2413, where 9 out of 16 instruments can be played at once, which improved the quality of the played game music at that time. From this advancement, the music played by YM2413 contains rich harmonic and rhythmic information.
- **Multi-modal**: The dataset contains 669 songs in various formats: binary video game music (VGM) files that contain FM synthesis parameters, converted MIDI files, and rendered audio files which restore the original game music sound. Also, the generated MIDI files can be rendered to audio using an FM-synthesizer plug-in.
- **Emotion-annotation**: We annotated the entire dataset with 19 emotion tags in the game context. To make a tag vocabulary with high agreement, each song was annotated by two dedicated annotators and verified by three reviewers. We also confirmed that these tags can be classified using the baseline classification approach.

We first describe the details of dataset collection and statistical analysis of musical features. Then, we provide

Name	Label Type	Music Style	# Inst	# Avg Tracks	# Songs	# Hours
LMD [8]	None	General	128	8.8 / no limit	178K	9K
DadaGP [9]	None	Band (Guitar Pro)	9	3.3 / 9	26.2K	1.2K
VGM-RNN [21]	None	Game music (NES, crowd source)	128	6.6 / no limit	4.2K	101.7*
JSB Chorale [10]	None	Classic (Bach Chorale)	4	4	382	3.9*
NES-MDB [19]	None	Game music (early 80s)	4	2.8 / max 4	5.2K	46.1
MOODetector [22]	28 adjectives	General	128	9.2 / no limit	196	12.1*
VGMIDI [20]	Valence	Piano-arranged game music	1	1	200	6.3*
EMOPIA [5]	Russell's 4Q	Pop piano covers	1	1	1,078	11
YM2413-MDB **(ours)**	19 adjectives	Game music (late 80s)	16	7.2 / max 9	669	15.4

Table 1: Comparison with existing multi-instrumental or emotion-labeled symbolic music datasets. When the dataset do not limit the instrument, # Inst is 128. When the dataset has no information of size in hours, it was estimated using the MIDI max tick (denoted as *).

baseline results of music emotion recognition in both audio and MIDI domains and emotion-conditioned symbolic music generation using a music generation model. We release YM2413-MDB in the online repository [1] . Examples of generated music are accessible in our demo webpage [2] .

2. RELATED WORK

2.1 Characteristics of Game Music

Starting with the single sound effect used in the first video game, *Pong* (1972), the development of game music is in line with the development of computer sound technology [23]. Today's game music is not much different from music from other media, such as dramas and movies. However, early video game music has a distinct characteristic from other genres of music due to technical constraints at the time. A musicological study on early game music found that some unique composition characteristics are found in the form of arpeggio patterns, compound melodies, and simulated reverb effects using delayed instruments [24]. Another characteristic of video game music is that it is composed to be played as a loop [25]. In addition, the transition of playing music occurs when the game context is changed, such as a triggered event: map transition, battle with a boss monster, mission completion, ending, and so on.

2.2 Game Music Generation

There are several studies that focus on computer-generated game music. For example, Prechtl used a music generation model based on Markov Chain, which changes the model state in real-time aligned with in-game parameters [25] They tested the model in a live game context. Mauthes generated video game music using an RNN-based model by collecting video game music crowd-sourced by the game user community [21]. Ferreira and Whitehead used piano-arranged video game music to generate music using LSTM and a genetic algorithm with sentiment as a condition [20]. Donahue et al. generated game music using Transformer-XL [26] with the Lakh dataset [8] pre-training and NES-MDB fine-tuning [19, 27].

[1] https://github.com/jech2/YM2413-MDB
[2] https://jech2.github.io/YM2413-MDB/

2.3 Multi-instrumental Symbolic Music Dataset

There are a handful of multi-instrumental datasets that cover different music genres, including pop [8, 22], classic [10, 28], band [9], and game [19, 21] genres. Table 1 compares the datasets to YM2413-MDB. Among them, the most widely used one is LMD [8]. Since LMD is a large-scale dataset and has a wide variety of instruments, most multi-instrumental research has used LMD with a set of simplified instruments [11–18] so far. Recently, large-scale datasets with a Band [9] or Orchestra [28] style have been released. In the game music genre, NES-MDB, the early 80's Nintendo Game Music Dataset [19] is most similar to our study. All songs in NES-MDB have four monophonic instruments because of the technical constraints of NES's Audio Processing Unit. Songs in NES-MDB have a unique style as retro game music and have expressiveness represented with note timings in the precise time units. However, such expressive characteristics require metric analysis such as beat tracking to extract score-level information.

2.4 Symbolic Music Dataset with Emotion Annotation

There are a few symbolic music datasets with emotion annotation [5, 20, 22]. MOODetector contains audio, MIDI, and lyrics for 193 music files [22]. The music was annotated according to the emotion category used in the MIREX Mood Classification Task [29]. VGMIDI contains 200 MIDI files of piano-arranged video game music with arousal-valence annotations [20]. EMOPIA contains 1087 pop piano music with annotations according to Russell's 4Q [5]. All of these datasets include both MIDI and audio formats.

3. THE YM2413-MDB

3.1 Dataset Description

The dataset consists of game music played by the YM2413 sound chip. YM2413 is a programmable sound generator based on FM synthesis developed by *Yamaha*. FM synthesis can generate diverse musical tones with rich harmonics. YM2413 can play a maximum of 9 instruments simultaneously from 16 instrument presets and one user tone register. The names of the instruments are shown in Figure 1(c). YM2413-MDB contains songs with a short

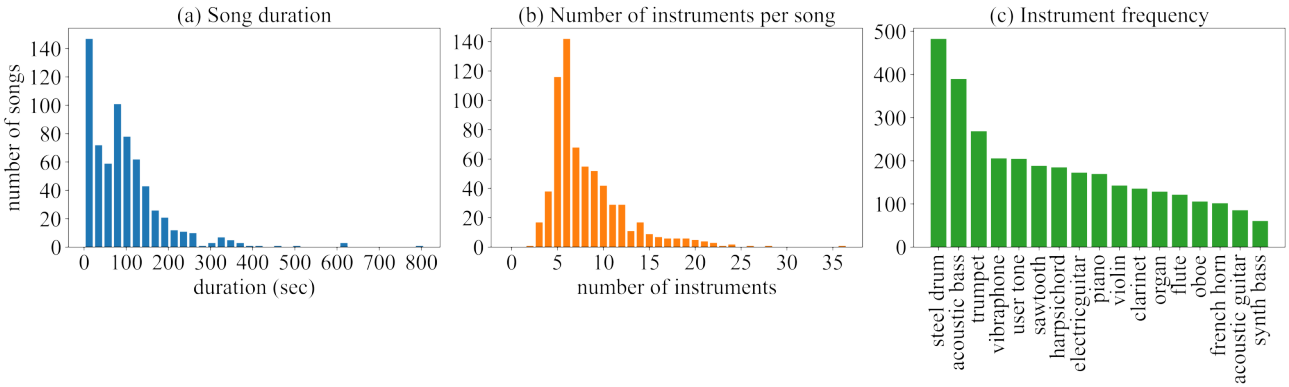


Figure 1: General statistics of YM2413-MDB. (a) Song duration of all songs, (b) Number of instruments per song considering the all program change in single channel, (c) Instrument frequency of whole dataset.

duration for specific game events similar to those in NES-MDB. The statistics are shown in Figure 1(a). In addition, we observed two unique composition patterns: unison playing and simulated reverb using delay. Unison playing is a pattern of the same melody with single or multiple instruments. This technique renders a rich tone by overlapping the same voices of the FM instrument. The delay pattern is also observed in the NES music to express the reverb by using the delayed same voice [24]. These patterns show how the composers at that time were concerned about expressing various tones. Therefore, the role and use of each instrument observed in YM2413-MDB seem to be quite different from those in today's composition style.

3.2 Dataset Collection & Preprocessing

We collected Video Game Music (VGM) files from the game music community websites [3] [4]. After data collection, we translated all the game music files to the MIDI format. Since the original VGM files are written as binary command sequences read by the sound chip, the VGM commands are different for all sound chips; for YM2413, there are non-standard commands for time wait, instrument note on/off, program change, volume change, and FM synthesis parameters for one user's custom tone. We conducted the conversion by emulating the YM2413 sound chip following the FM Operator Type-LL (OPLL) Application Manual [30]. For the VGM disassembly process and sound chip emulation, we referred to the source code of VGMPlay [5], a software that reads the VGM file and reproduces the sound of hardware sound chips. Also, we exported the VGM files as audio files using this software.

The original VGM file format expresses all time-related information as ticks with no tempo-related event. In the case of NES-MDB, they converted all MIDI files with 120 beats per minute (BPM) and 22050 ticks per beat to preserve the temporal resolution of 44100Hz without precise tempo calculation. However, the converted MIDI files were not aligned with the metrical structure that standard MIDI files have. This has hindered the use of bar-based music representation, chord recognition, and further music analysis [4, 16, 31, 32]. Therefore, we conducted the metrical alignment using the extracted down-beat information of rendered audio files using Madmom [14]. For each music file, tempo events were added using estimated tempos from all downbeats. Since most of the pieces had a fixed tempo, we used the median value of the beat-wise estimation of tempo.

In addition, the sound chip synthesizes sounds using frequency values rather than MIDI note numbers; in practice, a frequency is expressed as an F-Number, a discrete value that maps the frequency using 9 bits [30]. Therefore, some of the songs in YM2413-MDB were intentionally detuned for the song's ambiance, and instruments had pitch bends and vibratos for more expressiveness. These characteristics make it difficult to map the right MIDI pitch without careful consideration. We treated such cases using the MIDI pitch bend event, such that the integers and decimal points of the calculated MIDI note number are mapped to the MIDI pitch and pitch bend value.

3.3 Emotion Annotation

During the annotation and verification process and analysis, we referred to [33] for tag vocabulary refinement, verification procedure, and tag visualization. Two researchers participated in both initial tagging and annotation. Once the dataset was collected, we first listened to the entire game music audio files in the dataset and freely described them with the perceived emotion words. From this process, we found several game-specific emotion words that cannot be directly mapped into both the conventional emotion arousal-valence model and emotion tags used in MIREX music classification [22]: fluttered (war-inspiring), grand, bizarre, frustrating, comic, faint, and touching. As discussed in [34], the categorical approach is suitable for these complex emotions rather than the dimensional approach. Therefore, we decided to annotate the emotions as word tags.

From the emotion descriptions of all songs in the dataset, we obtained 35 emotion-related tags. We grouped similar emotion words according to a Korean crowdsource BGM repository [6] and finally obtained 19 emotion tags.

[3] https://www.smspower.org/Tags/FM
[4] https://vgmrips.net/packs/chip/ym2413
[5] https://github.com/vgmrips/vgmplay
[6] https://bgmstore.net/

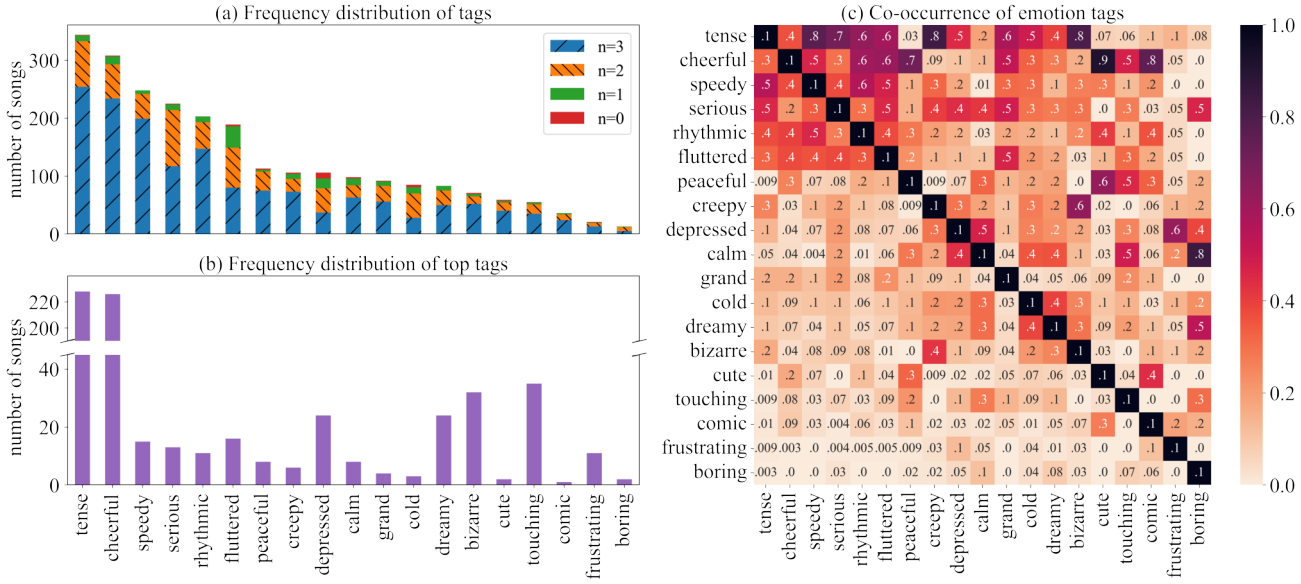


Figure 2: Emotion annotation and verification results. (a) Frequency distribution of emotion tags where n is the number of agreements of verifiers, (b) Frequency distribution of top tags, (c) Co-occurrence of emotion tags. Each column is normalized by the frequency of each tag.

After the tag vocabulary was refined, we conducted the labeling. While annotating the dataset, each of the annotators selected one dominant emotion tag for each music which we call "top tag" in this paper. Among the top tag annotated by the two annotators, the final top tag was selected as having a higher agreement after verification.

3.4 Emotion Verification

After the annotation step, each music file was validated by three verifiers; in total, ten people participated as verifier. During verification, verifiers were asked to mark the emotion tag if they disagree with the annotation. We provided verifiers with the annotator's description of each emotion tag to reduce the ambiguity of each emotion tag. After verification, tags with low agreement (lower than two) were excluded from the annotation of each music. When the top tag was excluded after verification (26 songs), the tag with the highest agreement was chosen as the new top tag.

Figure 2(a) shows the overall emotion tag distribution of the YM2413-MDB after verification. Also, the agreement portion of verification explains the subjectivity of each tag. In particular, 'serious', 'fluttered', 'depressed', and 'cold' tags have less agreement than other tags. The frequency distribution of top tags is shown in Figure 2(b), in which the imbalance is more noticeable than the frequency distribution of each tag. In Figure 2(c), we investigated the independence of each tag by calculating the co-occurrence of emotion tags. The words that often appear together(>0.8) are as follows: 'speedy'-'tense', 'creepy'-'tense', 'bizarre'-'tense', 'cute'-'cheerful', 'comic'-'cheerful', and 'boring'-'calm'.

Annotation and verification for the Korean tags were operated since all participants were native Korean. They are aged from twenty-four to thirty-one, with at least one retro video game-playing experience.

3.5 Data Representation

For the baseline symbolic-domain classification and generation, we used the event-based representation called Multi-Track Music Machine (MMM) [35]. The MMM representation can handle arbitrary instrument combinations. As discussed in Section 3.1, our dataset contains unison patterns; the same instrument appears in multiple tracks. To apply representations suggested for multi-instrumental music that assume each instrument appears at most once [16, 27, 32], we should discard instrument tracks that appear multiple times in a song. However, we did not apply the strategy due to the small dataset size. In our preliminary study, experiments with multi-instrumental representation suggested by [27] only generate a drum with a large number of notes. We speculate that it was due to the frequency imbalance of tokens; since the representation suggested by [27] combines musical instruments and pitch tokens, the number of tokens increases in proportion to the number of instruments, making the model difficult to learn pitch information. Therefore, we did not use those non-MMM style representations.

To re-implement MMM, we calculated the note density for each instrument by dividing the total number of note counts by the active bar number, which is the number of bars in which at least one note is played. Also, we quantized the music samples with 48 grids per bar. After removing the bar-fill-related vocabulary of MMM which is not necessary for our tasks, we used a total of 447 tokens: five tokens for the piece, track, and bar; 48 tokens for time delta tokens; 128 tokens for each instrument, note on, and note off. Since the MMM representation can express a fixed number of bars, we used four bars with a maximum of six instruments, which the transformer attention window size of 1024 can handle. Fixing the length of generated music is suitable for game music because the game

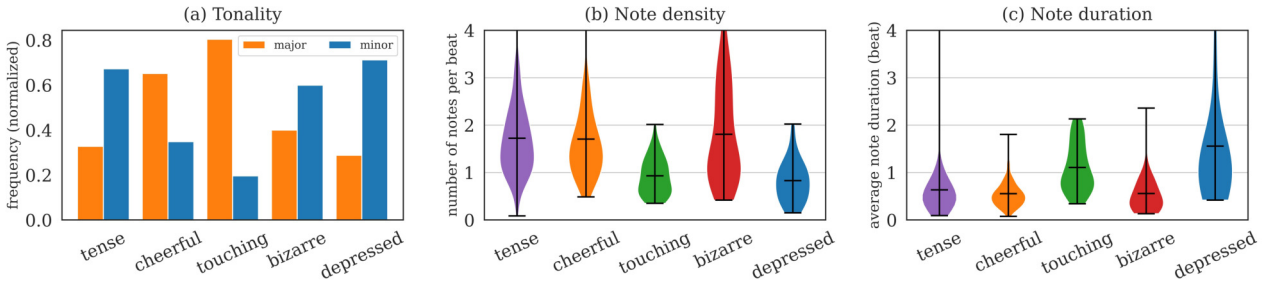


Figure 3: Data distribution for different top tag classes. (a) Normalized frequency of tonality histogram, (b) Note density, (c) Note duration. The outlier sample points were omitted for clear visualization.

music is loop-based, as discussed in Section 2. We sampled the instrument when the number of instruments was greater than six. Due to the sparsity issue, we adapted the dataset for training. The instruments were mapped into six categories during training: piano, string, woodwind, brass, guitar, bass, and drum. Tracks that change within the same channel are all adapted to be treated as one instrument.

4. DATASET ANALYSIS

Emotion is associated with various musical features such as harmony, rhythm, and loudness [36]. We investigated how the musical elements extracted from YM2413-MDB are correlated with the collected emotion tags. We used note density, note duration, and tonality following the previous work [5]. Due to the imbalanced distribution of top tags, we focused on songs with five top tags: 'tense', 'cheerful', 'touching', 'bizarre', and 'depressed'. When extracting the features, we excluded drum tracks to consider the pitch information only when estimating tonality and fairly compared songs with and without drums.

4.1 Tonality

The major/minor tonality is generally connected to positive/negative emotional states [37]. We measured tonality using the Krumhansl-Schmuckler key-finding algorithm [38] implemented in music21 [39]. Figure 3(a) shows a clear trend that the major tonality is frequently used for positive emotions (e.g. 'cheerful', 'touching'), and the minor tonality is frequently used for negative emotions (e.g. 'tense', 'bizarre', 'depressed').

4.2 Note Density and Duration

Another musical feature related to emotion is rhythm [40]. We use note density and note duration as indirect cues to extract the rhythmic characteristics. The note density refers to the number of notes per beat, which was obtained by dividing the total number of notes by the total number of beats. It was normalized by the number of instruments per song. The note duration was calculated by averaging the note duration value in beat units. As shown in Figure 3(b-c), songs with 'tense', 'cheerful', and 'bizarre' tags showed higher note density than the 'touching' and 'depressed' tags. Most songs with 'tense', 'cheerful', and 'bizarre' tags consisted of notes shorter than one beat, whereas songs with 'touching' and 'depressed' tags were widely distributed with longer notes. We also measured the note velocity, but all tag groups maintained a similar distribution. It is because songs in YM2413-MDB do not utilize the velocity features well; in most cases, the velocity of notes is changed together only when program change events occur.

Model	Emotion			Top Tag	
	4Q	Arousal	Valence	Tense	Cheerful
Logistic regression	0.48	0.76	0.56	0.66	0.66
LSTM-Attn [41] + MMM [35]	0.44	0.69	0.59	0.68	0.70

Table 2: Symbolic-domain classification accuracy. 4Q indicates Russell's four quadrants.

Model	Emotion			Top Tag	
	4Q	Arousal	Valence	Tense	Cheerful
Logistic regression	0.38	0.72	0.54	0.60	0.66
Short-chunk ResNet [42]	0.65	0.78	0.60	0.69	0.65

Table 3: Audio-domain classification accuracy. 4Q indicates Russell's four quadrants.

5. MUSIC EMOTION RECOGNITION

We provide our baseline symbolic and audio-domain emotion recognition results using YM2413-MDB. For both symbolic- and audio-domain classification, we used the source code and experimental setting (4Q, Arousal, Valence) of [5] and adapted it to our dataset configuration. We used the same model for classification (logistic regression, LSTM-Attn [41], and short-chunk ResNet [42]). For symbolic domain classification, we used a re-implemented MMM representation with 8 bars and 6 instrument chunks. Due to the small volume with imbalanced emotion tags, it was difficult to use the original emotion tags for emotion recognition directly. Instead, we mapped the top tags into Russell's four quadrants and classified a song into one of the four categories; for tags that do not appear in Russell's circumplex model, we manually mapped to one quadrant of the most similar emotion according to the subjectivity of the annotator [7]. We also provide the baseline binary classification results using 'tense' and 'cheerful' top tags. Other tags showed low classification results due to severe imbalance. We split the dataset with an 8:1:1 ratio for train, validation, and test via stratified sampling.

The emotion classification results of symbolic- and audio-domain are shown in Table 2 and Table 3. Both

[7] https://github.com/jech2/YM2413-MDB/blob/main/emo4Qmap.png

symbolic- and audio-domain classification results of top tags show that these emotions can be recognized by the baseline models. Since the dataset is multi-instrumental and the class imbalance is more severe than EMOPIA, the baseline classification was lower than those reported in [5]. Also, we note that the symbolic-domain classification performance of the LSTM-Attention model with MMM was poorer than that of the logistic regression. We conjectured that it was difficult to learn the temporal relations between instruments when using MMM representation.

6. EMOTION-CONDITIONED SYMBOLIC MUSIC GENERATION

6.1 Data Preparation

We provide the baseline emotion-conditioned symbolic music generation results using YM2413-MDB. Due to the emotion tag class imbalance, emotion conditioning was ineffective when using the entire set of emotion top tags. Instead, we selected 'cheerful' and 'depressed' as representative emotion conditions that showed contrasting features in the dataset analysis (Fig 3) and conducted conditional generation using the data with the two tags. Similar to [27], as our dataset size was small to train the deep learning model from scratch, we used LMD for pre-training the music generation model and fine-tuned it using YM2413-MDB. We filtered too short MIDI files which are less than 3 secs. We also filtered short samples annotated as sound effects. After filtering, we used 10K samples for validation and test sets when we trained the LMD. A total of 286 songs were used for fine-tuning, and four songs containing both 'cheerful' and 'depressed' tags were excluded.

6.2 Model & Train Setting

Transformer-XL [26] is widely used in music generation studies with language modeling approach [4, 9, 27, 43]. GPT2 [44] is a language model that showed high fine-tuning performance in NLP without changing the model structure and was also used in several music generation studies [35,45]. In our preliminary study, we observed that GPT2 generated better quality samples than Transformer-XL. In addition, the generation samples from lakh pre-train were better when adding more layers than [35] used (six layers).

Therefore, we used the 12-layer GPT2 [44] with an attention window size of 1024 for our baseline generation model. Before passing the main architecture, each token of event sequence input was embedded as 512 dimensions. To maintain the dimension while calculating attention, the number of attention heads and attention head dimensions were set to 8 and 64, respectively. For all layers, the dimension of the feed-forward layer after the attention module was 1024. The initial learning rate was set to 5e-5, and we used the AdamW optimizer with a linear scheduler. To generate music with emotion condition, we fine-tuned the pre-trained GPT2 model using YM2413-MDB.

Also, by using the in-attention mechanism proposed in [18], we projected the emotion tag embedding to the same space as the hidden states, then summed the projected condition with the hidden states at each self-attention layer. For emotion tag embedding, we used the pre-trained Word2Vec embedding [46] of 'cheerful' and 'depressed' tags, which were also updated during fine-tuning.

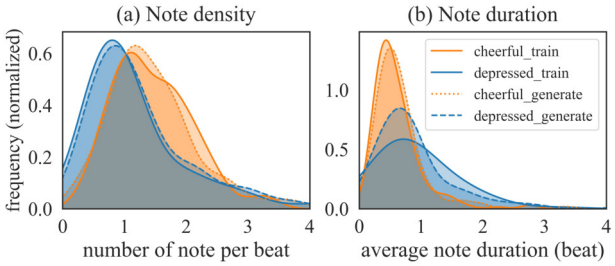


Figure 4: The note density and note duration distribution of the train set and generated results when emotion condition contains 'cheerful' and 'depressed'. Each histogram is smoothed using kernel density estimation.

6.3 Generated Results

Although we provided the baseline classification results, the performance was not sufficient to evaluate conditional music generation. Therefore, in this section, we show the overall tendency of the generation model through the objective measure. To see the validity of emotion conditioning in music generation, we compared 100 model-generated songs conditioned on 'cheerful' and 'depressed' emotions. We compared the histogram of note density and note duration of the generated results with the model inputs. For model inputs, the first 4 bars of each of the 284 songs were analyzed. As shown in Figure 4, given 'depressed' conditions, the model generates samples with a longer note duration and lower note density than when using 'cheerful' condition. However, when we listened to the generated samples, we found some of the generated samples struggled with temporal information, such as note onset time. It seems delayed notes in the dataset hindered learning the overall beat structure of the music. Although the model inputs had distinct distributions for major/minor tonalities, the generated results did not reflect these features. We suppose that it is due to the characteristics of game music that we observed in YM2413-MDB; the appearance of whole-tone or chromatic patterns makes it difficult for the model to learn tonality.

7. CONCLUSION

In this paper, we introduced YM2413-MDB, a multi-instrumental FM video game music dataset annotated with emotion tags. Using this dataset, we conducted a basic statistical analysis of musical features and provided the baseline results for emotion recognition and conditional music generation. We plan to study retro game music compositional styles further and improve music emotion classification and emotion-conditioned music generation. Also, we will investigate other uses of YM2413-MDB such as multi-instrumental music transcription, source separation, and structure analysis.

8. ACKNOWLEDGMENT

This work was supported by Year 2022 Culture Technology R&D Program by the Ministry of Culture, Sports and Tourism and Korea Creative Content Agency (Project Name: Research Talent Training Program for Emerging Technologies in Games, Project Number: R2020040211) and Institute of Information communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2019-0-00075, Artificial Intelligence Graduate School Program (KAIST)).

9. REFERENCES

[1] O. Lopez-Rincon, O. Starostenko, and G. Ayala-San Martín, "Algoritmic music composition based on artificial intelligence: A survey," in *Proceedings of 2018 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, 2018, pp. 187–193.

[2] S. Ji, J. Luo, and X. Yang, "A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions," *arXiv preprint arXiv:2011.06801*, 2020.

[3] Z. Wang*, K. Chen*, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "POP909: A pop-song dataset for music arrangement generation," in *Proceedings of 21st International Conference on Music Information Retrieval (ISMIR)*, 2020, pp. 38–45.

[4] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[5] H.-T. Hung, J. Ching, S. Doh, N. Kim, J. Nam, and Y.-H. Yang, "EMOPIA: A multi-modal pop piano dataset for emotion recognition and emotion-based music generation," in *Proceedings of 22nd International Conference on Music Information Retrieval (ISMIR)*, 2021, pp. 318–325.

[6] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *Proceedings of 7th International Conference on Learning Representations (ICLR)*, 2019.

[7] Q. Kong, B. Li, J. Chen, and Y. Wang, "GiantMIDI-Piano: A large-scale midi dataset for classical piano music," *Transactions of the International Society for Music Information Retrieval*, vol. 5, no. 1, pp. 87–98, 2022.

[8] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, Columbia University, 2016.

[9] P. Sarmento, A. Kumar, C. Carr, Z. Zukowski, M. Barthet, and Y.-H. Yang, "DadaGP: A dataset of tokenized guitarpro songs for sequence models," in *Proceedings of 22nd International Conference on Music Information Retrieval (ISMIR)*, 2021, pp. 610–617.

[10] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proceedings of the 29th International Conference on Machine Learning(ICML)*, 2012, pp. 1159–1166.

[11] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, 2018.

[12] H.-W. Dong and Y.-H. Yang, "Convolutional generative adversarial networks with binary neurons for polyphonic music generation," in *Proceedings of 19th International Conference on Music Information Retrieval (ISMIR)*, 2018, pp. 190–197.

[13] X. Liang, J. Wu, and Y. Yin, "MIDI-Sandwich: Multi-model multi-task hierarchical conditional vae-gan networks for symbolic single-track music generation," in *Proceedings of Australian Journal of Intelligent Information Processing Systems*, vol. 15, no. 2, 2019, pp. 1–9.

[14] X. Liang, J. Wu, and J. Cao, "MIDI-Sandwich2: Rnn-based hierarchical multi-modal fusion generation vae networks for multi-track symbolic music generation," *arXiv preprint arXiv:1909.03522*, 2019.

[15] T. Hirai and S. Sawada, "Melody2Vec: Distributed representations of melodic phrases based on melody segmentation," *Journal of Information Processing*, vol. 27, pp. 278–286, 2019.

[16] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "PopMAG: Pop music accompaniment generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1198–1206.

[17] H. Liang, W. Lei, P. Y. Chan, Z. Yang, M. Sun, and T.-S. Chua, "Pirhdy: Learning pitch-, rhythm-, and dynamics-aware embeddings for symbolic music," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 574–582.

[18] S.-L. Wu and Y.-H. Yang, "MuseMorphose: Full-song and fine-grained music style transfer with just one transformer VAE," *arXiv preprint arXiv:2105.04090*, 2021.

[19] C. Donahue, H. H. Mao, and J. McAuley, "The NES music database: A multi-instrumental dataset with expressive performance attributes," in *Proceedings of 19th International Conference on Music Information Retrieval (ISMIR)*, 2018, pp. 475–482.

[20] L. N. Ferreira and J. Whitehead, "Learning to generate music with sentiment," *Proceedings of 20th International Conference on Music Information Retrieval (ISMIR)*, 2019.

[21] N. Mauthes, "VGM-RNN: Recurrent neural networks for video game music generation," M.S. thesis, San Jose State University, 2018.

[22] R. E. S. Panda, R. Malheiro, B. Rocha, A. P. Oliveira, and R. P. Paiva, "Multi-Modal Music Emotion Recognition: A new dataset, methodology and comparative analysis," in *Proceedings of 10th International Symposium on Computer Music Multidisciplinary Research (CMMR)*, 2013, pp. 570–582.

[23] F. Gutierrez Rojas, "The impact of sound technology on video game music composition in the 1990s," M.S. thesis, Utrecht University, 2018.

[24] ——, "Programmed baroque'n'roll: Composition techniques for video game music on the nintendo entertainment system," B.S. thesis, San Jose State University, 2016.

[25] A. Prechtl, "Adaptive music generation for computer games," Ph.D. dissertation, Open University, 2016.

[26] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of the Association for Computational Linguistics (ACL)*, 2019.

[27] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," in *Proceedings of 20th International Conference on Music Information Retrieval (ISMIR)*, 2019, pp. 685–692.

[28] J. Liu, Y. Dong, Z. Cheng, X. Zhang, X. Li, F. Yu, and M. Sun, "Symphony generation with permutation invariant language model," *arXiv preprint arXiv:2205.05448*, 2022.

[29] X. Hu and J. S. Downie, "Exploring Mood Metadata: Relationships with genre, artist and usage metadata," in *Proceedings of 8th International Conference on Music Information Retrieval (ISMIR)*, 2007.

[30] Yamaha, "YM2413 FM operator Type-LL (OPLL) application manual," https://www.smspower.org/maxim/Documents/YM2413ApplicationManual, 1987.

[31] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound Word Transformer: Learning to compose full-song music over dynamic directed hypergraphs," in *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, 2021, pp. 178–186.

[32] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "MusicBERT: Symbolic music understanding with large-scale pre-training," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 791–800.

[33] K. L. Kim, J. Lee, S. Kum, C. L. Park, and J. Nam, "Semantic tagging of singing voices in popular music recordings," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1656–1668, 2020.

[34] J. S. Gómez-Cañón, E. Cano, T. Eerola, P. Herrera, X. Hu, Y.-H. Yang, and E. Gómez, "Music emotion recognition: Toward new, robust standards in personalized and context-sensitive applications," *IEEE Signal Processing Magazine*, vol. 38, no. 6, pp. 106–114, 2021.

[35] J. Ens and P. Pasquier, "MMM: Exploring conditional multi-track music generation with the transformer," *arXiv preprint arXiv:2008.06048*, 2020.

[36] R. Panda, R. M. Malheiro, and R. P. Paiva, "Audio features for music emotion recognition: a survey," *IEEE Transactions on Affective Computing*, pp. 1–1, 2020.

[37] M. P. Kastner and R. G. Crowder, "Perception of the major/minor distinction: Iv. emotional connotations in young children," *Music Perception*, vol. 8, no. 2, pp. 189–201, 12 1990.

[38] D. Temperley, "What's key for key? the krumhansl-schmuckler key-finding algorithm reconsidered," *Music Perception*, vol. 17, no. 1, pp. 65–100, 10 1999.

[39] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," 2010.

[40] E. G. Schellenberg, A. M. Krysciak, and R. J. Campbell, "Perceiving Emotion in Melody: Interactive effects of pitch and rhythm," *Music Perception*, vol. 18, no. 2, pp. 155–171, 2000.

[41] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *Proceedings of International Conference of Learning Representations (ICLR)*, 2017.

[42] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of cnn-based automatic music tagging models," in *Proceedings of the 17th Sound and Music Conference (SMC)*, 2020.

[43] S.-L. Wu and Y.-H. Yang, "The jazz transformer on the front line: Exploring the shortcomings of ai-composed music through quantitative measures," in *Proceedings of 21th International Conference on Music Information Retrieval (ISMIR)*, 2020, pp. 142–149.

[44] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[45] C. Payne, "MuseNet," https://openai.com/blog/musenet, 2019.

[46] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of International Conference on Learning Representations (ICLR), Workshop Track Proceedings*, 2013.

DETECTING SYMMETRIES OF ALL CARDINALITIES WITH APPLICATION TO MUSICAL 12-TONE ROWS

Anil Venkatesh **Viren Sachdev**
Department of Mathematics and Computer Science, Adelphi University
avenkatesh@adelphi.edu

ABSTRACT

Popularized by Arnold Schoenberg in the mid-20th century, the method of twelve-tone composition produces musical compositions based on one or more orderings of the equal-tempered chromatic scale. The work of twelve-tone composers is famously challenging to traditional Western tonal and structural sensibilities; even so, group theoretic approaches have determined that 10% of certain composers' works contain a highly unusual classical symmetry of music. We extend this result by revealing many symmetries that were previously undetected in the works of Schoenberg, Webern, and Berg. Our approach is computational rather than group theoretic, scanning each composition for symmetries of many different cardinalities. Thus, we capture partial symmetries that would be overlooked by more formal means. Moreover, our methods are applicable beyond the narrow scope of twelve-tone composition. We achieve our results by first extending the group-theoretic notion of symmetry to encompass shorter motives that may be repeated and reprised in a given composition, and then comparing the incidence of these symmetries between the work of composers and the space of all possible 12-tone rows. We present four candidate hierarchies of symmetry and show that in each model, between 75% and 95% of actual compositions contained high levels of internal symmetry.

1. INTRODUCTION

The Viennese composers Arnold Schoenberg, Anton Webern, and Alban Berg produced a combination of 86 twelve-tone compositions in the early-to-mid 20th century [1–3]. Each of these compositions is constructed from some permutation of the twelve pitch classes of the equal-tempered chromatic scale, which then guides the order of notes in the melody and harmonies. Figure 1 shows musical notation for one such permutation of the pitch classes that was selected by Schoenberg. Each such permutation is called a *tone row*, and we will take the *Viennese tone rows* to mean those that underlie the compositions of Schoenberg, Webern, and Berg, the principal members of the Second Viennese School. Using Hauer's arrangement of the pitch classes at the hour marks of an analog clock diagram [4], each tone row can be visualized as a directed graph that visits each vertex once, and the set of these directed graphs is in bijection with the set of tone rows [5,6].

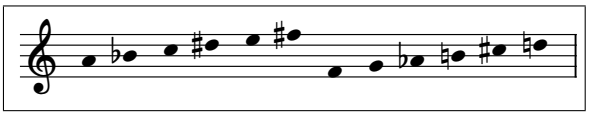


Figure 1. Musical notation for the row from Schoenberg's Serenade, opus 24, movement 5. Note that each pitch class is used once.

The strict rules of twelve-tone composition and its inherent relation to permutations have inspired formal mathematical study, both combinatorial and group theoretic. Recent work by von Hippel and Huron applied a key-finding algorithm to subsets of all lengths within certain tone rows in order to quantify their degree of "tonalness" [7]. This combinatorial approach construes the tone row as the union of many shorter overlapping subsets whose individual properties imply properties of the tone row as a whole. By contrast, Hunter and von Hippel treated tone rows more like indivisible algebraic objects in their group-theoretic study, showing that the action of the dihedral group (rotations and reflections) on clock diagrams, together with the operation of reversing the diagram's arrows, encompass the classical music-theoretic symmetries of *translation*, *inversion*, and *retrograde* [8].

The composers of the Second Viennese School considered tone rows to be equivalent under these symmetries so it makes sense to study equivalence classes of tone rows instead of the rows themselves. The resulting *row classes* of Schoenberg and Webern can therefore be visualized as unlabeled, undirected clock diagrams (start and end points need not be distinguished due to equivalence under retrograde). Berg also considered a fourth symmetry, *cyclic shift*, so his row classes are larger: they are equivalent to unlabeled, undirected clock diagrams whose endpoints connect [3]. Figure 2 shows clock diagrams of the twenty-one row classes used by Webern, with the four diagrams that possess dihedral symmetry set apart from the others. These four diagrams correspond to row classes that are isomorphic to their retrograde. The fact that Berg considered an extra symmetry has the practical effect of making his row classes larger, and hence the set of row classes smaller,

© A. Venkatesh and V. Sachdev. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** A. Venkatesh and V. Sachdev, "Detecting Symmetries of All Cardinalities With Application to Musical 12-Tone Rows", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

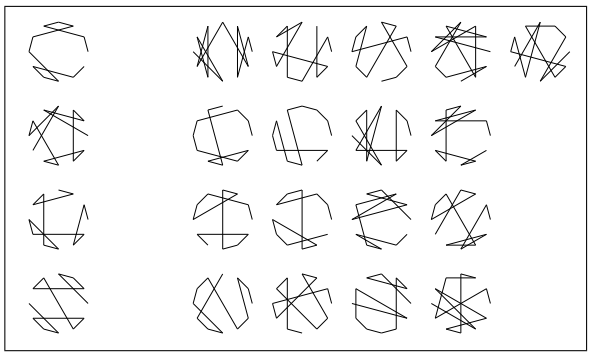


Figure 2. Clock diagrams of the twenty-one tone rows used by Webern, with the four symmetric diagrams on the left.

than those of Schoenberg and Webern. In this work, we follow convention by studying the two alternatives in parallel as the methods of study are otherwise unaffected by the use of an extra symmetry.

Hunter and von Hippel posed and resolved the following question: are there more symmetric Viennese tone rows than would be expected by chance alone? By the numbers, symmetric rows comprise just 5% of Schoenberg's works, 19% of Webern's, and 9% of Berg's. However, symmetric rows are *exceedingly* rare among all row classes: just 0.13% of row classes possess dihedrally symmetric clock diagrams (the figure is 1.16% when Berg's cyclic shift symmetry is included). Hunter and von Hippel made this idea rigorous by using a hypergeometric test to conclude that these composers showed a statistically significant preference for symmetry, whether intentionally or not [8].

Results of this kind can be concretely informative to music scholars as well as mathematicians. Most directly, Hunter and von Hippel offered analytical evidence that the three composers preferred symmetric row classes, a fact that had already been believed widely among music scholars [2, 3]. However, their results also contradicted certain beliefs, for example that Webern preferred non-palindromic symmetry [2]. They found that non-palindromic symmetry is simply much more common than palindromic symmetry, more than accounting for its higher incidence in Webern's corpus. In short, the enumeration of row classes and their symmetries is a truly interdisciplinary undertaking with promise to inform music scholars in ways that are relevant to their practice.

The main question left open by Hunter and von Hippel relates to the fact that just 10% of the Viennese tone rows are symmetric. While this figure is already surprisingly high compared to the incidence of symmetry among all row classes, it still leaves the fact that the Viennese twelve-tone composers evidently had some priorities beyond symmetry alone. Indeed, music scholars have posited many other properties of tone rows that may have attracted the composers' interest: "combinatorial," "all-interval," "tonally colored," and particularly "derived" rows have all been subject to study [9, 10]. Derived rows are those that consist of a sequence of $12/d$ notes repeated d times under the various transformations available to the composer [10]. The Gotham and Yust Serial Analyzer project [11] has yielded a catalog of rows in the repertoire classified by these traditional properties as well as more novel harmonic properties deduced from the discrete Fourier transform approach of Krumhansl [12]. The set of all possible derived rows has also been enumerated algebraically. Fripertinger and Lackner generalized the work of Hunter and von Hippel by constructing a group action and classified all tone rows according to the traditional properties in the Database of Tone Rows and Tropes [10]. To our knowledge, this database has yet to be studied in the sense of Hunter and von Hippel to determine whether the Viennese tone rows contain statistically significant quantities of derived rows, for example.

The combinatorial study of tone rows has led to the enumeration of rows with various traditionally studied properties as well as measurements of "tonalness" and other harmonic properties. Meanwhile on the group-theoretic side, Fripertinger and Lackner's group action has helped enumerate the derived rows, a much richer set than the fully symmetric rows of Hunter and von Hippel. Missing from the literature is a combinatorial attack on the *definition* of symmetry in tone rows: just as Yust [13] extended von Hippel and Huron's study of tonal fit to pairs of tonal, atonal, and mixed dimensions, this paper uses combinatorial means to generalize the concept of derived row, capturing nuances that are impractical to detect with group theory. For example, if we permute just the last two notes of a derived row, the resulting tone row will most likely not possess any group theoretic symmetries. The rigidity of group-theoretic symmetries causes this type of partial symmetry to be overlooked. In this work, we introduce a combinatorial alternative to the group-theoretic definitions of symmetry, scanning each tone row for partial symmetries of all cardinalities. Our approach results in a complete catalog of recurring motives within each tone row, from which we argue that the vast majority of the Viennese tone rows contain unusually high levels of symmetry. Moreover, we believe the flexibility of our approach grants it applications beyond the narrow scope of twelve-tone composition.

2. DETECTING INTERNAL SYMMETRY

For the purpose of this work, a *motive* within a tone row is any consecutive sequence of notes that recurs at least once within the same row (transposed and possibly inverted or in retrograde). For example, the row $(0, 1, 10, 5, 2, 3, 6, 7, 8, 11, 4, 9)$ contains the motive $(0, 1, 10, 5)$ which repeats transposed under retrograde inverse as $(10, 5, 2, 3)$, as shown in the top line of Figure 3. This particular row also contains the motive $(3, 6, 7, 8)$ and its transposed retrograde inverse $(6, 7, 8, 11)$. Motives can be as short as two notes or as long as twelve; whereas motives of length 2 occur whenever the same musical interval (or its inverse) is reused in a tone row, motives of length 12 are only found in the rare palindromic rows. The highest incidence of motives can be found in the *chromatic scale* and the *circle of fifths*, each of which consists of a single length-2 segment whose corresponding musical interval is

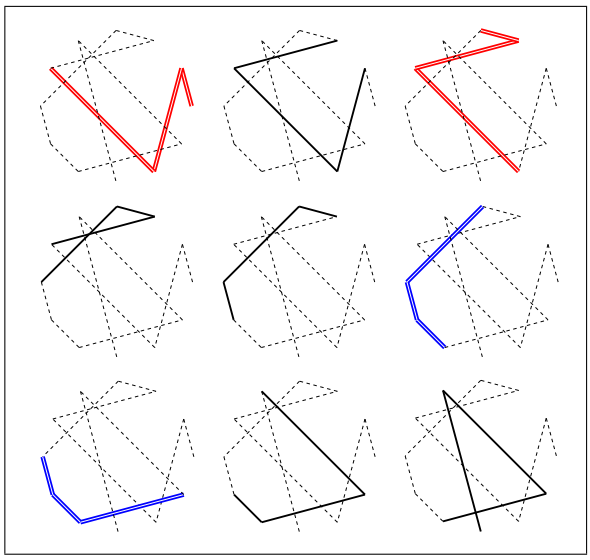


Figure 3. Clock diagrams of the tone row $(0, 1, 10, 5, 2, 3, 6, 7, 8, 11, 4, 9)$ showing the nine sequences of length 4. The two recurring motives are indicated with doubled lines.

repeated eleven times to form the row. In the opposite case, the *all-interval rows* consist of every different musical interval and consequently possess very few motives. Figure 4 shows clock diagrams for the circle of fifths (containing the maximum number of motives of all lengths) and an all-interval row that contains no motives above length 2 at all.

Our approach has two stages: first, we enumerate the motives of all lengths for every row class; then, we advance four plausible models for assigning an overall symmetry rating to any set of motives, with attention to the music-theoretic and mathematical literature to ensure consistency of our results with existing knowledge. We begin the first stage by treating each tone row as a collection of length-d sequences of notes, with d ranging from 2 to 12. Figure 3 shows a particular tone row's nine sequences of length 4, of which two pairs are found to be separate recurring motives. In general, the set of length-d motives of a tone row can be quantified by a partition of $12 - d + 1$ (the total number of length-d sequences) based on which sequences appear multiple times. Figure 3 shows that the depicted tone row corresponds to the partition $9 = 2+2+1+1+1+1+1$ since there are two motives that each appear twice, plus five other length-4 sequences that do not form motives. As d varies from 2 to 12, the corresponding sets of motives fully encode the symmetrical properties of a tone row at every cardinality. Therefore, the symmetrical properties of any tone row can be represented by a list of partitions of the integers from 1 to 11, corresponding to values of d ranging from 12 down to 2. We refer to such a list of partitions as the *full symmetry data* of a row class.

Generalizing symmetry in this way encodes a tremendous amount of information. Hunter and von Hippel identify three types of rows (eight, when cyclic shift is included) [8]. Friptertinger and Lackner extend this to a few thousand by introducing tropes [10]. But a naïve upper bound on the number of distinct full symmetry data is the product of the first eleven partition numbers, about 5.379×10^{10} or five thousand times the number of row classes! There are many principled ways to simplify the full symmetry data while maintaining consistency with the literature. One promising approach (not explored in this paper) would map each partition to its length, summarizing the symmetry data of a tone row with an integer 11-tuple. Under this scheme, rows whose 11-tuple values are small exhibit more repetitive motives (hence, greater symmetry) than those whose 11-tuple values are large. In an extreme example, the chromatic scale and the circle of fifths achieve scores of all 1's in this scheme. In this paper, we choose to map the full symmetry data to a differently defined 11-tuple: the maximum values of the partitions. The reason for using the maximum value instead of the length is subjective as both schemes capture essential information about the partition. Figure 5 depicts a tone row whose most-repeated motives of length 2, 3, and 4 have multiplicities of 3, 4, and 2, respectively. This particular tone row has no motives of length 5 or greater, meaning that its symmetries are quantified by the tuple $(3, 4, 2, 1, 1, 1, 1, 1, 1, 1, 1)$. We refer to this tuple as the *symmetry score* of the row class.

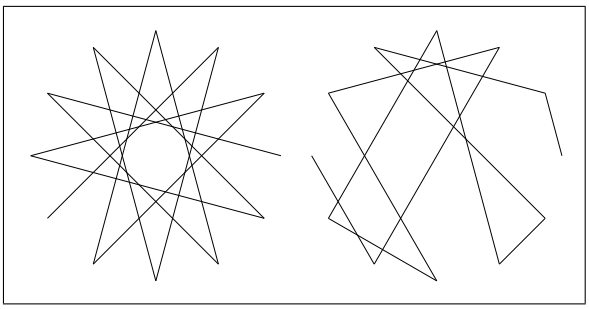


Figure 4. Clock diagrams of the maximally symmetric circle of fifths and a minimally symmetric all-interval row.

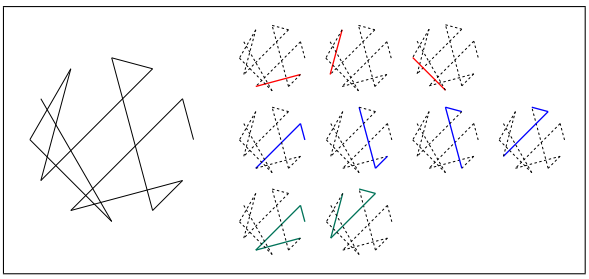


Figure 5. Clock diagrams of the tone row $(0, 1, 8, 11, 10, 3, 2, 7, 4, 6, 9, 5)$ showing that it achieves symmetry scores of 3, 4, and 2 for the lengths of 2, 3, and 4, respectively.

3. ANALYSIS

In Section 2 we established a map from the set of row classes to an 11-dimensional lattice with the property that greater values correspond to higher multiplicity of motives. The two clock diagrams in Figure 4 illustrate this property:

whereas the circle of fifths maps to a symmetry score of $(11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)$, the depicted all-interval row maps to a symmetry score of $(2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$ which is in fact the lowest possible multiplicity of motives among all row classes. The next step is to assess whether the Viennese tone rows contain unusually high levels of symmetry, but this depends on how we compare symmetry scores to each other. Instead of presenting just one possible interpretation of "high symmetry," we give four reasonable candidates that span a wide range of interpretations of symmetry while still respecting musical intuition and the mathematical literature on the topic. In each model, we conduct hypergeometric tests to gauge whether the incidence of high symmetry in the composers' corpora was significantly greater than would be predicted by chance alone. While these tests are exploratory in nature, the models in Sections 3.1 and 3.2 admit a conservative Bonferroni correction to adjust for the issue of multiple testing [14].

3.1 Reverse-Lexicographic Model

The most direct extension of Hunter and von Hippel's results is to introduce a sensible total order on the set of symmetry scores, then split the scores into "low" and "high" bins in music-theoretically sound ways and apply a hypergeometric test. A natural candidate for the total order is reverse-lexicographic order (RLEX), prioritizing longer motives over shorter ones. This choice has the advantage of seamlessly reproducing known results in its base case; moreover, it yields a natural definition of low and high symmetry: for each length d, the valid splits are the locations in the total order at which the d-symmetry score increments. For example, we can split the set of tone rows according to whether any length-4 motive is present; in the set of all row classes, 23.62% have a length-4 motive, yet fully 50% of Schoenberg's tone rows exhibit this symmetry. The associated hypergeometric test gives a p-value of $p < 0.0002$, so the prevalence of motives of length at least 4 in Schoenberg's work is unlikely to have arisen by chance alone. This model has 38 splits (40 for Berg) and the adjusted p-values are computed by multiplying by these numbers. Table 1 shows a selection of significant results for the three composers. Notably, more than 90% of Webern's corpus exhibits a statistically significant level of symmetry in the RLEX sense and both Schoenberg and Webern's corpora exhibit significant levels of symmetry under the multiple testing correction. Berg's use of a fourth symmetry and the correspondingly smaller space of row classes makes it more difficult to obtain significant results for his corpus, as already noted by Hunter and von Hippel. However, even in this case a few notable results are seen, although only on an exploratory basis.

3.2 Lattice Rank Model

The RLEX model faithfully reproduces known results, yet strongly penalizes row classes that contain many short motives but few long ones. The Lattice Rank model delivers on the opposite extreme by using a grading according to the sum of the symmetry score, thereby equally weighting motives of all lengths. Here, the natural divisions of "high" and "low" symmetry are determined by the rank function with 54 distinct levels (109 for Berg). For example, two-thirds of Webern's tone rows have lattice rank of 6 or greater (that is, 6 more multiplicity of motives than the lowest symmetry score), while only 11.17% of the set of all row classes has this property. The quantity of tone rows in Webern's corpus with at least this lattice rank almost certainly did not arise by chance alone ($p = 2.5 \times 10^{-9}$). Table 2 shows that the Lattice Rank model obtains remarkably similar results to the RLEX model in spite of being agnostic to motive length. Again, Schoenberg and Webern's corpora exhibit statistically significant levels of symmetry even under the multiple testing correction, while Berg's corpus only does so on an exploratory basis.

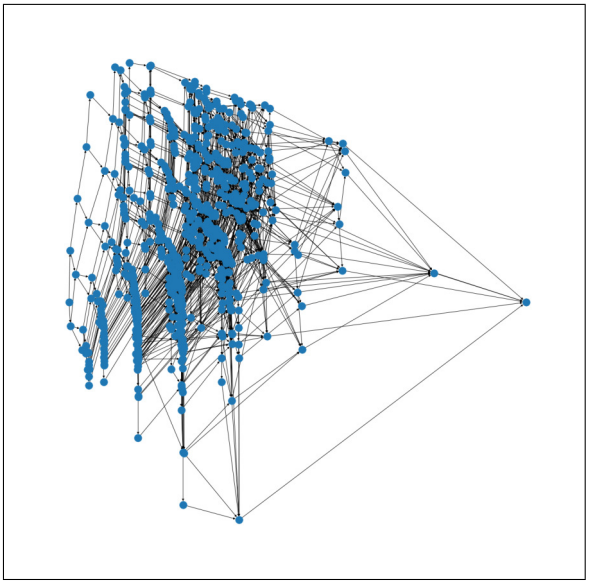


Figure 6. Directed acyclic graph representation for the partially ordered set of symmetry scores relevant to Schoenberg and Webern.

The fact that the RLEX and Lattice Rank models yield similar results in spite of applying diametrically opposed weighting schemes to motives of different lengths suggests that the Viennese tone rows simply contain a lot of internal symmetries no matter how you count them. Still, these models impose value judgments on the relative importance of each internal symmetry, possibly affecting the outcome of the analysis. To validate our findings, we offer a non-parametric alternative that makes no assumptions at all about the relative importance of the different motives. Instead, this alternative is based on the single intuitive assumption that the symmetry scores form a partially ordered set: a row class whose symmetry score is no greater in any component than the symmetry score of a different row class cannot be said to exhibit greater symmetry overall. Abstracting from the lattice structure in this way results in the diagrams in Figures 6 and 7, where each vertex is a symmetry score and the directed edges (all running left-to-right) run from lesser to greater in the partial order.

Schoenberg	All	p-value	Webern	All	p-value	Berg	All	p-value
76.19%**	45.50%	5.11×10^{-5}	90.48%***	39.15%	1.51×10^{-6}			
50.00%**	23.62%	1.88×10^{-4}	57.14%*	23.68%	1.01×10^{-3}	47.83%	28.84%	0.042
38.10%***	11.11%	5.47×10^{-6}	42.86%**	11.16%	2.23×10^{-4}			
11.90%	2.29%	2.67×10^{-3}	28.57%***	2.29%	5.89×10^{-6}	8.70%	1.04%	0.024
4.76%	0.13%	1.50×10^{-3}	19.05%***	0.13%	1.93×10^{-8}			

Table 1. Selected RLEX results for each composer, comparing their corpora to the set of all row classes with respect to various "high" and "low" symmetry bins. Asterisks indicate significance level under conservative Bonferroni correction. * : $p < .05$, ** : $p < .01$, *** : $p < 0.001$.

Schoenberg	All	p-value	Webern	All	p-value	Berg	All	p-value
83.33%	66.07%	0.011	95.24%	66.07%	1.96×10^{-3}			
47.62%**	20.87%	9.98×10^{-5}	71.43%***	20.87%	9.13×10^{-7}	56.52%	35.04%	0.029
33.33%**	11.17%	1.17×10^{-4}	66.67%***	11.17%	2.54×10^{-9}			
9.52%*	0.90%	5.48×10^{-4}	38.10%***	0.90%	7.48×10^{-12}	17.39%	4.50%	0.018
7.14%***	0.11%	1.47×10^{-5}	19.05%***	0.04%	1.38×10^{-10}			

Table 2. Selected Lattice Rank results for each composer, comparing their corpora to the set of all row classes with respect to various "high" and "low" symmetry bins. Asterisks indicate significance level under conservative Bonferroni correction. * : $p < .05$, ** : $p < .01$, *** : $p < 0.001$.

3.3 Partial Order Rating Models

In order to set up a hypergeometric test on the poset of symmetry scores, we still need to choose some linear extension of the poset to a total order. The RLEX and Lattice Rank models accomplish this by appealing to music intuitions on the relative importance of different kinds of motives. However, the fact that our poset has a unique least and greatest element gives us a non-parametric alternative. Given any symmetry score v, we count the number of row classes with strictly lesser scores (ascendants) and strictly greater scores (descendants) in the poset. If v has relatively few descendants compared to ascendants, then few row classes have strictly greater symmetry than v and we may say that v possesses a relatively high level of symmetry. To make this idea precise, let A_v be the number of row classes assigned to strictly lesser scores than v, and D_v be the number of row classes assigned to strictly greater scores than v. Lastly, let M be the total number of row classes. The formula

$$\frac{1}{2}(1 + (A_v - D_v)/M))$$

ranges from 0 to 1 as v ranges from the least to greatest elements of the poset. We define the Solo Poset Rating of v by renormalizing this formula to range from 0 to infinity, like so:

$$\begin{aligned} PR_s(v) &= -\log\left(1 - \frac{1}{2}(1 + (A_v - D_v)/M))\right) \\ &= -\log\left(\frac{1}{2}(1 - (A_v - D_v)/M)\right). \end{aligned} \quad (1)$$

In Figure 7, the symmetry score marked by the black vertex would receive a relatively high Solo Poset Rating because it has many ascendants but relatively few descendants. This imbalance is further amplified by the fact that scores near the minimum symmetry score are generally assigned to many more row classes than scores near the maximum symmetry score.

For example, two of Berg's tone rows have a symmetry score of $(6, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1)$, whose Solo Poset Rating is 0.695. An additional ten of Berg's tone rows achieve a greater Solo Poset Rating than this, so $12/23$ or 52.17% of Berg's tone rows exhibit a Solo Poset Rating of 0.695 or greater. Meanwhile, just 29.42% of all tone rows achieve a Solo Poset Rating of at least 0.695. If Berg had selected row classes at random, the probability that twelve or more of his corpus would have a Solo Poset Rating of at least 0.695 is just 0.018.

Figure 7 shows that for any given symmetry score, there are many other scores in the poset that are incomparable. We define the Cohort Poset Rating of a particular score as the average of the Solo Poset Ratings of all scores that are incomparable with the chosen score. Both variants of the Poset Rating quantify how close a given score is to the highest possible symmetry score, compared to its closeness to the lowest possible symmetry score. The two ratings yield nearly identical results when analyzing the corpora of Schoenberg and Webern (Tables 3 and 4). Due to Berg's smaller space of row classes, the only noteworthy finding for his corpus under the Poset Rating models is the one given in the previous paragraph (the Solo and Cohort models coincide in this case).

4. DISCUSSION

Our work takes a flexible approach to detecting patterns within tone rows, uncovering partial symmetries that cannot be detected by group theory. The trade-off is that we also find many internal structures that have no name in music theory. Whether this is a true disadvantage depends on

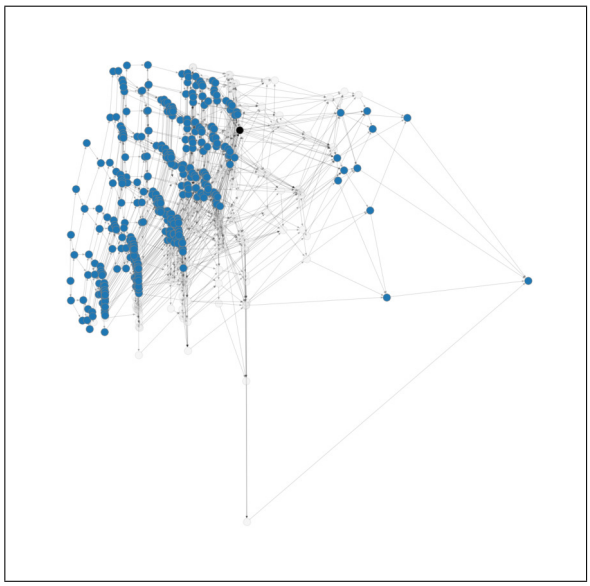


Figure 7. Directed acyclic graph representation of the symmetry scores relevant to Berg with highlighted ascendants and descendants of the black vertex, and incomparable scores greyed out.

Schoenberg	All Solo (Cohort)	p-value
83.33%	65.87%	0.010
52.38%	26.23%	2.73×10^{-4}
33.33%	12.85% (11.65%)	5.13×10^{-4}
9.52%	0.74% (0.60%)	2.73×10^{-4}
4.76%	0.74% (0.13%)	9.08×10^{-3}

Table 3. Selected Poset Rating results for Schoenberg, comparing his corpus to the set of all row classes with respect to various "high" and "low" symmetry bins. When the Solo and Cohort ratings do not coincide, the larger of the two p-values is displayed for brevity.

Webern	All Solo (Cohort)	p-value
95.24%	45.35%	1.62×10^{-6}
66.67%	12.85% (16.03%)	2.78×10^{-7}
52.38%	2.85%	2.67×10^{-12}
33.33%	1.03% (1.06%)	1.51×10^{-9}
19.05%	0.12% (0.25%)	2.39×10^{-7}

Table 4. Selected Poset Rating results for Webern, comparing his corpus to the set of all row classes with respect to various "high" and "low" symmetry bins. When the Solo and Cohort ratings do not coincide, the larger of the two p-values is displayed for brevity.

the validity of our basic premise: that the presence of a recurring motive is noteworthy in itself. We believe that this premise is supported by the preexisting scholarly interest in derived rows, those tone rows that are fully generated by a shorter repeating motive [10]. Since small adjustments to a derived row result in similar rows that yet lack any official symmetry, it seems natural to extend the concept of derived row to any tone row that is at least partially generated by a motive.

In bringing new methods to this topic, we have made particular effort to situate our work in the literature and reproduce known results where possible. Indeed, all our models agree with the existing knowledge that roughly 5% of Schoenberg's corpus and 19% of Webern's corpus have significant incidence of symmetry at the highest level. The RLEX model in particular achieves exact replication of known results. Where it was necessary for us to make a value judgment on the relative importance of different motives, we have followed the literature (in the case of RLEX) and also explored the diametrically opposed value judgment (in the case of Lattice Rank) in order to provide maximum contrast. In each case, we performed a multiple testing correction and still found significant levels of symmetry in much of the tone rows of Schoenberg and Webern. We also developed two non-parametric models to further validate our findings. Moreover, our approach to defining symmetry by the presence of shorter motives is analogous to von Hippel and Huron's measurement of the "tonalness" a row by the tonal properties of its subsets [7].

As for whether the Viennese twelve-tone composers favored symmetry in their work, our models support this across the board. We find that all three composers made extensive use of symmetry in their work, decidedly beyond what can be explained by chance alone. We found that 76%–83% of Schoenberg's corpus and 90%–95% of Webern's corpus contained significant levels of symmetry (up from 5% and 19%, respectively.) Even in the case of Berg, whose much smaller space of row classes limits the power of the hypergeometric test, we can extend the previously determined figure of 9% to the range of 48%–57% of his corpus although only on an exploratory basis.

Our combinatorial approach has one other advantage over the group-theoretic approaches in the literature. By forgetting the permutation group structure and studying motives on a purely combinatorial level, we no longer restrict ourselves to studying tone rows. Indeed, our methods apply just as well to any data series together with a relevant group of symmetries, with the one concession that there is no clear analog for the hypergeometric test. Still, the n-tuple symmetry score can serve as a feature for classification of melody and rhythm that encodes information of many different cardinalities, evoking concepts from persistent homology and wavelet analysis. While we assert no formal connection between these fields and our work, we take inspiration from the concept of studying an object at many different scales and hope that our success in the enumeration of tone row motives may find broader application in music information retrieval.

5. REFERENCES

[1] E. Haimo, "The evolution of the twelve-tone method," in *The Arnold Schoenberg Companion* (W. B. Bailey, ed.), pp. 101–128, Westport, CT: Greenwood Press, 1998

[2] K. Bailey, *The Twelve-Note Music of Anton Webern: Old Forms in a New Language*. Cambridge: Cambridge University Press, 1991.

[3] D. Headlam, *The Music of Alban Berg*. New Haven, CT: Yale University Press, 1996.

[4] J. M. Hauer, *Zwölftontechnik : die Lehre von den Tropen*. Wien: Universal Edition, 1925.

[5] E. Amiot, "Mathématiques et analyse musicale: une fécondation réciproque," in *Analyse Musicale*, vol. 28, pp. 37–41, 1992.

[6] E. Krenek, "Musik und Mathematik," in *Über neue Musik*, pp. 71–89, Vienna, Verlag der Ringbuchhandlung, 1937

[7] P. T. von Hippel and D. Huron, "Tonal and 'Anti-Tonal' Cognitive Structure in Viennese Twelve-Tone Rows," in *Empirical Musicology Review*, vol. 15, No. 1-2, 2020.

[8] D. J. Hunter and P. T. von Hippel, "How Rare is Symmetry in Musical 12-Tone Rows?," in *The American Mathematical Monthly*, vol. 110, No. 2, pp. 124–132, February 2003.

[9] M. Babbitt, "Some aspects of twelve-tone composition," in *The Score and I.M.A. Magazine*, vol. 12, pp. 53–61, 1955.

[10] H. Fripertinger and P. Lackner, "Tone rows and tropes," in *Journal of Mathematics and Music*, vol. 9, No. 2, pp. 111–172, 2015.

[11] M. Gotham and J. Yust, "Serial Analysis: A Digital Library of Rows in the Repertoire and their Properties, with Applications for Teaching and Research," in *DLfM '21: 8th International Conference on Digital Libraries for Musicology*, New York, NY: Association for Computing Machinery, 2021.

[12] C. Krumhansl, *Cognition foundations of musical pitch*, New York, NY: Oxford University Press, 1990.

[13] J. Yust, "Dimensions of Atonality: A Response and Extension of von Hippel and Huron (2020)," in *Empirical Musicology Review*, vol. 15, No. 1-2, pp. 119 - 119, 2020.

[14] M. Jafari and N. Ansari-Pour, "Why, When, and How to Adjust Your P Values?," in *Cell J.*, vol. 20, No. 4, 2019.

THE POWER OF DEEP WITHOUT GOING DEEP? A STUDY OF HDPGMM MUSIC REPRESENTATION LEARNING

Jaehun Kim
Delft University of Technology
Multimedia Computing Group
J.H.Kim@tudelft.nl

Cynthia C. S. Liem
Delft University of Technology
Multimedia Computing Group
C.C.S.Liem@tudelft.nl

ABSTRACT

In the previous decade, Deep Learning (DL) has proven to be one of the most effective machine learning methods to tackle a wide range of Music Information Retrieval (MIR) tasks. It offers highly expressive learning capacity that can fit any music representation needed for MIR-relevant downstream tasks. However, it has been criticized for sacrificing interpretability. On the other hand, the Bayesian nonparametric (BN) approach promises similar positive properties as DL, such as high flexibility, while being robust to overfitting and preserving interpretability. Therefore, the primary motivation of this work is to explore the potential of Bayesian nonparametric models in comparison to DL models for music representation learning. More specifically, we assess the music representation learned from the Hierarchical Dirichlet Process Gaussian Mixture Model (HDPGMM), an infinite mixture model based on the Bayesian nonparametric approach, to MIR tasks, including classification, auto-tagging, and recommendation. The experimental result suggests that the HDPGMM music representation can outperform DL representations in certain scenarios, and overall comparable.

1. INTRODUCTION

Deep learning became one of the most popular and successful methodologies to tackle a various range of MIR tasks; music classification [1], music generation [2], recommendation [3] and more. Some of the known benefits are 1) the high expressiveness towards any data structure, 2) effective ways to handle the overfitting, and finally, 3) the rapidly improving infrastructural supports, including both hardware (i.e., accelerators such as GPU) and software (i.e., deep learning frameworks). Fuelled by these benefits, DL models can learn useful representations of diverse data, which is one of the key reasons for its success [4]. On the other hand, DL models often are criticized for their lack of interpretability [5], also for applications within the MIR domain [6, 7].

© J. Kim and C. C. S. Liem. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J. Kim and C. C. S. Liem, "The power of deep without going deep? A study of HDPGMM music representation learning", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

According to [8, 9], interpretability of a machine learning model can be defined as the degree to which a user can understand the underlying mechanism of its operation. In this regard, DL models would be substantially less interpretable due to their complex internal structure than simpler—yet may be comparably expressive—alternatives such as hierarchical probabilistic models.

The Bayesian Nonparametric (BN) approach is such an alternative to achieving what DL does well in an interpretable way. As a nonparametric method, it can overcome underfitting by unlimited model capacity, while resisting overfitting through its Bayesian nature [10]. At the same time, the model can be interpretable, as the probabilistic model structure hypothesizes the data generation process in a relatively simple and humanly understandable manner.

In past decades, BN models have been applied in various MIR tasks. For example, they were applied to a latent representation of music audio in estimating music (self-) similarity [11, 12]. They also were employed to conduct harmonic and spectral analyses by decomposing the time-frequency representation of the music audio into a mixture of countable infinite latent components [13, 14]. These analyses also have shown to be useful for downstream tasks such as structural analysis [15] or music recommendation [16]. Further, a discriminative model based on the BN was developed for music emotion recognition [17].

While DL gained considerable popularity, it is striking BN models did not gain as much attention. As they promise similar high-level properties and strengths to DL, it will be worthwhile to explore to what extent they compare to modern DL models for MIR-relevant tasks. Therefore, in this work, we will assess music representations learned from BN models, and compare them under similar experimental control with modern deep neural network models. In particular, we consider the Hierarchical Dirichlet Process Gaussian Mixture Model (HDPGMM) [16, 18, 19] as our model of interest. To concretize the comparison, we employ the transfer learning experimental framework [20–22], which is commonly used for assessing the potential of learned representations. The contributions of this work can be listed as follows:

1. We explore and suggest insight into how "good" and transferable the HDPGMM representation is for a range of MIR tasks.
2. We provide an implementation of an efficient, GPU-

accelerated inference algorithm for HDPGMM, which can handle large-scale music datasets.

2. HDPGMM

In this section, [1] we introduce the Hierarchical Dirichlet Process Gaussian Mixture Model (HDPGMM) [12, 19]. We first discuss the Dirichlet Process Mixture Model (DPMM) upon which the HDPGMM is extended.

2.1 Dirichlet Process Mixture Model

The Dirichlet Process (DP) is an well-known stochastic process that draws random probability distributions, which often is described as a distribution over distributions [10]. One of the most common applications of DP is the infinite mixture model, thanks to the property of DP that one can draw distributions of an arbitrary dimensionality. Thus, the Dirichlet Process Mixture Model (DPMM) can find the appropriate number of components K based on the set of observations as part of the learning process.

Among several ways to represent DP, we introduce the stick-breaking construction [24] [2]. Stick-breaking constructs DP in a simple and general manner. Formally, it is as follows:

$$\begin{aligned} \beta'_k &\sim \text{Beta}(1,\gamma) & \phi_k &\sim H \\ \beta_k &= \beta'_k \prod_{l=1}^{k-1}(1-\beta'_l) & G_0 &= \sum_{k=1}^{\infty} \beta_k \delta_{\phi_k} \end{aligned} \tag{1}$$

where the two equations in the left column represent the draw of infinite dimensional weights β_k which sums to one. Notably, the distribution for β is also referred to $\beta \sim \text{GEM}(\gamma)$ [25]. In the right column, H denotes the base distribution from which variable $\phi_k \in \Phi$ is drawn. The right bottom equation defines the draw of the probability measure G_0, where δ_{ϕ_k} means the point mass centered at the component ϕ_k. Altogether, Eq. (1) constructs the DP $G_0 \sim \text{DP}(\gamma, H)$. Figure 1 depicts the process in a graphical way.

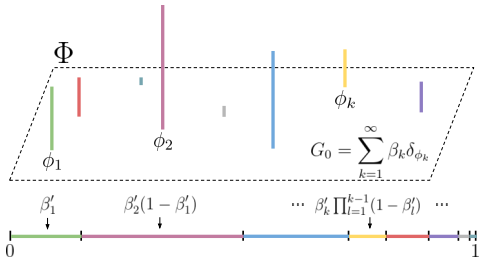


Figure 1: Illustration of stick-breaking construction.

In the mixture model context, we want to infer mixture components $\{\phi_1, \phi_2, \cdots, \phi_k, \cdots\}$ that fit to the data observations $\{x_1, x_2, \cdots, x_n\}$, which are assumed to be drawn from distributions $F(\phi)$ parameterized with variable ϕ (i.e., for a multivariate Gaussian F, $\phi = \{\mu, \Sigma\}$). We now can use DP for drawing the component ϕ that corresponds to x_i by introducing the cluster assignment variable $y_i \sim \text{Mult}(\beta)$:

$$\begin{aligned} \beta|\gamma &\sim \text{GEM}(\gamma) & \phi_k|H &\sim H \\ y_i|\beta &\sim \text{Mult}(\beta) & x_i|y_i, \{\phi_k\} &\sim F(\phi_{y_i}) \end{aligned} \tag{2}$$

where ϕ_{y_i} denotes the component parameter ϕ indexed by the assignment variable y_i corresponding to the ith observation x_i.

2.2 Hierarchical DPMM

In many data structures, groupings of atomic data points arise naturally (i.e., audio frames within a song, songs from an artist, words of lyrics). Hierarchical DP (HDP) is an extension of DP that models the "groupings" by imposing group-level DPs derived from the "corpus-level" DP as the global pool of components [19]. Following [18], the jth group-level DP can be expressed as follows:

$$\begin{aligned} \pi'_{jt} &\sim \text{Beta}(1,\alpha_0) & \psi_{jt} &\sim G_0 \\ \pi_{jt} &= \pi'_{jt} \prod_{s=1}^{t-1}(1-\pi'_{js}) & G_j &= \sum_{t=1}^{\infty} \pi_{jt} \delta_{\psi_{jt}} \end{aligned} \tag{3}$$

As seen above, HDP appears as the recursion of multiple levels of DPs [3]. Notably, the base distribution G_0 of each group-level DP is from the corpus-level DP. This relationship allows mapping group-level atoms ψ_{jt} to the corpus-level atoms ϕ_k. Wang et al. [18] introduce indicator variables $c_{jt} \sim \text{Mult}(\beta)$ which maps ψ_{jt} and ϕ_k by $\psi_{jt} = \phi_{c_{jt}}$, where β is drawn from the corpus-level DP in Eq. (1). It simplifies the model as we do not need to represent ψ_{jt} explicitly. Finally, we can represent HDPMM by introducing another indicator variable $z_{jn} \sim \text{Mult}(\pi_j)$ for the nth observation x_{jn} within the jth group, similarly to Eq. (2):

$$\begin{aligned} \pi_j|\alpha_0 &\sim \text{GEM}(\alpha_0) & \theta_{jn} &= \psi_{jz_{jn}} = \phi_{c_{jz_{jn}}} \\ z_{jn}|\pi_j &\sim \text{Mult}(\pi_j) & x_{jn}|z_{jn}, c_{jt}, \{\phi_k\} &\sim F(\theta_{jn}) \end{aligned} \tag{4}$$

where we use the indicator z_{jn} to select ψ_{jt}, which eventually is mapped as $\phi_{c_{jz_{jn}}}$ that represents the parameter θ_{jn} to draw the observation x_{jn}. HDPGMM is then defined by simply setting F as the (multivariate) Gaussian distribution and setting H as one of the distributions from which we can sample the mean and covariance (i.e., Gaussian-inverse Wishart distribution). Figure 2 depicts the HDPGMM graphically.

2.3 Inference Algorithm

We employ the online Variational Inference (VI) [18], which is a common and significantly faster choice than the Markov Chain Monte Carlo (MCMC) method to infer fully Bayesian models. VI approximates the true posterior by

[1] We adopted most of the notations from [18, 23].

[2] Literature commonly chooses the Chinese Restaurant Process (CRP) as an illustrative metaphor for DP, as it is intuitive and explains various properties of DP well [10]. We discuss DP with the stick-breaking construction, due to its further usage in the model inference within the work. Other metaphorical descriptions of DP are given in [10, 12]

[3] It implies naturally that multiple levels are possible (i.e., corpus - author - document) if it suits the data structure.

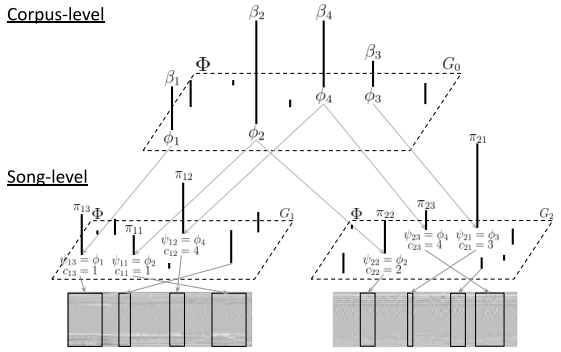


Figure 2: Illustration of 2-level HDPMM within corpus-song context. The top row depicts the corpus-level DP. The second row describes the draw of second-level (song-level) DPs per song from the corpus-level DP. Frame-level features are drawn from each song-level DPs, as depicted in the third row of the image.

minimizing the Kullback-Leibler (KL) divergence between the approximation $q(Z)$ and the true posterior $p(Z|X)$, where Z denotes the set of latent variables and parameters that we want to find, and X refers a set of observations [23]. One of the popular simplifications is the full factorization of the distribution $q(Z) = \prod_{i=1}^{|Z|} q_i(Z_i)$. In the context of HDPGMM, we have the following:

$$q(\beta', \pi', c, z, \phi) = q(\beta')q(\pi')q(c)q(z)q(\phi) \tag{5}$$

where β', π', c, z denote the corpus-level and group-level stick proportions, group-level component selection variable, and finally the observation-level component selection variable, respectively. ϕ refers to the parameter(s) for the F which draws the atomic observation, set as (multivariate) Gaussian in our context. Thus, ϕ includes the means μ and precision matrices Λ of each Gaussian component. We adopt the result from [23], where the Gaussian-Wishart distribution is used for the prior.

Each variational distribution further factorizes into:

$$\begin{aligned} q(\beta') &= \textstyle\prod_k^{K-1} \text{Beta}(\beta'_k|u_k, v_k) \\ q(\pi') &= \textstyle\prod_t^{T-1} \text{Beta}(\pi'_t|a_t, b_t) \\ q(c) &= \textstyle\prod_j \prod_t \text{Mult}(c_{jt}|\varphi_{jt}) \\ q(z) &= \textstyle\prod_j \prod_n \text{Mult}(z_{jn}|\zeta_{jn}) \end{aligned} \tag{6}$$

where u_k, v_k, a_t, b_t denote the variational parameters for the Beta distributions for corpus-level and group-level stick proportions, respectively. $\varphi_{jt} \in \mathbb{R}^K, \zeta_{jn} \in \mathbb{R}^T$ are the variational parameters for the Multinomial distribution to draw the selector c and z. Also, we truncate the infinite Beta distributions by K and T, which is a common way to avoid actually computing the infinite dimension [18]. With a sufficiently large number for the truncation, the model will still not be limited to the truncation and will only use the number of components optimal for the dataset. The final variational distribution is the Gaussian-Wishart distribution which draws the Gaussian parameters ϕ for F:

$$q(\phi) = \textstyle\prod_k^K \mathcal{N}(\mu_k|m_k, (\lambda_k\Lambda_k)^{-1})\mathcal{W}(\Lambda_k|W_k, \nu_k) \tag{7}$$

where we draw the precision $\Lambda_k \in \mathbb{R}^{d\times d}$ from the Wishart distribution with the variational parameter $W_k \in \mathbb{R}^{d\times d}$ and $\nu_k \in \mathbb{R}$, and the mean $\mu_k \in \mathbb{R}^d$ is drawn by the precision weighted by $\lambda_k \in \mathbb{R}$ and mean $m_k \in \mathbb{R}^d$.

We then can obtain the optimal model by maximizing the lower bound of the marginal log likelihood $\log p(X|Z)$ [23, 26, 27]:

$$\begin{aligned} &\log p(X|Z) \geq \mathbb{E}_q[\log p(X, \beta', \pi', c, z, \phi)] + H(q) \\ &= \textstyle\sum_j \big\{\mathbb{E}_q[\log\left(p(X_j|c_j, z_j, \phi)p(c_j|\beta')p(z_j|\pi'_j)p(\pi'_j|\alpha_0)\right)] \\ &\qquad + H(q(c_j)) + H(q(z_j)) + H(q(\pi'_j))\big\} \\ &\qquad + \mathbb{E}_q[\log p(\beta')p(\phi)] + H(q(\beta')) + H(q(\phi)) \end{aligned} \tag{8}$$

where $H(\cdot)$ denotes the entropy of given distribution, and $X_j = \{x_{j1}, x_{j2}, \cdots, x_{jN_j}\}$ is a set of observations within the jth group. [4]

Using the standard result of VI [18, 23, 27], the update rules for group-level parameters are derived as follows:

$$a_{jt} = 1 + \textstyle\sum_n \zeta_{jnt} \tag{9}$$

$$b_{jt} = \alpha_0 + \textstyle\sum_n \sum_{s=t+1}^T \zeta_{jns} \tag{10}$$

$$\varphi_{jtk} \propto \exp(\textstyle\sum_n \zeta_{jnt}\mathbb{E}_q[\log p(x_{jn}|\phi_k)] + \mathbb{E}_q[\log \beta_k]) \tag{11}$$

$$\zeta_{jnt} \propto \exp(\textstyle\sum_{k=1}^K \varphi_{jtk}\mathbb{E}_q[\log p(x_{jn}|\phi_k)] + \mathbb{E}_q[\log \pi_{jt}]) \tag{12}$$

Similarly, the update rules for the corpus-level parameters are as follows [23]:

$$u_k = 1 + \textstyle\sum_j \sum_{t=1}^T \varphi_{jtk} \tag{13}$$

$$v_k = \gamma + \textstyle\sum_j \sum_t^T \sum_{l=k+1}^K \varphi_{jtl} \tag{14}$$

$$\lambda_k = \lambda_0 + N_k \tag{15}$$

$$m_k = \lambda_k^{-1}(\lambda_0 m_0 + N_k \bar{x}_k) \tag{16}$$

$$W_k^{-1} = W_0^{-1} + N_k S_k + \frac{\lambda_0 N_k}{\lambda_0 + N_k}(\bar{x}_k - m_0)(\bar{x}_k - m_0)^\intercal \tag{17}$$

$$\nu_k = \nu_0 + N_k \tag{18}$$

where $\lambda_0 \in \mathbb{R}, m_0 \in \mathbb{R}^d, \nu_0 \in \mathbb{R}, W_0 \in \mathbb{R}^{d\times d}$ are the hyperparameters corresponding to the weight, location, degrees of freedom, and scale of Gaussian-Wishart distribution. The "sufficient statistics" and expectations in the update rules are defined as follows:

$$\begin{aligned} &\mathbb{E}_q[\log \beta_k] = \mathbb{E}_q[\log \beta'_k] + \textstyle\sum_{l=1}^{k-1} \mathbb{E}_q[\log(1 - \beta'_l)] \\ &\mathbb{E}_q[\log \beta'_k] = \Psi(u_k) - \Psi(u_k + v_k) \\ &\mathbb{E}_q[\log(1 - \beta'_k)] = \Psi(v_k) - \Psi(u_k + v_k) \\ &\mathbb{E}_q[\log \pi_{jt}] = \mathbb{E}_q[\log \pi'_{jt}] + \textstyle\sum_{s=1}^{t-1} \mathbb{E}_q[\log(1 - \pi'_s)] \\ &\mathbb{E}_q[\log \pi'_{jt}] = \Psi(a_{jt}) - \Psi(a_{jt} + b_{jt}) \\ &\mathbb{E}_q[\log(1 - \pi'_{jt})] = \Psi(b_{jt}) - \Psi(a_{jt} + b_{jt}) \\ &N_k = \textstyle\sum_j \sum_n r_{jnk} \\ &\bar{x}_k = \frac{1}{N_k} \textstyle\sum_j \sum_n r_{jnk} x_{jn} \\ &S_k = \frac{1}{N_k} \textstyle\sum_j \sum_n r_{jnk}(x_{jn} - \bar{x}_k)(x_{jn} - \bar{x}_k)^\intercal \end{aligned}$$

[4] Thus, the terms inside the sum over the group would further factorize into sums of per-group observations. We do not write these out for legibility and space considerations.

where $\Psi(\cdot)$ refers to the digamma function, and $r_{jnk} = \sum_{t=1}^{T} \zeta_{jnt}\varphi_{jtk}$ is the inferred "responsibility" of the nth observation of the jth group on the kth component. A standard batch update would compute statistics across the entire corpus, and update the corpus-level parameters. However, it may be slow or may suffer from too large or small numbers accumulated from a large-scale corpus.

We therefore employ the online VI, where corpus-level parameters are updated per mini-batch of groups passed. In this way, the early phase of model inference can be accelerated substantially compared to the full-batch update [18, 28]. The corpus-level update is then controlled by the learning rate $\rho_t = (\tau_0 + t)^{-\kappa}$, which is decayed over iterations parameterized by the offset $\tau_0 > 0$ and the rate $\kappa \in (0.5, 1]$:

$$Z_t = (1 - \rho_t)Z_{t-1} + \rho_t \tilde{Z}_t \tag{19}$$

where $\tilde{Z}_t$ denotes one of the corpus-level parameters from Eq. (13) to (18), which is updated by the given mini-batch at tth iteration, while Z_{t-1} is the current parameter. To scale the update with respect to the mini-batch size, we weight $\tilde{Z}$ by the factor of $w = \frac{|\tilde{X}|}{|X|}$, where $|X|, |\tilde{X}|$ denote the number of groups within the entire observation set X and the mini-batch of groups $\tilde{X}$. The overall inference algorithm is described in Algorithm 1.

Algorithm 1: Online VI for HDPGMM

```
Initialize φ = (φ_k)_{k=1}^K, u = (u_k)_{k=1}^K, v = (v_k)_{k=1}^K
  randomly and set t = 1.
while Stopping criterion is not met do
    Fetch a random mini-batch of groups X̃
    repeat
        Update a_j, b_j, ζ_j and φ_j using Eq. (9) to (12)
    until mini-batch likelihood stops improving;
    Compute u_k, v_k, λ_k, m_k, W_k, and ν_k using Eq. (13)
      to (18)
    Set ρ_t = (τ_0 + t)^{-κ}, t ← t + 1
    Update u_k, v_k, λ_k, m_k, W_k, and ν_k using Eq. (19)
end
```

Inspired by the implementation of [18, 27], we apply further regularization to the model. Specifically, we "splash" the uniform noise to the inferred responsibility r_{jn} as it can be biased if the groups are corrupted or incomplete, such as the preview audio of the entire song:

$$\tilde{r}_{jn} = (1 - \eta_t)r_{jn} + \eta_t e \tag{20}$$

where η_t is the regularization coefficient that determines the extent uniform noise $e = (e_k)_{k=1}^K$ is mixed into r_{jn}. $\eta_t = \frac{\eta_0 * 10^3}{(t+10^3)}$ also is defined as decaying function similar to the learning rate ρ_t.

3. EXPERIMENTAL SETUP

The overall experimental design is adopted from the recent music representation learning studies [20, 21, 29], where multiple downstream MIR tasks are tested with the feature set learned from the representation models. [5]

[5] The source code can be found at `https://github.com/eldrin/hdpgmm-music-experiments`.

3.1 Datasets, Features & Evaluation

We use a subset of Million Song Dataset (**MSD**) [30] as the dataset for the representation learning; more specifically, the audio preview excerpts from the MSD "train" subset introduced by [31, 32] [6]. For the evaluation of the representation, we employ three downstream tasks encompassing *music genre classification*, *music auto-tagging*, and *music recommendation*. For each target task, we chose the **GTZAN** dataset [33] with the fault-filtered split [34], the MagnaTagATune (**MTAT**) dataset [35] with the split from [36], and finally the Echo Nest taste profile subset as part of MSD (**Echonest**) [30] filtered by user and item frequency [37].

We evaluate the modeling using the cross-validation of task-specific prediction models. We fit the logistic regression models for GTZAN and MTAT datasets, taking the learned music representation as input and predicting the genre or music tags. We find the hyper-parameters of the logistic regression model by a randomized parameter search [38] for each run. As for the recommendation, we apply the item *K* Nearest Neighbor (*item-kNN*) method [39] where the song-song similarity is measured by the cosine similarity of the learned music representation. We adopt the data split introduced in [37]. For every trial, disjoint validation and test users are uniformly sampled, which includes 1142 users for each. Then 30% of listening records of these users are held out as testing records. We find the best *K* by conducting a grid search on the range $K \in \{2^4, 2^5, 2^6, 2^7, 2^8, 2^9\}$ by measuring the accuracy on the validation users. Using the best *K*, we compute the final score on the held out test records of testing users. We take the average score over 5 repetitions for each run to incorporate the random split effect.

We repeat 5 evaluation runs for each learning trial to consider the various random effects (i.e., initialization, randomized search). We employ the commonly used accuracy measures in each task to assess them, which are further elaborated in Table 1.

Id	# Samples	# Classes/# Users	Acc. Measure
MSD	$213,354$	N/A	N/A
Echonest	$40,980$	$571,355$ [7]	nDCG [40]
GTZAN	$1,000$	10	F1 [41]
MTAT	$25,863$	50	AUROC [42]

Table 1: Details on the datasets. We use the *macro* average strategy for F1 and Area Under Receiver Operating Characteristic curve (AUROC) measure, and we consider the top 500 recommended songs for computing the normalized Discounted Cumulative Gain (nDCG).

We select a set of audio features to infer the HDPGMM and non-DL baseline models inspired by [17]. The further details of the features can be found in Table 2. We use the implementation of `librosa` [43] with the default parameters for all the features.

[6] Length of preview audio differs per clip, where about 70% of samples are approximately 30 seconds and the rest are 1 minute, with a small subset longer than 1 minute.

[7] This refers the number of users in the dataset.

Feature	Aspect	Dim.	Transform
MFCC	Timbre	13	-
ΔMFCC	Timbre	13	-
ΔΔMFCC	Timbre	13	-
Onset Strength [44]	Rhythm	1	log
Chroma [45]	Tonal	12	log

Table 2: Details on the base audio features we employed for the experiment.

3.2 Comparisons

We evaluate 4 comparisons against HDPGMM, including DL-based and non-DL-based models.

- **G1** is the simplest model among the comparisons. The song-wise feature is represented by the concatenated parameters $\phi = \{\mu, \Sigma\}$ of the single multivariate Gaussian fitted on the frame-level base audio feature vectors within a song. We chose the square root of the diagonal in covariance, which means the feature is the concatenation of mean $\mu \in \mathcal{R}^{52}$ and standard deviation $\sigma \in \mathcal{R}^{52}$ of features.
- **VQ Codebook** can be deemed as the approximation of HDPGMM models [12]. First, a corpus-wise K-means clustering is fitted on the frame level features. Then, song-level feature is represented by the normalized frequency of cluster assignment of each frame-level feature within the song. [8] We set $K = 256$.
- **KIM** is a DL-based music representation trained in a simple self-supervised learning framework with a *VGG-like* [46] architecture [20]. We use the original implementation of the work, which takes the stereo mel-spectrogram of a 2.5 second-length clip as input. The representation is extracted from the last hidden layer before the prediction and summarized with global average and standard deviation pooling through the entire song.
- **CLMR** [29] is a recent DL-based music representation employing self-supervised learning through the contrastive learning method [47]. We employ the original implementation of [29] that uses the 2.67 second-length raw audio samples as input feature. [9] The representation is drawn from the last hidden layer and pooled with the global average over the entire song excerpt.
- **HDPGMM** uses the base audio feature with the whitening following [48]. As for song-level representation y_j, we employ the exponentiation of $\tilde{y}_{jk} = \exp(\mathbb{E}_q[\log p(X_j|c_j, z_j, \phi_k)])$ and normalize it over components and the length of the clip $y_{jk} = N_j^{-1} \frac{\tilde{y}_{jk}}{\sum_{k'=1}^{K} \tilde{y}_{jk'}}$. We adopt many hyper-parameter setups from the result of [18]; we set the truncation of components on the corpus level K and the song level T as 256 and 32 respectively, mini-batch size to 1024, learning rate parameters $\kappa = 0.6$ and $\tau_0 = 64$. We, however, explore the regularization rate $\eta_0 \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ and total corpus size $|X| \in \{2 \times 10^3, 2 \times 10^4, 213354\}$ to examine their effect on the representation. [10]

For all the comparisons except G1, we repeat 3 learning trials in consideration of random effects. Within a trial, we iterated the update for 100 epochs for KIM, CLMR, and HDPGMM. Further, we set the representation's dimensionality to 256, homogeneous across all comparisons except $G1$. Finally, we take the logarithm of the representation $\hat{y}_j = \log(\max(y_j, 10^{-8}))$ if it is given as the probability simplex (i.e., VQCodebook, HDPGMM).

4. RESULT AND DISCUSSION

4.1 Effect of Learning Factors

Overall, we observe that the learning factors of HDPGMM can make a substantial difference. As Figure 3 shows, the result suggests that the additional regularization in Eq. (20) generally affects positively the performance of all the downstream tasks we tested. It implies that the entire song inputs would likely improve the representation further, as the regularization crudely masks the effect of the missing data due to the preview clipping to some extent.

Further, we find that the number of training samples positively affects the performances in general. We measured the effect by repeatedly sub-sampling the training data 5 times in two different sizes mentioned in Section 3.2, and conducted the same evaluation applied for the full training set. The result shows that the positive effect is logarithmic, suggesting that the model should be learned with an exponentially larger dataset to get a linear improvement in the performance. The recommendation task, on the other hand, indicates the effect may not be linear, as we observe that the learning with 2×10^3 samples outperforms the 2×10^4 samples. However, the largest dataset leads to a better representation than the smallest datasets.

4.2 Model Comparison

Dataset	Measure	Model	Mean (±SD)
Echonest	nDCG	G1	0.0237 (± 0.0011)
		VQCodebook	0.0155 (± 0.0003)
		KIM	0.0257 (± 0.0015)
		CLMR	**0.0362 (± 0.0012)**
		HDPGMM	0.0221 (± 0.0006)
GTZAN	F1	G1	0.5396 (± 0.0049)
		VQCodebook	0.5777 (± 0.0022)
		KIM	0.5420 (± 0.0290)
		CLMR	**0.6375 (± 0.0368)**
		HDPGMM	**0.6467 (± 0.0069)**
MTAT	AUROC	G1	0.8441 (± 0.0006)
		VQCodebook	0.8386 (± 0.0013)
		KIM	0.8014 (± 0.0045)
		CLMR	0.8262 (± 0.0026)
		HDPGMM	**0.8482 (± 0.0020)**

Table 3: Evaluation results on downstream tasks.

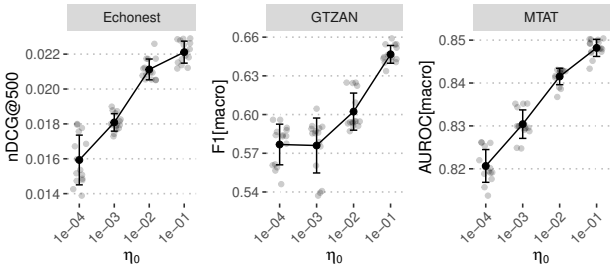
[8] It can be interpreted similarly to the inferred cluster assignment variable π_j in Eq. (4).

[9] Unlike the original work, we reduce the dimensionality of the hidden layer, from which we extract the representation, from 512 to 256 to control the representation dimensionality with other comparisons. Also, we do not apply the data augmentation for a fair comparison and also as it is beyond the main scope of the work. However, as it is reported that it can meaningfully improve the representation, we assume that applying the method would benefit other models similarly.

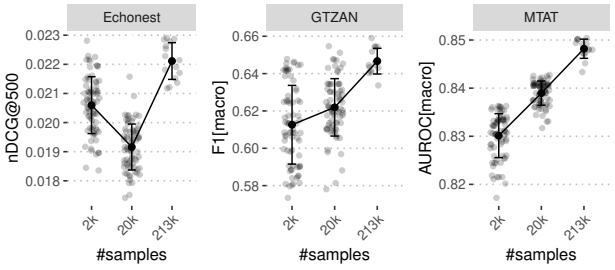
[10] The implementation can be found at `https://github.com/eldrin/pytorch-hdpgmm`.

Hip-Hop	country	female vocalists	pop	electronic	pop	oldies
pop	rock	singer-songwriter	female vocalists	dance	soul	blues
rnb	pop	pop	female vocalist	electronica	female vocalists	country
soul	oldies	acoustic	rock	funk	rnb	60s
male vocalists	indie	Mellow	Love	electro	dance	soul
rock	singer-songwriter	folk	dance	Hip-Hop	Hip-Hop	rock
0.0394	0.0308	0.0299	0.0269	0.0268	0.0248	0.0227

Table 4: Top tags for a few mostly "loaded" components (column-wise). The last row is the normalized total responsibility $\tilde{N}_k = \frac{N_k}{\sum_{k'}^{K} N_{k'}}$, meaning the proportional amount of songs having the component.



(a) Effect of regularization factor η_0. ($|X| = 213,354$)



(b) Effect of the number of training samples. ($\eta_0 = 10^{-1}$)

Figure 3: Effect of the learning factors on HDPGMM. Error bars indicate the standard deviations, and the grey dots are the raw data points.

Model comparison suggests that the HDPGMM model is comparable to the DL representation on average and can outperform it in a few specific scenarios. For the comparison, we chose the best HDPGMM model according to the result in Section 4.1 by setting $\eta_0 = 10^{-1}$ and using the entire dataset. As reported in Table 3, KIM achieves worse performance than HDPGMM on GTZAN and MTAT while performing better at recommendation (Echonest). On the other hand, CLMR, which adopts a more sophisticated learning strategy, achieves the best performance in Echonest, but performs worse than HDPGMM on MTAT and comparable on GTZAN. It is notable that the mixture model variants (VQCodebook, HDPGMM) perform particularly worse on the recommendation task. We hypothesize that the cosine similarity might not be an optimal choice for the probability simplex representation, or the feature set we consider may lack aspects that are crucial for the recommendation; this requires further study.

Except for this case, HDPGMM outperforms the other non-DL comparisons in general. It especially achieves significant improvement over its simplified version (VQCodebook) for all tasks, and mostly outperforms G1.

4.3 Component Interpretability

Finally, we explore the interpretability aspect of the HDPGMM model. To achieve it, we employ the *Lastfm-MSD* music tag dataset [30] where we find the mapping between the MSD songs and the social music tags. All our MSD training samples have mappings to the social tags, whose vocabulary size reaches roughly half a million. We compute the most relevant music tags for each corpus-level component found by the HDPGMM model. We estimate the relevance by a proxy measure $\alpha_{tk} = \sum_{j \in X: t \in j} N_j^{-1} \sum_n r_{jnk}$, where t denotes the tag index and r_{jnk} refers the responsibility computed in Eq. (20). To improve the clarity, we filter the most popular tags by weighting the measure by the Inverse Document Frequency (IDF) for each tag.

Table 4 shows an example from one of the HDPGMM with high regularization ($\eta_0 = 10^{-1}$). It suggests that the most frequent component (the first column) can be interpreted as the corpus-wide, universal topic. From second to the rest it represents a few other topics of music, such as *country-rock (column 1)*, *pop-female vocalist (column 2, 3, 5)*, and *electronic-dance (column 4)*. Notably, several compatible topics (i.e., columns 2, 3, and 5) exist, which can be interpreted as the collection of slightly different sub-clusters of an umbrella topic. It suggests that the advanced BN methods such as the hierarchical latent component models using the nested HDP [49] would further improve the representation.

5. CONCLUSION AND FUTURE WORK

In this work, we explore the potential of a BN model in the MIR task space. The results suggest that the HDPGMM representation achieves comparable effectiveness over the DL equivalents and outperforms a few scenarios. It implies that the method can be as effective as DL models when used in the supervised learning setup as well, by, for instance, jointly maximizing the variational lower bound both for the data generation and the prediction tasks. It has already been shown that such a joint objective approach can outperform the separated feature learning and transfer strategy that we demonstrate in this work [17].

Notably, HDPGMM is one of the more simple models in the BN approach. As mentioned above, there are models that are "deeper" such as nested DP models, which allow for a hierarchical structure of latent components [49]. The model also can be improved by introducing a sequential model such as Hidden Markov Models (HMM) as a base distribution [11], as the HDPGMM model assumes the exchangeability of tokens (feature frames), which is not ideal for a music signal. We left these possibilities to future work.

6. ACKNOWLEDGEMENT

A part of this work was carried out on the Dutch national e-infrastructure with the support of the SURF Coorperative.

7. REFERENCES

[1] M. Won, J. Spijkervet, and K. Choi, *Music Classification: Beyond Supervised Learning, Towards Real-world Applications*. https://music-classification.github.io/tutorial, 2021. [Online]. Available: https://music-classification.github.io/tutorial

[2] J. Briot, G. Hadjeres, and F. Pachet, *Deep Learning Techniques for Music Generation*, ser. Computational Synthesis and Creative Systems. Springer International Publishing, 2019.

[3] M. Schedl, "Deep learning in music recommendation systems," *Frontiers in Applied Mathematics and Statistics*, vol. 5, 2019.

[4] E. J. Humphrey, J. P. Bello, and Y. LeCun, "Moving beyond feature design: Deep architectures and automatic feature learning in music informatics," in *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, F. Gouyon, P. Herrera, L. G. Martins, and M. Müller, Eds. FEUP Edições, 2012, pp. 403–408.

[5] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. A. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *5th IEEE International Conference on Data Science and Advanced Analytics, DSAA 2018, Turin, Italy, October 1-3, 2018*, F. Bonchi, F. J. Provost, T. Eliassi-Rad, W. Wang, C. Cattuto, and R. Ghani, Eds. IEEE, 2018, pp. 80–89.

[6] B. L. Sturm, "A simple method to determine if a music information retrieval system is a "horse"," *IEEE Trans. Multim.*, vol. 16, no. 6, pp. 1636–1644, 2014.

[7] ——, "The "horse" inside: Seeking causes behind the behaviors of music content analysis systems," *Comput. Entertain.*, vol. 14, no. 2, pp. 3:1–3:32, 2016.

[8] C. Molnar, *Interpretable Machine Learning*, 2nd ed., 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book

[9] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artif. Intell.*, vol. 267, pp. 1–38, 2019.

[10] Y. W. Teh, "Dirichlet process," in *Encyclopedia of Machine Learning and Data Mining*, C. Sammut and G. I. Webb, Eds. Springer, 2017, pp. 361–370.

[11] Y. Qi, J. W. Paisley, and L. Carin, "Dirichlet process HMM mixture models with application to music analysis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2007, Honolulu, Hawaii, USA, April 15-20, 2007*. IEEE, 2007, pp. 465–468.

[12] M. D. Hoffman, D. M. Blei, and P. R. Cook, "Content-based musical similarity computation using the hierarchical dirichlet process," in *ISMIR 2008, 9th International Conference on Music Information Retrieval, Drexel University, Philadelphia, PA, USA, September 14-18, 2008*, J. P. Bello, E. Chew, and D. Turnbull, Eds., 2008, pp. 349–354.

[13] ——, "Finding latent sources in recorded music with a shift-invariant hdp," in *in International Conference on Digital Audio Effects (DAFx) (under review)*, 2009.

[14] M. Nakano, J. L. Roux, H. Kameoka, N. Ono, and S. Sagayama, "Infinite-state spectrum model for music signal analysis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*. IEEE, 2011, pp. 1972–1975.

[15] M. Nakano, Y. Ohishi, H. Kameoka, R. Mukai, and K. Kashino, "Bayesian nonparametric music parser," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2012, Kyoto, Japan, March 25-30, 2012*. IEEE, 2012, pp. 461–464.

[16] K. Yoshii and M. Goto, "Continuous plsi and smoothing techniques for hybrid music recommendation," in *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, K. Hirata, G. Tzanetakis, and K. Yoshii, Eds. International Society for Music Information Retrieval, 2009, pp. 339–344.

[17] J. Wang, Y. Lee, Y. Chin, Y. Chen, and W. Hsieh, "Hierarchical dirichlet process mixture model for music emotion recognition," *IEEE Trans. Affect. Comput.*, vol. 6, no. 3, pp. 261–271, 2015.

[18] C. Wang, J. W. Paisley, and D. M. Blei, "Online variational inference for the hierarchical dirichlet process," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, ser. JMLR Proceedings, G. J. Gordon, D. B. Dunson, and M. Dudík, Eds., vol. 15. JMLR.org, 2011, pp. 752–760.

[19] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Hierarchical dirichlet processes," *Journal of the American Statistical Association*, vol. 101, no. 476, pp. 1566–1581, 2006.

[20] J. Kim, J. Urbano, C. C. S. Liem, and A. Hanjalic, "One deep music representation to rule them all? A comparative analysis of different representation learning strategies," *Neural Comput. Appl.*, vol. 32, no. 4, pp. 1067–1093, 2020.

[21] K. Choi, G. Fazekas, M. B. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull, Eds., 2017, pp. 141–149.

[22] S. Dieleman, P. Brakel, and B. Schrauwen, "Audio-based music classification with a pretrained convolutional network," in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, A. Klapuri and C. Leider, Eds. University of Miami, 2011, pp. 669–674.

[23] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, January 2006.

[24] J. Sethuraman, "A constructive definition of Dirichlet priors," *Statistica Sinica*, vol. 4, pp. 639–650, 1994.

[25] J. Pitman, "Poisson-dirichlet and gem invariant distributions for split-and-merge transformations of an interval partition," *Comb. Probab. Comput.*, vol. 11, no. 5, pp. 501–514, 2002.

[26] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Mach. Learn.*, vol. 37, no. 2, pp. 183–233, 1999.

[27] D. M. Blei and M. I. Jordan, "Variational inference for Dirichlet process mixtures," *Bayesian Analysis*, vol. 1, no. 1, pp. 121 – 143, 2006.

[28] M. D. Hoffman, D. M. Blei, and F. R. Bach, "Online learning for latent dirichlet allocation," in *Advances in Neural Information Processing Systems 23: 24th Annual Conference on Neural Information Processing Systems 2010. Proceedings of a meeting held 6-9 December 2010, Vancouver, British Columbia, Canada*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 856–864.

[29] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, J. H. Lee, A. Lerch, Z. Duan, J. Nam, P. Rao, P. van Kranenburg, and A. Srinivasamurthy, Eds., 2021, pp. 673–681.

[30] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[31] J. Pons, O. Nieto, M. Prockup, E. M. Schmidt, A. F. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, E. Gómez, X. Hu, E. Humphrey, and E. Benetos, Eds., 2018, pp. 637–644.

[32] J. Lee, J. Park, K. L. Kim, and J. Nam, "Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, no. 1, 2018.

[33] G. Tzanetakis and P. R. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech Audio Process.*, vol. 10, no. 5, pp. 293–302, 2002.

[34] C. Kereliuk, B. L. Sturm, and J. Larsen, "Deep learning and music adversaries," *IEEE Trans. Multim.*, vol. 17, no. 11, pp. 2059–2071, 2015.

[35] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, K. Hirata, G. Tzanetakis, and K. Yoshii, Eds. International Society for Music Information Retrieval, 2009, pp. 387–392.

[36] J. Lee and J. Nam, "Multi-Level and Multi-Scale Feature Aggregation Using Pretrained Convolutional Neural Networks for Music Auto-Tagging," *IEEE Signal Processing Letters*, vol. 24, no. 8, pp. 1208–1212, Aug. 2017.

[37] D. Liang, R. G. Krishnan, M. D. Hoffman, and T. Jebara, "Variational autoencoders for collaborative filtering," in *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, P. Champin, F. Gandon, M. Lalmas, and P. G. Ipeirotis, Eds. ACM, 2018, pp. 689–698.

[38] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[39] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," *ACM Trans. Inf. Syst.*, vol. 22, no. 1, pp. 143–177, 2004.

[40] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, 2002.

[41] C. J. van Rijsbergen, *Information Retrieval*. Butterworth, 1979.

[42] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, 2006.

[43] B. McFee, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, and et al., "librosa/librosa: 0.9.1," Feb 2022.

[44] S. Böck and G. Widmer, "Maximum filter vibrato suppression for onset detection," in *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx-13)*, Maynooth, Ireland, September 2013.

[45] D. P. Ellis, Apr 2007, accessed: 2022-05-12. [Online]. Available: https://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[47] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, ser. Proceedings of Machine Learning Research, vol. 119. PMLR, 2020, pp. 1597–1607.

[48] J. Nam, J. Herrera, M. Slaney, and J. O. S. III, "Learning sparse feature representations for music annotation and retrieval," in *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, F. Gouyon, P. Herrera, L. G. Martins, and M. Müller, Eds. FEUP Edições, 2012, pp. 565–570.

[49] J. W. Paisley, C. Wang, D. M. Blei, and M. I. Jordan, "Nested hierarchical dirichlet processes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 2, pp. 256–270, 2015.

POP MUSIC GENERATION WITH CONTROLLABLE PHRASE LENGTHS

Daiki Naruse[1] **Tomoyuki Takahata**[1] **Yusuke Mukuta**[1,2] **Tatsuya Harada**[1,2]
[1] The University of Tokyo, Japan
[2] RIKEN, Japan
{naruse, takahata, mukuta, harada}@mi.t.u-tokyo.ac.jp

ABSTRACT

Research on music generation using deep learning has attracted more attention; in particular, Transformer-based models have succeeded in generating coherent musical pieces. Recently, an increasing number of studies have focused on phrases that are smaller musical units, and several studies have addressed phrase-level control. In this study, we propose a method for sequentially generating a piece that enables the control of each phrase length and, consequently, the length of the entire piece. We added PHRASE and a new event, BAR COUNTDOWN, which indicates the number of bars remaining in the phrase, to the existing event-based music representations. To reflect user input indicating the phrase lengths of the piece being generated, we used an autoregressive generation model that adds these two events to the generated event-token sequence based on the user input and uses it as input for the next time step. Subjective listening tests revealed that the pieces generated by our methods possessed designated phrase lengths and ended naturally at the determined length. [1]

1. INTRODUCTION

Music generation has been studied for more than half a century [1] and has advanced significantly in recent years with the development of deep learning. In many studies, deep neural sequence models such as recurrent neural networks (RNNs) and Transformers [2] have been used to model music. Transformer-based methods [3–7] have succeeded in generating coherent music throughout a piece. To apply these sequence models to music generation, it is necessary to represent a piece as a sequence of tokens. Event-based representations such as MIDI-like [8] and its advanced versions, REMI [6] and CP [7], have been used.

More recently, an increasing number of studies have focused on phrases and sections [9–14], which are smaller musical segments. These studies aimed to generate a structured piece that was divided into several segments and developed through repetition and transformation. Several studies addressed phrase-level control [11–13] and allowed the control of phrase attributes such as melody, rhythm, and harmonic fullness. Length is another important phrase attribute and is controllable with a phrase-by-phrase generation policy [13], which means that each phrase is generated independently and joined. However, this phrase-by-phrase generation policy has a limitation in that natural transitions between phrases are not guaranteed.

Therefore, we worked on controlling the phrase lengths with a sequential generation policy. The sequential generation policy, unlike the phrase-by-phrase or section-by-section generation policy [9, 13], is a method of sequentially generating an entire piece at once. We aim to create a model that outputs a piece according to user input regarding the configuration of the phrases and the length of each phrase, as shown in Figure 1 (the detailed generation process is described in Sections 3.3 and 3.4). The controllability of each phrase length implies that the length of the entire piece can be controlled. To control the length of each phrase and the entire piece, we extended two recently used event-based music representations, REMI [6] and CP [7]. The random timing of the switching of phrases and the end of the generation in the existing representations is likely due to the model not knowing which phrase and where it is generating. Therefore, we added PHRASE and a new event, BAR COUNTDOWN, which indicates the number of bars remaining in the phrase, to REMI and CP. To reflect the user input, we used an autoregressive generation method in which these two events were added based on the user input to the generated event-token sequence, and the sequence was entered into the model again. To evaluate this approach, two subjective listening tests were conducted for the length of each phrase and the entire piece. By comparing our methods with the dataset and existing methods, we demonstrate our methods are effective in length control.

Our contributions are summarized as follows:

- We extended the existing music representations (REMI [6] and CP [7]) by adding PHRASE and BAR COUNTDOWN events and showed that both events are necessary for length control.
- We enabled the reflection of the user input by an autoregressive generative model that adds the PHRASE and BAR COUNTDOWN events to the generated event-token sequence based on the user input and uses it as input for the next time step.

© D. Naruse, T. Takahata, Y. Mukuta, and T. Harada. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** D. Naruse, T. Takahata, Y. Mukuta, and T. Harada, "Pop Music Generation with Controllable Phrase Lengths", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] Samples of the generated pieces are available at https://mil-tokyo.github.io/phrase-length-designated-music-generation/.

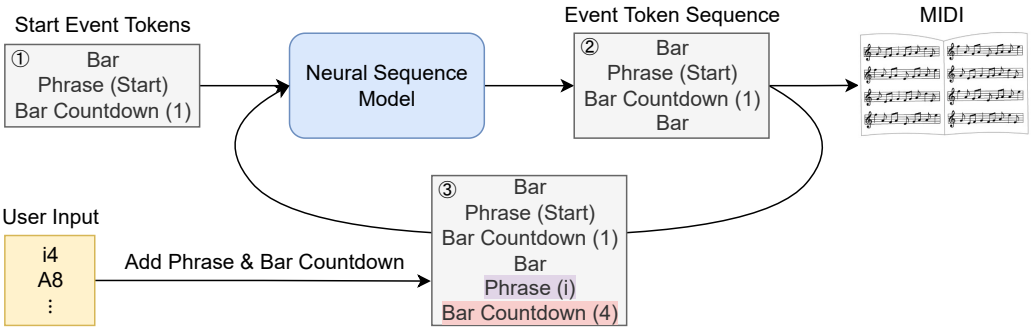


Figure 1: Generation process reflecting user input regarding phrase lengths.

2. RELATED WORK

2.1 Event-based Music Representations

To apply neural sequence models to music generation, music must be represented using a token sequence. Many studies [3–5, 15] adopted MIDI-like [8]. NOTE ON and NOTE OFF events indicate the start and end of a note, respectively, and a TIME SHIFT event advances the time step.

In MIDI-like, bars and beats are implicit, which are clearly indicated in the score, and it is difficult for the model to learn beat regularity and rhythmic structure. The model also has difficulty learning that the NOTE ON and NOTE OFF events must exist in pairs. To address these problems, an improved music representation called REMI (revamped MIDI-derived events) [6] was proposed. In REMI, the TIME SHIFT event in MIDI-like is replaced with BAR and BEAT events, and the NOTE OFF event is replaced with a NOTE DURATION event. TEMPO and CHORD events are added for clear harmony and expressive rhythmic freedom.

Later, a further extension of REMI called CP (compound word representation) [7] was suggested. In CP, consecutive and related events are grouped and placed in the same time step. Specifically, BAR, BEAT, CHORD, and TEMPO events are grouped into a METRICAL family and note-related events into a NOTE family. Additionally, a new event, EOS, is added to mark the end of a piece.

2.2 Latest Transformer-based Music Generation

Recently, Transformer-based methods have successfully generated coherent music throughout a piece and have become common in automatic composition in the symbolic domain. The Music Transformer study [3] was the first to apply the Transformer model to music generation. This study used MIDI-like to generate pieces by autoregressively predicting an event token at each time step. The Pop Music Transformer study [6] proposed REMI and generated pieces with a better rhythmic structure. The Jazz Transformer study [16] addressed the generation of Jazz. An attempt was made to introduce structure by adding the following four structure-related events to REMI: PHRASE, MLU, PART, and REPETITION. The CP Transformer study [7] proposed CP. Predicting events of the same family simultaneously at each time step significantly reduces the length of the token sequence, resulting in faster learning and inference. In addition, the EOS event allowed the model to complete the generation with the natural closure.

HAT (Harmony-Aware Hierarchical Music Transformer) [14] focused on phrases and sections. It represents music in CP and uses three Transformers hierarchically to allow event tokens to interact at different levels. The structure of the pieces was improved by learning the texture and the form jointly bridged by the harmony.

MusicFrameworks [13] is a monophonic melody generation system that enables phrase-level control. Music is described using music frameworks, a hierarchical music structure representation, and melodies are generated through a multi-level generative process. The manipulation of music frameworks allows control over phrase attributes such as structure, melody, and rhythm. Phrase length can also be controlled by modifying the structural information and length of the rhythm and chord information in each phrase. However, because of the phrase-by-phrase generation policy, there is no guarantee that transitions between phrases are natural.

3. METHOD

3.1 Phrase-Length Designated Music Generation

In this study, we worked on an automatic composition task that allowed control over the length of each phrase (and the length of the entire piece) with a sequential generation policy. The user inputs the configuration of the phrases and the length of each phrase in units of bars, and a piece with the designated phrase lengths and total length is generated. For example, if the input is "i4 A8 B8 o4," then a piece is generated with four bars for the intro phrase, eight bars for phrase A, eight bars for phrase B, and four bars for the outro phrase, with a total length of 24 bars.

3.2 PHRASE and BAR COUNTDOWN Events

We added the following two events to REMI [6] and CP [7]: PHRASE and BAR COUNTDOWN.

The first one, PHRASE, is an event that indicates to which phrase a bar belongs, e.g., PHRASE (i) refers to the intro phrase. This event was first proposed in the Jazz Transformer study [16]. In our study, PHRASE (Start) and PHRASE (End) were used in addition to phrase labels in the

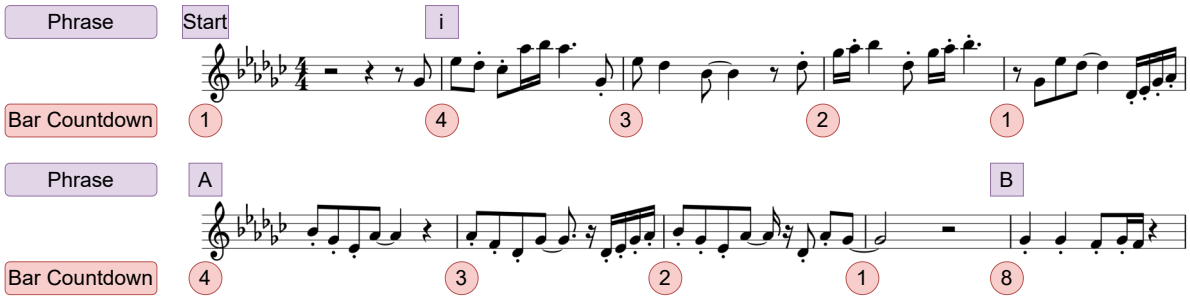


Figure 2: PHRASE and BAR COUNTDOWN events in the staff notation. This is the preprocessed melody part of '001.wav' in POP909 [17].

score. PHRASE (Start) represents one bar before the piece begins. This bar exists in all pieces to consider an anacrusis (or auftakt). Pieces that begin with an anacrusis have some notes in this bar, while those without an anacrusis promptly move to the next bar. PHRASE (End) is placed at the end of the event-token sequence and represents the end of the piece.

The second event, BAR COUNTDOWN, indicates the number of bars remaining in a phrase. If four bars remain, it is expressed as BAR COUNTDOWN (4), and the number of bars is counted for each bar.

These two events are expected to allow the model to know which phrase and where it is generating and to adjust the generation toward the turn of the phrase and the end of the piece. The correspondence between the two events and the musical score is shown in Figure 2. In REMI, the PHRASE and BAR COUNTDOWN events are placed just after the BAR event, which represents a bar line. In CP, these two events are placed along all the time steps. We slightly modified the original settings of CP in the CP Transformer [7] to decompose the METRICAL family into BAR and POS families. The BAR family represents a bar line instead of the BAR event, and the POS family groups the BEAT and CHORD events. In both REMI and CP, performance-related events, NOTE VELOCITY and TEMPO, were not used to reduce the burden of learning. We refer to these extended REMI and CP as **REMI + Ph&BC** and **CP + Ph&BC**, respectively (Ph and BC represent PHRASE and BAR COUNTDOWN, respectively). A list of events used in each representation is shown in Table 1, and an example of each event-token sequence is shown in Figure 3.

3.3 Reflection of User Input

We propose an autoregressive generation method to reflect user input regarding phrase lengths when generating pieces. As shown in Figure 1, event tokens are first predicted using the trained neural sequence model. Before using the predicted event-token sequence as the model input, the PHRASE and BAR COUNTDOWN events are added to the appropriate places based on the user input. This method is expected to enable the input of the phrase lengths that do not exist in the training data because they are present in BAR COUNTDOWN events.

REMI + Ph&BC	**CP + Ph&BC**	
Event	**Event**	**Family**
Note On Note Duration	Note On Note Duration	Note
Bar	-	Bar
Beat Chord	Beat Chord	Pos
Phrase **Bar Countdown**	**Phrase** **Bar Countdown**	[All]
-	Conti	[All]

Table 1: Events in REMI + Ph&BC and CP + Ph&BC.

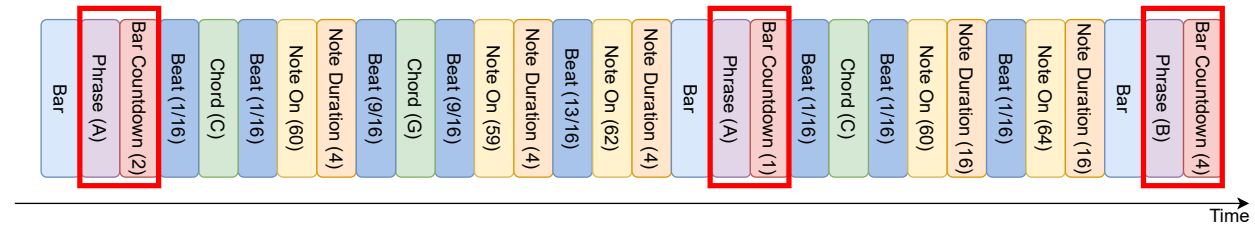
3.4 Pipeline

The same Transformer-based model was used as in the Pop Music Transformer [6] and the CP Transformer [7]. During the training stage, the MIDI file, chord annotation, and phrase annotation of each piece in the dataset are converted into the REMI + Ph&BC or CP + Ph&BC event-token sequence. The model is trained to predict the next event tokens from the input event-token sequence. The pipeline during the generation stage is illustrated in Figure 1. First, BAR, PHRASE (Start), and BAR COUNTDOWN (1) are input to the model to start the generation. Then, as described in Section 3.3, the next event tokens are predicted by the model, and after adding the PHRASE and BAR COUNTDOWN events based on user input, the event-token sequence is again entered into the model. By repeating this process to predict the event tokens sequentially, a piece is generated probabilistically. Once the model has generated the designated number of bars, it adds PHRASE (End) and terminates the generation.

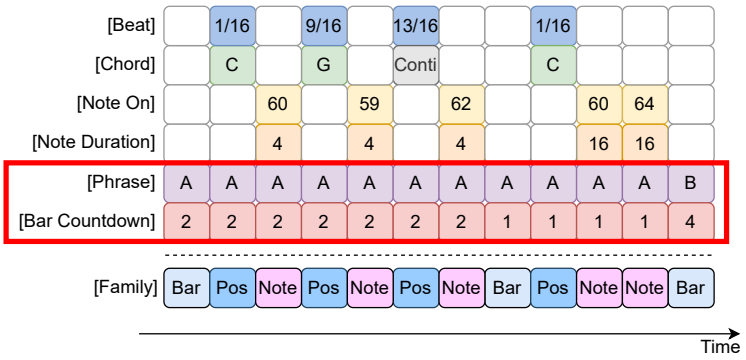
4. EXPERIMENTS

4.1 Dataset

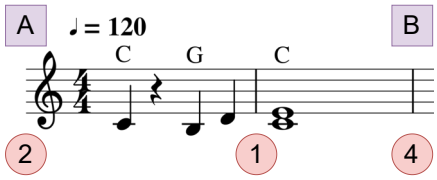
In this study, POP909 [17] is used as the MIDI dataset. The dataset consists of 909 pieces that are piano arrangements of pop songs by professional musicians and is divided into three parts: vocal melody, secondary melody or lead instrument melody, and piano accompaniment. We also used algorithmic chord annotations included in POP909 and the



(a) REMI + Ph&BC



(b) CP + Ph&BC



(c) Musical score

Figure 3: Examples of REMI + Ph&BC (a) and REMI + Ph&BC (b) event-token sequences and the corresponding staff notation (c).

human phrase annotations provided by Dai et al. [18].

After selecting pieces with 4/4 time signatures and excluding those whose downbeats were not aligned with the bar lines, 763 MIDI files were obtained. As a preprocessing step, we merged the three parts and quantized each piece into a 16th-note grid. Next, we shifted all pieces so that the first complete bar was the second to consider the pieces with an anacrusis, as described in Section 3.2. Furthermore, as the number of the intro and outro phrases was smaller than that of the other phrases, we extracted the first and last parts of the pieces and performed data augmentation by transforming their keys in the range of -3 to $+3$. We also modified the chord annotations. All flats on the root note were changed to sharps, and the chord types were limited to the following six types: maj, min, dim, aug, sus4, and sus2.

4.2 Overview of Evaluation Methods

In this study, the length of each phrase and the entire piece was evaluated. In the evaluation of each phrase length, we determined whether each phrase had a designated length by locating the boundary at which the phrase changed. The controllability of the overall length was determined based on whether it ended naturally or abruptly.

For an objective evaluation, methods that can automatically detect phrase boundaries and calculate a naturalness score for the end of a piece are required; however, no suitable methods are available (details are discussed in Section 4.6). In this study, we did not conduct an objective evaluation; rather, we conducted a subjective evaluation.

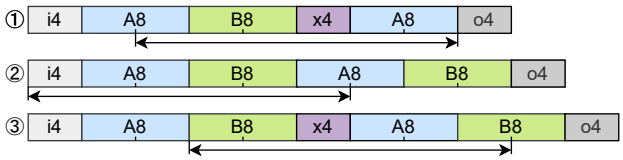
For the subjective evaluation, we administered two listening tests: one to divide a piece into several phrases and the other to evaluate the naturalness of the end. The details are provided in Section 4.4.

4.3 Comparative Methods

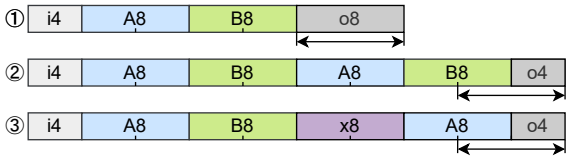
First, we compared our REMI + Ph&BC and CP + Ph&BC with the existing music representations, REMI [6] and CP [7]: **REMI** and **CP**. Since these cannot control the phrase lengths, we evaluated them only for closure. In these methods, we used the events and families shown in Table 1, excluding PHRASE and BAR COUNTDOWN, for comparison under the same conditions. In REMI, generation cannot be terminated by the model; instead, the model is forced to terminate when the number of generated bars reaches the target number. In CP, the model can naturally end a piece, although it cannot control the length of the piece. The final parts of the generated pieces were used for the evaluation.

Next, for the ablation studies, we compared REMI + Ph&BC and CP + Ph&BC with methods with a lower number of events. First, we compared REMI + Ph&BC with **REMI + Ph$_{\text{fewer}}$&BC**, a method that places PHRASE events only at the beginning of phrases. Note that in CP, the number of time steps does not change even if the number of PHRASE events is reduced, so CP + Ph$_{\text{fewer}}$&BC was not evaluated. We also compared our method with methods that used only one of the two events: **REMI + BC**, **REMI + Ph**, **CP + BC**, and **CP + Ph**.

The pieces in the POP909 dataset were also evaluated: **POP909**. Additionally, we intentionally created pieces that ended abruptly by cutting them off in the middle: **POP909$_{\texttt{cut}}$**. This method was used only when evaluating the closure.



(a) Test 1



(b) Test 2

Figure 4: Inputs of the generated pieces used for the evaluation. One division represents four bars. The segments indicated by arrows were extracted and used.

4.4 Subjective Evaluation

We conducted the following two online listening tests using Amazon Mechanical Turk:

Test 1 Phrase Boundary Detection
We investigated whether each phrase had a designated length or not. The participants listened to a piece while looking at the score and divided it into phrases. They were told in advance how many phrases there were, and they were asked to identify the phrase boundaries.

Test 2 4-Grade Evaluation of Closure
We examined whether the pieces ended naturally or abruptly. The subjects listened to a piece and answered whether the closure was natural or abrupt on a 4-point Likert scale: "Natural," "Somewhat natural," "Somewhat abrupt," and "Abrupt."

Since the majority of phrases in POP909 are four and eight bars in length [12], three inputs, consisting of four- and eight-bar phrases, were used for generation. The concrete inputs are shown in Figure 4. To reduce the burden on the subjects, in Test 1, 24 bars from the middle of a piece containing four phrases, and in Test 2, eight bars from the end were extracted and used for the evaluation. We generated 50 pieces per input and randomly selected two pieces, i.e., six pieces (three inputs $\times$ two pieces) were evaluated for each method.

First, qualification tests were conducted using the dataset to select those who understood each task and performed well, i.e., conformed to the dataset. In Test 1, 24 subjects correctly identified at least six of the nine phrase boundaries for three POP909 pieces, 13 of whom had more than one year of musical experience. In Test 2, 29 subjects answered "Natural" or "Somewhat natural" for two POP909 pieces and "Abrupt" or "Somewhat abrupt" for two $\text{POP909}_{\text{cut}}$ pieces, 12 of whom had more than one year of experience.

We then tested eight methods (except REMI, CP, and $\text{POP909}_{\text{cut}}$) in Test 1 and all 11 methods in Test 2. Each test was performed 60 times, and one piece per method was evaluated per test. In Test 1, the score was based on the percentage of correct answers, whereas in Test 2, "Natural" was scored as 4 points and "Abrupt" as 1 point.

Method	Test 1	Test 2
POP909	0.778	(3.67)
$\text{POP909}_{\text{cut}}$		1.29
REMI + Ph&BC (ours)	**0.633**	**3.00**
REMI + Ph_{fewer}&BC	0.550	2.34
REMI + BC	0.400	1.92
REMI + Ph	0.356	1.27
CP + Ph&BC (ours)	**0.583**	**3.28**
CP + BC	0.461	2.25
CP + Ph	0.306	1.78
REMI		1.35
CP		(3.17)

Table 2: Average percentage of correct answers for phrase boundaries in Test 1 and the average score of the four-grade evaluation of the closure in Test 2. For both tests, higher scores indicate better performance. The bracketed score in Test 2 indicates that it was evaluated with pieces that were not of the designated length.

4.5 Results

The average percentage of correct answers for phrase boundaries in Test 1 and the average score of the four-grade evaluation of the closure in Test 2 are listed in Table 2. In addition, the one-tailed t-test scores comparing our two methods to the other methods are shown in Table 3.

In Test 1, the scores of REMI + Ph&BC and CP + Ph&BC were much higher than 0.130 ($= 3/23$), the score when the phrase boundaries were answered at random. Compared with POP909, the scores were significantly lower (Table 3). It is suggested that our methods are effective in controlling the phrase lengths. When compared to methods used in the ablation studies, our methods scored significantly higher than methods with one of the two events. This indicates that both PHRASE and BAR COUNTDOWN events are required to control the phrase lengths. No significant differences were found between REMI + Ph&BC and REMI + Ph_{fewer}&BC. Thus, to control the phrase lengths, the number of events can be reduced by placing the PHRASE event only at the beginning of the phrase.

In Test 2, our scores exceeded 2.5, which was in the middle of the score range. They were significantly lower than the POP909 score but significantly higher than the $\text{POP909}_{\text{cut}}$ score (Table 3). Furthermore, they did not differ from the CP score, where the generated pieces ended naturally. Therefore, it can be said that our methods can end a piece naturally at a designated length. When compared to the methods used in the ablation studies, our methods scored significantly higher than all other methods. This indicates that both PHRASE and BAR COUNTDOWN events are required to control the length of the piece. These results

Method	Test 1		Test 2	
POP909	$\mu < \mu_m{}^{**}$	$(p = 0.0022)$	$\mu < \mu_m{}^{***}$	$(p = 3.2 \times 10^{-5})$
$\text{POP909}_{\text{cut}}$			$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 2.0 \times 10^{-17})}$
REMI + Ph_{fewer}&BC	$\mu > \mu_m$	$(p = 0.082)$	$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 7.6 \times 10^{-4})}$
REMI + BC	$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 1.1 \times 10^{-4})}$	$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 2.5 \times 10^{-7})}$
REMI + Ph	$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 7.4 \times 10^{-7})}$	$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 1.1 \times 10^{-17})}$
REMI			$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 6.9 \times 10^{-16})}$
CP			$\mu < \mu_m$	$(p = 0.18)$

(a) Results of the t-test between REMI + Ph&BC and other methods.

Method	Test 1		Test 2	
POP909	$\mu < \mu_m{}^{***}$	$(p = 1.6 \times 10^{-4})$	$\mu < \mu_m{}^{***}$	$(p = 5.8 \times 10^{-4})$
$\text{POP909}_{\text{cut}}$			$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 1.0 \times 10^{-31})}$
CP + BC	$\boldsymbol{\mu > \mu_m{}^{*}}$	$\boldsymbol{(p = 0.024)}$	$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 3.3 \times 10^{-9})}$
CP + Ph	$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 1.6 \times 10^{-6})}$	$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 3.2 \times 10^{-17})}$
REMI			$\boldsymbol{\mu > \mu_m{}^{***}}$	$\boldsymbol{(p = 1.6 \times 10^{-27})}$
CP			$\mu > \mu_m$	$(p = 0.22)$

(b) Results of the t-test between CP + Ph&BC and other methods.

Table 3: One-tailed t-test scores comparing our two methods to the other methods in Tests 1 and 2. μ and μ_m denote the average score of our method and each of the other methods, respectively. ${}^{*}p < .05, {}^{**}p < .01, {}^{***}p < .001$. A p-value less than 0.05 was considered statistically significant.

also highlight the importance of placing the PHRASE event in every bar (REMI + Ph&BC), not just at the beginning of the phrase (REMI + Ph_{fewer}&BC).

4.6 Discussion

Comparing the method that uses only the PHRASE event and the one that uses only the BAR COUNTDOWN event, the BAR-COUNTDOWN-only method scored higher, regardless of the test or representation (Table 2). This means that the BAR COUNTDOWN event was more effective in controlling the length of each phrase and the entire piece. This is consistent with the role of the BAR COUNTDOWN event in teaching the model the number of bars until the end of the phrase. The reasons why adding the PHRASE event to BAR-COUNTDOWN-only method would improve scores are inferred as follows. In Test 1, the PHRASE event indicates that the phrase has changed and may play a role in making the boundaries of the phrase more distinct. In Test 2, the event can tell the model that the phrase being generated is the outro phrase, and the piece is almost over.

A two-tailed t-test was performed to determine whether there was a significant difference between REMI + Ph&BC and CP + Ph&BC. The results showed that there was no significant difference between these methods, with $p = 0.42$ for Test 1 and $p = 0.11$ for Test 2. We can say that both representations achieve equally good results. These two events could potentially be used for new event-based representations derived from REMI and CP. In addition, although the Transformer was used as the model in this study, it is expected to be widely applied to sequence models that perform autoregressive generation.

As mentioned in Section 4.2, there are no objective evaluation metrics suitable for this study; therefore, we did not conduct an objective evaluation but only a careful subjective evaluation. Although the fitness scape plot [19, 20] has been used in studies focusing on the generation of music structures [14, 16], it cannot be used for phrases that do not necessarily repeat, as in this study. Although several algorithms have been proposed to determine phrase boundaries [21], they cannot be adopted because of the low correctness rate when applied to POP909 pieces. The development of phrase-segmentation research and the establishment of objective evaluation methods are required.

One limitation of this study is that it was not possible to create repetitions by designating the same phrase labels in the input. For example, if the input is "A4 B4 A4," the two phrases A are completely different. A possible reason is that the model cannot refer to the next phrase label; therefore, the piece is not connected to the beginning of the next phrase, which was previously defined. A mechanism for making such distant phrases with the same label alike is necessary and is an issue for the future.

In addition, the following points need to be addressed in future studies: (1) combining our methods with music theory to achieve more natural pieces, especially in terms of phrase transition and closure. (2) evaluating length diversity because only a few types of lengths were used because of the convenience of the evaluation. (3) controlling other phrase attributes such as emotions.

5. CONCLUSION

In this study, we proposed a method to control the length of each phrase and the entire piece using a sequential generation policy. In this method, two events are added to the existing event-based music representations: the PHRASE event, which indicates the phrase to which the bar belongs, and the BAR COUNTDOWN event, which indicates the number of bars remaining in the phrase. To reflect the user input, an autoregressive generative model is used that adds these two events based on the user input to the previously generated event-token sequence and uses it as input for the next time step. Subjective listening tests indicated that adding two events effectively controlled the length of each phrase and the entire piece.

6. ACKNOWLEDGEMENT

This work was partially supported by JST AIP Acceleration Research JPMJCR20U3, Moonshot R&D Grant Number JPMJPS2011, CREST Grant Number JPMJCR2015, JSPS KAKENHI Grant Number JP19H01115, and Basic Research Grant (Super AI) of Institute for AI and Beyond of the University of Tokyo.

7. REFERENCES

[1] L. A. Hiller Jr and L. M. Isaacson, "Musical Composition with a High Speed Digital Computer," in *Audio Engineering Society Convention 9*. Audio Engineering Society, 1957.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *NeurIPS*, 2017.

[3] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer: Generating Music with Long-Term Structure," in *ICLR*, 2019.

[4] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," in *ISMIR*, 2019.

[5] C. Payne, "MuseNet," *OpenAI Blog*, 2019.

[6] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions," in *ACM Multimedia*, 2020.

[7] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs," in *AAAI*, 2021.

[8] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: learning expressive musical performance," *Neural Computing and Applications*, 2018.

[9] Y. Zhou, W. Chu, S. Young, and X. Chen, "BandNet: A Neural Network-based, Multi-Instrument Beatles-Style MIDI Music Composition Machine," in *ISMIR*, 2019.

[10] S. Dai, X. Ma, Y. Wang, and R. B. Dannenberg, "Personalized Popular Music Generation Using Imitation and Structure," *arXiv preprint arXiv:2105.04709*, 2021.

[11] S.-L. Wu and Y.-H. Yang, "MuseMorphose: Full-Song and Fine-Grained Music Style Transfer with One Transformer VAE," *arXiv preprint arXiv:2105.04090*, 2021.

[12] J. Zhao and G. Xia, "AccoMontage: Accompaniment Arrangement via Phrase Selection and Style Transfer," in *ISMIR*, 2021.

[13] S. Dai, Z. Jin, C. Gomes, and R. B. Dannenberg, "Controllable deep melody generation via hierarchical music structure representation," in *ISMIR*, 2021.

[14] X. Zhang, J. Zhang, Y. Qiu, L. Wang, and J. Zhou, "Structure-Enhanced Pop Music Generation via Harmony-Aware Learning," *arXiv preprint arXiv:2109.06441*, 2021.

[15] J. Ens and P. Pasquier, "MMM: Exploring Conditional Multi-Track Music Generation with the Transformer," *arXiv preprint arXiv:2008.06048*, 2020.

[16] S.-L. Wu and Y.-H. Yang, "The Jazz Transformer on the Front Line: Exploring the Shortcomings of AI-composed Music through Quantitative Measures," in *ISMIR*, 2020.

[17] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "POP909: A Pop-song Dataset for Music Arrangement Generation," in *ISMIR*, 2020.

[18] S. Dai, H. Zhang, and R. B. Dannenberg, "Automatic Analysis and Influence of Hierarchical Structure on Melody, Rhythm and Harmony in Popular Music," in *CSMC-MuMe*, 2020.

[19] M. Müller, P. Grosche, and N. Jiang, "A Segment-Based Fitness Measure for Capturing Repetitive Structures of Music Recordings." in *ISMIR*, 2011.

[20] M. Müller and N. Jiang, "A Scape Plot Representation for Visualizing Repetitive Structures of Music Recordings," in *ISMIR*, 2012.

[21] O. Nieto, G. J. Mysore, C. i Wang, J. B. L. Smith, J. Schlüter, T. Grill, and B. McFee, "Audio-Based Music Structure Analysis: Current Trends, Open Challenges, and Applications," *Trans. Int. Soc. Music. Inf. Retr.*, vol. 3, pp. 246–263, 2020.

EXPLOITING PRE-TRAINED FEATURE NETWORKS FOR GENERATIVE ADVERSARIAL NETWORKS IN AUDIO-DOMAIN LOOP GENERATION

Yen-Tung Yeh[1,2] **Bo-Yu Chen**[1,2] **Yi-Hsuan Yang**[1,3]

[1] Academia Sinica, [2] National Taiwan University, [3] Taiwan AI Labs

ytsrt66589@gmail.com, bernie40916@gmail.com, yang@citi.sinica.edu.tw

ABSTRACT

While generative adversarial networks (GANs) have been widely used in research on audio generation, the training of a GAN model is known to be unstable, time consuming, and data inefficient. Among the attempts to ameliorate the training process of GANs, the idea of Projected GAN emerges as an effective solution for GAN-based image generation, establishing the state-of-the-art in different image applications. The core idea is to use a pre-trained classifier to constrain the feature space of the discriminator to stabilize and improve GAN training. This paper investigates whether Projected GAN can similarly improve audio generation, by evaluating the performance of a StyleGAN2-based audio-domain loop generation model with and without using a pre-trained feature space in the discriminator. Moreover, we compare the performance of using a general versus domain-specific classifier as the pre-trained audio classifier. With experiments on unconditional one-bar drum loop and synth loop generation, we show that a general audio classifier works better, and that with Projected GAN our loop generation models can converge around 5 times faster without performance degradation.

1. INTRODUCTION

Generative adversarial networks (GANs) [1] are deep generative models that are composed of a *generator* and a *discriminator*. The discriminator can be considered as a classifier (or a "critic") which judges whether its input is a real sample, or a synthetic one created by the generator counterpart; the generator takes as input a random vector (sometimes plus additional conditional inputs [2]) and aims to create a synthetic sample that "fools" (i.e., appears to be realistic to) the discriminator. During the GAN training process, the discriminator and generator are updated in an iterative manner, fixing the parameters of one and updating those of the other each time. After the training converges, the generator can be used to generate original samples, leading to state-of-the-art (SOTA) results in image generation [3–7] and many other domains.

GANs have been extensively applied to music generation as well, including symbolic-domain generation [8–17] and audio-domain generation [18–28]. In particular, GAN-based models represent the SOTA in audio-domain music generation tasks such as single-note generation [18, 26], drum track generation [21], and loop generation [27].

Despite its widespread applications, GANs are notoriously difficult to train [29–33]. This is partly due to the fact that the generator and discriminator have opposite goals by design, with the generator aiming to *maximize* the discriminator loss and the discriminator aiming to *minimize* the same loss in different iterations, making the training dynamics complicated. As the parameters of the discriminator are constantly being updated over the training process, the generator has no single critic to improve its own performance over time. And, while the parameters of the generator and discriminator are typically initialized *randomly*, the GAN training process can be time-consuming. Using a pre-trained feature network as part of the discriminator has been shown to speed up the training process [34–36], but some modifications of the pre-trained feature network is needed to avoid the discriminator from being too strong and causing the gradients of the generator to vanish.

Projected GAN [37] is a new approach that is shown to effectively leverage the benefits of pre-trained feature networks, leading to SOTA results in unconditional image generation [7]. Projected GAN adds two *random projection* modules after the pre-trained feature networks, named "cross-channel mixing" (CCM) and "cross-scale mixing" (CSM), to prevent the discriminator from focusing only on a subset of features and thereby avoid mode collapse. They not only reduce the time of GAN training, but also improve the quality of the generated images.

This paper studies whether Projected GAN can also improve audio generation, by incorporating it to unconditional audio-domain loop generation [27] as a case study. We note that there are well-studied pre-trained feature networks in the image domain [34–36]. There are also studies on the choice of pre-trained network for retrieval and analysis of musical audio [38] and soundtracks [39]. However, we are less sure which pre-trained feature networks to use in the context of musical audio generation. Therefore, besides pioneering the use of Projected GAN for audio generation, an interesting aspect of our research is that we in-

© Y.-T. Yeh, B.-Y. Chen, and Y.-H. Yang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Y.-T. Yeh, B.-Y. Chen, and Y.-H. Yang, "Exploiting Pre-trained Feature Networks for Generative Adversarial Networks in Audio-domain Loop Generation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

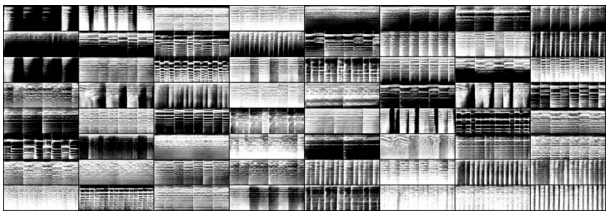


Figure 1. Mel-spectrograms of examples of synth loops generated by Projected GAN with a general classifier.

vestigate different types of pre-trained feature networks for Projected GAN-based musical audio generation, including general classifier and domain-specific classifiers.

Specifically, for the **general** classifier, we use the pre-trained VGGish feature networks trained on YouTube-100M [39], a huge collection of sounds and musical audio from YouTube. For **domain-specific** classifiers, we use short-chuck convolutional neural network (SCNN)-based models [40], which have led to SOTA accuracy across multiple music auto-tagging datasets. We use two SCNNs, one trained on the MagnaTagATune (MTAT) dataset [41] for tagging music of a wide range of genres, and the other trained on a collection of loops for genre classification of loops, which are domain-wise even closer to our generation task. We elaborate on the datasets and classifiers in Section 3, and then the usage of the classifiers as the pre-trained feature networks for loop generation in Section 4.

We report objective and subjective evaluations in Sections 5 and 6 studying the influence of different pre-trained feature networks for drum loop generation and synth loop generation following the approach of Projected GAN, finding that the use of a general classifier works consistently better. Moreover, we find slightly better result can be obtained if we "fuse" general and domain-specific pre-trained feature networks in our model. Objective and subjective evaluations both demonstrate that Projected GAN improves training speed and the final generation quality.

We share our code at `https://github.com/Arthurddd/pjloop-gan`, and examples of the generated loops at `https://arthurddd.github.io/PjLoopGAN/`. Illustrations of the Mel-spectrograms of the generated synth loops and drum loops can be found respectively in Figure 1 and later on in Figure 3.

2. BACKGROUND

Related Work. GANs have garnered great interest in recent years, leading to models that generate high-fidelity audio waveforms. WaveGAN [19] pioneers the use of GAN for audio; it can synthesize one-second raw audio waveforms of speech and music with coherence. Follow-up research applies GAN to other audio synthesis tasks. For example, MelGAN [42] and Parallel WaveGAN [43] use GAN to build neural vocoders that can reconstruct a waveform from the corresponding Mel-spectrogram. GAN-TTS [44] applies GAN to convert text to natural human speech. UNAGAN [23] uses GAN to synthesize singing voice.

GAN has also been used to synthesize audio samples of musical materials, including percussive and harmony ones, that can be used in music production. GANSynth [18] can synthesize harmony music notes with diverse timbre and controllable pitches. DrumGAN [24] can synthesize one-shot percussion sounds, allowing for the use of condition perceptual features to intuitively control the timbre of the percussion sounds. StyleWaveGAN [45] improves the audio quality and inference time of DrumGAN. However, the aforementioned models deal with only single notes or one-shot samples instead of longer phrases such as musical loops, limiting their applicability to creating loop-based music. In view of this need, Hung *et al.* [27] study the task of one-bar drum loop generation and benchmark the performance of UNAGAN [23], StyleGAN [4] and StyleGAN2 [5] for this task over the FreeSound Loop dataset [46], showing that the model based on StyleGAN2 [5] performs the best. However, Hung *et al.* [27] do not consider the training efficiency of their models. Moreover, while they focus on drum loops only, we consider also the generation of synth loops to encompass not only percussive but also harmonic musical materials in our work.

Specific to the GAN training technique, there are three main approaches to improve the discriminator. First, modifying the *architecture* of the discriminator to improve the discriminator's ability [47–49]. Second, assembling *multiple discriminators* to capture more comprehensive features, an approach that as been widely investigated in the audio domain for building the vocoder [42, 50–52]. The third approach introduces a *pre-trained feature network* to the discriminator to avoid learning its parameters completely from scratch, which helps speed up the convergence time and prevent model overfitting. Zhao *et al.* [34] use the pre-trained network in the large-scale dataset, adapt it to a small dataset, and propose adaptive filter modulation to deal with domain shift. Grigoryev *et al.* [35] and Kumari *et al.* [36] both find that pre-trained network selection can largely influence performance and propose a recipe to choose a suitable pre-trained network for a particular image-domain task. Projected GAN [37] is a very recent idea that combines the aforementioned approaches, using multiple discriminators and employing pre-trained feature networks simultaneously to improve the training stability and efficiency of the GAN. However, to our best knowledge, adapting Projected GAN to the audio-domain generation is still unexplored, for either speech or music.

Projected GAN. Projected GAN [37] introduces a set of $L = 4$ feature projectors $\{P_l, l = 1, \ldots, L\}$, each maps either real samples $\mathbf{x}$ or fake samples $G(\mathbf{z})$ to a fixed pre-trained feature space. It aims to minimize the following objective function to match the data distribution in the feature space, instead of matching data distributions directly:

$$\min_G \max_{\{D_l\}} \sum\nolimits_{l=1}^{L} \exp\big(E_x[\log D_l(P_l(\mathbf{x}))] + E_z[\log(1 - D_l(P_l(G(\mathbf{z}))))]\big), \quad (1)$$

where $\{D_l, l = 1, \ldots, L\}$ denotes the set of independent discriminators operating on different feature projections,

and G denotes the generator that converts a random vector $\mathbf{z}$ into a fake sample. Each projector P_l consists of three components: a pre-trained feature network C_l, the CCM modules, and the CSM modules. First, we extract features from L different layers of the feature network C_l, leading to L set of feature maps in different scales. Second, for each scale, the CCM mixes the features across channels by *random* (i.e., not-learned) 1×1 convolutions with an equal number of input and output channels, which can be viewed as the generalization of random permutation. Third, CSM further mixes features across scales by *random* 3×3 convolutions and bilinear upsampling layers, yielding a U-Net architecture that combines the feature maps of different scales. The features fused by CCM and CSM in L different scales would then be fed to each discriminator D_l. The random projections introduced by CCM and CSM have the effect of encouraging each D_l to take into account the entire feature space instead of overfitting to a sub-set of feature space, thereby avoiding mode collapse. Without using gradient penalties and any sophisticated training strategies, Projected GAN updates its loss simply by summing the output of the L discriminators.

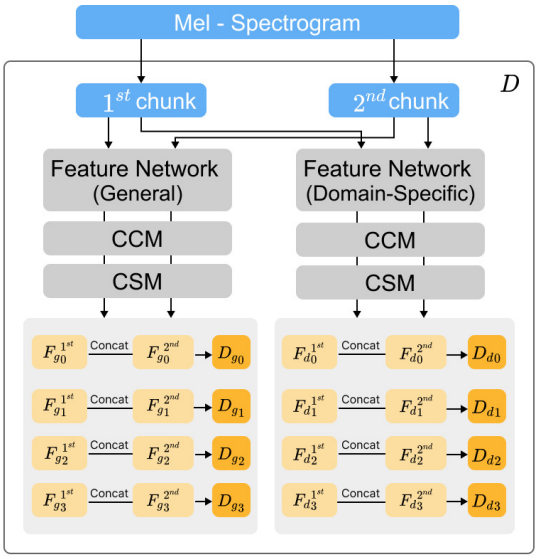


Figure 2. Diagram of the proposed discriminator architecture. A Mel-spectrogram is divided into two chunks and then fed to either a general or a domain-specific pre-trained feature network, or both (for "fusion"). Features computed for both chunks are aggregated at each scale l before feeding to the corresponding discriminator D_l.

3. DATSETS & CLASSIFIERS

Our work is built upon the use of the following datasets.

Youtube-100M [39] is a private dataset of Google, containing 100 million Youtube videos. Each video has on average 5 manually assigned tags, out of 30,871 possible labels. Hershey *et al.* [39] train a VGGish pre-trained network for large-scale audio classification using the dataset. While we are not able to have a copy of the dataset, we can use the pre-trained weights Gemmeke *et al.* [53] share publicly as our general audio classifier.

MagnaTagATune (MTAT) [41] is a dataset commonly-used in research on music auto-tagging. It contains 25,863 music clips, each 29-seconds long. We use MTAT to train one of our domain-specific pre-trained feature networks. We follow the original data split [41] and use only the top 50 tags, including genre and instrumentation labels, as well as decades (e.g., '80s' and '90s') and moods.

Looperman dataset is an in-house collection of loops from `https://www.looperman.com/`, a website hosting free music loops. We get the audio and uploader-provided metadata of 23,983 drum loops and 22,625 synth loops. According to the metadata, the drum loops and synth loops can be categorized to 66 and 58 genres, respectively. We use the Looperman dataset for not only building a domain-specific pre-trained feature network but also for building our audio-domain loop generation model.

Following the preprocessing steps of Hung *et al.* [27], we first apply downbeat tracking via `madmom` [54] to split every loop into single bars and then time-stretch each to 120 BPM (beats-per-minute) by `pyrubberband` [55] to unify their length to be always **2 seconds per loop**. After this preprocessing, we have 128,122 and 42,570 one-bar loops for drum and synth, respectively. We split the data by 80/10/10 for training the loop genre classifier, and use the entire data for music loop generation.

3.1 Pre-trained Feature Networks for Music

General. As aforementioned, we use the VGGish network trained on Youtube-100M [53] as the general pre-trained feature network. Taking the Mel-spectrogram as input, the model uses 2D convolutional blocks and 2D max-pooling layers to compute 128-dimensional features at the output.

Domain specific. Short-Chuck CNN (SCNN) [40] has a simple 2D CNN architecture with 3×3 filters and residual module but it has been shown to be remarkably effective for music auto-tagging. We use in our implementation 6 layers of CNN blocks with a fully connected layer and the residual module. Each CNN block comprises a 2×2 max-pooling layer. We use SCNN to train separate classifiers for MTAT and Looperman from scratch, and then used the trained classifiers as our domain-specific pre-trained feature networks. For MTAT, SCNN achieves 0.909 ROC-AUC and 0.445 PR-AUC. For genre classification of the looperman drum loops, the classification accuracy reaches 0.792. For synth loops, the accuracy attains 0.700.

Fusion. Fusing multiple pre-trained feature networks has also been shown useful for Projected GAN-based image generation [7]. Accordingly, we also experiment with a simple fusion strategy that allows the discriminator to consider both general and domain-specific features at the same time, as depicted in Figure 2.

4. AUDIO GENERATION BY PROJECTED GAN

Following Hung *et al.* [27], we use StyleGAN2 [5] as the backbone of our loop generation model. However, we improve their model in a number of aspects, regarding to not

only the discriminator (i.e., with Projected GAN) but also the generator. We provide the details below.

Improving the generator. The generator G consists of a mapping network G_m and a synthesis network G_s. We implement G_m with only 6 fully-connected layers instead of the 8 in the original StyleGAN2 architecture. Furthermore, echoing the finding in the image domain [56], we find that the length of the vectors $\mathbf{z}$ (i.e., the input of G_m) largely affects the model performance. Setting the length of $\mathbf{z}$ to 512 as done in [27] leads to mode collapse in our preliminary experiments, as shown in Figure 3(a). This may be related to the so-called "intrinsic dimension" of the data [56]; an overly large latent space of $\mathbf{z}$ introduces redundancy that distracts the generator. As smaller $\mathbf{z}$ empirically works better, we set its length to 32 in the experiments reported below. We similarly set the length of the "style code" $\mathbf{w}$ (i.e., the output of G_m) to a small value of 64.

Pipeline The training follows the procedure of the Projected GAN, but we use the following pipeline to match the input shape expected by the pre-trained feature networks. First, we compute the Mel-spectrogram with 512-point window size and 160-point hop size for short-time Fourier Transform (STFT) and 64 Mel channels. Second, we feed a latent $\mathbf{z}$ to the mapping network G_m to generate style codes that modulate the convolutions of the synthesis network G_s, using four upsampling blocks to generate a Mel-spectrogram that corresponds to a synthesized *2-second* loop. We note that our pre-trained feature networks are on *1-second* audio. To address the size mismatch, we split the Mel-spectrogram into **two chunks**, with the first half corresponding to the first second and the other the rest. As illustrated in Figure 2, we feed the two chunks separately to the discriminators, going through the pre-trained feature network, CCM and CSM. Moreover, we concatenate the features with the same scales but in different chunks to aggregate the features from individual chunks, yielding $L = 4$ aggregated features $\{F_l, l = 1...L\}$. We feed each of these features independently to the corresponding discriminators D_l. Additionally, if we consider two pre-trained feature networks simultaneously, referred to as "fusion" in Section 3.1, we have in total $2L$ aggregated features and $2L$ discriminators. Eventually, we sum the loss from these discriminators to update the network.

Similar to Hung *et al.* [27], at inference time, the Mel-spectrogram generated by the generator would go through a MelGAN vocoder [42] to become waveforms.

5. OBJECTIVE EVALUATION

We use the following objective metrics to evaluate the quality and diversity of the generated loops.

Inception Score (IS) [58, 59] measures the quality of the generated loops and detects whether there is a mode collapse by using a pre-trained domain-specific classifier, namely the loop genre classifier for each type of loops (i.e., drum or synth). It penalizes models whose samples cannot be reliably classified into a single class or that only belong to a few from all possible classes.

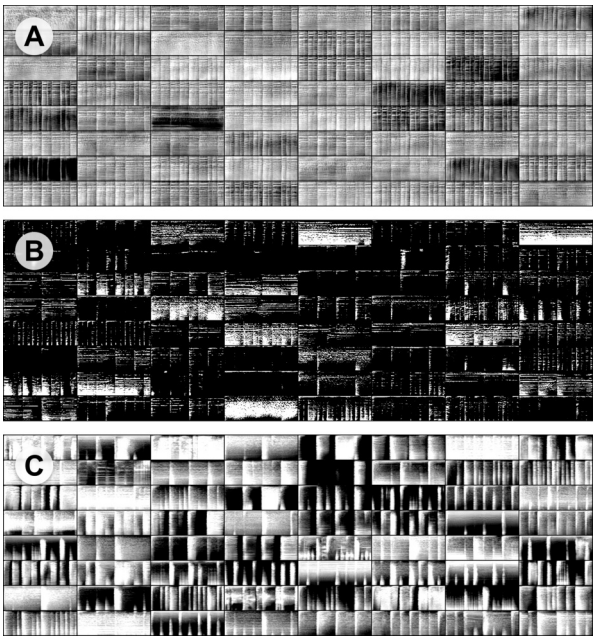


Figure 3. Mel-spectrograms of examples of drum loops generated (a) by a failed case using an overly large latent space for $\mathbf{z}$, leading to mode collapse (see Section 4); (b) Projected GAN with the MTAT domain-specific classifier; (c) Projected GAN with the VGG general classifier.

Fréchet Audio Distance (FAD) [60] reflects both quality and diversity as it measures the distance between continuous multivariate Gaussians fitted to the embeddings of the real and generated loops.

Density & Coverage (D&C) [57] are new metrics that measure respectively the quality and diversity of the generated data. D&C are considered to be more robust against the influence of outliers compared to older metrics such as precision and recall (P&R) [61]. 'D' is the average number of real-sample neighborhood spheres that contain each of the fake samples. It may be greater than 1 depending on the density of reals around the fakes. 'C' is the fraction of real samples whose neighborhoods contain at least one fake sample. In our implementation, we use the hyperparameters suggested by [57]. Moreover, following [57], we calculate D&C with a randomly-initialized VGG16 model that projects samples to VGG16 *fc2* space whose dimension is deliberately set to a small value 64.

We evaluate the following models here:

- **StyleGAN2**: the SOTA for drum loop generation [27].
- **StyleGAN2 (early-stop)**: the StyleGAN2 that stops training at the same point when Projected GAN converges, providing a reference demonstrating the possible advantage of the training efficiency of Projected GAN.
- **Projected StyleGAN2** with different pre-trained feature networks, including 1) the general VGG classifier alone, 2) the domain-specific SCNN classifier trained on MTAT alone (denoted as $\text{SCNN}_{\text{MTAT}}$), 3) the SCNN classifier trained on Looperman alone, using the drum or synth classifier depending on whether it is for generating drum or synth loops (denoted as $\text{SCNN}_{\text{Loop}}$), 4) "fusion" of

Models	Drum loops				Synth loops			
	IS ↑	FAD ↓	D ↑	C ↑	IS ↑	FAD ↓	D ↑	C ↑
A Real data	16.30	0.01	1.00	1.00	14.95	0.01	1.00	1.00
B StyleGAN2 [27]	5.58	2.50	1.01	**0.89**	4.80	3.60	1.19	**0.91**
C StyleGAN2 (early-stop)	5.21	3.40	0.56	0.43	4.55	7.97	1.01	0.75
D Projected StyleGAN2 (VGG)	5.87	3.03	1.06	0.70	4.70	**2.34**	1.31	0.82
E Projected StyleGAN2 ($SCNN_{MTAT}$)	4.75	8.18	0.00	0.00	3.97	7.14	0.001	0.002
F Projected StyleGAN2 ($SCNN_{Loop}$)	4.45	7.21	0.00	0.00	3.08	8.32	0.001	0.002
G Projected StyleGAN2 (VGG+$SCNN_{MTAT}$)	6.22	2.45	**1.11**	0.74	4.73	3.13	1.67	0.85
H Projected StyleGAN2 (VGG+$SCNN_{Loop}$)	**6.31**	**2.34**	1.08	0.73	**4.82**	2.56	**1.70**	0.85

Table 1. Objective evaluation result for the different settings of the Projected GANs trained on the Looperman dataset for loop generation. 'D' and 'C' stand for Density & Coverage [57] (↓/↑: the lower/higher the better; the best in bold).

VGG, $SCNN_{MTAT}$, and 5) "fusion" of VGG, $SCNN_{Loop}$.

Table 1 presents the objective evaluation results of models trained on drum loops and synth loops. Each model generates 10,000 random loops to compute the scores. We also compute the scores using the real data for setting a high anchor of the objective scores.

We see that the best-performing configurations of Projected StyleGAN2 can achieve higher IS and lower FAD than StyleGAN2 (i.e., row B). For drum loops, the IS can be improved from 5.58 to 6.31 (row H) ; for synth loop the FAD can be reduced from 3.60 to 2.34 (row D).

Interestingly and somehow surprisingly, when only a single pre-trained feature network is used (i.e., rows D–F), we find that *only* Projected StyleGAN2 (VGG) performs nicely. The domain-specific pre-trained feature networks actually degrade the performance of loop generation; either Projected StyleGAN2 ($SCNN_{MTAT}$) or Projected StyleGAN2 ($SCNN_{Loop}$) obtains lower IS and higher FAD, and even close-to-zero D&C. This suggests that a general pre-trained feature network works much better than a domain-specific pre-trained feature network in the context of Projected GAN-based unconditional loop generation. This is likely because the discriminator employing a domain-specific pre-trained feature network is too strong compared to the generator, leading to gradient vanishing. Figure 3(b) shows samples generated by Projected StyleGAN2 ($SCNN_{MTAT}$); we see that the model weirdly generates sparse samples most of the time.

Table 1 also shows that the "fusion" of general and domain-specific pre-trained feature networks (rows G & H) performs slightly better in some metrics than using a general pre-trained feature network alone (row D). In particular, we see that Projected StyleGAN2 (VGG+$SCNN_{Loop}$) achieves the highest IS score in both drum and synth loop generation. It also reaches the lowest FAD for drum loops and the second lowest FAD for synth loops. We show examples of its generated synth and drum loops in Figures 1 and 3(c), respectively.

Projected StyleGAN2 appears to have lower D&C compared to StyleGAN2, suggesting that Projected StyleGAN2 sacrifices a little diversity for higher quality of the generated samples. This is presumably because the feature space has been constrained by the pre-trained feature networks throughout the whole training process, making it

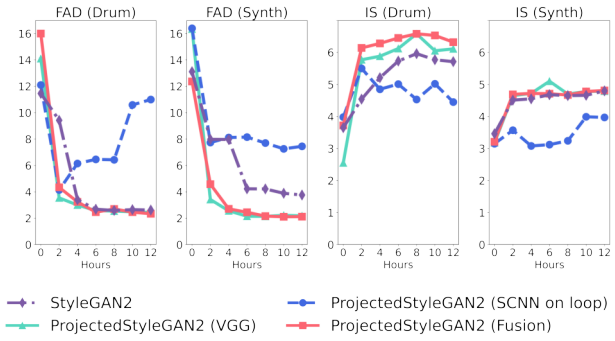


Figure 4. The FAD and IS as a function of training hours.

hard to capture the whole distributions. Future work can be done to remedy this and further improve the diversity of Projected StyleGAN2.

Figure 4 shows how the FAD and IS of four selected models (rows B, D, F, H in Table 1) varies as a function of training time, when all the models are trained separately and independently on a single V100 GPU. For both drum and synth loop generation, Projected GAN leads to lower FAD much faster than the StyleGAN2 baseline. For synth loops, Projected GAN with only 2-hour training reaches lower FAD than the StyleGAN2 baseline with 12-hour training. In general, Projected GAN converges at approximately 2 hours for both drum and synth loops, while StyleGAN2 converges x5 times longer at about 10 hours. Echoing the result in Table 1, using the epoch of StyleGAN2 corresponding to 2-hour training time (i.e., row C) obtains worse scores than the models corresponding to rows B and D in almost all the metrics for both drum and synth loops. Overall, these results nicely demonstrate how Projected GAN speeds up and improves GAN training.

6. SUBJECTIVE EVALUATION

To further assess the loops generated by the models, we conduct a subjective listening test through an anonymous online questionnaire. Each subject is presented with a randomly picked human-made loop from the Looperman dataset (i.e., 'Real') and a randomly-generated loop by each of the following four models: StyleGAN2, StyleGAN2 (early-stop), Projected StyleGAN2 ($SCNN_{Loop}$)

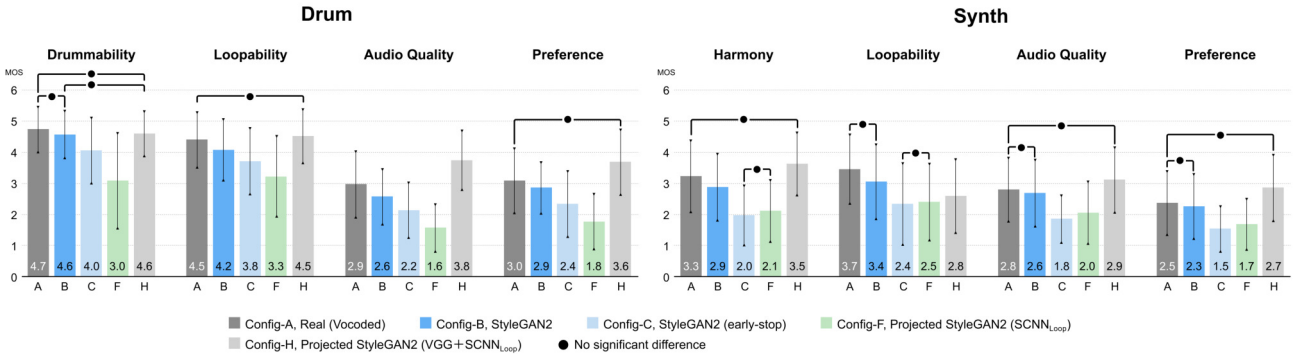


Figure 5. Subjective evaluation results. The performance difference between any pair of models in any metric is statistically significant (p-value< 0.001) under the Wilcoxon signed-rank test, except for the pairs that are explicitly highlighted.

and Projected StyleGAN2 (VGG+SCNN$_{\text{Loop}}$). The subject is then asked to rate each of these loops in terms of the following metrics, all on a five-point Likert scale.

- **Drummability or Harmony**: Drummability (for drum loops) means whether the sample contains percussion sounds, and Harmony (for synth loops) means whether the sample contains harmonic sounds.
- **Loopability**: whether the sample can be played repeatedly in a seamless manner.
- **Audio quality**: whether the sample is free of unpleasant noises or artifacts.
- **Preference**: how much you like it.

To evaluate loopability, we repeat each sample four times in the audio recording. Because the output of the models all goes through the MelGAN vocoder [42] to become waveforms, we do the same for 'Real' to be fair.

Figure 5 shows the averaged results from 35 participants. The responses indicate an acceptable level of reliability, Cronbach's $\alpha = 0.721$. The subjective evaluation result aligns nicely with the objective evaluation result. Projected StyleGAN2 (VGG+SCNN$_{\text{Loop}}$) performs the best and Projected StyleGAN2 (SCNN$_{\text{Loop}}$) performs the worst. Overall, Projected StyleGAN2 (VGG+SCNN$_{\text{Loop}}$) achieves results comparable to StyleGAN2 and even 'Real' for both types of loops. Somehow surprisingly, Projected StyleGAN2 (VGG+SCNN$_{\text{Loop}}$) can even outperform 'Real' in audio quality and preference for both drum and synth loops. We conjecture that this is because the 'Real' here actually stands for the "MelGAN-vocoded" version of the real data, whose quality may have suffered from the artefacts introduced by the vocoder.

Almost all models lead to higher subjective scores for drum loops than for synth loops, suggesting that synth loop generation is more challenging. We note that, for synth loops, Projected StyleGAN2 (VGG+SCNN$_{\text{Loop}}$) actually obtains higher harmony scores than StyleGAN2 and 'Real', but its loopability scores is lower. To further improve its performance for synth loop generation, future work may be done to explicitly consider loopability in model training.

7. LIMITATIONS

We only focus on one-bar loop generation with a specific BPM of 120 in our study, which is "convenient" as it gives us fixed-size Mel-spectrograms to be treated as images by StyleGAN2. However, to be applicable to loop-based music production, future work needs to consider variable BPMs and loops with more bars. As the size of the Mel-spectrogram would not be fixed when we consider variable BPMs, future work may need to adopt generative models other than StyleGAN2 as the backbone. Possible candidates are UNAGAN [23] and VQGAN [62], both of which may also benefit from the ideas of Projected GAN.

To apply Projected GAN, we need to take care of the *shape matching* between the pre-trained networks and the generative model. For example, in our work the pre-trained networks (e.g., the VGGish one) is trained on 1-second audio chunks, but our generator is to generate 2-second loops. The size mismatch can be easily addressed by simply splitting the Mel-spectrogram into two chunks in our work, but this is trickier if we consider BPMs other than 120.

8. CONCLUSION

In this paper, we have demonstrated that Projected GAN can be used to improve the training efficiency and overall performance of GAN-based models for audio generation, specifically the generation of drum loops and synth loops. Moreover, we demonstrated that using a domain-specific pre-trained feature network alone does not work well; we need to use either a general pre-trained feature network, or the fusion of multiple pre-trained feature networks.

This work can be extended in many ways. First, we can expand our work to generate variable-length loops, or to other GAN-based audio-related tasks. Next, we can explore unsupervised or self-supervised approaches (e.g., [63, 64]) to get the pre-trained feature network. Third, we are also interested in using class conditions or attributes for more controllable loop generation. Doing so may help improve the diversity of the generated loops as well, according to related work on image generation [7]. Finally, using diffusion probabilistic models [65, 66] as the generative model for loop generation also worth trying.

9. ACKNOWLEDGEMENT

We are grateful to Tun-Min Hung for helpful discussions during the project. We also thank the anonymous reviewers for their constructive feedbacks. Our research is funded by grant NSTC 109-2628-E-001-002-MY2 from the National Science and Technology Council of Taiwan.

10. REFERENCES

[1] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Advances in neural information processing systems*, 2014, pp. 2672–2680.

[2] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv:1411.1784*, 2014.

[3] A. Brock, J. Donahue, and K. Simonyan, "Large scale GAN training for high fidelity natural image synthesis," *arXiv preprint arXiv:1809.11096*, 2018.

[4] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2019, pp. 4396–4405.

[5] T. Karras *et al.*, "Analyzing and improving the image quality of StyleGAN," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2020.

[6] T. Karras, M. Aittala, S. Laine, E. Härkönen, J. Hellsten, J. Lehtinen, and T. Aila, "Alias-free generative adversarial networks," in *Proc. Conf. Neural Information Processing Systems*, 2021.

[7] A. Sauer, K. Schwarz, and A. Geiger, "StyleGAN-XL: Scaling StyleGAN to large diverse datasets," in *Proc. ACM SIGGRAPH Conf.*, 2022.

[8] O. Mogren, "C-RNN-GAN: Continuous recurrent neural networks with adversarial training," in *Proc. Constructive Machine Learning Workshop*, 2016.

[9] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2017.

[10] H.-W. Dong *et al.*, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proc. AAAI Conference on Artificial Intelligence*, 2018.

[11] G. Chen *et al.*, "Musicality-novelty generative adversarial nets for algorithmic composition," in *Proc. ACM Int. Conf. Multimedia*, 2018, p. 1607–1615.

[12] N. Trieu and R. M. Keller, "JazzGAN: Improvising with generative adversarial networks," in *Proc. Int. Workshop on Musical Metacreation*, 2018.

[13] I.-C. Wei, C.-W. Wu, and L. Su, "Generating structured drum pattern using variational autoencoder and self-similarity matrix," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2019.

[14] H. Jhamtani and T. Berg-Kirkpatrick, "Modeling self-repetition in music generation using generative adversarial networks," in *Proc. Machine Learning for Music Discovery Workshop*, 2019.

[15] N. Zhang, "Learning adversarial Transformer for symbolic music generation," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2020.

[16] Y. Yu, A. Srivastava, and S. Canales, "Conditional LSTM-GAN for melody generation from lyrics," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 17, no. 1, 2021.

[17] A. Muhamed, L. Li, X. Shi, S. Yaddanapudi, W. Chi, D. Jackson, R. Suresh, Z. C. Lipton, and A. J. Smola, "Symbolic music generation with Transformer-GANs," in *Proc. AAAI Conference on Artificial Intelligence*, 2021.

[18] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial neural audio synthesis," in *Proc. Int. Conf. Learning Representations*, 2019.

[19] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *Proc. Int. Conf. Learning Representations*, 2019.

[20] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "WGANSing: A multi-voice singing voice synthesizer based on the Wasserstein-GAN," in *Proc. European Signal Processing Conf.*, 2019, pp. 1–5.

[21] S. Lattner and M. Grachten, "High-level control of drum track generation using learned patterns of rhythmic interaction," in *Proc. IEEE Work. Applications of Signal Processing to Audio and Acoustics*, 2019.

[22] J. Drysdale, M. Tomczak, and J. Hockman, "Adversarial synthesis of drum sounds," in *Proc. Int. Conf. Digital Audio Effects*, 2020.

[23] J.-Y. Liu, Y.-H. Chen, Y.-C. Yeh, and Y.-H. Yang, "Unconditional audio generation with generative adversarial networks and cycle regularization," in *Proc. INTERSPEECH*, 2020.

[24] J. Nistal, S. Lattner, and G. Richard, "DrumGAN: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2020.

[25] J. Nistal, C. Aouameur, S. Lattner, and G. Richard, "VQCPC-GAN: Variable-length adversarial audio synthesis using vector-quantized contrastive predictive coding," in *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2021.

[26] J. Nistal, S. Lattner, and G. Richard, "DarkGAN: Exploiting knowledge distillation for comprehensible audio synthesis with GANs," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2021, pp. 484–492.

[27] T. Hung, B. Chen, Y. Yeh, and Y. Yang, "A benchmarking initiative for audio-domain music generation using the Freesound Loop Dataset," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2021.

[28] B.-Y. Chen, W.-H. Hsu, W.-H. Liao, M. A. M. Ramírez, Y. Mitsufuji, and Y.-H. Yang, "Automatic DJ transitions with differentiable audio effects and generative adversarial networks," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2022.

[29] T. Salimans *et al.*, "Improved techniques for training GANs," in *Proc. Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[30] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," in *Proc. Int. Conf. Learning Representations*, 2017.

[31] N. Kodali *et al.*, "On convergence and stability of GANs," *arXiv preprint arXiv:1705.07215*, 2017.

[32] I. Gulrajani *et al.*, "Improved training of Wasserstein GANs," in *Proc. Advances in Neural Information Processing Systems*, 2017, pp. 5769–5779.

[33] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for GANs do actually converge?" in *Proc. Int. Conf. Machine Learning*, 2018.

[34] M. Zhao, Y. Cong, and L. Carin, "On leveraging pretrained GANs for generation with limited data," in *Proc. Int. Conf. Machine Learning*, 2020.

[35] T. Grigoryev, A. Voynov, and Babenko, "When, why, and which pretrained GANs are useful?" in *Proc. Conf. Neural Information Processing Systems*, 2022.

[36] N. Kumari, Z. R., E. Shechtman, and J. Y. Zhu, "Ensembling off-the-shelf models for GAN training," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2022.

[37] A. Sauer, K. Chitta, J. Müller, and A. Geiger, "Projected GANs converge faster," in *Proc. Advances in Neural Information Processing Systems*, 2021.

[38] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2017.

[39] S. Hershey *et al.*, "CNN architectures for large-scale audio classification," in *Proc. Int. Conf. Acoustics, Speech and Signal Processing*, 2017.

[40] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of CNN-based automatic music tagging models," in *Proc. Sound and Music Computing Conf.*, 2020.

[41] E. Law, K. West, M. Mandel, M. Bay, and J. Downie, "Evaluation of algorithms using games: The case of music tagging," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2009, pp. 387–392.

[42] K. Kumar *et al.*, "MelGAN: Generative adversarial networks for conditional waveform synthesis," *arXiv preprint arXiv:1910.06711*, 2019.

[43] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel WaveGAN: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *IEEE Int. Conf. Acoustics, Speech and Signal Proc.*, 2020, pp. 6199–6203.

[44] M. Bińkowski *et al.*, "High fidelity speech synthesis with adversarial networks," *arXiv preprint arXiv:1909.11646*, 2019.

[45] A. Lavault, A. Roebel, and M. Voiry, "Style-based synthesis of drum sounds with extensive controls using generative adversarial networks," in *Proc. Int. Sound and Music Computing Conf.*, 2022.

[46] A. Ramires *et al.*, "The Freesound Loop Dataset and annotation tool," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2020, pp. 287–294.

[47] J. Zhao, M. Mathieu, and Y. LeCun, "Energy-based generative adversarial network," in *Proc. Int. Conf. Learning Representations*, 2017.

[48] E. Schonfeld, B. Schiele, and A. Khoreva, "A u-net based discriminator for generative adversarial networks," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2020, pp. 8207–8216.

[49] Y. Jiang, S. Chang, and Z. Wang, "TransGAN: Two pure Transformers can make one strong GAN, and that can scale up," in *Proc. Conf. Neural Information Processing Systems*, 2021.

[50] J. Yang *et al.*, "VocGAN: A high-fidelity real-time vocoder with a hierarchically-nested adversarial network," in *Proc. INTERSPEECH*, 2020.

[51] J. Kong, J. Kim, and J. Bae, "Hifi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Proc. Conf. Neural Information Processing Systems*, 2020.

[52] J. Kim, S. Lee, J. Lee, and S. Lee, "Fre-GAN: Adversarial frequency-consistent audio synthesis," in *Proc. INTERSPEECH*, 2021.

[53] J. F. Gemmeke *et al.*, "Audio Set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2017.

[54] S. Böck *et al.*, "Madmom: A new Python audio and music signal processing library," in *Proc. ACM Multimedia Conf.*, 2016.

[55] "pyrubberband," https://pyrubberband.readthedocs.io/.

[56] P. Pope, C. Zhu, A. Abdelkader, M. Goldblum, and T. Goldstein, "The intrinsic dimension of images and its impact on learning," *arXiv preprint arXiv:2104.08894*, 2021.

[57] M. F. Naeem *et al.*, "Reliable fidelity and diversity metrics for generative models," in *Int. Conf. Machine Learning*, 2020.

[58] T. Salimans *et al.*, "Improved techniques for training GANs," in *Proc. Conf. Neural Information Processing Systems*, 2016, pp. 2226–2234.

[59] S. Barratt and R. Sharma, "A note on the inception score," in *Proc. ICML Works. Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[60] K. Kilgour *et al.*, "Fréchet Audio Distance: A metric for evaluating music enhancement algorithms," *arXiv preprint arXiv: 1812.08466*, 2019.

[61] T. Kynkäänniemi *et al.*, "Improved precision and recall metric for assessing generative models," in *Proc. Conf. Neural Information Processing Systems*, 2019, p. 3929–3938.

[62] P. Esser, R. Rombach, and B. Ommer, "Taming Transformers for high-resolution image synthesis," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition*, 2021.

[63] X. Chen and K. He, "Exploring simple Siamese representation learning," *arXiv preprint arXiv:2011.10566*, 2020.

[64] K. He *et al.*, "Masked autoencoders are scalable vision learners," *arXiv preprint arXiv:2111.06377*, 2021.

[65] S. Liu, D. Su, and D. Yu, "DiffGAN-TTS: High-fidelity and efficient text-to-speech with denoising diffusion GANs," *arXiv preprint arXiv:2201.11972*, 2022.

[66] Z. Wang, H. Zheng, P. He, W. Chen, and M. Zhou, "Diffusion-GAN: Training GANs with diffusion," *arXiv preprint arXiv:2206.02262*, 2022.

MODELING THE RHYTHM FROM LYRICS FOR MELODY GENERATION OF POP SONG

Daiyu Zhang Ju-Chiang Wang Katerina Kosta Jordan B. L. Smith Shicen Zhou
ByteDance
{daiyu.zhang, ju-chiang.wang, katerina.kosta, jordan.smith, zhoushicen}@bytedance.com

ABSTRACT

Creating a pop song melody according to pre-written lyrics is a typical practice for composers. A computational model of how lyrics are set as melodies is important for automatic composition systems, but an end-to-end lyric-to-melody model would require enormous amounts of paired training data. To mitigate the data constraints, we adopt a two-stage approach, dividing the task into lyric-to-rhythm and rhythm-to-melody modules. However, the lyric-to-rhythm task is still challenging due to its multimodality. In this paper, we propose a novel lyric-to-rhythm framework that includes part-of-speech tags to achieve better text-setting, and a Transformer architecture designed to model long-term syllable-to-note associations. For the rhythm-to-melody task, we adapt a proven chord-conditioned melody Transformer, which has achieved state-of-the-art results. Experiments for Chinese lyric-to-melody generation show that the proposed framework is able to model key characteristics of rhythm and pitch distributions in the dataset, and in a subjective evaluation, the melodies generated by our system were rated as similar to or better than those of a state-of-the-art alternative.

1. INTRODUCTION

Setting lyrics to a melody is a common but complex task for a composer. The form, articulation, meter, and symmetry of expression in lyrics can inspire, or set constraints on, the melodic arrangement. Given the importance of melody, it is unsurprising that the decades-long history of Music Metacreation systems includes countless melody-creation systems (see [1] for a review). However, less attention has been paid to the lyric-to-melody generation task (i.e., generating a melody for given input lyrics). The task is challenging for many reasons, including but not limited to: the need to handle the prosody of the text correctly (e.g., one should avoid setting an unstressed word like 'the' on a stressed note in the melody); the need to reflect the structure of the lyrics in the melody; and the need to create a good melody to begin with.

© D. Zhang, J.-C. Wang, K. Kosta, J. B. L. Smith, and S. Zhou. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** D. Zhang, J.-C. Wang, K. Kosta, J. B. L. Smith, and S. Zhou, "Modeling the rhythm from lyrics for melody generation of pop song", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

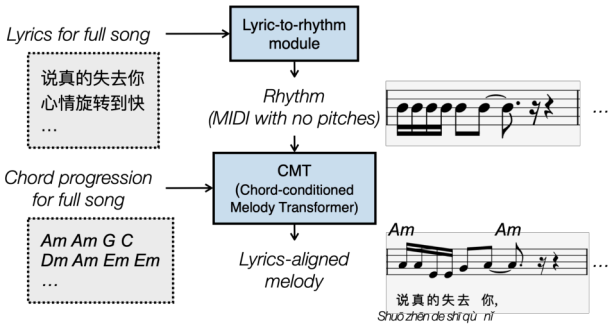


Figure 1: Diagram of the proposed system.

With the rapid growth of deep learning tools, this task has gained more attention, and there are many recent examples of lyric-to-melody creation systems, most using an end-to-end approach [2–5]. Modeling the relationship between lyric syllables and musical notes is a complex, cross-modal task, but it is hoped that we can succeed with a large amount of paired examples (i.e., lyrics aligned to their corresponding melodies). However, acquiring such data is expensive, and using unsupervised learning has shown limited performance gains [3]. All the systems mentioned here are trained on fewer than 200,000 examples of song lyrics; by contrast, the text-to-image system DALL-E has 12 billion parameters and involved hundreds of millions of paired text-image training data [6].

One alternative, suggested in [7], is to pick an intermediate representation and adopt a two-stage approach: one model to convert lyrics to the chosen representation, and a second to convert that to a melody. The motivation is that there is sufficient data to train each model separately, without the paired lyrics-melody data required by the end-to-end approach.

We choose 'rhythm' as the intermediate step because, if we disregard melismas and expressive singing techniques, we can assume there is a one-to-one correspondence between syllables and onsets, and between onsets and melody pitches. Also, there is plenty of data to model each step: first, from karaoke-style scrolling lyrics data, we can obtain an alignment between syllables in lyrics and note onsets in music, and thus note durations and metrical positions, too. Second, there are multiple public datasets from which to learn to assign pitches for each note given their duration. Our goal is then to solve two sub-tasks, namely lyric-to-rhythm and rhythm-to-melody, with an as-

sumption that the rhythm generation process is independent of the pitch generation one [7].

There are many recent melody generation models [8–11], but lyric-to-rhythm modeling is rarely attempted. In this paper, we introduce a novel framework for converting lyrics to rhythms using an encoder-decoder Transformer architecture [12]. The proposed system is outlined in Fig. 1: given an input set of lyrics, a lyric-to-rhythm module assigns onset times and durations for each syllable. This rhythm, along with a user-provided chord progression, is fed into a Chord-conditioned Melody Transformer (CMT) [13], a state-of-the-art melody generation system, to predict the pitch for each note. The details of the lyric-to-rhythm module and the CMT are provided in Sections 3.3 and 3.2, respectively.

2. BACKGROUND

Lyrics and melody are not arbitrarily combined; common sense suggests and prior analysis [14] indicates that patterns in lyrics and melodies are related and can be modeled, in part, with features of the melody (e.g., note duration) and lyrics (e.g., syllable stress). One of the earliest lyric-to-melody systems was designed to handle Japanese prosody [15]: first, the input text was segmented into phrases; next, a set of pre-composed rhythms was searched for one that fit the syllable count and matched the accent pattern of the text; finally, pitches were assigned using dynamic programming to optimise the interval directions with the natural prosody of the words. An earlier lyric-to-rhythm system also leveraged a dataset of pre-composed rhythms that were scored based on their match to the input syllable-stress and word-rarity patterns [16]. Although our system has little in common with these works, we do share the use of rhythm as an intermediate representation.

Algorithms for automatic music generation are a subset of Music Metacreation systems [1], which have been present in Western music in many forms, including being used for the creation of standalone pieces and, either offline or in real-time, as part of the human composition process. With the help of machine learning and deep learning architectures, many such systems have shown to be capable of generating a plausible outcomes that match the musical characteristics of given datasets. Supervised generative models aim to learn a representation of the underlying characteristics of a training set distribution. Depending on the model, this representation can be either explicitly depicted or implicitly used to generate samples from the learned distribution [17].

Some systems aim to generate a part of a musical piece with the aid of another given part (including melody-to-lyrics creation [18], the inverse of the task we consider). Conditioning the choice of parameters in a generative model on data from other modalities, such as a bass line or a structure, can yield controllable generation systems [19][p.82-83]. For the case of using chords to condition melody generation, a recent system adjusting a general adversarial network architecture has been presented in [20] with the option of generating melody lines over a given

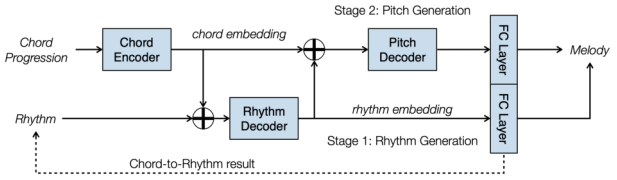


Figure 2: A two-stage structure of CMT, where $\oplus$ represents concatenation. Stage 1: chord-to-rhythm. Stage 2: rhythm+chord-to-pitch based on the result of Stage 1.

accompaniment. The Chord-conditioned Melody Transformer (CMT) [13] is the most recent effort in this area; we adapt much of the design of this system, extending it to accept both lyrics and chords as input. Details of this system, and how we adapt it, follow in Section 3.

3. METHODOLOGY

3.1 System Overview

Our system design is motivated by the Chord-conditioned Melody Transformer (CMT) [13]. The authors of CMT proposed a two-stage system, assuming a hierarchy that the process of generating melodies is two-phase, as depicted in Fig. 2: *Stage 1*, generating the rhythm of notes from chord progressions; *Stage 2*, generating the pitch for each note depending on the chord progressions and generated rhythm. Our proposed system augments CMT by replacing chord-to-rhythm (i.e., Stage 1) with a novel *lyric-to-rhythm module*. As a result, users can input the lyrics and chord progression of a full song in our system (see Fig. 1). Then, the lyric-to-rhythm module generates the MIDI (with empty pitches). Second, CMT processes the MIDI and chord progression to generate the melody. As a result, the rhythm is generated with a global view of the lyrics, while the melody is generated with a causal view of the rhythm and chords.

In the following subsections, we will first review CMT and explain the difficulties of modifying it to handle the lyric-to-melody task in Section 3.2. Then, we will detail our solution in Section 3.3.

3.2 Chord-Conditioned Melody Transformer (CMT)

CMT adopts a pianoroll-like representation [13, 21] that includes chord, rhythm, and pitch (CRP) information. It splits the timeline into semiquaver-length frames (1/4 of a beat), each described by three vectors: a 12-dimensional binary *chord* vector (pitch classes in the chord get a 1); a 3-dimensional one-hot *rhythm* vector (onset, hold state, rest state); and a 22-dimensional one-hot melodic *pitch* vector (for this part we restrict MIDI pitches to between 48 and 67, plus a hold state and a rest state, giving a total dimension of 22). Please refer to [13][Fig. 1] for an illustration.

CMT contains three main modules: *Chord Encoder (CE)*, a bidirectional LSTM [22]; *Rhythm Decoder (RD)*, a stack of self-attention blocks; and *Pitch Decoder (PD)*, another stack of self-attention blocks. In Stage 1, given an input chord progression, the chord embedding encoded

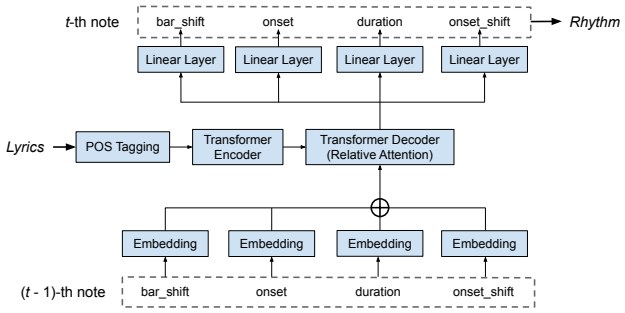


Figure 3: The proposed lyric-to-rhythm framework.

by CE is autoregressively sent into RD to output the rhythm embedding, followed by a fully-connected layer ("FC Layer" in Fig. 2) to predict the sequence of rhythm vectors for the entire song. In Stage 2, the concatenation of the chord and rhythm embeddings is autoregressively fed into PD, followed by a fully-connected layer to predict the sequence of pitch vectors. Finally, rhythm and pitch vectors are combined and converted to the melody.

However, to leverage CMT for the lyric-to-melody task, we face three problems. (1) *Multimodality*: CMT was designed to take the input of a chord progression to generate the melody. However, it is non-trivial to directly add a lyric encoder for lyrics input, as lyrics are more complicated sequential data than chords. (2) *Representation*: CMT uses a pianoroll-like (i.e. CRP) representation to encode melody, where the time axis is evenly scaled (e.g., 1/16 beat), so a note may require multiple tokens to carry the duration. This makes it difficult to create a one-to-one mapping that ties a syllable (or character) to a single note token. (3) *Constraint on length*: in CMT, the CE generates the melody on a segment-to-segment basis (e.g., 8 bars at a time) without exploiting the global context of a full-song chord progression. However, we believe the structural information carried in the input lyrics is crucial to determine the repetitive pattern for the output melody. The next subsection details how we address these problems: (1) is addressed with a POS tagger that compactly encodes useful lyrics information; and (2) and (3) are addressed by adapting a Compound Word representation.

3.3 Lyric-to-Rhythm Framework

Fig. 3 shows our lyric-to-rhythm framework, which is analogous to a language translation task: i.e., an input sequence of lyrics is translated into an output sequence of notes. In this work, we assume that each syllable (or character) is mapped onto one note as a simplification; handling melismas remains a future challenge. To this end, we adapt an encoder-decoder Transformer architecture [12]. To enhance the repetitive coherence modeling in note sequences, we incorporate relative self-attention [23, 24].

To extract the features of lyrics, we employ *part-of-speech (POS) tagging* with a Transformer encoder. Following prior works [25, 26], we characterize the rhythmic features of a note with a tuple of `(bar_shift, position, duration, onset_shift)`, and model the sequence of tuples using the *Compound Word (CP)* Transformer decoder [27]. The lyric-to-rhythm module generates the t-th note based on the full context of lyrics and the previously generated notes (from the first to (t-1)-th) in an auto-regressive manner, with the future notes being masked. We describe the POS tag representation and CP Transformer in the next subsections.

Token Name	Vocab.	Description
`bar_shift`	0, 1, 2	Time shift in bar to current bar
`onset`	0 – 15	Onset in 1/4 beat in current bar
`duration`	0 – 31	Duration in 1/4 beat
`onset_shift`	0 – 15	Time shift in 1/4 beat to previous note's onset

Table 1: Rhythmic features for a note.

3.3.1 Part-of-Speech (POS) Tagging

In natural language processing, POS tagging refers to the process of labeling every word in a text with its part of speech. The taxonomy of POS tags varies by language, but commonly includes 'noun,' 'verb,' 'adjective,' 'adverb,' and others. POS tags can augment the text information by indicating the structure of sentences [28], and thus plays an important role in tokenizing the input words in conventional text-to-speech (TTS) systems [29, 30].

POS are word-level descriptors, but we want syllable-level descriptors in order to align the lyrics with the rhythm. (When dealing with Chinese lyrics, we can also say 'character-level' since each Chinese character is one syllable.) Thus, we combine each POS tag with the syllable index to create a POS 'token': e.g., the input English sentence "Why not tell someone," would result in: ['adverb-0', 'adverb-0', 'verb-0', 'noun-0', 'noun-1'], where the two syllables in "someone" are represented by ['noun-0', 'noun-1'].

3.3.2 Compound Word Transformer

In contrast to the CP proposed in [27], we do not distinguish between *note*- and *metre*-related events. Instead, we include four tokens in every compound word (see Table 1), so that we can have one set of tokens per syllable. From karaoke scrolling lyrics data, we can obtain the onset and duration of each syllable, and by tracking the downbeats, we can obtain the metric position in the bar. Following [27], each of the four tokens is converted into an embedding, and then the embeddings are concatenated before being sent to the Transformer decoder. Each of the four output embeddings is linearly projected to predict the value for the associated token of the t-th note.

Once all the notes are ready, we convert them to MIDI (with unspecified pitch) with the following steps:

1. Place an empty note at bar 0 and position 0.
2. Determine the onset by shifting (`bar_shift`$\times 16+$ `onset_shift`) units from the previous onset.
3. Set the duration by $min($`duration`, next note's `onset_shift`$)$.
4. Repeat 2 and 3 until all the notes are processed.

We note that `onset` is not used for generating MIDIs. Instead, we use the position shifted to determine the onset so that notes are placed in an incremental order. Nevertheless, we suspect that `onset` can help regularization in training. Using the CP representation addresses the "constraint on length" issue mentioned in Section 3.2, as it permits a more compact sequence of tokens that can model a longer duration, such as a full song.

4. SYSTEM CONFIGURATION

This section describes how we trained the lyric-to-rhythm and rhythm-to-melody models. For each model we explain what data were used and how they were collected. We focus on Chinese pop songs to validate our system, but the framework could be adaptable to other languages since parts of speech and syllables are broadly useful concepts.

4.1 Lyric-to-Rhythm Model

We collected data for 45K Chinese pop songs using a similar pipeline as [31] and [7][Appendix A]. That is, we crawled online to obtain paired lyrics and audio, with timestamps indicating the onset of each line of the lyrics. Then, for each song, we performed the following steps: (1) isolate the vocal audio using source separation; (2) convert lyrics to phoneme sequences; (3) estimate the phoneme onset timestamps using forced phoneme alignment; and (4) estimate the time signature and beat and downbeat times. From the phoneme and beat data, we can derive the syllable onsets and thus the bar-shift, onset, duration and onset-shift attributes required by the model. The steps were performed using in-house tools comparable to those used in [7]: Spleeter [32], Phonemizer [33], Montreal Forced Aligner [34], and Madmom [35], respectively.

We kept songs with a detected time signature of 4/4 (around 90%), and quantized the timestamp of each syllable in quarter beats. Errors in automatic lyrics alignment, in beat tracking, and in the detected time signature can all degrade the model quality, so we selected 330 songs to manually adjust the timestamps. This subset was used to fine-tune the model.

For POS tagging, we adopted Jieba [1], an open-source tool that supports 56 tags commonly used in Chinese. Without POS tags, the vocabulary size for our dataset was 5,368 unique characters. Reducing to POS and then adding the syllable index resulted in a vocabulary of 123 unique POS tokens. In Sec. 5.2, we will compare a model using this 123-dimensional POS vector to an ablated version of the system that encodes the raw characters index in a 5368-dimensional vector.

In Chinese pop songs, symmetric expression of text structure is commonly reflected in melody repetition. Fig. 4 shows one example: the chorus melody of "Goodbye Kiss" [2] by Jacky Cheung. The two phrases outlined in solid boxes are identical in melody, and nearly identical in text; but even where the lyrics are different (in the dotted boxes), they have the same POS tags. With the POS tagging representation, we believe the lyric-to-rhythm model can learn to generate similar rhythms for two text phrases if they have a common structure.

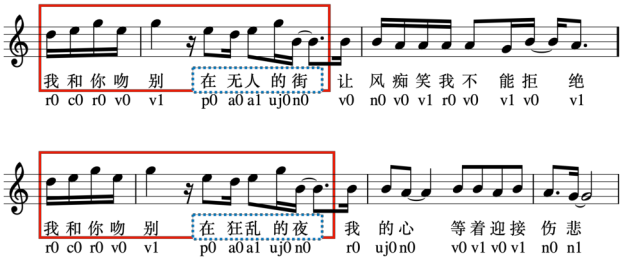


Figure 4: The melody-lyrics-POS example in the chorus section of "Goodbye Kiss". POS abbreviation key: {'r': pronoun, 'c': conjunction, 'v': verb, 'p': preposition, 'a': adjective, 'n': noun, 'uj': auxiliary}.

4.2 Rhythm-to-Melody Model

To train the rhythm-to-melody model, we used POP909 [36] and Lead-Sheet-Dataset [37]. POP909 contains data on 909 Chinese pop songs including chords, melody in MIDI format, and other information not used here. Lead-Sheet-Dataset (LSD), a collection of symbolic content sourced from HookTheory, [3] contains lead-sheets (i.e., melodies and chords) of 16K song segments in MIDI format. We used the songs in 4/4 time (dropping roughly 10% of the data) and transposed all pieces to the key of C major or A minor. This resulted in about 30K bars of music from POP909 and about 40K from LSD.

We followed [13] to train the model, setting the input length to be 8 bars but reducing the pitch range from 48 to 20; any pitches outside the range were octave-shifted to lie within the range. After obtaining the rhythm MIDI of a full song from the lyric-to-rhythm module, pitches were generated for 8 bars autoregressively, with a 4-bar sliding window, i.e., the model composes the next 4 bars given the previous 4 bars already composed.

5. EVALUATION

We would like to answer two questions: first, does our system succeed in emulating basic musical qualities of the training data? And second, does it produce pleasing, viable

[1] `https://github.com/fxsjy/jieba`
[2] `https://www.youtube.com/watch?v=bJRkEmrkIO4`
[3] `https://www.hooktheory.com/`

settings of lyrics? To answer the first, we compare the output melodies of our model (denoted 'pop-melody') to the held-out training data and discuss their similarity. For the second, we conducted a listening test in which participants rated the quality of the lyric settings of our model as well as those of a state-of-the-art alternative, TeleMelody [7].

5.1 Objective Results

We analyze the melodies created by our system in two objective evaluation strands. The first one is to demonstrate how similar the rhythms generated by our model are to the original data (see Fig. 5); the second is to look for and characterize the differences between the melodies produced by the two models (see Fig. 6). We compare statistics over several musical quantities computed on the dataset and compositions generated by both systems. For this comparison, we have generated 400 scores from each system and used the same amount of scores from the dataset.

Most of the musical quantities we compute are adapted from [38] and [39]. These symbolic descriptors have been shown to enhance melodic expectation when embodied in a cognitively plausible system for music prediction. Expectation and memorability have been shown to be important characteristics for identifying a plausible melody, and surprise and repetition are measurable elements that relate to these charactertistics. (For more background on such descriptors and on the concepts of predictability and uncertainty in the pleasure of music, see [40, 41].)

We showcase two sets of descriptors, one for each evaluation strand. The first contains: the *duration* of the melody notes; their *inter-onset intervals* (IOIs; the distance between the start of a note to the start of the preceding one); and their metrical *position in bar*. Fig. 5 shows the distributions of these descriptors for the dataset and for the outputs of our system (tagged as "pop-melody") before and after fine-tuning (see Section 4.1). Note that we exclude TeleMelody from this comparison since it was trained on a different dataset (of around 110K samples), so it is not meaningful to compare it to our training data.

Judging from the distributions, the outputs of both models are broadly similar to the melodies in the dataset. However, there is clearly a surfeit of short notes (0.25 crochets, or sixteenth notes) in the generated melodies, which skews the distribution of IOIs as a result. Also, regarding note position in the bar, there is a subtle variation of the likely onset positions in the dataset that is not reflected in the generated data, which, prior to fine-tuning, has an almost uniform distribution.

The other set of descriptors contains: the *pitch contour*, which gives the likelihood that the next note in the melody will be lower (descending), higher (ascending) or the same; *note sparsity*, which gives the fraction of the timeline which has no note in the melody (a value of 0 indicates no rests in the melody); and the *pitch-in-chord-triads ratio*, a kind of 'consonance' metric, calculated as the fraction of notes in the melody that belong to the accompanying chord triad.

These descriptors are illustrated in Fig. 6, comparing

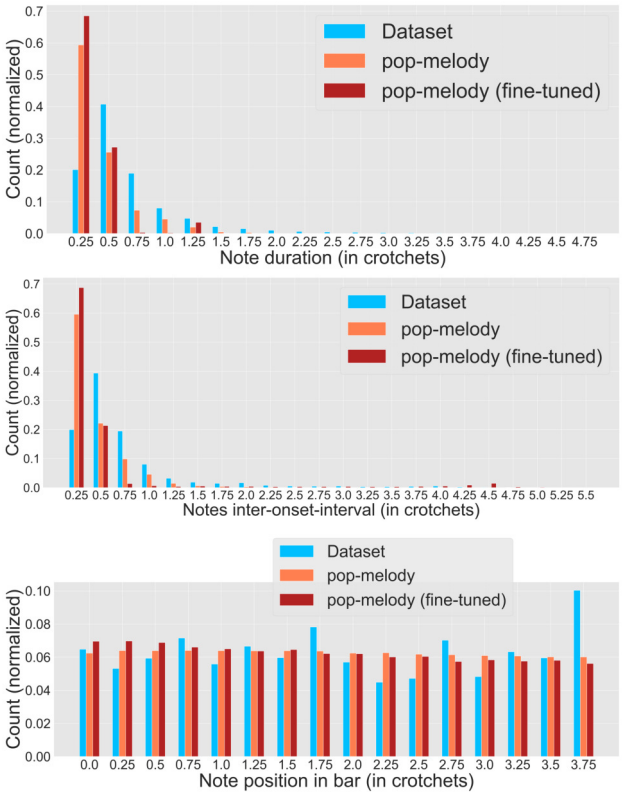


Figure 5: Distributions of descriptors values derived from melodies from the dataset and melodies generated by the proposed pop-melody system.

the melodies from our fine-tuned system ("pop-melody") with TeleMelody. Here, the purpose is not to compare the systems to the data—they were trained on different data, and may each reflect their training set well—but to assess how the melodies of the systems differ. From the contour descriptors, it is clear that TeleMelody is more likely to generate many consecutive notes with the same pitch, whereas melodies from the proposed system have more variation. The melodies from our system also tend to have fewer rests, and tend to include more notes that appear in the underlying chord. The latter can be interpreted as a tendency to stay in consonance and limiting the space of "dissonant" or "unexpected" moments.

5.2 Subjective Results

We conducted a subjective listening test using a similar design as [3, 7, 42]: we selected lyrics from ten random songs from the test portion of the dataset of 45K songs and used these as input to three systems to generate melodies: "TeleMelody"; "Pop-melody", the proposed system; and "Baseline", an ablated version of our system that does not use POS tokens (see Sec 4.1). This resulted in 30 full songs: ten triples with the same lyrics, chords, and tempo. We rendered the lyrics and melodies to audio with an in-house singing voice synthesis comparable to Xiaoice [42] and rendered a simple accompaniment with the chords.

We had 20 participants, all of whom had some musical background and could read musical scores and play an in-

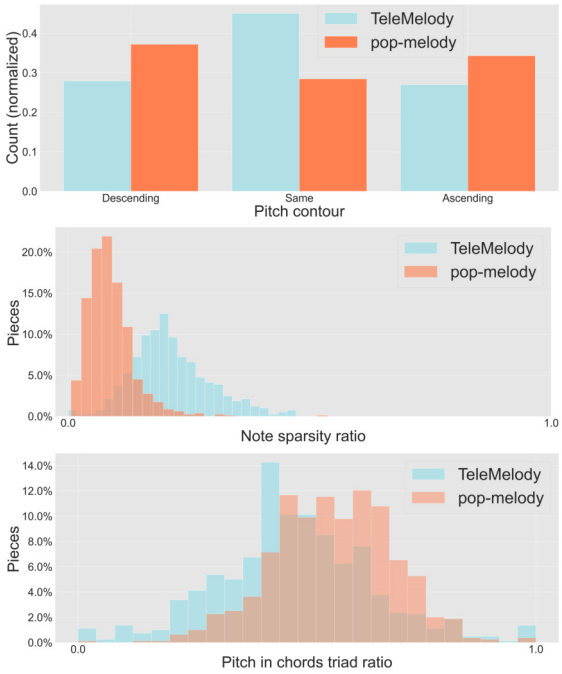


Figure 6: Distributions of descriptor values derived from melodies generated by TeleMelody [7] and by the proposed pop-melody system.

strument. Participants listened to a triple at a time, where the system identities were masked and the order is at random. Then, they rated each song on four criteria on a Likert scale from *Bad* (1) to *Excellent* (5):

1. **Rhythm**: is the timing of notes suitable for the lyrics?
2. **Harmony**: do the pitches fit the chords and key?
3. **Melody**: does the melody line sound natural with the lyrics?
4. **Overall**: what is the overall quality of the melody?

After rating each triple, listeners also rated their familiarity with the original song of the input lyrics on a 5-point Likert scale. The average rating here was 1.5: somewhere between "1. Never heard the title or melody" and "2. Heard the song title, but not the melody".

The results of the study are shown in Table 2. Overall, listeners gave the three systems similar average ratings: all lie within 3.6 ± 0.25. However, Wilcoxon signed-rank tests reveal small but consistent differences between the systems; see Table 3 for the p-values of all comparisons. First, we see that the proposed system is consistently better than Baseline, suggesting that POS-based tokenization is effective. Second, we find that the proposed system also matches or outperforms TeleMelody; the difference is greatest for rhythmic quality. Despite the broadly positive ratings, mostly between *Fair* (3) and *Good* (4), comments from the participants mostly cited shortcomings of the output. TeleMelody and Pop-melody both earned comments that the "melody is a little weird" and sometimes "too repetitive", but only the TeleMelody outputs earned comments that the "rhythm is a little weird" and "fragmented".

System	Rhythm	Harmony	Melody	Overall
Baseline	3.42(.93)	3.67(.86)	3.46(.93)	3.42(.82)
TeleMelody	3.58(.96)	3.69(.90)	3.38(.85)	3.57(.73)
Pop-melody	3.84(.87)	3.87(.81)	3.64(.89)	3.68(.72)

Table 2: Subjective result and comparison.

Comparison	Rhythm	Harmony	Melody	Overall
Pop vs Tele	0.0006	0.007	0.001	0.1
Pop vs Baseline	1.1e-08	0.002	0.006	1.8e-05
Tele vs Baseline	0.01	0.89	0.22	0.02

Table 3: P-values of the subjective result comparison.

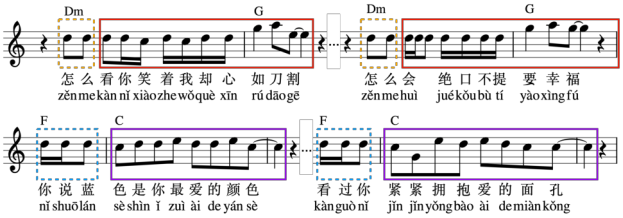


Figure 7: Output examples (a) above and (b) below.

Two output examples from our system are shown in Figs. 7(a) and 7(b). In both cases the melodies follow the input chord progressions and we find parallelism and variation in the melody when the lyric structure recurs. E.g., in Fig. 7(a), the similar lyrics begin with the same two melody notes (dashed boxes), and the remainders have similar rhythm and contour (solid boxes). Similarly, in Fig. 7(b), the similar lyrics are given identical openings (dashed boxes) with rhythmically identical continuations (solid boxes).

6. CONCLUSION AND FUTURE WORK

In this paper we proposed a new approach to generate melody for a given lyric by combining lyric-to-rhythm and rhythm-to-melody modules. We found that listeners rated the long-term text-settings provided by our system as acceptable, and at least as good as a competing system.

In order to achieve a cross-modal mapping from syllables to onsets to melody notes, we made the simplifying assumption that each syllable is sung on one note. There is a clear way to improve this in the lyric-to-rhythm module by adding a syllable-state token to the Compound Word, indicating whether we are at the onset of a syllable, or the continuation of one. However, allowing a one-to-many syllable-to-note mapping would also complicate the automatic syllable alignment step, making the hand-corrected data even more precious.

We also found that POS tags were valuable text tokens; using them led to a boost in text-setting quality. Given this success, we ought to leverage more linguistic information, such as syllable stress and word frequency, as in [15, 16]. Music structure labels (e.g., verse and chorus) could also prove valuable. This is an under-explored area, but may become feasible with the introduction of more datasets, or with automatic labelling systems [43] to further augment existing data. if more datasets become available.

7. REFERENCES

[1] P. Pasquier, A. Eigenfeldt, O. Bown, and S. Dubnov, "An introduction to musical metacreation," *Computers in Entertainment (CIE)*, vol. 14, no. 2, pp. 1–14, 2017.

[2] Y. Yu, A. Srivastava, and S. Canales, "Conditional LSTM-GAN for melody generation from lyrics," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 1, pp. 1–20, 2021.

[3] Z. Sheng, K. Song, X. Tan, Y. Ren, W. Ye, S. Zhang, and T. Qin, "Songmass: Automatic song writing with pre-training and alignment constraint," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 35, no. 15, 2021, pp. 13 798–13 805.

[4] H. Bao, S. Huang, F. Wei, L. Cui, Y. Wu, C. Tan, S. Piao, and M. Zhou, "Neural melody composition from lyrics," in *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, 2019, pp. 499–511.

[5] H.-P. Lee, J.-S. Fang, and W.-Y. Ma, "iComposer: An automatic songwriting system for Chinese popular music," in *Proc. Conference of the North American Chapter of the Association for Computational Linguistics*, 2019, pp. 84–88.

[6] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," in *Proc. 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8821–8831.

[7] Z. Ju, P. Lu, X. Tan, R. Wang, C. Zhang, S. Wu, K. Zhang, X. Li, T. Qin, and T.-Y. Liu, "TeleMelody: Lyric-to-melody generation with a template-based two-stage method," *arXiv preprint arXiv:2109.09617*, 2021.

[8] K. Chen, C.-i. Wang, T. Berg-Kirkpatrick, and S. Dubnov, "Music SketchNet: Controllable music generation via factorized representations of pitch and rhythm," in *Proc. ISMIR*, 2020, pp. 77–84.

[9] S. Dai, Z. Jin, C. Gomes, and R. B. Dannenberg, "Controllable deep melody generation via hierarchical music structure representation," in *Proc. ISMIR*, 2021, pp. 143–150.

[10] K. Chen, G. Xia, and S. Dubnov, "Continuous melody generation via disentangled short-term representations and structural conditions," in *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*. IEEE, 2020, pp. 128–135.

[11] H. H. Tan, "ChordAL: A chord-based approach for music generation using Bi-LSTMs." in *Proc. International Conference on Computational Creativity (ICCC)*, 2019, pp. 364–365.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[13] K. Choi, J. Park, W. Heo, S. Jeon, and J. Park, "Chord conditioned melody generation with transformer based decoders," *IEEE Access*, vol. 9, pp. 42 071–42 080, 2021.

[14] E. Nichols, D. Morris, S. Basu, and C. Raphael, "Relationships between lyrics and melody in popular music," in *Proc. ISMIR*, 2009, pp. 471–476.

[15] S. Fukayama, K. Nakatsuma, S. Sako, T. Nishimoto, and S. Sagayama, "Automatic song composition from the lyrics exploiting prosody of the Japanese language," in *Proc. 7th Sound and Music Computing Conference (SMC)*, 2010, pp. 299–302.

[16] E. Nichols, "Lyric-based rhythm suggestion," in *Proc. International Computer Music Conference*, 2009.

[17] I. J. Goodfellow, "NIPS 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[18] X. Ma, Y. Wang, M.-Y. Kan, and W. S. Lee, "AI-Lyricist: Generating music and vocabulary constrained lyrics," in *Proc. 29th ACM International Conference on Multimedia*, 2021, pp. 1002–1011.

[19] J.-P. Briot, G. Hadjeres, and F. D. Pachet, *Deep learning techniques for music generation*. Springer Cham, 2020.

[20] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proc. ISMIR*, 2017, pp. 324–331.

[21] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proc. ISMIR*, 2019, pp. 596–603.

[22] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional LSTM networks for improved phoneme classification and recognition," in *International Conference on Artificial Neural Networks*. Springer, 2005, pp. 799–804.

[23] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proc. Conference of the North American Chapter of the Association for Computational Linguistics*, 2018, pp. 464–468.

[24] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[25] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "MusicBERT: Symbolic music understanding with large-scale pre-training," *arXiv preprint arXiv:2106.05630*, 2021.

[26] Y.-H. Chou, I. Chen, C.-J. Chang, J. Ching, Y.-H. Yang *et al.*, "MidiBERT-Piano: Large-scale pre-training for symbolic music understanding," *arXiv preprint arXiv:2107.05223*, 2021.

[27] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, 2021, pp. 178–186.

[28] M. Sun and J. R. Bellegarda, "Improved POS tagging for text-to-speech synthesis," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 5384–5387.

[29] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition*, 2nd ed. Prentice Hall, 2008.

[30] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda, "The HMM-based speech synthesis system (HTS) version 2.0." in *SSW*, 2007, pp. 294–299.

[31] L. Xue, K. Song, D. Wu, X. Tan, N. L. Zhang, T. Qin, W.-Q. Zhang, and T.-Y. Liu, "DeepRapper: Neural rap generation with rhyme and rhythm modeling," in *Proc. Association for Computational Linguistics and International Joint Conference on Natural Language Processing*, 2021, pp. 69–81.

[32] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: A fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020.

[33] M. Bernard and H. Titeux, "Phonemizer: Text to phones transcription for multiple languages in Python," *Journal of Open Source Software*, vol. 6, no. 68, p. 3958, 2021.

[34] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal Forced Aligner: Trainable text-speech alignment using Kaldi." in *Proc. Interspeech*, 2017, pp. 498–502.

[35] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proc. 24th ACM International Conference on Multimedia*, 10 2016, pp. 1174–1178.

[36] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proc. ISMIR*, 2020, pp. 38–45.

[37] Y.-C. Yeh, W.-Y. Hsiao, S. Fukayama, T. Kitahara, B. Genchel, H.-M. Liu, H.-W. Dong, Y. Chen, T. Leong, and Y.-H. Yang, "Automatic melody harmonization with triad chords: A comparative study," *Journal of New Music Research (JNMR)*, vol. 50, no. 1, pp. 37–51, 2021.

[38] D. Conklin and I. H. Witten, "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, 1995.

[39] M. T. Pearce, "The construction and evaluation of statistical models of melodic structure in music perception and composition," Ph.D. dissertation, City University London, 2005.

[40] B. P. Gold, M. T. Pearce, E. Mas-Herrero, A. Dagher, and R. J. Zatorre, "Predictability and uncertainty in the pleasure of music: A reward for learning?" *Journal of Neuroscience*, vol. 39, no. 47, pp. 9397–9409, 2019.

[41] D. Müllensiefen, "Fantastic: Feature analysis technology accessing statistics (in a corpus): Technical report v1," *London, England: Goldsmiths, University of London. Retrieved from http://www.doc.gold.ac.uk/isms/m4s*, 2009.

[42] H. Zhu, Q. Liu, N. J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, and E. Chen, "Xiaoice band: A melody and arrangement generation framework for pop music," in *Proc. 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2837–2846.

[43] J.-C. Wang, Y.-N. Hung, and J. B. L. Smith, "To catch a chorus, verse, intro, or anything else: Analyzing a song with structural functions," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 416–420.

Papers - Session II

VISUALIZATION FOR AI-ASSISTED COMPOSING

Simeon Rau[1*] **Frank Heyen**[1*] **Stefan Wagner**[2] **Michael Sedlmair**[1]
[1] VISUS, University of Stuttgart, Germany
[2] ISTE, University of Stuttgart, Germany
[*] Equal contribution
simeon.rau@visus.uni-stuttgart.de, frank.heyen@visus.uni-stuttgart.de

ABSTRACT

We propose a visual approach for interactive, AI-assisted composition that serves as a compromise between fully automatic and fully manual composition. Instead of generating a whole piece, the AI takes on the role of an assistant that generates short melodies for the composer to choose from and adapt. In an iterative process, the composer queries the AI for continuations or alternative fill-ins, chooses a suggestion, and adds it to the piece. As listening to many suggestions would take time, we explore different ways to visualize them, to allow the composer to focus on the most interesting-*looking* melodies. We also present the results of a qualitative evaluation with five composers.

1. INTRODUCTION

Composing music is a challenging task, especially for beginners. Complex music theory, different rules and patterns in a wide range of genres, missing experience, or even stagnating inspiration make it difficult to express intended ideas and emotions. These aspects can result in frustration and leave the composer unsatisfied or even unable to finish a piece. Artificial intelligence (AI) might potentially mitigate such situations by generating music that provides inspiration or even fits the composer's needs directly.

Current AI-driven approaches for music generation allow creating parts or whole pieces in audio [1] or symbolic form [2]. Although automatic composition can be useful, for example as background music in videos or games [3], it often lacks personality and structure [3, 4]. A fine-grained steering of the generation process, beyond parameters such as overall tempo and feel, is often not possible. Therefore, a piece that is fully generated by AI might not satisfy expectations [5]. Although AI-generated music can show potential creativity [6], the current state of AI music will most likely not replace human composers as a whole [5].

To address above problems, we propose a user-centered approach for AI-assisted composing. Here, users have more control, acting as a leading composer who always has the last word, while the AI serves as assistant that supports composers instead of replacing them [4, 5, 7]. When assisted with suggestions for continuations, fill-ins, and replacements, users can iteratively elaborate on these and make progress towards a personal creation. Composing step by step often yields better results than selecting from multiple completely generated ones and gives a greater feeling of satisfaction and authorship [8].

To support the communication between AI and human, we propose using interactive visualization. As it allows to quickly spot and filter the most interesting suggestions, visualization facilitates choice. This means users do not have to spend time sifting through heaps of less interesting suggestions and can spend their time more efficiently, for example by only listening to a few melodies per group of similar ones. Furthermore, visualization can improve understanding of how the underlying AI works [9], and allow for a more effective usage and better steering of the AI.

We make two primary contributions: (1) We designed four interactive visualizations to assist the user in choosing and investigating AI-generated suggestions. The first two represent a graph structure of suggestions [10–12], while the third displays larger numbers of melody samples, by encoding their similarities and characteristics. Our fourth visualization shows the correlation between melody samples. (2) We evaluated our approach through a qualitative study with five composers, who generally liked our approach and were curious in analyzing the AI and using visualizations to find interesting samples. Results show that our representations were unfamiliar to most participants, but made interaction with the AI more accessible.

Our supplemental material [1] contains a live version of our prototype, the full source code, and additional details.

2. RELATED WORK

Briot et al. [13] surveyed deep learning for music generation and compare objectives, data representations, architectures, and the output that consists of audio [1] or symbolic data [14]. They cover recurrent neural networks [2, 15], variational autoencoders [16–18], and generative adversarial networks [19, 20]. Other work [14, 21] shows combinations of above techniques for different tasks. Recently, transformers [22–24] showed promising results and there also exist agent-based or heuristic algorithms [25].

© S. Rau, F. Heyen, S. Wagner, and M. Sedlmair. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** S. Rau, F. Heyen, S. Wagner, and M. Sedlmair, "Visualization for AI-Assisted Composing", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] github.com/visvar/vis-ai-comp

Figure 1. Overview of our design: (A) Piano roll to show/edit the composition. (B) Icicle plot and (C) node-link chart for tree structures. (D) Similarity-based scatterplot as overview of all samples. Color encodes the AI's temperature parameter.

A survey by Huang et al. [7] on usage and requirements of AI for music co-creation found that users desire control, authorship, and creative freedom and often create multiple samples to choose from. Our approach addresses these requirements and supports sample choice through visual overviews of samples and second-level continuations. We allow for fine-grained control and creative freedom by letting the user replace parts with fill-ins and edit individual notes. Suh et al. [26] found users to be more willing to take risks and try different styles when using AI, but also to be pushed to progress faster. Our approach uses AI as an assistant to provide similar benefits with more user control.

Magenta Studio [27] allows generating music with different tools but limited steering options. Our design uses similar options but allows generating many samples at once and gives full control over note selection and adaption. *Bach Doodle* [28] allows users to harmonize an input melody in the style of Bach via *CoCoNet* [29] and was later improved [30] in its usability for novice users. Our approach focuses on creating the melody itself, which could then later be harmonized through similar models. *SketchNet* [31] gives users control in music generation, allowing to roughly sketch melodies. For editing melodies, Tsuchiya et al. [32] proposed a non-notewise method. In contrast, we allow choosing and notewise editing suggestions before combining them into a composition and could integrate above work to make suggestions more relevant.

Steering interfaces (selecting from options part-by-part) are often preferred over radio options (selecting from longer options), as they allow for more control and efficacy, giving a stronger sense of authorship [8]. Following that idea, Huang [2] uses a tree structure of generated continuations for interactive composition. We use a similar idea, but additionally support cognition and control through visualization and more interaction.

There are many interactive tools that use AI to generate music [23, 33–35] or fill-in missing parts [36, 37] that provide various ways of control, but lack exploration of different choices and fine-grained editing. In contrast, we provide visual exploration techniques and options, such as single-note edits or overviews of many options at once.

Our visualizations make use of similarity metrics for melodies. Such metrics have been studied in prior work, for example recursive metrics [38] and edit distances [39] for symbolic melodic similarity. Instead of comparing complete pieces, we use our metrics to produce overviews of a collection of short melodies by visually sorting or positioning them by these metrics. Abstractly, we follow the idea of visual parameter analysis [40] by visualizing AI output for different parameters to get insights about their impact, which has not yet been explored systematically for semi-automatic music composition. For example, we use glyphs [41, 42] to show melody samples in a scatterplot, similar to music collections [43]. We also use tree visualisations such as icicle plots and node-link diagrams, which have been used for event sequences before [12], but have not yet been explored for a combination of music and AI.

3. DESIGN

With our approach, we target two primary user groups, namely amateur composers and developers/analysts of

[2] youtu.be/vnRrGCB04WE?t=1293

composer AI models. We focused on amateurs as a starting point, as we hypothesized that they could benefit most from a continuous interaction with a composing AI. In our approach, users query the AI for suggestions but make all decisions, keeping the composition personalized.

The AI analyst, on the other hand, is interested in investigating a composing AI's behaviour and the influence of different input melodies and parameters on output suggestions. Understanding this behaviour is crucial for the user's trust and leads to more efficient steering.

The main workflow of our approach looks as follows: Users start with recording or generating a seed melody. Based on it, they query the AI for multiple continuation suggestions with chosen parameters. We then visualize these samples to help users filter, listen to, select, and customize the most interesting continuations for a personal composition. Repeating these steps, or even creating multiple levels of continuations, results in a tree of melody samples, where a sample's children are the possible continuations for the remaining composition. Using the same steps to replace an existing part (fill-in) splits the composition node in the graph and adds the fill-in samples as nodes in between, resulting in a directed acyclic graph (Figure 2).

This approach is abstractly inspired by the idea of visual parameter space analysis [40], and also practically by the artist MJx Music. He used an AI to create an album by manually generating hundreds of samples, listening to all, selecting the most interesting, and combining them into songs [3]. In contrast to this brute-force workflow, we add visualization to provide similar flexibility but more usability, by showing overviews from which users then pick the most interesting samples to further investigate.

To make this general idea possible, we needed to investigate metrics and aggregations to be able to relate different samples to each other, as well as visualization designs which allow users to interact with the space of samples.

3.1 Metrics and Aggregations

We designed a range of metrics that help our visualizations arrange or summarize a collection of melody samples. A user might have some idea of prioritization when looking for samples with specific properties. Some of our visualizations therefore allow sorting by different aspects, such as the variance of interval sizes as a statistical metric to detect lively, varied, or monotone melodies. We can also sort by two metrics at once, with a scatterplot that uses each metric as one axis. As a starting point, we chose the following metrics: mean note duration, variance of interval sizes, similarity to composition, distinct pitches, range of pitches, temperature (AI parameter affecting randomness [4]), and number of noteshapes from composition. The latter indicates how often a shape of three consecutive notes (intervals between them) occur in the composition.

The similarity between samples can also be used for sorting, for example to focus on samples that are more similar to parts from the composition. Furthermore, pairwise similarities between all samples can be used to place them on a "map" where similar ones are closer to each other. We designed a rather simple similarity metric that takes both pitch and rhythm into account, and allows the user to choose a weight for controlling the impact of both.

Above metrics can also be used to compactly represent samples, by only visualizing the metrics instead of all notes, as we do in our glyphs and histograms.

Our example metrics are not necessarily complete or optimal, but serve as a starting point for further research. For some tasks, what is optimal can even be hard to define at all or be subjective, such as sorting by "happiness" or determining the degree of "similar feeling", possibly requiring users to fine tune or train metrics by themselves. Our main approach works with any kind of metric and could be easily extended in the future.

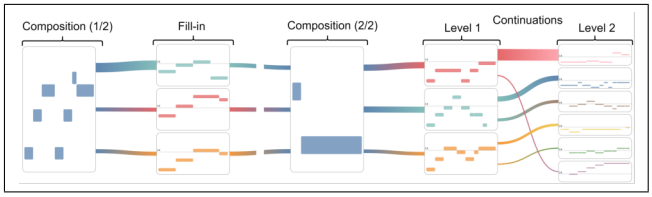


Figure 2. Node-link diagram with one fill-in and two levels of continuations. Colors help differentiate between melody samples. Here, nodes in the same level are sorted by variance of intervals descending, which is also encoded in the link width.

3.2 Visualizations

Piano Rolls as Note Representation: Piano rolls serve both as representation and editor for notes. We chose piano rolls over staff notation for multiple reasons: They are more easy to read for beginners, represent each pitch (MIDI note) in its own row, visually show rhythm and breaks through block size and gaps, and are better readable when small, as there are less fine details. The central view of our tool is a large piano roll at the top that shows all melody samples and the composition at all time (Figure 1A). In this view, we allow adjusting pitch, start, and duration of notes, as well as adding or deleting them.

Sample Relation Graphs: Users can create a tree structure of melody samples to see different continuation paths for a given melody. Listening to all paths takes time, so we want to make selection more efficient by extending and supporting the sequential listening process with a parallel visual approach. To this end, we visualize the tree structure and its melody samples together (Figure 2).

Displaying all samples in a single piano roll at once would lead to overlap and clutter. Therefore, we show the tree structure in an icicle plot, using most space to display piano rolls (Figure 1B). Children (continuations) of a sample are displayed on the right of its node, dividing its height equally. All piano rolls share a common timeline on the X-axis while the Y-axis is different for each. With different color encodings, users can differentiate between melodies or get additional information, for example about

[3] magenta.tensorflow.org/mj-hip-hop-ep
[4] magenta.tensorflow.org/performance-rnn

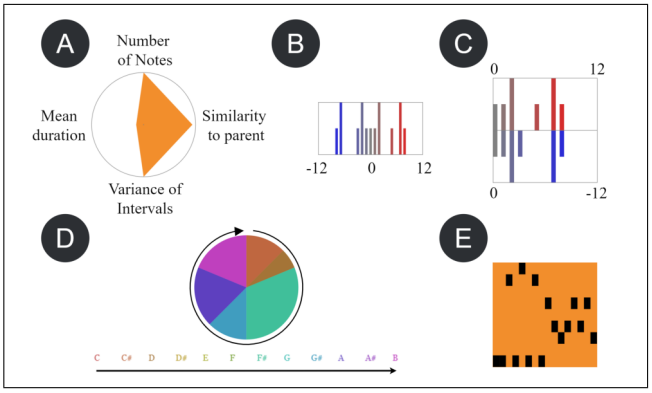


Figure 3. Comparison of our glyphs on the same data. (A) star glyph, (B, C) occurrences of intervals between consecutive notes, (D) chroma distribution, and (E) piano roll.

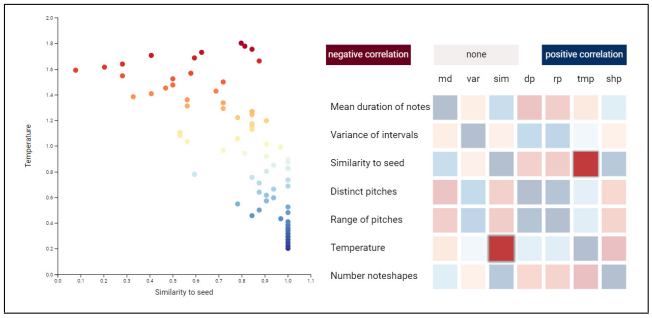


Figure 4. Negative correlation between temperature and similarity. Colors encode temperature (left) and correlation (right).

the parameters used to generate them. We allow listening to single samples or selecting a whole path or parts of it. Users can add a path to the composition and subsequently adjust notes manually or generate new continuations.

Sorting all melodies of the same tree level, which are alternatives for the same time span, could be useful when focusing on some priority. Since an icicle plot already encodes relations through position, sorting would break this encoding. Therefore, we added links between nodes to allow sorting melodies of each level while keeping relations visible, resulting in a node-link graph (Figure 2). Besides showing relationships, these links also encode the value of the selected sorting metric in their width.

Similarity-Based Layout and Glyphs: A composer can only use an AI efficiently if they roughly understand how it behaves. In our case, this means knowing which parameter values (e.g., temperature) to choose for an intended output. Since there is usually some uncertainty in the results, more than just a few samples are needed to draw conclusions.

Above visualizations do not scale to large numbers of samples, as small piano rolls are hard to read. While sorting and scrolling help, the overview gets lost. To provide an overview for hundreds of samples at once, we visualize them as dots in a scatterplot, placing similar once closer together via dimensionality reduction (Figure 1D).

Using a circular brush, users can select a neighborhood and take a look at its melodies, represented as piano rolls and statistical aggregations of all selected samples. To indicate the overall variety of the selection, we show two histograms for the occurrences of pitches and note durations.

Because selecting neighborhoods at random is inefficient, the user needs an impression of melodies directly inside the scatterplot. We therefore replaced the dots by glyphs, small symbols showing some kind of data, such as statistics or the melodic structure itself. Users can switch between these glyphs at any time to visually filter interesting melodies before looking at their neighbourhood in detail. As overlapping glyphs are unreadable, we apply gridification to produce a regular, occlusion-free layout.

We designed four types of glyphs to show different aspects of a melody (Figure 3): The first type shows multiple metrics in a *starglyph*. As default, four metrics are shown that convey both rhythm and melody: note count, mean note duration, variance of intervals between two notes, and similarity to notes in the current composition. Jumps between notes are often interesting for composers, for example when looking for a climax or a calm section after it, inspiring us to show the occurrences of intervals between two notes in a *histogram glyph*. Users can directly see when a melody mostly uses repeating notes and small changes or has larger jumps in it. The third type of glyph represents the tonality of melodies by a *pie chart of chroma occurrences*, i.e., one slice for each of the twelve notes. It helps selecting or investigating melodies based on contained notes and therefore tonality, as well as looking for outliers with unusual pitches. For example, a pie chart showing mostly C, E, and G likely fits into a C major composition. Because above glyphs represent metrics that can be hard to grasp for unfamiliar users, we added a fourth type that shows melodies as *small piano rolls*. With these glyphs, the rough contour of the melody is directly visible, so users can look for those matching their intention.

Correlation Analysis: An analyst can use the above scatterplot and glyphs to investigate sample metrics, but could also be interested in investigating correlation between metrics. We therefore provide a correlation matrix as overview of pairwise correlations between all metrics, which encodes Pearson correlation as color (Figure 4). We show negative, positive, and no correlation in red, blue, and white, so users can quickly filter interesting pairs. After choosing a pair of metrics by clicking, a 2D scatterplot shows all samples as points, positioned based on their metric values. One insight we made was a positive correlation between temperature and pitch range, so higher temperature values often lead the AI to generate more outlier notes.

Attribution: A common problem when co-creating with AI is authorship, as users are often hesitant to claim generated melodies as their own. To analyze this problem and provide a feeling for how much authorship a composer has, we added a coloring for the notes' attribution (Figure 5). We assign each note one of five attribution classes, covering the range between human- and AI-generated, and display the percentage of each class, to show composers how much and what they contributed to the composition.

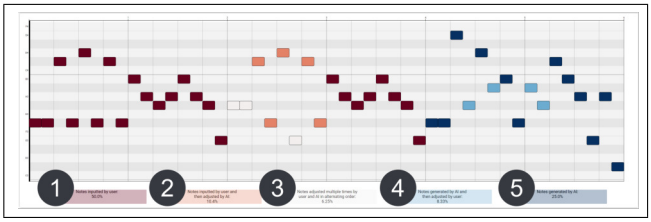


Figure 5. Color indicates the attribution of notes, from red (by user) to blue (by AI) along five steps: (1) completely human-created notes, (2) human-created but AI-adapted notes (fill-in), (3) notes changed by both parties multiple times, (4) AI-generated but user-updated notes, and (5) AI-generated, unedited notes.

3.3 Implementation

We implemented our prototype as a web app build with React, which allows publishing it as a website [1] . It works completely inside the browser with an AI model [5] made by Magenta [6] , which we chose due to easy access and usage, but other models from any framework or language can be hosted by others and integrated through plugins.

For our visualizations, we use D3 [44] for colormaps and scales, DruidJS [45] and MDS [46, 47] for computing dimensionality reduction layouts, and Hagrid [48] to avoid overlapping glyphs. Users can record melodies using MIDI [49] devices and export finished compositions as MIDI files for further processing with other tools.

4. EVALUATION

As music composition is a complex and time-consuming task without clear measures for efficiency and success, we chose a qualitative study design with five domain experts (P1 to P5). P1, P3, and P4 were pursuing a degree in composition for experimental music, P2 holds a degree in composition and studies performance, and P5 studied a music instrument major and composes occasionally. Although our approach primarily targets beginners, we chose experts, as they are more familiar with composition workflows. We therefore expected more detailed, thorough, and critical feedback from this group.

Our design brings potentially unfamiliar technology to participants, such as machine learning, visualization, and dimensionality reduction. To avoid them having to learn using our design, we conducted pair analytics [50], where domain experts and visualization designers jointly analyze data, while the designers serve as technical assistants.

We recorded screen and voice throughout the study with consent. The study concluded with a semi-structured interview to further inquire about general thoughts. On average, participants took 2.2 hours for the study. Due to space limits, we only discuss the main findings here, full details can be found in the supplemental material.

[5] MusicRNN, magenta.github.io/magenta-js/music/classes/
[6] github.com/magenta/magenta-js

Results: All participants started with recording and editing a short melody. They were interested in how the AI would react to their melodies, especially P3: *"I was curious, so I put in some short and some long notes"*. Next, they generated some continuations for their melody.

Participants found our icicle plot helpful *"when generating many samples, [this] can help [to] see more quickly which samples are interesting and which are not"* (P4). We found that the icicle plot led to a faster selection of samples by just looking at them: *"I can see that these [points to two samples] are very interesting"* (P5). Especially P4 and P5 liked this representation due to the miniature piano rolls that helped P5 find certain intervals in melodies.

Participants used the node-link diagram to sort samples, inspect them, and select continuations. They liked sorting by metrics: *"I think it's good to sort by intervals [... it is] essential when selecting a melody with specific characteristics"* (P1), and found it helpful to use different metrics: *"I can imagine for complex music and many samples [...] and having an rough idea, [sorting] can help using parameters"* (P4), *"[When] composing with dissonance [...] sorting by dissonance would help"* (P3).

While composing, P3 repeatedly asked where recently added samples began within the composition. Our attribution visualization showed the difference between notes added by the AI and edited notes, which helped P3: *"where was the last note? [switching to attribution] This visual is very comfortable"* (P3).

We were specifically interested in how participants would select samples to interact with them. They often had some idea for what they were looking for, such as setting a contrast (P1), wanting *"to hear the most randomness"* (P2), or variations and less randomness (P4), where our visualization helped: *"[This] is the only one that goes down"* (P5). Participants sometimes did not like samples (*"I liked the first part, but the second was not good"* (P2)) and edited them after adding to the composition (P5) or even discarded all to *"change something on my own"* (P3).

As we also wanted to evaluate AI analysis, participants then generated 50 to 80 continuations to analyze with our visualizations. P1 found all glyphs helpful and told us these could *"extend the intuitive analysis of composers [... and] lead to a different style"* in composing. The participants were *"not used to look at the data"* (P2, P3) and these kinds of visualizations, but most learned quickly and found the scatterplot *"very interesting as an analysis tool, even independent of the AI"* (P1). Especially P1 was very interested in all the analysis possibilities and mentioned that to their knowledge, the lack of analysis is a drawback to algorithmic composition that works with randomness: *"I did see that really rarely and this is a huge shortcoming [...] when composing with the computer"* (P1).

Surprisingly, P1 told us how their professor answered the question of *"how to select the best samples?"*: *"You could generate as often as you like and listen to all and decide while listening intuitively"*. P1 complained about this strategy: *"You would listen to all and have no overview"*. Our scatterplot provides such an overview (P1).

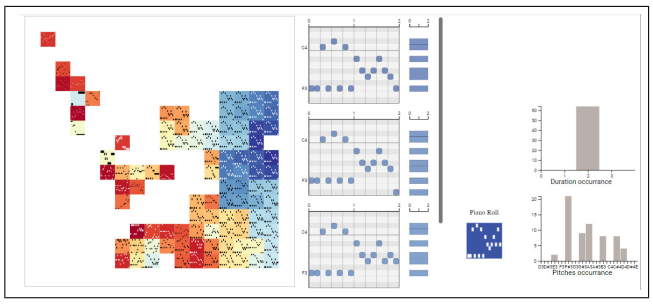


Figure 6. Similarity based scatterplot with piano roll glyphs colored by temperature (left). Selected samples are shown as piano rolls (center) and summarized as histograms (right).

We evaluated how users select neighborhoods in the scatterplot and found that the color (encoding temperature) impacted their decision. For example, P2 *"clicked there because of the color [...] ones with higher and lower temperature"* and P3 investigated *"how melodies vary"* based on temperature. Especially interesting to us was that P5 changed the similarity metric to rhythm only, to look for samples in a cluster with the original rhythm. P5 then clicked on an outlier sample: *"Why is this separate?"*.

The glyph representations were used by all participants. Piechart glyphs helped P3 select a melody sample, when they were looking for something calm that contrasts the composition. While analyzing, P5 searched for perfect fourths with the interval histogram glyphs, as their melody contained a lot of these. *"I am interested in finding melodies with perfect fourths [showing occurrences of interval glyph]. We can directly see [...] there is almost nothing"* (P5). Other participants used this glyph to select samples *"when I have a structure in mind [such as a rising line]"* (P1), which showed more positive intervals than negative, or to investigate the extreme melodies with many larger intervals. All participants used piano roll glyphs to compare melodic structures and find common pattern in clusters: *"I can see the same musical structure (melodic bows) [...] and further away it turns different"* (P1).

All participants found a relationship between temperature, shown through the color of piano rolls, and the structure (Figure 6): *"The blue ones have clearer structure while red ones are jumpier without that much connection"* (P1), *"I can see the randomness"* (P2). We found the piano roll to be more intuitive than other glyph types and therefore a good default, as it shows melodies directly: *"For me [the piano roll glyph] is very intuitive, I can directly imagine how they approximately could sound like"* (P4).

Especially P5 found that *"[correlation visualization] is fantastic [...]. This would make analysis much easier even without using the AI"*. After investigating some metric combinations, P5 surprised us by finding a pattern regarding similarity and temperature, where the correlation between an arbitrary metric and the similarity would always be the opposite to when combining the metric with temperature (Figure 4).

Our study concluded with short semi-structured interviews, summarized below. None of the participants preferred our approach over their current workflow – which we expected, as experts learned their workflow over years – but many could image using it for inspiration or certain tasks. All participants found the visualizations helpful for comparing, filtering, and selecting melody samples, especially for beginners, even when P2 had problems imaging the sound of a melody by its visuals. Most mentioned that it takes time to get used to the unfamiliar visualizations and metrics. Despite general scepticism against AI, all participants saw potential value in an AI-assisted approach.

5. LIMITATIONS

The sample relation visualizations do not scale for larger numbers of samples or levels, which could be addressed by not showing all samples at once but using scrolling or filtering. As alternative, the scatterplot can show many samples, but struggles with displaying similarities and complete glyphs. Using a gridified layout poses a trade-off between overlap and inexact positions.

Our example metrics and glyphs might not be optimal and miss some key characteristics. The current implementation is limited to monophonic melodies, but can be extended to polyphony through other AIs and metrics. Some glyphs only represent one aspect of music, but composers often need to consider multiple at once, such as pitch *and* time. For sorting, we only used statistical metrics of melodies, which are interesting for some experimental composers, but could be hard to interpret for beginners or other composers. These users would need more musical characteristics like mood, which is harder to calculate.

Another limitation of our approach is learnability, as users have to learn reading and interpreting a set of visual encodings. Based on our evaluation, we believe that regular users of our design can quickly get an intuition by looking at and listening to a range of samples.

6. CONCLUSION

As AI will likely not fully replace human composers in the foreseeable future [5], we advocate for an approach in which both, human and AI, cooperate. To this end, we propose a user-centered, interactive, and visual approach for iterative, AI-assisted music composition aimed at hobby musicians and experimental composers. We designed different visualizations as interface for interaction with generative models. Since these visualizations are AI-agnostic, a composer could use any kind of AI or general algorithm in combination with our approach.

In the future, we want to test different models [16, 22] to generate polyphonic music, harmonize melodies, extend metrics, and allow better steering. Analyzing the composition process could benefit from a history visualization that shows which notes were added or changed when and by whom. We further plan to explore aggregations and glyphs for more compact representations and extend our evaluation to more diverse composers, including beginners.

7. ACKNOWLEDGEMENTS

This work was funded by the Cyber Valley Research Fund and the Artificial Intelligence Software Academy (AISA).

8. REFERENCES

[1] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," in *Proc. International Conf. Learning Representations (ICLR)*, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1612.07837

[2] S. Oore, I. Simon, S. Dieleman, and D. Eck, "Learning to create piano performances," in *NIPS Workshop on Machine Learning and Creativity*, 2017. [Online]. Available: https://nips2017creativity.github.io/doc/Learning_Piano.pdf

[3] D. Herremans, C.-H. Chuan, and E. Chew, "A functional taxonomy of music generation systems," *ACM Comput. Surv.*, 2017. [Online]. Available: https://doi.org/10.1145/3108242

[4] C. Hernandez-Olivan and J. R. Beltran, "Music composition with deep learning: A review," 2021. [Online]. Available: https://doi.org/10.48550/ARXIV.2108.12290

[5] S. Knotts and N. Collins, "A survey on the uptake of music AI software," in *Proc. International Conf. New Interfaces for Musical Expression (NIME)*, 2020, pp. 499–504. [Online]. Available: https://doi.org/10.5281/zenodo.4813499

[6] T. Mejtoft, L. Lagerhjelm, U. Söderström, and O. Norberg, "Creative capabilities of machine learning: Evaluating music created by algorithms," in *European Conf. Cognitive Ergonomics (ECCE)*, 2021, pp. 1–4. [Online]. Available: https://doi.org/10.1145/3452853.3452863

[7] C. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, and C. J. Cai, "AI song contest: Human-AI co-creation in songwriting," in *Proc. International Society for Music Information Retrieval Conf. (ISMIR)*, 2020, pp. 708–716. [Online]. Available: https://doi.org/10.5281/zenodo.4245530

[8] R. Louie, J. Engel, and A. Huang, "Expressive communication: A common framework for evaluating developments in generative models and steering interfaces," 2021. [Online]. Available: http://arxiv.org/abs/2111.14951

[9] F. Heyen, T. Munz, M. Neumann, D. Ortega, N. T. Vu, D. Weiskopf, and M. Sedlmair, "ClaVis: An interactive visual comparison system for classifiers," in *Proc. International Conf. Advanced Visual Interfaces (AVI)*, 2020, pp. 1–9. [Online]. Available: https://doi.org/10.1145/3399715.3399814

[10] S. Bruckner and T. Möller, "Result-driven exploration of simulation parameter spaces for visual effects design," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, pp. 1468–1476, 2010. [Online]. Available: https://doi.org/10.1109/TVCG.2010.190

[11] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman, "LifeFlow: Visualizing an overview of event sequences," in *Proc. Conf. Human Factors in Computing Systems (CHI)*, 2011, p. 1747–1756. [Online]. Available: https://doi.org/10.1145/1978942.1979196

[12] Z. Liu, B. Kerr, M. Dontcheva, J. Grover, M. Hoffman, and A. Wilson, "CoreFlow: Extracting and visualizing branching patterns from event sequences," *Computer Graphics Forum (CGF)*, pp. 527–538, 2017. [Online]. Available: https://doi.org/10.1111/cgf.13208

[13] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, *Deep learning techniques for music generation*. Springer, 2020. [Online]. Available: https://doi.org/10.1007/978-3-319-70163-9

[14] E. S. Koh, S. Dubnov, and D. Wright, "Rethinking recurrent latent variable model for music composition," in *Proc. IEEE International Workshop Multimedia Signal Processing (MMSP)*, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/MMSP.2018.8547061

[15] G. Hadjeres and F. Nielsen, "Anticipation-RNN: Enforcing unary constraints in sequence generation, with application to interactive music generation," *Neural Computing and Applications*, pp. 995–1005, 2020. [Online]. Available: https://doi.org/10.1007/s00521-018-3868-4

[16] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proc. International Conf. Machine Learning (PMLR)*, 2018, pp. 4364–4373. [Online]. Available: https://doi.org/10.48550/arXiv.1803.05428

[17] A. Weber, L. N. Alegre, J. Torresen, and B. C. da Silva, "Parameterized melody generation with autoencoders and temporally-consistent noise," in *Proc. International Conf. New Interfaces for Musical Expression (NIME)*, 2019, pp. 174–179. [Online]. Available: https://doi.org/10.5281/zenodo.3672914

[18] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," in *Proc. International Society for Music Information Retrieval Conf. (ISMIR)*, 2019, pp. 343–351. [Online]. Available: http://doi.org/10.5281/zenodo.3527814

[19] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proc. AAAI Conf. Artificial*

Intelligence, vol. 32, no. 1, 2018. [Online]. Available: https://doi.org/10.48550/arXiv.1709.06298

[20] L. C. Yang, S. Y. Chou, and Y. H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proc. International Society for Music Information Retrieval Conf. (ISMIR)*, 2017, pp. 324–331. [Online]. Available: https://doi.org/10.5281/zenodo.1415990

[21] R. T. Dean and J. Forth, "Towards a deep improviser: a prototype deep learning post-tonal free music generator," *Neural Computing and Applications*, pp. 969–979, 2020. [Online]. Available: https://doi.org/10.1007/s00521-018-3765-x

[22] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer: Generating music with long-term structure," in *International Conf. Learning Representations (ICLR)*, 2019, pp. 1–13. [Online]. Available: https://doi.org/10.48550/arXiv.1809.04281

[23] J. A. T. Lupker, "Score-Transformer: A deep learning aid for music composition," in *Proc. International Conf. New Interfaces for Musical Expression (NIME)*, 2021. [Online]. Available: https://doi.org/10.21428/92fbeb44.21d4fd1f

[24] T. Nuttall, B. Haki, and S. Jorda, "Transformer neural networks for automated rhythm generation," in *Proc. International Conf. New Interfaces for Musical Expression (NIME)*, 2021. [Online]. Available: https://doi.org/10.21428/92fbeb44.fe9a0d82

[25] O. Lopez-Rincon, O. Starostenko, and G. A.-S. Martín, "Algoritmic music composition based on artificial intelligence: A survey," in *Proc. International Conf. Electronics, Communications and Computers (CONIELECOMP)*, 2018, pp. 187–193. [Online]. Available: https://doi.org/10.1109/CONIELECOMP.2018.8327197

[26] M. Suh, E. Youngblom, M. Terry, and C. J. Cai, "AI as social glue: Uncovering the roles of deep generative AI during social music composition," in *Proc. Conf. Human Factors in Computing Systems (CHI)*, 2021, pp. 1–11. [Online]. Available: https://doi.org/10.1145/3411764.3445219

[27] A. Roberts, J. Engel, Y. Mann, J. Gillick, C. Kayacik, S. Nørly, M. Dinculescu, C. Radebaugh, C. Hawthorne, and D. Eck, "Magenta Studio: Augmenting creativity with deep learning in Ableton Live," in *Proc. International Workshop on Musical Metacreation (MUME)*, 2019. [Online]. Available: https://doi.org/10.5281/zenodo.4285266

[28] C. A. Huang, C. Hawthorne, A. Roberts, M. Dinculescu, J. Wexler, L. Hong, and J. Howcroft, "The Bach Doodle: Approachable music composition with machine learning at scale." in *Proc. International Society for Music Information Retrieval Conf. (ISMIR)*, 2019, pp. 793–800. [Online]. Available: http://doi.org/10.5281/zenodo.3527930

[29] T. Chakraborti, B. McCane, S. Mills, and U. Pal, "CoCoNet: A collaborative convolutional network applied to fine-grained bird species classification," in *35th International Conf. Image and Vision Computing New Zealand (IVCNZ)*, 2020, pp. 1–6. [Online]. Available: http://doi.org/10.1109/IVCNZ51579.2020.9290677

[30] R. Louie, A. Coenen, C. Z. Huang, M. Terry, and C. J. Cai, "Novice-AI music co-creation via AI-steering tools for deep generative models," in *Proc. Conf. on Human Factors in Computing Systems (CHI)*. ACM, 2020, p. 1–13. [Online]. Available: https://doi.org/10.1145/3313831.3376739

[31] K. Chen, C. Wang, T. Berg-Kirkpatrick, and S. Dubnov, "Music SketchNet: Controllable music generation via factorized representations of pitch and rhythm," in *Proc. International Society for Music Information Retrieval Conf. (ISMIR)*, 2020, pp. 77–84. [Online]. Available: http://doi.org/10.5281/zenodo.4245372

[32] Y. Tsuchiya and T. Kitahara, "A non-notewise melody editing method for supporting musically untrained people's music composition," *Journal Creative Music Systems (JCMS)*, 2019. [Online]. Available: https://doi.org/10.1007/BF02289565

[33] Y. Zhang, G. Xia, M. Levy, and S. Dixon, "COSMIC: A conversational interface for human-AI music co-creation," in *Proc. International Conf. New Interfaces for Musical Expression (NIME)*, 2021. [Online]. Available: https://doi.org/10.21428/92fbeb44.110a7a32

[34] C. A. Huang, D. Duvenaud, and K. Z. Gajos, "ChordRipple: Recommending chords to help novice composers go beyond the ordinary," in *Proc. International Conf. Intelligent User Interfaces (IUI)*, 2016, p. 241–250. [Online]. Available: https://doi.org/10.1145/2856767.2856792

[35] N. Privato, O. Rampado, and A. Novello, "A creative tool for the musician combining LSTM and Markov chains in Max/MSP," in *Proc. International Conf. EvoMUSART, Held as Part of EvoStar*, 2022, p. 228–242. [Online]. Available: https://doi.org/10.1007/978-3-031-03789-4_15

[36] R. Guo, I. Simpson, C. Kiefer, T. Magnusson, and D. Herremans, "MusIAC: An extensible generative framework for music infilling applications with multi-level control," in *Proc. International Conf. EvoMUSART, Held as Part of EvoStar*, 2022, pp. 341–356. [Online]. Available: https://doi.org/10.1007/978-3-031-03789-4_22

[37] T. Bazin and G. Hadjeres, "NONOTO: A model-agnostic web interface for interactive music composition by inpainting," in *Proc. International Conf. Computational Creativity (ICCC)*, 2019, pp. 89–91. [Online]. Available: http://doi.org/10.48550/ARXIV.1907.10380

[38] S. Park, T. Kwon, J. Lee, J. Kim, and J. Nam, "A cross-scape plot representation for visualizing symbolic melodic similarity," in *Proc. International Society for Music Information Retrieval Conf. (ISMIR)*, 2019, pp. 423–430. [Online]. Available: https://doi.org/10.5281/zenodo.3527834

[39] M. Gotham and M. Ireland, "Taking form: A representation standard, conversion code, and example corpus for recording, visualizing, and studying analyses of musical form," in *Proc. International Society for Music Information Retrieval Conf. (ISMIR)*, 2019, pp. 693–699. [Online]. Available: https://doi.org/10.5281/zenodo.3527904

[40] M. Sedlmair, C. Heinzl, S. Bruckner, H. Piringer, and T. Möller, "Visual parameter space analysis: A conceptual framework," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, pp. 2161–2170, 2014. [Online]. Available: https://doi.org/10.1109/TVCG.2014.2346321

[41] M. Ward, C.-h. Chen, W. K. Härdle, and A. Unwin, "Multivariate data glyphs: Principles and practice," in *Handbook of Data Visualization.* Springer, 2008, pp. 179–198.

[42] A. Klippel, F. Hardisty, and C. Weaver, "Star Plots: How shape characteristics influence classification tasks," *Cartography and Geographic Information Science (CaGIS)*, pp. 149–163, 2009. [Online]. Available: https://doi.org/10.1559/152304009788188808

[43] O. Gomez, K. K. Ganguli, L. Kuzmenko, and C. Guedes, "Exploring music collections: An interactive, dimensionality reduction approach to visualizing songbanks," in *Proc. International Conf. Intelligent User Interfaces (IUI)*, 2020, p. 138–139. [Online]. Available: https://doi.org/10.1145/3379336.3381461

[44] M. Bostock, V. Ogievetsky, and J. Heer, "D^3: Data-driven documents," *IEEE Trans. Visualization and Computer Graphics (TVCG)*, vol. 17, no. 12, pp. 2301–2309, 2011. [Online]. Available: https://doi.org/10.1109/TVCG.2011.185

[45] R. Cutura, C. Kralj, and M. Sedlmair, "Druid$_{JS}$ —— a javascript library for dimensionality reduction," in *Proc. IEEE Visualization Conf. (VIS)*, 2020, pp. 111–115. [Online]. Available: https://doi.org/10.1109/VIS47514.2020.00029

[46] J. B. Kruskal, "Nonmetric multidimensional scaling: A numerical method," *Psychometrika*, pp. 115–129, 1964. [Online]. Available: https://doi.org/10.1007/BF02289694

[47] ——, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, pp. 1–27, 1964. [Online]. Available: https://doi.org/10.1007/BF02289565

[48] R. Cutura, C. Morariu, Z. Cheng, Y. Wang, D. Weiskopf, and M. Sedlmair, "Hagrid —— gridify scatterplots with Hilbert and Gosper curves," in *International Symp. Visual Information Communication and Interaction (VINCI)*, 2021, pp. 1–8. [Online]. Available: https://doi.org/10.1145/3481549.3481569

[49] R. A. Moog, "MIDI: Musical instrument digital interface," *Journal Audio Engineering Society (AES)*, pp. 394–404, 1986. [Online]. Available: https://doi.org/10.4324/9780203484272-524

[50] R. Arias-Hernandez, L. T. Kaastra, T. M. Green, and B. Fisher, "Pair analytics: Capturing reasoning processes in collaborative visual analytics," in *Hawaii International Conf. System Sciences (HICSS)*, 2011, pp. 1–10. [Online]. Available: https://doi.org/10.1109/HICSS.2011.339

RETRIEVING MUSICAL INFORMATION FROM NEURAL DATA: HOW COGNITIVE FEATURES ENRICH ACOUSTIC ONES

Ellie Bean Abrams[1,2,3] **Eva Muñoz Vidal**[1,2,3] **Claire Pelofi***[1,2] **Pablo Ripollés***[1,2,3]

[1] Music and Audio Research Laboratory, New York University
[2] Center for Language, Music, and Emotion, New York University
[3] Department of Psychology, New York University
* denotes shared last authorship
{ea84,elm8254,cp2830,pr82}@nyu.edu

ABSTRACT

Various features – from low-level acoustics, to higher-level statistical regularities, to memory associations – contribute to the experience of musical enjoyment and pleasure. Recent work suggests that *musical surprisal*, that is, the unexpectedness of a musical event given its context, may directly predict listeners' experiences of pleasure and enjoyment during music listening. Understanding how surprisal shapes listeners' preferences for certain musical pieces has implications for music recommender systems, which are typically content- (both acoustic or semantic) or metadata-based. Here we test a recently developed computational algorithm, called the Dynamic-Regularity Extraction (D-REX) model, that uses Bayesian inference to predict the surprisal that humans experience while listening to music. We demonstrate that the brain tracks musical surprisal as modeled by D-REX by conducting a decoding analysis on the neural signal (collected through magnetoencephalography) of participants listening to music. Thus, we demonstrate the validity of a computational model of musical surprisal, which may remarkably inform the next generation of recommender systems. In addition, we present an open-source neural dataset which will be available for future research to foster approaches combining MIR with cognitive neuroscience, an approach we believe will be a key strategy in characterizing people's reactions to music.

1. INTRODUCTION

Musical surprisal, or, the relative expectations listeners have of ongoing musical events, is essential in understanding humans' engagement and experience with music [1]. Decades of theoretical and experimental work have defined the study of expectation and surprisal as cognitive processes, often in the context of language processing [2, 3]. Importantly, this line of inquiry has crucial implications for the study of music preference and pleasure [4, 5]. Recent studies have shown that the relationship between information-theoretic measures such as surprisal, entropy, or complexity, and enjoyment and pleasure may be described by an inverted U-shape curve (also referred to as the Wundt effect) where stimulus enjoyment is enhanced with an increase in complexity of the song. But as surprisal increases to higher levels, its effect becomes unpleasant [4, 6]. The perceptual measures (e.g. complexity, familiarity/novelty, surprisal) that have been shown to modulate musical preferences are referred to as *collative* variables [7–9]. Supposedly, when other high-level variables are controlled, collative variables explain a large portion of variance in listeners' musical preference [8, 10] following the aforementioned parabolic function. The "sweet spot" of this inverse U-shaped curve may shift left or right, with respect to surprisal measures, depending on factors such as personality, openness-to-experience, or genre preferences [9, 11]. While most music recommender systems rely on acoustic (extracted from a user's music library) and semantic features (derived from subjective behavioral ratings) to predict listeners' preferences, the use of cognitive measures such as music surprisal can remarkably improve their performance, especially because these are known to accurately predict musical pleasure. The results presented here suggest that cognitive neuroscience methods can be leveraged to elucidate new ways of extracting information from music and better predict user preferences.

Early theoretical work on bottom-up and top-down processes modulating expectations by Leonard Meyer and Eugene Narmour [12–14] brought about efforts to model musical prediction, evolving more concretely into explorations of musical tension [15, 16], entropy [17], and the neural bases of surprisal [18, 19]. Crucially, computational models may be tested against both subjective behavioral and objective neurophysiological measures in order to provide a deeper understanding of whether – and how precisely – information-theoretic measures of music inform the cognitive processes underlying music perception and enjoyment. The Information Dynamics of Music (IDyOM) model [20] generates variable-order Markov

© E.B. Abrams, E.M. Vidal, C. Pelofi, and P. Ripollés. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** E.B. Abrams, E.M. Vidal, C. Pelofi, and P. Ripollés, "Retrieving musical information from neural data: how cognitive features enrich acoustic ones", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

probability distributions for each note in a melodic sequence by extracting statistics from both a music corpora and a short-term musical context, thereby incorporating long-term (top-down) and short-term (bottom-up) musical regularities. The IDyOM model has been shown to predict behavioral and physiological markers of listeners' expectations [21–25] and has recently been related to brain activity, showing that melodic expectation is also directly encoded in the neural signal [26–31]. While IDyOM is a well-validated, efficient, and widely used computational model of musical surprisal, it operates only on symbolic (MIDI) data and requires a training set of stimuli (the long-term component of the model) to generate predictions.

To circumvent these shortcomings and explore the behavioral and neural response to continuous audio signals, we turn to a computational model of surprisal recently developed by Skerritt-Davis and Elhilali at Johns Hopkins, the Dynamic Regularity Extraction (D-REX) model [32, 33]. D-REX uses a Bayesian framework to generate predictions and was originally designed to evaluate prediction errors over time for stochastic sound sequences. This model is relevant to the MIR community, as it can be run on any continuous audio input, which broadens its usability for the analysis of large, diverse collections of music. Our previous behavioral results have validated D-REX as predictive of subjective ratings of surprisal, showing that surprisal as calculated by D-REX predicted subjective behavioral surprisal ratings for 80 music excerpts [34]. Given the important role that prediction plays in musical pleasure, enjoyment, and engagement [4, 6, 35], D-REX is used here to explore whether *the brain* tracks musical surprisal. Concretely, we go beyond subjective behavioral responses and test whether musical surprisal as calculated by D-REX is directly represented in an objective neurophysiological signal. Specifically, we present an experimental work in which we recorded brain activity using magnetoencephalography (MEG) while twenty participants listened to musical excerpts. We then relate the neural signal to the D-REX model output using a decoding algorithm to determine whether surprisal is represented at the brain level.

2. DATA COLLECTION AND PROCEDURE

Twenty participants with self-reported normal hearing completed the experiment (11 female, 24.8 $\pm$ 2.9 years of age). Participants were presented with 30 one-minute-long musical excerpts (described in Section 3.1) while their brain activity was recorded using MEG. Participants began each trial by clicking a button to start playing each musical excerpt. At the end of each excerpt, participants moved to the next stage where they provided ratings across five measures using a 4-point scale (1 lowest to 4 highest): pleasure, valence, recognition, familiarity, and surprisal.

MEG measures the magnetic fields generated by the electrical activity of neurons in the brain. Unlike electrical currents captured by EEG devices, magnetic activity can pass through the cortex and skull without distortion, resulting in higher spatial resolution [36]. Continuous MEG data was collected using a 157-channel axial gradiometer system at NYU, at a sampling rate of 1000Hz with an online low-pass filter of 200Hz. Prior to conducting the main analysis, MEG data underwent preprocessing, starting with the noise-reduction of the signal using the continuously adjusted least squares method (CALM) with the MEG160 software [37]. The data was then exported into MNE-Python [38] and bad channels (e.g., channels which saturated during the recording) were removed through visual inspection and interpolated using a weighted sum of signals from neighboring channels. An independent component analysis (ICA) was fitted on the data using FastICA in MNE-Python to isolate independent sources of noise contaminating the channels. Components corresponding to system noise, heartbeat, and eye-blinks were removed from the raw recording after inspection of the topography and time-course of magnetic activity for each component. Finally, epochs were extracted from one second before stimulus onset to stimulus offset, resulting in 61-second-long epochs. For an additional data quality check, we inspected each participant's auditory response to a set of randomized 1000Hz tones and 250Hz tones and observed a higher amplitude response to the higher tone, thus confirming satisfactory data quality.

3. STIMULI

The musical stimuli used here have been previously used to validate D-REX as a predictor of behavioral subjective ratings of surprisal using different genres of music (classical and elevator music) [34]. Classical stimuli were taken from a list of musical excerpts rated by 65 participants for pleasantness in a previous study [39]. The other music stimuli were selected from a range of sources, including songs from Muzak Orchestra's Stimulus Progression albums, as well as more contemporary elevator music compositions (see Supplementary Table S1 [1] for a list of stimuli). The most interesting minute (highest accumulated surprisal) of each piece was selected using a shifting window of 60s across D-REX's surprisal output, so that any results showing lower correlations with brain activity would not simply be due to choosing a particularly unsurprising minute of the piece (see [34]). All excerpts were normalized to 70dB using Praat and python's AudioSegment package, and the sound faded 3s in 3s out.

4. MODELING MUSICAL SURPRISE

4.1 Acoustic Features

D-REX takes as input a set of acoustic features, which can be extracted using any existing MIR techniques. In our case, we extracted features using the NSL Auditory-Cortical Matlab Toolbox developed by the Neural Systems Laboratory at the University of Maryland. The toolbox is an implementation of a cortical model of sound processing, and outputs an estimate of sound as it is represented

[1] Supplementary materials may be downloaded at `https://osf.io/dbm49/`.

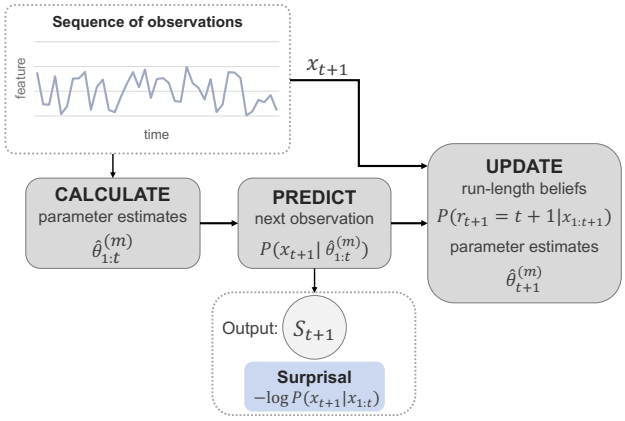


Figure 1. Simplified schematic of D-REX model. The model collects parameter estimates from run-lengths 0 until memory constraint m and generates a prediction for the next observation. At x_{t+1}, the model updates its run-length beliefs and parameter estimates. The output of the model used in the current project is surprisal S_{t+1}.

at various stages of the auditory pathway. We followed the same procedure as detailed in Huang and Elhilali [40], where each acoustic waveform was processed through log-spaced asymmetrical cochlear filters (from 255 Hz to 10.3 KHz). A cortical, or multi-resolution spectrotemporal representation was then generated for a 4-D output (scale-rate-time-frequency). *Rate* and *scale* refer to the two bandwidths (temporal and spectral, respectively) which characterize the filter making up the typical spectrotemporal receptive field of an auditory neuron [41]. *Time* and *frequency* refer to the dimensions of a sound's spectrogram. Fifteen features were extracted from the resulting auditory cortical representation, including average spectral energy, average rate energy, average scale energy, bandwidth, average bark loudness, pitch value, pitch salience (harmonicity), spectral brightness, spectral flatness, spectral irregularity, maximum rate (maximum of temporal variations along each frequency channel), maximum scale (maximum of the spread of spectral energy along the logarithmic frequency axis), centroid rate (centroid of temporal variations along each frequency channel), centroid scale (centroid of the spread of spectral energy along the logarithmic frequency axis), and centroid rate using absolute value of rate (see Supplementary Table S2 and [40] for a complete list of how features were extracted) [34, 40].

4.2 Surprisal Output: Dynamic Regularity Extraction (D-REX) Model

D-REX uses Bayesian sequential prediction with perceptual constraints, such as memory (m) and observation noise (n) to model the brain processes which govern expectation-realization over time in response to sound sequences (as depicted in Figure 1) [32, 33]. Memory m represents the working memory constraints of the listener, determining the maximum previous time points included in context hypotheses for the next time point. Observation noise n consists of adding independent Gaussian noise with zero-mean and a constant variance n^2 to the input. The model takes as input a continuous feature over time x_t extracted directly from the waveform as described in Section 4.1. From each feature, the model predicts the distribution for the next time point x_{t+1} given previous ones $x_{1:t}$. Concretely, previous context is estimated after collecting local statistics $\hat{\theta}$, which correspond to the sample mean and sample variance of the input. Predictions for future time-points are based on these statistical representations of previous events:

$$P(x_{t+1}, |x_{1:t}) = P(x_{t+1}|\hat{\theta}_t) \tag{1}$$

The model assumes the parameters θ can change at any time. A run represents the number of time points between change points of θ. Thus, the model generates multiple hypotheses by gathering statistics across runs and integrates over all possible run lengths (r_t) to predict the observation at the next time point:

$$P(x_{t+1}, |x_{1:t}) = \sum_{r_t} P(x_{t+1}|r_t, x_{t-r_t+1:t}) P(r_t|x_{1:t}) \tag{2}$$

The output of interest is surprisal, S_{t+1}, which refers to how well the new observation was predicted by the model. Specifically, it is the mismatch between the observation x_{t+1} and its predictive probability in bits:

$$S_{t+1} = -\log P(x_{t+1}|x_{1:t}) \tag{3}$$

As conveyed in Eqn (3), surprisal is inversely related to the event's probability: an event with low probability has high surprisal, an event with high probability has low surprisal, and an event with a probability of 1 has zero surprisal.

The D-REX model in this case takes a matrix of features (the 15 extracted features described in Section 4.1) and calculates surprisal as above *separately* across each feature vector. Then, D-REX combines all the outputs by getting the product of predictive probability (described in Eqn (3)) *across* features for a *summary measure* of joint surprisal. In our analysis, we use Surprisal to refer to *joint* surprisal, that is, the summary measure of surprisal across all 15 feature inputs (as in [34, 40]).

5. ANALYSIS

5.1 Decoding Method: Temporal Response Function

To decode acoustic (e.g., envelope) and cognitive (e.g., surprisal) features from the neural data, we used a Temporal Response Function (TRF) approach [42]. This decoding algorithm (`https://github.com/mickcrosse/mTRF-Toolbox`) describes the linear mapping of stimuli features onto a set of channels of neural activity. In this context, the input consists of the time-series collected at each MEG electrode n sampled at times $t = 1, ...T$. The assumption is made that the neural response at some time-point $r(t, n)$ can be described as the convolution of a specific stimulus feature $s(t)$ with a channel-specific kernel w_n, as described in Eqn (4):

$$r(t,n) = (s * w_n)(t) + \varepsilon(t,n) \quad (4)$$

where $\epsilon(t,n)$ is the residual noise at each channel that is not explained by the model. The kernel w_n, thus, is essentially a filter that describes the linear transformation of the stimulus feature into the neural response. Its weights are estimated for a specified range of time lags, τ, relative to the instantaneous occurrence of the stimulus feature $s(t)$, in order to capture the typical input-output activity of interest. In the context of auditory processing, the range of time lags over which to estimate $w(\tau,n)$ should be those used to capture the neural components of an auditory Event-Related Potential (ERP; e.g. -100-600 ms). Therefore, the weight at τ 100, for instance, describes how one unit of change in amplitude of a given input feature affects the neural response 100 ms later [43].

The weights $w(\tau,n)$ are estimated by minimizing the unexplained residual response $\epsilon(t,n)$, in this case by minimizing the difference (in terms of mean-squared error) between the actual neural response, $r(t,n)$, and the predicted $\hat{r}(t,n)$ response:

$$min\ \varepsilon(t,n) = \sum_t [r(t,n) - \hat{r}(t,n)]^2 \quad (5)$$

Concretely, this is achieved using ridge regression. This decoding method takes into account the high autocorrelation property of continuous stimuli such as music, thus avoiding the temporal smearing that would result from less sophisticated decoding approach, such as cross-correlation. We can interpret the continuous weights $w(n)$ for each τ as a *modeled-ERP*: a marker of the feature encoding into the neural data.

To summarize, the modeled-ERPs we will discuss describe the linear transformation of the stimulus feature into the neural response, and it can provide insights into the *dynamics* of the neural response to this specific feature [42]. As such, a modeled-ERP's shape may reveal cognitive processes usually observed through ERP analyses, such as the P1 and P2 components [43]. This method presents a useful alternative to averaging across segments of the neural response to derive ERP profiles from continuous signals, which is not precise nor informative enough. Speech envelope [44–46], phonemes [47, 48], semantics [49], and more recently, musical syntax [46] have been successfully decoded from brain data using TRF modeling [46–51].

5.2 Decoding Model Input

The decoding analysis was conducted twice using two different inputs: a purely acoustic signal (the first derivative of the amplitude envelope of the audio) and a cognitive signal (the surprisal D-REX output). The amplitude envelope of music carries modulations occurring in the 2-10 Hz range that capture the temporal patterns of critical rhythmic information [52]. Consistent results demonstrate a reliable cortical tracking of the music envelope [46, 53]. Therefore, we first conducted the decoding analysis using the envelope information, as a baseline for cortical tracking of low-level acoustical features [51, 53]. The broadband amplitude envelope was extracted using the Hilbert transform for each individual excerpt. The obtained signal was low-pass filtered at 30 Hz and down-sampled at 100 Hz. Then, the first derivative Envelope' was extracted (see Figure 2A), as it is well-known to enhance stimulus-neural response mapping when using linear system identification methods [54].

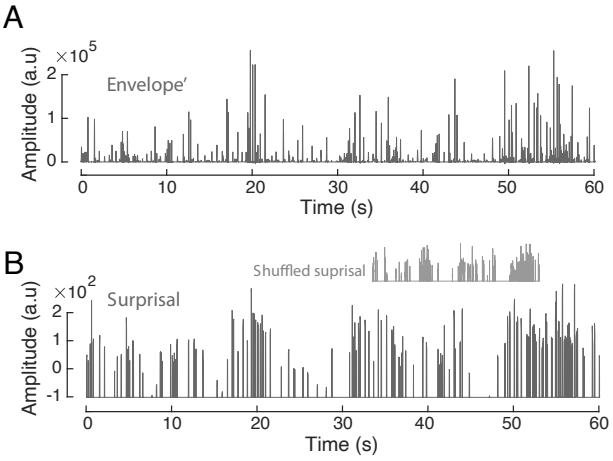


Figure 2. Example of decoding model input. **(A)** The first derivative of the envelope for one excerpt. **(B)** Surprisal as calculated using D-REX and indexed at each onset. Shuffled surprisal, shown in grey, is used for the null model.

The decoding analysis was then conducted using the Surprisal signal as input. Because the D-REX output is a smooth, continuous signal that does not constitute a good input for a TRF-based decoding method [42, 54], we used onset information to extract surprisal values associated with note onsets. From the Envelope', we extracted onset indexes by selecting indexes of amplitudes above a threshold determined individually for each excerpt. We then used the resulting onset vector to index Surprisal values (see Figure 2B), which resulted in a discrete, onset-based Surprisal signal.

5.3 Decoding Model Output

The modeled-ERP is estimated on a portion of the stimulus and neural data, and tested on an unseen part of the data through a leave-one-out cross-validation method. To evaluate the performance of the model in predicting neural data from stimulus features, a Pearson's correlation is then computed between the actual neural data (averaged across participants, which is typical for this type of experiment [51, 55]) and the data predicted by the convolution $s * w_n$. Thus, r-values are obtained for each channel and each musical piece and are then averaged across pieces to obtain one average r-value per channel. The distribution of these r-values is indicative of how well the particular feature used to obtain the modeled-ERP is encoded in the brain. For a visual representation of the flow of data processing, refer to Figure 3.

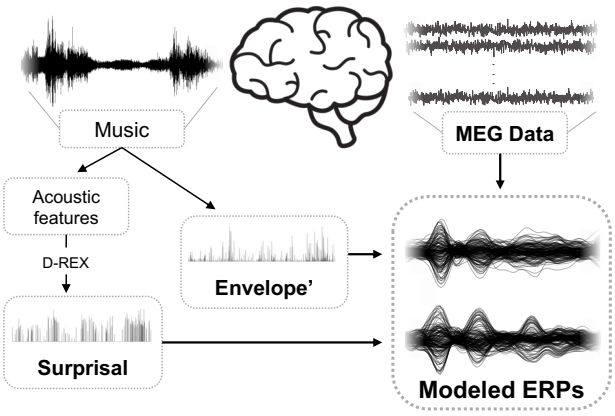


Figure 3. Diagram depicting the flow of the music to neural mapping described above.

6. RESULTS

The TRF decoding method capture the neural response to a feature (e.g. Envelope' or Surprisal) and informs us as to how well it is represented in the neural data. Here, this is evaluated by estimating a modeled-ERP that describes the mapping, at each channel, between a set of input features and the neural data. A correlation between predicted data obtained from the modeled-ERP and actual data is then computed as a marker of how well a feature is represented in the brain activity. Here we conducted two analyses to characterize the encoding of surprisal as modeled by D-REX in the neural data: we first tested the hypothesis that the encoding of surprisal was significantly different from its baseline, by computing a null model of Surprisal encoding and comparing the obtained r-value distribution between the null and the real model. Second, we sought to gain insight from the specific shape of the modeled-ERP by comparing it to the one obtained using the Envelope'.

6.1 Cortical Encoding of Surprisal

To properly baseline the encoding of features, the distribution of Pearson's correlation *r*-values obtained by decoding surprisal was tested against a distribution of r-values obtained by shuffling each song's surprisal and conducting the same decoding analysis on this shuffled surprisal. A significant enhancement of r-values obtained from the real model, as compared to the null one, would confirm that surprisal as predicted by D-REX is significantly encoded in the neural data. In practice, we examined the statistical difference between the Pearson's *r*-values for each MEG channel resulting from a real Surprisal model and its baseline. The real model was constructed using the Surprisal signal (plotted for one song in Figure 2B) as input to the TRF. The baseline model was obtained by shuffling the Surprisal values attributed to each onset over 100 permutations (one permutation is illustrated in Figure 2B).

For each electrode, we obtained an r-value by conducting a TRF analysis on the neural data averaged across participants, using either the real Suprisal input or a shuffled version of it (averaged over 100 permutations). We thus obtained one r-value for each channel and each musical piece, and the values obtained were consistent with previous reported musical stimuli encoding [46, 51]. We averaged these values across musical pieces and obtained two distributions of r-values (e.g. real and null model) for each channel, plotted in Figure 4D. They reveal that r-values obtained from the real model were higher, which was confirmed by a paired t-test where r-values for each MEG channel ($t(156) = 16.25; p < .001, d_s = 1.29$).

6.2 Higher-Level Processing

The second analysis consisted of examining the particular shape of modeled-ERPs obtained from the Envelope' and the Surprisal model, to gain insight into the differences in neural responses elicited by the two features. After averaging participants' neural data, the Envelope' and Surprisal modeled-ERPs were estimated for each individual MEG channel and each musical excerpt, and then averaged across musical excerpts. One modeled-ERP per MEG channel was thus obtained (see Figure 4A and 4B). As expected, both modeled-ERPs exhibit typical auditory responses at [50-100] ms and [150-200] ms [43]. However, the modeled-ERPs obtained from the Surprisal input exhibit an amplitude peak at around 350 ms. To statistically evaluate differences between the responses to the Envelope' and the Surprisal inputs, we first computed the absolute value of the difference Surprisal-Envelope' (z-scored). This difference signal is plotted in Figure 4C. The significant peaks were assessed over the entire signal ([-100-600] ms) using a series of t-tests that were conducted on each individual time sample, using the distribution of values over channels at each time point. The shaded grey areas on the *x*-axis indicate significance corrected for multiple comparisons (FDR correction, $p < .05$). This analysis revealed significant differences for the two main auditory components (at [50-100] ms and [150-200] ms). This may reflect the fact that the Surprisal signal was obtained by modulating values at note onsets, while the Envelope' model used a more continuous signal, yielding a more attenuated early modeled response. The third significant peak that occurs at around 350 ms indicates an enhanced neural response in the modeled-ERP obtained from the Surprisal. This response resembles a P3 response, a well-characterized, consistent, and reliable neurophysiological marker of syntactic processing in both the music [56, 57] and language [58] domains.

7. CONCLUSIONS

The central goal of the current project was to determine whether musical surprisal, a high-level, cognitive measure, enhances decoding performance of neural signal above and beyond what may be explained by acoustic features alone, thus showing that the human brain is tracking statistical regularities as modeled by D-REX. By validating D-REX, a computational model which simulates musical surprisal not from symbolic stimuli (e.g., MIDI) but directly from any audio file, we provide a valuable tool that can integrate perceptual and cognitive musical components relevant to

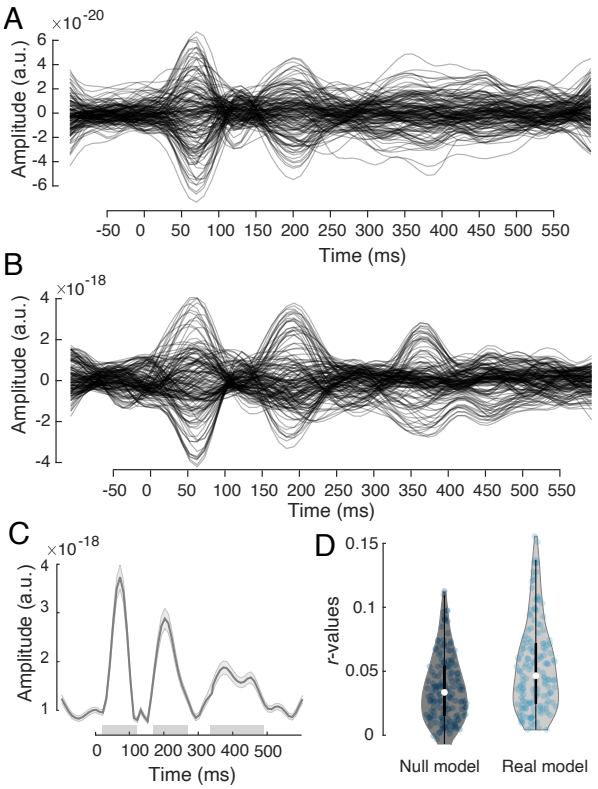


Figure 4. Results of decoding analysis. **(A)** The modeled-ERPs for each channel from the decoding model using Envelope' (the acoustic feature) as input. The weights for each channel are averaged across songs. **(B)** The modeled-ERPs for each channel from the decoding model using Surprisal (the cognitive feature) as input. **(C)** The absolute values of the Surprisal - Envelope' (z-scored) signals averaged across channels over time. Shaded areas around the curve indicate SEM over channels. Grey bars on the *x*-axis represent time windows during which a significant difference between the two averaged modeled-ERPs was found ($p < .005$, FDR-corrected). **(D)** Violin plots of *r*-value distributions for the real and baseline (i.e. shuffled) Surprisal models. Each dot corresponds to the *r*-value for one MEG channel.

recommender systems. Additionally, we provide an open-source database of neural data (twenty participants listening to 30 one minute long musical excerpts along with the audio files for these excerpts) with an excellent temporal resolution (at a sampling frequency of 1000 Hz) that can be used to further investigate the neural correlates of music processing. [2] In this paper, we used a decoding method to validate a Bayesian algorithm, D-REX, which models the continuous surprisal experienced by listeners during music listening. By describing a music-to-neural mapping between music stimuli and MEG data, we demonstrate that D-REX is an effective model to explain brain signals above and beyond mere acoustical features, capturing neural responses indicative of higher-level processing.

Our analysis also highlights a specific time window of auditory processing in the musical domain which is crucial to higher-level mechanisms involved in music listening. In the modeled-ERP derived from the Surprisal input (see Figure 4B), there is a significant peak around 350 ms, which undoubtedly evokes the P3 ERP [59, 60]. This ERP is implicated in syntax processing and context updating operations in both language [58] and music processing [56,57]. This further suggests that D-REX is capturing higher-level expectations generated by gathering statistics over time from acoustic information.

In the context of music listening, a recorded neural signal may be conceptualized as "a mid-level representation of the original music piece that has been heavily distorted by two consecutive black-box filters—the brain and the [MEG] equipment" [61]. However, while the brain does extract and represent the acoustic features contained in the musical excerpt, it also combines them to generate purely cognitive components, such as surprisal. In this experiment, we extract musical features and interface them with a cognitive model which abstracts beyond acoustic features into the psychological domain. Importantly, this cognitive model represents a measure which is one of the many factors predicting individual music preference and pleasure [4, 6]. Thus, we show that combining music information retrieval with cognitive neuroscience is an optimal way to measure people's responses to music.

8. FUTURE DIRECTIONS

8.1 Further Validation of D-REX Model

The current analysis was executed using *joint surprisal*, which is a summary measure across all 15 extracted acoustic features. Though our results show that D-REX joint surprisal is effectively encoded in the neural data recorded during music listening, it is possible that surprisal across specific features, such as pitch value or harmonicity, may be more relevant to cognitive aspects of music listening. Also, these representations may differ across participants. Thus, it is worth exploring across which dimensions surprisal is better represented. In addition, the present study used Western musical genres for listeners to validate D-REX, but future work may expand this exploration to non-Western genres.

8.2 Music Recommender Systems

This project presents a new line of inquiry for music recommender systems, which have typically been content-based (using extracted auditory features or subjective semantic ratings) or user-item/metadata-based [62]. Given the aforementioned inverse U-relationship between music surprisal and musical preferences, D-REX may be used to characterize individuals' musical corpora and, compiled with other variables, efficiently predict liking for new music. More broadly, we propose that harnessing cognitive neuroscience methods is potentially fruitful in improving and generating new methods for recommender systems by including cognitively relevant outcomes such as surprisal [4, 6].

[2] https://osf.io/dbm49/

9. REFERENCES

[1] D. B. Huron, *Sweet Anticipation: Music and the psychology of expectation*. MIT Press, 2007.

[2] R. Levy, "Expectation-based syntactic comprehension," *Cognition*, vol. 106, no. 3, pp. 1126–1177, Mar. 2008.

[3] R. M. Willems, S. L. Frank, A. D. Nijhof, P. Hagoort, and A. van den Bosch, "Prediction during natural language comprehension," *Cereb. Cortex*, vol. 26, no. 6, pp. 2506–2516, Jun. 2016.

[4] B. P. Gold, M. T. Pearce, E. Mas-Herrero, A. Dagher, and R. J. Zatorre, "Predictability and uncertainty in the pleasure of music: A reward for learning?" *J. Neurosci.*, vol. 39, no. 47, pp. 9397–9409, Nov. 2019.

[5] V. N. Salimpoor, D. H. Zald, R. J. Zatorre, A. Dagher, and A. R. McIntosh, "Predictions and the brain: how musical sounds become rewarding," *Trends Cogn. Sci.*, vol. 19, no. 2, pp. 86–91, Feb. 2015.

[6] V. K. M. Cheung, P. M. C. Harrison, L. Meyer, M. T. Pearce, J.-D. Haynes, and S. Koelsch, "Uncertainty and surprise jointly predict musical pleasure and amygdala, hippocampus, and auditory cortex activity," *Curr. Biol.*, vol. 29, no. 23, pp. 4084–4092.e4, Dec. 2019.

[7] D. E. Berlyne, "Aesthetics and psychobiology," *Journal of Aesthetics and Art Criticism*, vol. 31, no. 4, 1973.

[8] A. Chmiel and E. Schubert, "Unusualness as a predictor of music preference," *Music Sci.*, vol. 23, no. 4, pp. 426–441, Dec. 2019.

[9] Y. Güçlütürk, R. H. A. H. Jacobs, and R. van Lier, "Liking versus complexity: Decomposing the inverted u-curve," *Front. Hum. Neurosci.*, vol. 10, p. 112, Mar. 2016.

[10] A. Chmiel and E. Schubert, "Back to the inverted-u for music preference: A review of the literature," *Psychol. Music*, vol. 45, no. 6, pp. 886–909, Nov. 2017.

[11] P. G. Hunter and E. G. Schellenberg, "Interactive effects of personality and frequency of exposure on liking for music," *Pers. Individ. Dif.*, vol. 50, no. 2, pp. 175–179, Jan. 2011.

[12] L. B. Meyer, *Emotion and meaning in music*. University of Chicago Press, 1961.

[13] E. Narmour, "The "genetic code" of melody: Cognitive structures generated by the implication-realization model," *Contemporary Music Review*, vol. 4, no. 1, pp. 45–63, 1989.

[14] ——, "The top-down and bottom-up systems of musical implication: Building on meyer's theory of emotional syntax," *Music Percept.*, vol. 9, no. 1, pp. 1–26, Oct. 1991.

[15] E. H. Margulis, "A model of melodic expectation," *Music Percept.*, vol. 22, no. 4, pp. 663–714, Apr. 2005.

[16] M. M. Farbood, "A parametric, temporal model of musical tension," *Music Percept.*, vol. 29, no. 4, pp. 387–428, Apr. 2012.

[17] D. Temperley, *Music and probability*. MIT Press, 2007.

[18] S. Koelsch, P. Vuust, and K. Friston, "Predictive processes and the peculiar case of music," *Trends Cogn. Sci.*, vol. 23, no. 1, pp. 63–77, Jan. 2019.

[19] L. Gebauer, M. L. Kringelbach, and P. Vuust, "Ever-changing cycles of musical pleasure: The role of dopamine and anticipation," *Psychomusicology*, vol. 22, no. 2, pp. 152–167, Dec. 2012.

[20] M. T. Pearce and G. A. Wiggins, "Auditory expectation: the information dynamics of music perception and cognition," *Top. Cogn. Sci.*, vol. 4, no. 4, pp. 625–652, Oct. 2012.

[21] N. Politimou, P. Douglass-Kirk, M. Pearce, L. Stewart, and F. Franco, "Melodic expectations in 5-and 6-year-old children," *Journal of Experimental Child Psychology*, vol. 203, p. 105020, 2021.

[22] D. Omigie, M. T. Pearce, and L. Stewart, "Tracking of pitch probabilities in congenital amusia," *Neuropsychologia*, vol. 50, no. 7, pp. 1483–1493, 2012.

[23] K. A. Corrigall, B. Tillmann, and E. G. Schellenberg, "Measuring children's harmonic knowledge with implicit and explicit tests," *Music Perception: An Interdisciplinary Journal*, vol. 39, no. 4, pp. 361–370, 2022.

[24] R. Bianco, B. Gold, A. Johnson, and V. Penhune, "Music predictability and liking enhance pupil dilation and promote motor learning in non-musicians," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[25] R. Bianco, L. E. Ptaszynski, and D. Omigie, "Pupil responses to pitch deviants reflect predictability of melodic sequences," *Brain and Cognition*, vol. 138, p. 103621, 2020.

[26] G. M. Di Liberto, C. Pelofi, R. Bianco, P. Patel, A. D. Mehta, J. L. Herrero, A. de Cheveigné, S. Shamma, and N. Mesgarani, "Cortical encoding of melodic expectations in human temporal cortex," *Elife*, vol. 9, Mar. 2020.

[27] G. Marion, G. M. Di Liberto, and S. A. Shamma, "The music of silence: Part i: Responses to musical imagery encode melodic expectations and acoustics," *Journal of Neuroscience*, vol. 41, no. 35, pp. 7435–7448, 2021.

[28] D. Omigie, M. Pearce, K. Lehongre, D. Hasboun, V. Navarro, C. Adam, and S. Samson, "Intracranial recordings and computational modeling of music reveal the time course of prediction error signaling in frontal and temporal cortices," *Journal of Cognitive Neuroscience*, vol. 31, no. 6, pp. 855–873, 2019.

[29] D. R. Quiroga-Martinez, N. C. Hansen, A. Højlund, M. Pearce, E. Brattico, and P. Vuust, "Decomposing neural responses to melodic surprise in musicians and non-musicians: evidence for a hierarchy of predictions in the auditory system," *NeuroImage*, vol. 215, p. 116816, 2020.

[30] D. R. Quiroga-Martinez, N. C. Hansen, A. Højlund, M. Pearce, E. Brattico, and P. Vuust, "Musical prediction error responses similarly reduced by predictive uncertainty in musicians and non-musicians," *European Journal of Neuroscience*, vol. 51, no. 11, pp. 2250–2269, 2020.

[31] D. Omigie, M. T. Pearce, V. J. Williamson, and L. Stewart, "Electrophysiological correlates of melodic processing in congenital amusia," *Neuropsychologia*, vol. 51, no. 9, pp. 1749–1762, 2013.

[32] B. Skerritt-Davis and M. Elhilali, "Detecting change in stochastic sound sequences," *PLoS Comput. Biol.*, vol. 14, no. 5, p. e1006162, May 2018.

[33] ——, "A model for statistical regularity extraction from dynamic sounds," *Acta Acust. United Acust.*, vol. 105, no. 1, pp. 1–4, Jan. 2019.

[34] E. B. Abrams, P. Ripolles, and D. Poeppel, "The rewards of muzak: elevator music as a tool for the quantitative characterization of emotion and preference," Oct 2021. [Online]. Available: psyarxiv.com/xqs8b

[35] R. Bianco, B. P. Gold, A. P. Johnson, and V. B. Penhune, "Music predictability and liking enhance pupil dilation and promote motor learning in non-musicians," *Nature News*, Nov 2019. [Online]. Available: https://www.nature.com/articles/s41598-019-53510-w

[36] P. Hansen, M. Kringelbach, and R. Salmelin, *MEG: An Introduction to Methods*. Oxford University Press, Jun. 2010. [Online]. Available: https://doi.org/10.1093/acprof:oso/9780195307238.001.0001

[37] Y. Adachi, M. Shimogawara, M. Higuchi, Y. Haruta, and M. Ochiai, "Reduction of non-periodic environmental magnetic noise in meg measurement by continuously adjusted least squares method," *IEEE Transactions on Applied Superconductivity*, vol. 11, no. 1, pp. 669–672, 2001.

[38] A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, and M. S. Hämäläinen, "MNE software for processing MEG and EEG data," *Neuroimage*, vol. 86, pp. 446–460, Feb. 2014.

[39] N. Martínez-Molina, E. Mas-Herrero, A. Rodríguez-Fornells, R. J. Zatorre, and J. Marco-Pallarés, "Neural correlates of specific musical anhedonia," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 113, no. 46, pp. E7337–E7345, Nov. 2016.

[40] N. Huang and M. Elhilali, "Auditory salience using natural soundscapes," *J. Acoust. Soc. Am.*, vol. 141, no. 3, pp. 2163–2176, Mar. 2017.

[41] E. Hemery and J.-J. Aucouturier, "One hundred ways to process time, frequency, rate and scale in the central auditory system: a pattern-recognition meta-analysis," *Frontiers in Computational Neuroscience*, vol. 9, 2015. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fncom.2015.00080

[42] M. J. Crosse, G. M. Di Liberto, A. Bednar, and E. C. Lalor, "The multivariate temporal response function (mtrf) toolbox: a matlab toolbox for relating neural signals to continuous stimuli," *Frontiers in human neuroscience*, vol. 10, p. 604, 2016.

[43] E. C. Lalor, A. J. Power, R. B. Reilly, and J. J. Foxe, "Resolving precise temporal processing properties of the auditory system using continuous stimuli," *Journal of neurophysiology*, vol. 102, no. 1, pp. 349–359, 2009.

[44] S. Akram, A. Presacco, J. Z. Simon, S. A. Shamma, and B. Babadi, "Robust decoding of selective auditory attention from meg in a competing-speaker environment via state-space modeling," *NeuroImage*, vol. 124, pp. 906–917, 2016.

[45] D. D. Wong, S. Fuglsang, J. Hjortkjær, E. Ceolini, M. Slaney, and A. de Cheveigné, "A comparison of temporal response function estimation methods for auditory attention decoding," *Biorxiv*, pp. 1–22, 2018.

[46] G. M. Di Liberto, C. Pelofi, S. Shamma, and A. de Cheveigné, "Musical expertise enhances the cortical tracking of the acoustic envelope during naturalistic music listening," *Acoustical Science and Technology*, vol. 41, no. 1, pp. 361–364, 2020.

[47] G. M. Di Liberto, J. A. O'Sullivan, and E. C. Lalor, "Low-frequency cortical entrainment to speech reflects phoneme-level processing," *Current Biology*, vol. 25, no. 19, pp. 2457–2465, 2015.

[48] G. M. Di Liberto, V. Peter, M. Kalashnikova, U. Goswami, D. Burnham, and E. C. Lalor, "Atypical cortical entrainment to speech in the right hemisphere underpins phonemic deficits in dyslexia," *NeuroImage*, vol. 175, pp. 70–79, 2018.

[49] M. P. Broderick, A. J. Anderson, G. M. Di Liberto, M. J. Crosse, and E. C. Lalor, "Electrophysiological correlates of semantic dissimilarity reflect the comprehension of natural, narrative speech," *Current Biology*, vol. 28, no. 5, pp. 803–809, 2018.

[50] N. Ding and J. Z. Simon, "Neural coding of continuous speech in auditory cortex during monaural and dichotic listening," *Journal of neurophysiology*, vol. 107, no. 1, pp. 78–89, 2012.

[51] G. M. Di Liberto, C. Pelofi, R. Bianco, P. Patel, A. D. Mehta, J. L. Herrero, A. de Cheveigné, S. Shamma, and N. Mesgarani, "Cortical encoding of melodic expectations in human temporal cortex," *eLife*, vol. 9, 2020.

[52] N. Ding, A. D. Patel, L. Chen, H. Butler, C. Luo, and D. Poeppel, "Temporal modulations in speech and music," *Neuroscience & Biobehavioral Reviews*, vol. 81, pp. 181–187, 2017.

[53] K. B. Doelling and D. Poeppel, "Cortical entrainment to music and its modulation by expertise," *Proceedings of the National Academy of Sciences*, vol. 112, no. 45, pp. E6233–E6242, 2015.

[54] C. Daube, R. A. Ince, and J. Gross, "Simple acoustic features can explain phoneme-based predictions of cortical responses to speech," *Current Biology*, vol. 29, no. 12, pp. 1924–1937, 2019.

[55] D. H. Baker, G. Vilidaite, F. A. Lygo, A. K. Smith, T. R. Flack, A. D. Gouws, and T. J. Andrews, "Power contours: Optimising sample size and precision in experimental psychology and human neuroscience." *Psychological methods*, vol. 26, no. 3, p. 295, 2021.

[56] P. Regnault, E. Bigand, and M. Besson, "Different brain mechanisms mediate sensitivity to sensory consonance and harmonic context: evidence from auditory event-related brain potentials," *J. Cogn. Neurosci.*, vol. 13, no. 2, pp. 241–255, Feb. 2001.

[57] B. Poulin-Charronnat, E. Bigand, and S. Koelsch, "Processing of musical syntax tonic versus subdominant: an event-related potential study," *J. Cogn. Neurosci.*, vol. 18, no. 9, pp. 1545–1554, Sep. 2006.

[58] D. Friedman, R. Simson, W. Ritter, and I. Rapin, "The late positive component (p300) and information processing in sentences," *Electroencephalogr. Clin. Neurophysiol.*, vol. 38, no. 3, pp. 255–262, Mar. 1975.

[59] E. Donchin and M. G. H. Coles, "Is the p300 component a manifestation of context updating?" *Behavioral and Brain Sciences*, vol. 11, no. 3, p. 357–374, 1988.

[60] J. Polich, "Updating p300: An integrative theory of p3a and p3b," *Clinical Neurophysiology*, vol. 118, no. 10, pp. 2128–2148, 2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1388245707001897

[61] S. Stober, T. Prätzlich, and M. Müller, "Brain beats: Brain beats: Tempo extraction from eeg data," 01 2016, pp. 276–282.

[62] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, and M. Elahi, "Current challenges and visions in music recommender systems research," *Int. J. Multimed. Inf. Retr.*, vol. 7, no. 2, pp. 95–116, Jun. 2018.

BEAT TRANSFORMER: DEMIXED BEAT AND DOWNBEAT TRACKING WITH DILATED SELF-ATTENTION

Jingwei Zhao[2,4] **Gus Xia**[3,5] **Ye Wang**[1,2,4]
[1] School of Computing, NUS [2] Institute of Data Science, NUS [3] Music X Lab, NYU Shanghai
[4] Integrative Sciences and Engineering Programme, NUS Graduate School [5] MBZUAI
jzhao@u.nus.edu, gxia@nyu.edu, wangye@comp.nus.edu.sg

ABSTRACT

We propose Beat Transformer, a novel Transformer encoder architecture for joint beat and downbeat tracking. Different from previous models that track beats solely based on the spectrogram of an audio mixture, our model deals with *demixed* spectrograms with multiple instrument channels. This is inspired by the fact that humans perceive metrical structures from richer musical contexts, such as chord progression and instrumentation. To this end, we develop a Transformer model with both time-wise attention and instrument-wise attention to capture deep-buried metrical cues. Moreover, our model adopts a novel *dilated self-attention* mechanism, which achieves powerful hierarchical modelling with only linear complexity. Experiments demonstrate a significant improvement in demixed beat tracking over the non-demixed version. Also, Beat Transformer achieves up to 4% point improvement in downbeat tracking accuracy over the TCN architectures. We further discover an interpretable attention pattern that mirrors our understanding of hierarchical metrical structures.

1. INTRODUCTION

Music audio beat and downbeat tracking, which aims to infer the very basic metrical structure of music, is a long-standing central topic in music information retrieval (MIR). A good beat estimation benefits various downstream MIR tasks, including transcription and structure analysis [1–5]. Moreover, beat tracking can be applied to human-computer interaction [6, 7], music therapy [8], and more scenes, as beats echo with human perceptual and motor sensitivity to musical rhythms.

We see significant progress in beat tracking with the development of deep neural networks. Current mainstream methods utilize temporal convolutional networks (TCNs) to extract frame-wise beat activations from an input spectrogram [9]. We further see successful efforts in boosting beat tracking performance, including phase-informed post-processing with dynamic Bayesian networks (DBNs) [10], multi-task learning for joint beat, downbeat and tempo estimation [11–13], and explicit beat phase modelling [14].

Recently, Transformer has demonstrated highly competitive performances over a range of MIR tasks [15–21]. In this paper, we propose Beat Transformer, a novel Transformer encoder architecture for joint beat and downbeat tracking. To better accommodate Transformer to our purpose, we introduce two extra inductive biases. Firstly, our model is constructed with *short-windowed dilated self-attention*. An exponentially increasing dilation rate enables our model to discern beats from non-beats in a hierarchical manner. With a fixed window size, our model maintains a linear complexity to the input sequence length.

Another inductive bias is *demixed beat tracking*. This strategy is inspired by the fact that human beat tracking is always accompanied by and enhanced by a deep understanding of the musical contexts. For example, the coordination of instruments enforces the progression of chords and bass notes, thus implying metrical accents, and such cues can be easily identified by human listeners. To capture this relation, we use Spleeter [22] to demix an input music piece into multiple instrument channels, and our model performs both *time-wise* and *instrument-wise* attention in alternate Transformer layers to excavate metrical cues.

We evaluate Beat Transformer on a wide range of beat- and downbeat-annotated datasets. Besides competing with state-of-the-art works, we present a thorough ablation study to illustrate the effectiveness of dilated self-attention and demixing. Moreover, our model learns highly interpretable representations. We demonstrate that our model can be interpreted as a learner over finite-state Markov chains, and we observe beat phase transition through visualization of the transition (attention) matrix.

In brief, the contributions of our paper are as follows:

- We propose Beat Transformer [1], a novel Transformer encoder architecture for joint beat and downbeat tracking in music audio.
- We devise dilated self-attention, which demonstrates powerful sequential modelling with linear complexity, potentially adaptable to more general MIR tasks.
- We make use of music demixing to complement and enhance beat tracking, shedding light on future MIR research towards universal music understanding.

[1] Available at https://github.com/zhaojw1998/Beat-Transformer.

© J. Zhao, G. Xia, and Y. Wang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J. Zhao, G. Xia, and Y. Wang, "Beat Transformer: Demixed Beat and Downbeat Tracking with Dilated Self-Attention", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

2. RELATED WORKS

We review two topics related to our work: beat tracking, and Transformer. For beat tracking, we focus on its development with deep neural networks. For Transformer, we address its application in MIR. For more general review of both topics, we refer readers to [23] and [24], respectively.

2.1 Music Audio Beat Tracking

Beat tracking has been formulated as a two-stage sequential learning task. The first stage aims to determine the likelihood of beat presence, or beat activation, at each frame of an input spectrogram. The initial deep learning approach for this purpose was based on long short-term memory networks (LSTM) [25, 26]. To better pick up the beat sequence from raw model output, at the second stage, a dynamic Baysian network (DBN) is introduced to infer tempo and beat phase transition from beat activation [10].

The current mainstream methods substitute the LSTM with a TCN architecture [9, 12–15]. Specifically, the convolutional kernels have a dilation rate exponential to the depth of layer. This hierarchical structure facilitates the network to model various scales, functionally similar to pooling, but maintains the same input and output size [27]. Besides architecture, another breakthrough of beat tracking is the formulation of multi-task learning [11–15]. Specifically, beat, downbeat, and tempo are strongly correlated metrical features. Sharing model weights among all three sub-tasks helps each to reach better convergence.

A recent trend of beat tracking is to deal with *demixed* music sources. Chiu *et. al.* leverages demixed drum and non-drum streams to enhance model adaptability to different drum source conditions [28, 29]. In fact, humans can track beats while switching their attention among different instrument parts. Hence the coordination of instrumental sources can be explored for useful metrical information.

In our work, we inherit the fashion of multi-task learning and the use of DBN, while proposing a novel Transformer encoder architecture to replace TCN. We formalize *dilated self-attention* [30, 31] for efficient modeling of long metrical structures. Moreover, we seek to enhance beat tracking by introducing *instrumental attention* among drum, piano, bass, vocal, and other demixed sources.

2.2 Transformer in MIR

Transformer has established itself *de facto* state-of-the-art in natural language and symbolic music domain. Recently, it has also demonstrated outstanding performance over audio-based MIR tasks, including music transcription [17–19], music tagging [20, 21], and other analysis [16]. For these tasks, a Transformer model is trained over short spectrogram clips typically of 2-5 seconds as a compromise to the quadratic complexity computing self-attention.

Transformer is first applied to beat tracking by Hung *et. al.* [15] using SpecTNT blocks [16]. For each SpecTNT block, a spectral Transformer encoder first aggregates spectral information at each time step, and then a temporal encoder exchanges information in time and pays attention to beat and downbeat positions. Such a time-frequency design makes an effective use of spectral features and achieves the state-of-the-art performance.

Our work is also a Transformer-based architecture that strives to enhance beat tracking with richer musical contexts. Instead of aggregating spectral features as in [15], we resort to demixed instrumental attention as a more explicit inductive bias to exploring spectral information. Our design of dilated self-attention is also a crucial step to accommodate Transformer to beat tracking. With only linear complexity, our model handles full-length songs at a time.

3. METHOD

The core of our method is a Transformer encoder based on *1) dilated self-attention* (DSA), and *2) demixed instrumental attention*, to extract a framewise beat activation from input spectrograms. In this section, we first formalize DSA in Section 3.1. We then present our design of demixed beat tracking in Section 3.2. We further interpret our method with Markov chain properties in Section 3.3.

3.1 Dilated Self-Attention (DSA)

3.1.1 Background of Self-Attention (SA)

We first recall that, for vanilla Transformer layers, self-attention (SA) is computed via the scaled dot product:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^\top}{\sqrt{d_\mathrm{f}}})V, \qquad (1)$$

where $Q_{1:T}, K_{1:T}, V_{1:T} \in \mathbb{R}^{T\times d_\mathrm{f}}$ are query, key, and value sequences, each linearly mapped from input $x_{1:T}$. T is the sequence length, and d_f is the feature dimension.

The partial attention from position i to j is explicitly:

$$e_{ij} = \frac{Q_i K_j^\top}{\sqrt{d_\mathrm{f}}}, \qquad (2)$$

where $1 \leq i, j \leq T$. Such computation leads to quadratic complexity $\mathcal{O}(T^2)$ in terms of both time and space.

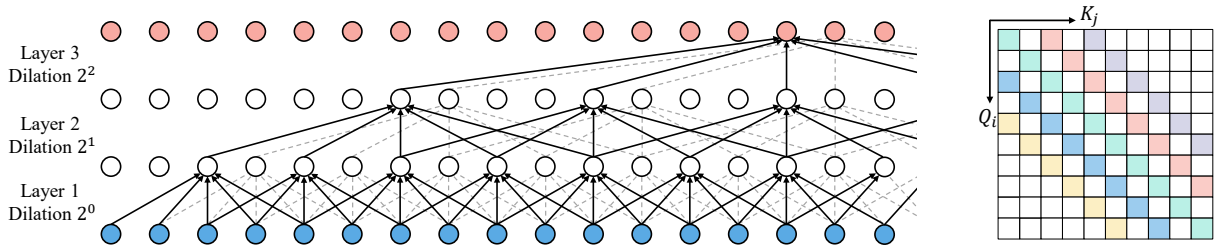
3.1.2 Dilated Self-Attention (DSA)

An illustration of DSA is shown in Figure 1. DSA is computed over a short window of size $l_\mathrm{win} = m + n + 1$, where m and n are the length of non-causal and causal components of the window (in Figure 1, $m = n = 2$). Each Transformer layer has a dilation rate $r \geqslant 1$, and r increases exponentially as the layer goes deeper.

Formally, given Q, K, and $V \in \mathbb{R}^{T\times d_\mathrm{f}}$, DSA first computes Q-K attention by:

$$e_{ik} = \frac{Q_i K_{i+rk}^\top}{\sqrt{d_\mathrm{f}}}, \qquad (3)$$

where $1 \leq i \leq T$ and $-m \leq k \leq n$. Specifically, $i + rk$ refers to the positions in $K_{1:T}$ that are attainable by Q_i under the dilated window of rate r. When $i + rk$ exceeds the sequence range $[1, T]$, we fill e_{ik} with $-\texttt{inf}$.



(a) Layer-wise view of dilated self-attention (Adapted from TCN structures [9, 27])

(b) Attention matrix view at layer 2

Figure 1: Illustration of dilated self-attention (with a non-causal short window of size 5) over a three-layer Transformer. Part (a) shows the hierarchical connectivity across layers, which shares the same pattern as TCN in [9]. Part (b) shows the attention matrix at layer 2, with colours indicating relative position. The white colour indicates unattainable positions.

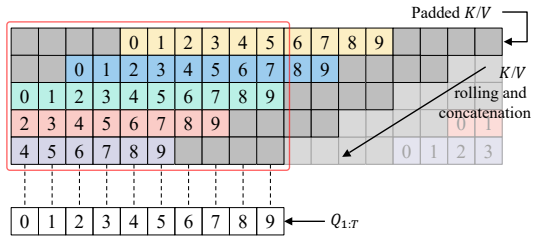


Figure 2: Illustration of efficient DSA implementation, with dilation rate $r = 2$ and window size $l_{\text{win}} = 5$.

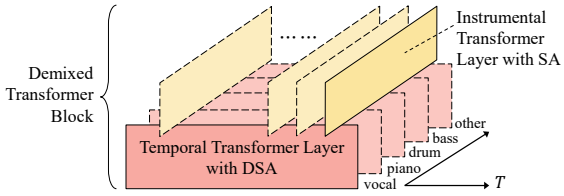


Figure 3: Demixed Transformer block. Two Transformer layers are stacked "orthogonally", each handling time-wise dilated self-attention and instrument-wise self-attention.

Then, the Q-K attention e_{ik} is normalized via softmax:

$$p_{ik} = \frac{\exp(e_{ik})}{\sum_{k=-m}^{n} \exp(e_{ik})} \tag{4}$$

The output of DSA is a sequence $z_{1:T}$, where z_i is the weighted average of V under the same dilated window:

$$z_i = \sum_{k=-m}^{n} p_{ik}(V_{i+rk}) \tag{5}$$

We apply DSA with exponentially increasing dilation rates and relative positional embedding (RPE) [32] to a stack of Transformer layers. Each layer consists of two sub-layers: DSA, and a position-wise feed-forward layer. We place residual connections across each sub-layer and perform layer normalization [33] before each sub-layer.

3.1.3 Memory Efficient Implementation of DSA

A straightforward implementation of DSA is to mask SA. Concretely, a square mask takes the same form as in Figure 1b, where all uncolored positions are filled with $-\texttt{inf}$, rendering a rather sparse attention matrix. This way, however, still requires quadratic complexity, because e_{ij} is explicitly computed for all masked positions.

Our implementation takes a "rolling" strategy to eliminate redundant computation. As in Figure 2, l_{win} copies of $K_{1:T}$ sequences are padded and rolled along the time axis, and then concatenated on a new axis. In this way, each Q_i sees K_j directly and only at $j = i + rk$ for $-m \le k \le n$, which is exactly the coverage of the dilated window.

Formally, given $K_{1:T} \in \mathbb{R}^{T \times d_{\text{f}}}$, dilation rate r, and window size l_{win}, the rolling strategy takes three steps:

1. Pad $K_{1:T}$ with $\lfloor \frac{l_{\text{win}}}{2} \rfloor \times r$ steps on both sides;
2. Make l_{win} copies of padded K. Starting from 0, each copy is cyclically rolled r more steps;
3. Concatenate each copy along a new axis and retrieve the first T steps. The output has shape $T \times l_{\text{win}} \times d_{\text{f}}$.

The same procedure applies to $V_{1:T}$ as well. In this way, computing DSA is essentially as simple as Equation (1). Here, instead of $Q, K, V \in \mathbb{R}^{T \times d_{\text{f}}}$, we have $Q \in \mathbb{R}^{T \times 1 \times d_{\text{f}}}$ and $K, V \in \mathbb{R}^{T \times l_{\text{win}} \times d_{\text{f}}}$, with T treated as a batch dimension. The computation complexity is $\mathcal{O}(T \times l_{\text{win}})$, while l_{win} is fixed and small enough to be left uncounted.

3.2 Demixed Beat Tracking

3.2.1 Demixed Transformer Block

We use Spleeter 5-stems model [22] to demix an input piece into spectrograms with $|C|$ instrument channels, where $C = \{\texttt{vocal}, \texttt{piano}, \texttt{drum}, \texttt{bass}, \texttt{other}\}$. As shown in Figure 3, we stack two Transformer layers to perform time-wise and instrument-wise attention in turn. Let the input at layer l be $x^l_{1:T,1:|C|} \in \mathbb{R}^{T \times |C| \times d_{\text{f}}}$, a *temporal* Transformer layer (TTL) first takes $x^l_{1:T,c}$ for $1 \le c \le |C|$:

$$x^{l+1}_{1:T,c} = \text{TTL}(x^l_{1:T,c}) \tag{6}$$

Then, an *instrumental* Transformer layer (ITL), on the *orthogonal* direction, takes $x^{l+1}_{t,1:|C|}$ for $1 \le t \le T$:

$$x^{l+2}_{t,1:|C|} = \text{ITL}(x^{l+1}_{t,1:|C|}) \tag{7}$$

A TTL followed by ITL forms a *demixed Transformer block*. TTL consists of DSA as described in Section 3.1.2.

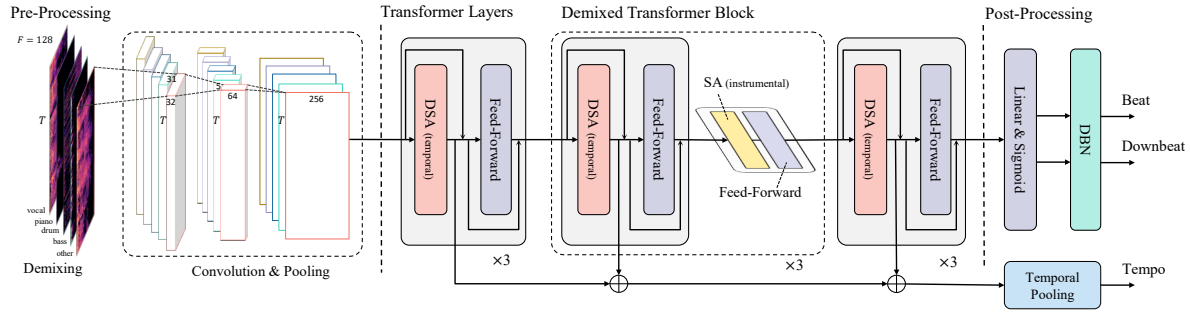


Figure 4: Beat Transformer architecture. For conciseness, layer normalization and dropout layers are not shown.

For ITL, we use vanilla SA because there is only 5 instrument channels. As instruments are not sequentially ordered, we do not add any positional encoding to ITL.

Through demixed Transformer blocks, our model can capture the rhythmic evolution of each instrument, as well as the harmonic coordination among all instruments.

3.2.2 Partial Demix Augmentation

Spleeter may produce empty channels when a certain instrument does not present. To avoid potential effects of such a situations, we develop a partial demix strategy for data augmentation. Partial demix creates new stems by summing up existing instrument channels of a default 5-stem demixed input sample. For example, an augmented data sample may have three channels corresponding to $C' = \{\texttt{vocal\&piano}, \texttt{drum}, \texttt{bass\&other}\}$.

In our case, we randomly sum up 2, 3, or 4 instrument channels of a 5-stem input with a probability 30%, 10%, and 10% during training. In this way, our model is encouraged to pay attention to *instrument-agnostic* musical contents and thus is less affected by empty channels where no valid music content is present. As our augmentation strategy also adds to the demix diversity and the data quantity, we believe it brings general benefits to training as well.

3.3 Markov Chain Interpretation

In Equation (4), we formulate the attention matrix of DSA as $P = [p_{ij}]_{1 \le i,j \le T}$, where $p_{ij} \ge 0$ if and only if $j = i + rk$ for $-m \le k \le n$. Here, r is the dilation rate, and m, n are components of the attention window. Moreover, P satisfies $\sum_{j=1}^{T} p_{ij} = 1$ for all i. Therefore, P can be regarded as the transition matrix of a finite-state Markov chain, where each state is a position of the input sequence.

For a stack of temporal Transformer layers (TTL), where DSA is employed, layer l essentially learns a unique one-step transition P^l, by which our model can attend to local neighbours covered by the attention window. Through L layers, our model makes an L-step transition, during which it attends to global positions hierarchically. The overall L-step transition matrix $P^{(L)}$ satisfies:

$$P^{(L)} = \prod_{l=1}^{L} P^l \tag{8}$$

Note that P^l itself is a rather sparse matrix (due to *short* attention window), while $P^{(L)}$ is densely connected. Its components $[p_{ij}^{(L)}]_{1 \le i,j \le T}$ represent the hierarchical attention weights across the whole L layers, which can tell us much richer attention patterns (more in Section 4.4).

3.4 Complete Architecture

A complete view of Beat Transformer is presented in Figure 4. The inputs are log-scaled spectrograms demixed by Spleeter, with $F = 128$ mel-bins and $|C| = 5$ instrument channels. Subject to Spleeter, our frame rate is 43.07 fps and the frequency range is up to 11 kHz. We then use three 2D convolutional layers, shared by each demixed channel, as a front-end feature extractor. The convolutional design is the same as in [13] except that we employ more filters to reach feature dimension $d_{\text{model}} = 256$.

Beat Transformer comprises 9 temporal Transformer layers (TTL) with DSA. Each TTL has 8 attention heads with window size $l_{\text{win}} = 5$, four of which have skewed window ranges, where $m = 0, 1, 3, 4$, respectively, and $n = 4 - m$. The dilation rate grows exponentially from 2^0 to 2^8, stretching to a receptive field of 47.51 seconds. Among the 9 TTLs, the middle three are expanded to demixed Transformer blocks by interleaving ITLs. We found it sufficient to perform instrumental attention only in the middle layers, which has a proper scale of 1-5 seconds. Each Transformer layer has 8 heads ($d_{\text{f}} = 32$) followed by a feed-forward layer with hidden dimension $d_{\text{ff}} = 1024$.

We sum up the instrument channels of the output of the last Transformer layer and obtain a frame-wise beat representation of shape $T \times d_{\text{model}}$. Following multi-task learning practice [11–14], we use a linear layer to map the representation to beat and downbeat activations respectively, and add a regularization branch predicting global tempo via "skip connections" [12]. We apply DBN in Madmom package [34] as the post-processor to pick up the beat and downbeat sequence from raw activations. For DBN parameters, we set $\texttt{observation\_lambda} = 6$, $\texttt{transition\_lambda} = 100$, and $\texttt{threshold} = 0.2$.

4. EXPERIMENTS

4.1 Datasets

We utilize a total of 7 datasets for model training and evaluation: *Ballroom* [35, 36], *Hainsworth* [37], *RWC Popular* [38], *Harmonix* [39], *Carnetic* [40], *SMC* [41],

Dataset	Model	Beat Accuracy			Downbeat Accuracy		
		F-Measure	CMLt	AMLt	F-Measure	CMLt	AMLt
Ballroom	TCN+Demix	0.960	0.942	0.960	0.925	0.924	0.956
	Ours w/o Demix	0.968	0.946	0.965	0.930	0.925	0.963
	Ours w/o Aug.	0.967	0.949	**0.967**	0.928	0.931	0.958
	Ours	**0.968**	**0.954**	0.966	**0.941**	**0.944**	**0.969**
	Böck *et al.* [13]	0.962	0.947	0.961	0.916	0.913	0.960
	Hung *et al.* [15]	0.962	0.939	0.967	0.937	0.927	0.968
Hainsworth	TCN+Demix	0.887	0.827	0.918	0.739	0.708	0.861
	Ours w/o Demix	0.902	0.844	0.934	0.721	0.688	0.843
	Ours w/o Aug.	0.892	0.831	0.908	0.742	0.703	0.837
	Ours	0.902	0.842	0.918	**0.748**	0.712	0.841
	Böck *et al.* [13]	**0.904**	0.851	**0.937**	0.722	0.696	**0.872**
	Hung *et al.* [15]	0.877	**0.862**	0.915	0.748	**0.738**	0.870
Harmonix	TCN+Demix	0.954	0.903	0.956	0.901	0.866	0.923
	Ours w/o Demix	0.954	0.902	0.958	0.887	0.846	0.916
	Ours w/o Aug.	0.952	0.901	0.950	0.897	0.863	0.919
	Ours	**0.954**	0.905	0.957	0.898	0.863	0.919
	Böck *et al.* [13]*	0.933	0.841	0.938	0.804	0.747	0.873
	Hung *et al.* [15]	0.953	**0.939**	**0.959**	**0.908**	**0.872**	**0.928**
SMC	TCN+Demix	0.596	0.455	0.625			
	Ours w/o Demix	0.589	0.448	0.621			
	Ours w/o Aug.	0.595	0.450	0.626			
	Ours	0.596	0.456	0.635			
	Böck *et al.* [13]	0.552	0.465	0.643			
	Hung *et al.* [15]	**0.605**	**0.514**	**0.663**			
GTZAN	TCN+Demix	0.873	0.780	0.907	0.700	0.646	0.842
	Ours w/o Demix	0.876	0.787	0.914	0.686	0.633	0.834
	Ours w/o Aug.	0.881	0.797	0.921	0.703	0.653	0.845
	Ours	0.885	0.800	0.922	0.714	0.665	0.844
	Böck *et al.* [13]	0.885	**0.813**	**0.931**	0.672	0.640	0.832
	Hung *et al.* [15]	**0.887**	0.812	0.920	**0.756**	**0.715**	**0.881**

Table 1: Testing results of beat and downbeat tracking under 8-fold cross-validation. GTZAN is unseen from training and held out for test only. Böck *et. al.* [13] on Harmonix is reproduced by [15], as indicated by the * symbol. We use underscore to denote best results comparing with our ablation models and use **boldface** to compare with state-of-the-art models.

and *GTZAN* [42, 43]. We acquire *Harmonix* in mel-spectrogram and invert each piece to audio using Griffin-Lim Algorithm [44, 45] with Librosa package [46]. Following convention, we leave *GTZAN* for testing only and use the other datasets in 8-fold cross validation [11–13].

4.2 Training

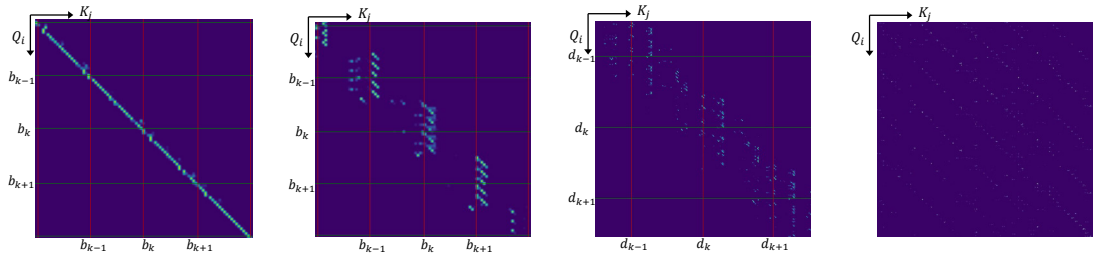
Our model is supervised in a multi-task learning fashion, where beat, downbeat, and tempo are predicted jointly [13]. Beat and downbeat annotations are each represented as a 1D binary sequence that indicates beat (1) and non-beat (0) states at each input frame. Following [13], we widen beat and downbeat states to ± 2 neighbours of annotated frames with weights 0.5 and 0.25. Following [12], we derive tempo target from beat annotation for the tempo prediction branch. We found the use of tempo branch generally beneficial to beat tracking, as it may serve as a regularization term that helps reaching better convergence.

For training, we combine the binary cross entropy loss over beat, downbeat, and tempo by weighing them equally. We use a batch size of 1 to train on whole sequences with different lengths. For excessively long songs, we split them into 3-minute (8k-frame) clips. We apply RAdam [47] plus Lookahead [48] optimizer with an initial learning rate of $1\mathrm{e}{-3}$, which is reduced by a factor of 5 whenever the validation loss gets stuck for 2 epochs before being capped at a minimum value of $1\mathrm{e}{-7}$. We use dropout [49] with rate 0.5 for the tempo branch and 0.1 for other parts of the network. We apply *partial demix augmentation* described in Section 3.2.2 as the only means of data augmentation. Our model has 9.29M trainable parameters and is trained with an RTX-A5000-24GB GPU. Each training fold generally takes 20 epochs (in 11 hours) to fully converge.

4.3 Evaluation

4.3.1 Baseline Methods

We first compare three ablation models to validate our module design. The first ablation model is *Ours* trained without partial demix augmentation (***Ours w/o Aug.***). The second model removes all ITL layers and tracks beat with a single channel of non-demixed mixture (***Ours w/o Demix***). The last model replaces each TTL layer with a TCN layer [13] of the same dilation rate (***TCN+Demix***). We imple-



(a) $P^{(1)}$ lower scale (b) $P^{(3)}$ beat scale (c) $P^{(5)}$ downbeat scale (d) $P^{(9)}$ higher scale

Figure 5: Visualization of temporal attention matrix based on the product rule in Equation (8) for L-step transition on a Markov chain. $L = 1, 3, 5$, and 9 from part (a) to (d), which model different hierarchies of the metrical structure.

ment TCN layers following [23] while setting the input and output shape to be the same as our model. The resulting model yields a comparable amount of 10.25M parameters and is trained without augmentation.

In addition, we compare our model with two recent works that have achieved state-of-the-art performance. Specifically, Böck *et al.* [13] is based on TCN architectures and Hung *et al.* [15] is based on SpecTNT.

4.3.2 Results and Discussion

In Table 1, we first observe *Ours w/o Aug.* yields generally better performance than *TCN+Demix*, especially on the unseen *GTZAN* dataset. As both models share a comparable amount of parameters, this result demonstrates the capability of Transformer (DSA) versus TCN (dilated convolution), which also corroborates with previous findings on Transformer's comparability to convolution on general tasks [50–52]. Considering that Transformer is notoriously data-inefficient to train, it is remarkable that our model is well-trained with limited data without augmentation. We owe this merit to DSA, which not only prevents redundant computation but also makes musical sense in terms of the hierarchical structure of music metrical modelling.

Comparing *Ours w/o Demix* to *Ours w/o Aug.*, while both models are highly competitive in beat tracking, the latter demonstrates more superiority in downbeat tracking. Downbeat tracking is generally more difficult than beat tracking because it is involved with deeper musical knowledge, such as chord and bass progression, behind the apparent spectrogram energy. In our model, the instrumental attention captures the instrumental coordination as hints to the harmonic cues that are orthogonal to the temporal axis, and thus acquires better metrical modelling.

Comparing *Ours w/o Aug.* to *Ours*, we observe a consistent improvement across datasets brought by partial demix augmentation, which indicates the general usefulness of this augmentation strategy to model training.

Compared to state-of-the-art models, our improvement in downbeat accuracy is more significant than that in beat accuracy. On the test-only *GTZAN* dataset, we obtain 4% point gain in *F-measure* over Böck *et al.* [13] in downbeat tracking. Compared to Hung *et al.* [15], which is also based on Transformer, our model can be more flexibly trained (owing to the efficient DSA mechanism) on a 24GB GPU in contrast to four 32GB GPUs reported in [15].

4.4 Attention Matrix Visualization

We visualize the attention matrix that our model learns by interpreting it as a *multi-step Markov transition matrix* as defined in Equation (8). Specifically, the L-step matrix is the product of L one-step matrices through L layers. Here we only consider TTLs with dilated self-attention, as ITLs work on an orthogonal axis. Figure 5 shows the attention matrix $P^{(L)}$ for $L = 1, 3, 5$, and 9 of the `drum` channel, inferred from the piece `hiphop.00090` chosen from *GTZAN*.

Figure 5a shows a one-step transition. Each position Q_i can only attend to its neighbours covered by the attention window. Still, we observe that beat positions (denoted by b_k) are likely to get more attention. In Figure 5b where we step to the beat scale, most attention spots are aligned with beats. Moreover, we observe that the attention at b_k is typically prolonged after Q_i leaves b_k, and is formed before Q_i reaches b_k for every k. This means that our model learns to transition its attention from the *offbeat* phase following the last beat to the *upbeat phase* preceding the next beat. Figure 5c further stretches the view to the downbeat scale, and we can see similar patterns aligned with downbeat positions (denoted by d_k). Finally, in Figure 5d, the attention reaches further positions and displays a structural pattern.

The above visualization demonstrates the inner logic that our model exploits for beat and downbeat tracking. We see that our model gathers information from both local and global scales with an organized hierarchy.

5. CONCLUSION

In conclusion, we contribute a novel Transformer architecture for audio beat and downbeat tracking. The main novelty lies first in our design of dilated self-attention, which brings down the computation complexity of Transformer from quadratic to linear level. In addition, we successfully enhance beat and downbeat tracking by utilizing off-the-shelf progress in music demixing. Our model not only captures deeper harmonic cues for better metrical inference but also discerns beat and downbeat in a visualizable hierarchical manner. Our model is efficient, interpretable, and potentially generalizable with highly competitive sequential modelling power. We hope our model encourages future MIR research toward universal music understanding.

6. REFERENCES

[1] A. Holzapfel and E. Benetos, "The sousta corpus: Beat-informed automatic transcription of traditional dance tunes," in *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR*, 2016, pp. 531–537.

[2] G. Shibata, R. Nishikimi, and K. Yoshii, "Music structure analysis based on an LSTM-HSMM hybrid model," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020, pp. 23–29.

[3] R. Nishikimi, E. Nakamura, M. Goto, K. Itoyama, and K. Yoshii, "Bayesian singing transcription based on a hierarchical generative model of keys, musical notes, and F0 trajectories," *IEEE ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1678–1691, 2020.

[4] R. Ishizuka, R. Nishikimi, and K. Yoshii, "Global structure-aware drum transcription based on self-attention mechanisms," *Signals*, vol. 2, no. 3, pp. 508–526, 2021.

[5] C. Donahue and P. Liang, "Sheet sage: Lead sheets from music audio," 2021, Late Breaking Demo in the 22nd International Society for Music Information Retrieval Conference, ISMIR. [Online]. Available: https://archives.ismir.net/ismir2021/latebreaking/000049.pdf

[6] T. Bi, P. Fankhauser, D. Bellicoso, and M. Hutter, "Real-time dance generation to music for a legged robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*. IEEE, 2018, pp. 1038–1044.

[7] K. Yamamoto, "Human-in-the-loop adaptation for interactive musical beat tracking," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 794–801.

[8] Z. Cai, R. J. Ellis, Z. Duan, H. Lu, and Y. Wang, "Basic evaluation of auditory temporal stability (beats): A novel rationale and implementation," in *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR*, 2013, pp. 541–546.

[9] E. P. MatthewDavies and S. Böck, "Temporal convolutional networks for musical audio beat tracking," in *27th European Signal Processing Conference, EUSIPCO*. IEEE, 2019, pp. 1–5.

[10] F. Krebs, S. Böck, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking," in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR*, 2015, pp. 72–78.

[11] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR*, 2016, pp. 255–261.

[12] S. Böck, M. E. P. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 486–493.

[13] S. Böck and M. E. P. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020, pp. 574–582.

[14] T. Oyama, R. Ishizuka, and K. Yoshii, "Phase-aware joint beat and downbeat estimation based on periodicity of metrical structure," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 493–499.

[15] Y. Hung, J. Wang, X. Song, W. T. Lu, and M. Won, "Modeling beats and downbeats with a time-frequency transformer," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2022, pp. 401–405.

[16] W.-T. Lu, J.-C. Wang, M. Won, K. Choi, and X. Song, "SpecTNT: A time-frequency transformer for music audio," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 396–403.

[17] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, "Sequence-to-sequence piano transcription with transformers," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 246–253.

[18] L. Ou, Z. Guo, E. Benetos, J. Han, and Y. Wang, "Exploring transformer's potential on automatic piano transcription," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2022, pp. 776–780.

[19] J. Gardner, I. Simon, E. Manilow, C. Hawthorne, and J. H. Engel, "MT3: multi-task multitrack music transcription," in *The Tenth International Conference on Learning Representations, ICLR*. OpenReview.net, 2022.

[20] M. Won, S. Chun, and X. Serra, "Toward interpretable music tagging with self-attention," *arXiv preprint arXiv:1906.04972*, 2019.

[21] M. Won, K. Choi, and X. Serra, "Semi-supervised music tagging transformer," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 769–776.

[22] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020, deezer Research. [Online]. Available: https://doi.org/10.21105/joss.02154

[23] M. E. Davies, S. Böck, and M. Fuentes, *Tempo, Beat and Downbeat Estimation*, 2021. [Online]. Available: https://tempobeatdownbeat.github.io/tutorial/intro.html

[24] T. Lin, Y. Wang, X. Liu, and X. Qiu, "A survey of transformers," *arXiv preprint arXiv:2106.04554*, 2021.

[25] S. Böck and M. Schedl, "Enhanced beat tracking with context-aware neural networks," in *Proceedings of the 14th International Conference on Digital Audio Effects, DAFx*, 2011, pp. 135–139.

[26] S. Böck, F. Krebs, and G. Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles," in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, 2014, pp. 603–608.

[27] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *The 9th ISCA Speech Synthesis Workshop*. ISCA, 2016, p. 125.

[28] C. Chiu, J. Ching, W. Hsiao, Y. Chen, A. W. Su, and Y. Yang, "Source separation-based data augmentation for improved joint beat and downbeat tracking," in *29th European Signal Processing Conference, EUSIPCO*. IEEE, 2021, pp. 391–395.

[29] C. Chiu, A. W. Su, and Y. Yang, "Drum-aware ensemble architecture for improved joint musical beat and downbeat tracking," *IEEE Signal Processing Letters*, vol. 28, pp. 1100–1104, 2021.

[30] R. Dai, S. Das, L. Minciullo, L. Garattoni, G. Francesca, and F. Brémond, "PDAN: pyramid dilated attention network for action detection," in *IEEE Winter Conference on Applications of Computer Vision, WACV*. IEEE, 2021, pp. 2969–2978.

[31] N. Moritz, T. Hori, and J. L. Roux, "Capturing multi-resolution context by dilated self-attention," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2021, pp. 5869–5873.

[32] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, Volume 2*. Association for Computational Linguistics, 2018, pp. 464–468.

[33] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[34] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: A new python audio and music signal processing library," in *Proceedings of the 2016 ACM Conference on Multimedia Conference, MM*. ACM, 2016, pp. 1174–1178.

[35] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1832–1844, 2006.

[36] F. Krebs, S. Böck, and G. Widmer, "Rhythmic pattern modeling for beat and downbeat tracking in musical audio," in *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR*, 2013, pp. 227–232.

[37] S. W. Hainsworth and M. D. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP Journal on Advances in Signal Processing*, vol. 2004, no. 15, pp. 1–11, 2004.

[38] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical and jazz music databases," in *ISMIR 2002, 3rd International Conference on Music Information Retrieval*, 2002.

[39] O. Nieto, M. McCallum, M. E. P. Davies, A. Robertson, A. M. Stark, and E. Egozy, "The harmonix set: Beats, downbeats, and functional segment annotations of western popular music," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 565–572.

[40] A. Srinivasamurthy and X. Serra, "A supervised approach to hierarchical metrical cycle tracking from audio music recordings," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2014, pp. 5217–5221.

[41] A. Holzapfel, M. E. P. Davies, J. R. Zapata, J. L. Oliveira, and F. Gouyon, "Selective sampling for beat tracking evaluation," *IEEE Transactions on Speech and Audio Processing*, vol. 20, no. 9, pp. 2539–2548, 2012.

[42] G. Tzanetakis and P. R. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[43] U. Marchand and G. Peeters, "Swing ratio estimation," in *Proceedings of the 18th International Conference on Digital Audio Effects, DAFx*, 2015.

[44] D. Griffin and J. Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on acoustics, speech, and signal processing*, vol. 32, no. 2, pp. 236–243, 1984.

[45] N. Perraudin, P. Balázs, and P. L. Søndergaard, "A fast griffin-lim algorithm," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*. IEEE, 2013, pp. 1–4.

[46] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8. Citeseer, 2015, pp. 18–25.

[47] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *8th International Conference on Learning Representations, ICLR*. OpenReview.net, 2020.

[48] M. R. Zhang, J. Lucas, J. Ba, and G. E. Hinton, "Lookahead optimizer: k steps forward, 1 step back," in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, 2019, pp. 9593–9604.

[49] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[50] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *9th International Conference on Learning Representations, ICLR*, 2021.

[51] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *Proceedings of the 38th International Conference on Machine Learning, ICML*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 10 347–10 357.

[52] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *2021 IEEE/CVF International Conference on Computer Vision, ICCV*. IEEE, 2021, pp. 9992–10 002.

SKETCHING THE EXPRESSION: FLEXIBLE RENDERING OF EXPRESSIVE PIANO PERFORMANCE WITH SELF-SUPERVISED LEARNING

Seungyeon Rhyu[1] **Sarah Kim**[2] **Kyogu Lee**[1,3]
[1] Music and Audio Research Group (MARG), Seoul National University, South Korea
[2] Krust Universe, South Korea
[3] Graduate School of AI, AI Institute, Seoul National University, South Korea
rsy1026@snu.ac.kr, estelle.kim@krustuniverse.com, kglee@snu.ac.kr

ABSTRACT

We propose a system for rendering a symbolic piano performance with flexible musical expression. It is necessary to actively control musical expression for creating a new music performance that conveys various emotions or nuances. However, previous approaches were limited to following the composer's guidelines of musical expression or dealing with only a part of the musical attributes. We aim to disentangle the entire musical expression and structural attribute of piano performance using a conditional VAE framework. It stochastically generates expressive parameters from latent representations and given note structures. In addition, we employ self-supervised approaches that force the latent variables to represent target attributes. Finally, we leverage a two-step encoder and decoder that learn hierarchical dependency to enhance the naturalness of the output. Experimental results show that our system can stably generate performance parameters relevant to the given musical scores, learn disentangled representations, and control musical attributes independently of each other.

1. INTRODUCTION

Computational modeling of expressive music performance focuses on mimicking human behaviors that convey the music [1, 2]. For piano performance, one common task is to *render* an expressive performance from a quantized musical score. It aims to reproduce the loudness and timing of musical notes that fits to the given score. Most of the conventional studies have used musical scores of Western piano music that includes sufficient amount of guidelines for musical expressions [3–6]. Recent studies using deep learning methods have successfully rendered plausible piano performances that are comparable to those of professional pianists from the given Classical scores [7–9].

More recently, it has increased attention to *controlling* music performance by manipulating one or more *disentangled* representations from a generative model. These representations are sensitive to the variation of certain factors while invariant to other factors [10]. Maezawa *et al.* aimed to control a performer's interpretation through a conditional variational recurrent neural network (CVRNN) [11]. They intended to disentangle a time-variant representation of the personal interpretation. In the acoustic domain, Tan *et al.* proposed a generative model based on a Gaussian mixture variational autoencoder (GM-VAE) that separately controlled dynamics and articulations of the notes [12]. Their novelty lied in learning multiple representations of high-level attributes from the low-level spectrogram.

However, these studies have constrained musical creativity. Maezawa *et al.* controlled musical expression only through quantized features from the musical scores. Tan *et al.* did not consider controlling tempo or timing with a latent representation. These methods may have restricted any potential for rendering piano performances with flexible musical expression. Musical creativity can be expanded not only by composers but also by performers who can elastically choose various strategies to highlight multiple nuances or emotions [13–15]. Moreover, the music generation field can be also broadened if static music created by automatic composition systems can be easily colored with realistic and elastic expression [16].

Therefore, we attempt a new approach that renders piano performances with flexible musical expressions. We disregard a typical assumption from previous studies that a performer must follow a composer's intent [4, 17–19]. According to the literature, performers learn to identify or imitate "expressive models", or *explicit planning*, of existing piano performances [20]. We focus on this attribute, defining it as a higher-level *sketch* of the expressive attributes (i.e. dynamics, articulation, and tempo [21]) that the performer draws based on a personal interpretation of the musical piece [4, 11, 20]. We also assume that the remaining attribute represents common performing strategies that are connected to certain musical patterns, while these strategies slightly differ across performers [22, 23]. We call this attribute as a *structural attribute* that belongs to given note structures of a musical piece.

In this study, we propose a generative model that can

© S. Rhyu, S. Kim, and K. Lee. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** S. Rhyu, S. Kim, and K. Lee, "Sketching the Expression: Flexible Rendering of Expressive Piano Performance with Self-Supervised Learning", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

flexibly control the entire musical expression, or the explicit planning, of symbolic piano performance [1]. Our system is based on a conditional variational autoencoder (CVAE) that is modified for sequential data [11, 24]. The system generates multiple parameters of piano performance from a note structure of a musical passage, using disentangled representations for the explicit planning and structural attribute.

We employ a self-supervised learning framework to force the latent representations to learn our target attributes [24–26]. In addition, we facilitate independent control of the three expressive attributes–dynamics, articulation, and tempo–by utilizing an existing method that aligns the latent code with a target attribute [27, 28]. Finally, we design a novel mechanism that intuitively models a polyphonic structure of piano performance. In particular, we insert intermediate steps for *chordwise* encoding and decoding of the piano performance to our encoder-decoder architecture, where a *chord* denotes a group of simultaneous notes.

Our approach has several contributions as follows: 1) Our system aims to control musical expression while maintaining any characteristics induced by a given musical structure; 2) We use self-supervised learning where new supervisory signals are involved in regularizing the latent representations effectively; 3) Our system aims to control multiple expressive attributes independently of each other; 4) Lastly, we leverage an intermediate step that projects a notewise representation into the chordwise in the middle of our system to intuitively model the polyphonic structure of piano performance.

2. PROPOSED METHODS

We aim to build a generative model that factorizes expressive piano performance as the explicit planning and structural attribute. The model is based on a conditional variational autoencoder (CVAE) that reproduces performance parameters based on a given musical structure.

2.1 Data Representation

We extract features that represent a human performance and the corresponding musical score, following the conventional studies [11, 19, 29].

Performance Features. We extract three features that represent the expressive attributes of each performed note, respectively: **MIDIVelocity** is a MIDI velocity value that ranges from 24 to 104. **IOIRatio** represents an instantaneous variation in tempo. We compute an inter-onset-interval (IOI) between the onset of a note and the mean onset of the *previous* chord for both a performed note and the corresponding score note. Then, a ratio of performed IOI to score IOI is calculated, clipped between 0.125 and 8, and converted into a logarithmic scale [4]. **Articulation** represents how much a note is shortened or lengthened compared to the instantaneous tempo. It is a ratio of a performed duration to an IOI value between the onset of a note and mean onset of the *next* chord [19]. It is clipped between 0.25 and 4 and converted into a logarithmic scale.

Score Features. The features for a musical score represent eight categorical attributes for how the notes are composed: **Pitch** is a MIDI index number that ranges from 21 to 108. **RelDuration** and **RelIOI** are 11-class attributes of a quantized duration and IOI between a note onset and a previous chord, respectively. They range from 1 to 11, and each class represents a multiple of a 16th note's length with respect to a given tempo [30, 31]. **IsTopVoice** is a binary attribute of whether the note is the uppermost voice. It is heuristically computed regarding pitches and durations of surrounding notes. **PositionInChord** and **NumInChord** are 11-class attributes of a positional index of a note within its chord and the total number of notes in that chord, respectively, that range from 1 to 11. An index 1 for PositionInChord denotes the most bottom position. **Staff** is a binary attribute of the staff of a note, either of the G clef or F clef. **IsDownbeat** is a binary attribute of whether a note is at a downbeat or not.

2.2 Modeling Musical Hierarchy

Inspired by previous studies [4, 8, 9, 32], we build a two-step encoder and decoder: An encoder models both notewise and chordwise dependencies of the inputs, and a decoder reconstructs the notewise dependency from the chordwise representation and the notewise condition. We denote a *chord* as a group of notes that are hit simultaneously, regardless of the staff, so that they sound together at an instant time [33]. Thus, learning the chordwise dependency is analogous to direct modeling of the temporal progression of the piano performance. Let $\mathcal{M} \in \mathbb{R}^{C\times N}$ be a matrix that aligns serialized notes to their polyphonic structure, where C and N are the number of chords and the number of notes, respectively. Within the encoder, the notewise representation is sequentially average-pooled by $\mathcal{M}$ with dynamic kernel sizes where each size represents the number of notes in each chord. We denote this operation as *N2C*. In this way, we can directly model chord-level dependency of the note-level expressive parameters [32]. In contrast, the decoder extends the chordwise representation from the encoder back to the notewise using the transposed alignment matrix $\mathcal{M}^T$, of which process we denote as *C2N*. Along this, the notewise embedding of the score features replenishes the notewise information for the output. Consequently, notes in the same chord *share* any information of their corresponding chord, while maintaining their differences by the conditional score features:

$$\text{N2C}(e) = \frac{\mathcal{M}\cdot e}{\sum_{n=1}^{N}\mathcal{M}_{n,1:C}}, \quad \text{C2N}(e) = \mathcal{M}^{\text{T}}\cdot e \qquad (1)$$

where e denotes a notewise or chordwise representation.

2.3 Overall Network Architecture

Our proposed network is generally based on the conditional VAE framework [34, 35]. Concretely, we use the sequential VAE that is modified for generation of sequential data [11,24,36]. Let $x = \{x_n\}_{n=1}^{N}$ be a sequence of the

[1] https://github.com/rsy1026/sketching_piano_expression

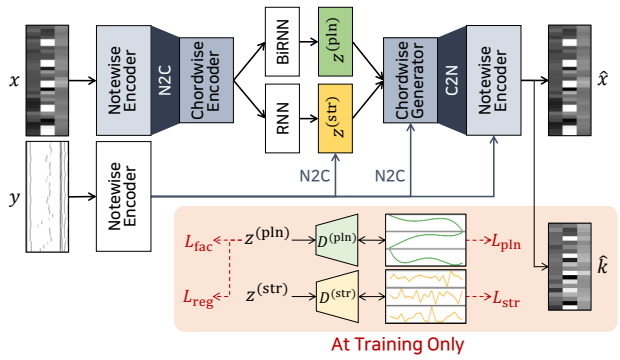


Figure 1: Overall architecture of the proposed system. The orange box includes the auxiliary tasks only for training.

performance features, and $y = \{y_n\}_{n=1}^{N}$ be a sequence of the conditional score features. Our network has two *chord-wise* latent variables $z^{(\text{pln})} = \{z_c^{(\text{pln})}\}_{c=1}^{C} \in \mathbb{R}^{C \times d^{(\text{pln})}}$ and $z^{(\text{str})} = \{z_c^{(\text{str})}\}_{c=1}^{C} \in \mathbb{R}^{C \times d^{(\text{str})}}$ that represent explicit planning and structural attribute, where $d^{(\text{pln})}$ and $d^{(\text{str})}$ are the sizes of $z^{(\text{pln})}$ and $z^{(\text{str})}$, respectively. Our network generates notewise performance parameters x from these latent variables and given score features y. The overall architecture of our proposed system is illustrated in Figure 1.

Generation. A probabilistic generator parameterized by θ produces the note-level performance parameters x from the two latent variables $z^{(\text{pln})}$ and $z^{(\text{str})}$ with the given condition y. We note that the latent variables are in chord-level. This decreases a computational cost and also enables intuitive modeling of polyphonic piano performance where each time step represents a stack of notes and the simultaneous notes share common characteristics [8]:

$$p_\theta(x, y, z^{(\text{pln})}, z^{(\text{str})}) = p_\theta(x|z^{(\text{pln})}, z^{(\text{str})}, y) \\ p_\theta(z^{(\text{pln})}) \prod_{c=1}^{C} p_\theta(z_c^{(\text{str})}|z_{<c}^{(\text{str})}, y_{\leq c}^{(\text{chd})}) \quad (2)$$

where $y^{(\text{chd})} = \text{N2C}(e_y)$ is the chordwise embedding, and e_y is the notewise embedding for y. We assume that the prior of $z_c^{(\text{pln})}$ is a standard normal distribution. In contrast, $z_c^{(\text{str})}$ is sampled from a sequential prior [24, 36, 37], conditioned on both previous latent variables and chordwise score features: $z_c^{(\text{str})} \sim \mathcal{N}(\mu^{(\text{prior})}, \text{diag}(\sigma^{(\text{prior})^2})$, where $[\mu^{(\text{prior})}, \sigma^{(\text{prior})}] = f^{(\text{prior})}(z_{<c}^{(\text{str})}, y_{\leq c}^{(\text{chd})})$, and $f^{(\text{prior})}$ is a unidirectional recurrent neural network. The latent representations and $y^{(\text{chd})}$ pass through the decoder as shown in Figure 1. During training, the model predicts the intermediate chordwise output that is computed as $\text{N2C}(x)$. This is to enhance reconstruction power of our system, propagating accurate information of chord-level attributes to the final decoder. The intermediate activation is then extended to the notewise through the C2N operation. The note-level parameters are generated autoregressively based on this activation and the notewise score feature. We use teacher forcing during training [38].

Inference. A probabilistic encoder parameterized by ϕ approximates the posterior distibutions of the latent representations $z^{(\text{pln})}$ and $z^{(\text{str})}$ from the performance input x and conditional score input y:

$$q_\phi(z^{(\text{pln})}, z^{(\text{str})}|x, y) = q_\phi(z^{(\text{pln})}|x^{(\text{chd})}) \\ \prod_{c=1}^{C} q_\phi(z_c^{(\text{str})}|x_{\leq c}^{(\text{chd})}, y_{\leq c}^{(\text{chd})}) \quad (3)$$

where $x^{(\text{chd})} = \text{N2C}(e_x)$ is the chordwise embedding, and e_x is the notewise embedding for x. The posterior distributions of $z_c^{(\text{pln})}$ and $z_c^{(\text{str})}$ are approximated by distribution parameters encoded by $f^{(\text{pln})}(x^{(\text{chd})})$ and $f^{(\text{str})}(x^{(\text{chd})}, y^{(\text{chd})})$, where $f^{(\text{pln})}$ and $f^{(\text{str})}$ are bidirectional and unidirectional recurrent neural networks, respectively. We note that $z^{(\text{pln})}$ is independent of the score features y. This allows a flexible transfer of the explicit planning among other musical pieces. On the other hand, $z^{(\text{str})}$ is constrained by y since the structural attributes are dependent on the note structure.

Training. We train the models p_θ and q_ϕ by approximating marginal distributions of the performance features x conditioned on the score features y. This requires to maximize negative evidence lower bound (ELBO) that includes regularization force by Kullback–Leibler divergence [34]:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi(z^{(\text{pln})}, z^{(\text{str})}|x,y)} \left[\log p_\theta(x|z^{(\text{pln})}, z^{(\text{str})}, y)\right] \\ + \mathbb{E}_{q_\phi(z^{(\text{pln})}, z^{(\text{str})}|x,y)} \left[\log p_\theta(k|z^{(\text{pln})}, z^{(\text{str})}, y)\right] \\ - \text{KL}(q_\phi(z^{(\text{pln})}|x) \| p_\theta(z^{(\text{pln})})) \\ - \sum_{c=1}^{C} \text{KL}(q_\phi(z_c^{(\text{str})}|x_{\leq c}^{(\text{chd})}, y_{\leq c}^{(\text{chd})}) \| p_\theta(z_c^{(\text{str})}|z_{<c}^{(\text{str})}, y_{\leq c}^{(\text{chd})})) \quad (4)$$

where $k = \text{N2C}(x)$ is the chordwise performance features.

2.4 Regularizing the Latent Variables

We enhance disentanglement of the latent representations $z^{(\text{pln})}$ and $z^{(\text{str})}$ using four regularization tasks [24].

Prediction Tasks. We extract new supervisory signals for additional prediction tasks from the input data [24]. We define a signal of explicit planning $I^{(\text{pln})}$ as a set of smoothed contours of the expressive parameters. It is extracted as a polynomial function predicted from the chordwise performance parameters k. We also derive a signal of structural attribute as $I^{(\text{str})} = \text{sign}(k - I^{(\text{pln})})$ which represents normalized directions of the performance parameters. We train two discriminators $D^{(\text{pln})}$ and $D^{(\text{str})}$ that directly receive $z^{(\text{pln})}$ and $z^{(\text{str})}$, respectively. $D^{(\text{pln})}$ is composed of A sub-discriminators where each discriminator $D_a^{(\text{pln})}$ predicts a signal $I_a^{(\text{pln})}$ for each expressive attribute a from $z_a^{(\text{pln})} \in \mathbb{R}^{C \times (d^{(\text{pln})}/A)}$, where $z_a^{(\text{pln})}$ is a constituent part of $z^{(\text{pln})}$, and A is the number of expressive attributes. This setting is for a clear disentanglement among the expressive attributes. On the other hand, $D^{(\text{str})}$ predicts the signal $I^{(\text{str})}$ at once for all expressive attributes that belong to the same musical structure. All discriminators are jointly trained with the generative model, and the costs $\mathcal{L}_{\text{pln}}$ and $\mathcal{L}_{\text{str}}$ are minimized

as $\mathcal{L}_{\text{pln}} = \frac{1}{A}\sum_a \text{MSE}(D_a^{(\text{pln})}(z_a^{(\text{pln})}), I_a^{(\text{pln})})$ and $\mathcal{L}_{\text{str}} = \text{MSE}(D^{(\text{str})}(z^{(\text{str})}), I^{(\text{str})})$, respectively.

Factorizing Latent Variables. We further constrain a generator to guarantee that $z^{(\text{pln})}$ delivers correct information regardless of $z^{(\text{str})}$ [39]. During training, we sample a new output $\tilde{x}$ using $z^{(\text{pln})} \sim q_\phi(z^{(\text{pln})}|x)$ and $\tilde{z}^{(\text{str})} \sim p_\theta(z^{(\text{str})})$. Then, we re-infer $\tilde{z}^{(\text{pln})} \sim q_\phi(\tilde{z}^{(\text{pln})}|\tilde{x})$ to estimate the superversory signal $I^{(\text{pln})}$. This prediction loss is backpropagated only through the generator:

$$\mathcal{L}_{\text{fac}} = \frac{1}{A}\sum_a \text{MSE}(D_a^{(\text{pln})}(\tilde{z}_a^{(\text{pln})}), I_a^{(\text{pln})}) \quad (5)$$

Aligning Latent Variables with Factors. Finally, we enable the "sliding-fader" control of the expressive attributes [28]. To this end, we employ the regularization loss proposed by Pati *et al.* [27] that aligns specific dimensions of $z^{(\text{pln})}$ with the target expressive attributes. This method assumes that a latent representation can be disentangled through its monotonic relationship with a target attribute. Let d_i and d_j be a target dimension d of ith and jth latent representations, respectively, where $d \in z_a^{(\text{pln})}$, $i, j \in [1, M]$, and M is the size of a mini-batch. A distance matrix $\mathcal{D}_d$ is computed between d_i and d_j within a mini-batch, where $\mathcal{D}_d = d_i - d_j$. A similar distance matrix $\mathcal{D}_a$ is computed for the two target attribute values a_i and a_j. We minimize a MSE between $\mathcal{D}_d$ and $\mathcal{D}_a$ as follows:

$$\mathcal{L}_{\text{reg}} = \text{MSE}(\tanh(\mathcal{D}_d), \text{sign}(\mathcal{D}_a)) \quad (6)$$

2.5 Overall Objective

The overall objective of our proposed network aims to generate realistic performance features with properly disentangled representations for the intended factors:

$$\mathcal{L} = \mathcal{L}_{\text{VAE}} + \lambda_{\text{pln}}\mathcal{L}_{\text{pln}} + \lambda_{\text{str}}\mathcal{L}_{\text{str}} + \lambda_{\text{fac}}\mathcal{L}_{\text{fac}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}} \quad (7)$$

where λ_{pln}, λ_{str}, λ_{fac}, and λ_{reg} are hyperparameters for balancing the importance of the loss terms.

3. EXPERIMENTAL SETUPS

3.1 Dataset and Implementation

We use Yamaha e-Competition Dataset [8] and Vienna 4x22 Piano Corpus [40]. From these datasets, we collect 356 performances of 34 pieces by Frédéric Chopin, which have been representative research subjects for analyzing the Western musical expression [6, 22, 41, 42]. We use 30 pieces (108,738 batches) for training and the rest for testing. To verify the generality of model performances, we also collect the external dataset from ASAP dataset [43]. We use 116 performances for 23 pieces by 10 composers who represent various eras of Western music. For subjective evaluation, we collect 42 songs of non-Classical songs from online source [2] which are less constrained to written expression than most Classical excerpts.

[2] http://www.ambrosepianotabs.com/page/library

We basically follow Jeong *et al.* [8] to compute the input features from the aligned pairs of performance and score data. We set MIDI velocities and Beat Per Minute (BPM) of all notes in the score data to be 64 and 120, respectively. We also remove any grace notes for simplicity and manually correct any errors. The performance features are further normalized into a range from -1 to 1 for training. We use an ADAM optimizer [44] with an initial learning rate of 1e-5, which is reduced by 5% every epoch during backpropagation. We empirically set λ_{pln}, λ_{str}, λ_{fac}, and λ_{reg} to be 1000, 100, 1, 10, respectively. We set a degree of the polynomial function computing $I^{(\text{pln})}$ as 4 through an ablation study described in the supplementary material.

3.2 Comparative Methods

To the best of our knowledge, there is no existing method that does not intentionally follow the written guidelines in the musical score. Therefore, we use variants of our proposed network as comparing methods that differ in model architecture: **Notewise** denotes the proposed model without the hierarchical learning. **CVAE** denotes a variant of Notewise where $z^{(\text{pln})}$ is substituted with the supervisory signal $I^{(\text{pln})}$. We also conduct an ablation study that investigates necessity of the four loss terms.

4. EVALUATION

We evaluate the proposed network in terms of both objective and subjective criteria.

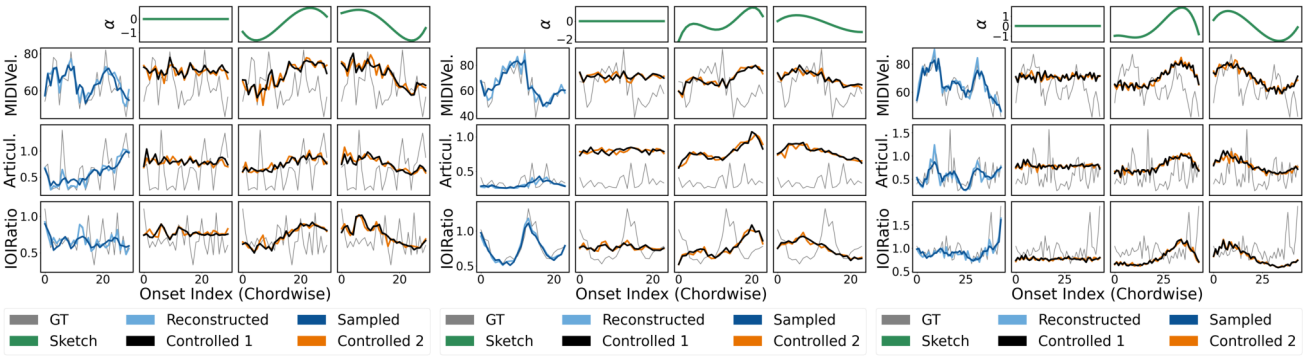
4.1 Generation Quality

We compute Pearson's correlation coefficients between the reconstructed or generated samples and human piano performances [6, 9, 11, 19]. We first measure the reconstruction quality of the test samples ("R_{recon}"). Then, we evaluate the samples generated from $\tilde{z}^{(\text{str})} \sim p_\theta(z^{(\text{str})})$ and either of : 1) $z^{(\text{pln})} \sim q_\phi(z^{(\text{pln})}|x)$ ("$\text{R}_{x|\text{pln}}$") and 2) $z_0^{(\text{pln})} \sim q_\phi(z_0^{(\text{pln})}|x_0)$ ("$\text{R}_{x|\text{pln}_0}$"), where x_0 is a zero matrix.

The results are shown in Table 1. Notewise shows the best scores in both datasets, and our method outperforms CVAE in R_{recon}. It indicates that our proposed architecture where a latent representation is used instead of a direct condition is generally good at reconstructing the human data. When using the randomly sampled $\tilde{z}^{(\text{str})}$, our method and the model without $\mathcal{L}_{\text{reg}}$ show stable scores compared to other baseline models. The model without $\mathcal{L}_{\text{reg}}$ also shows the highest scores in $\text{R}_{x|\text{pln}}$ for both datasets. It indicates that $\mathcal{L}_{\text{reg}}$ may contribute the least to generation power among other loss terms. CVAE and the model only with $\mathcal{L}^{(\text{pln})}$ also show high scores in $\text{R}_{x|\text{pln}_0}$. This may be due to the posterior collapse that makes the decoder depends mostly on the score condition [45], which is demonstrated in the supplementary material.

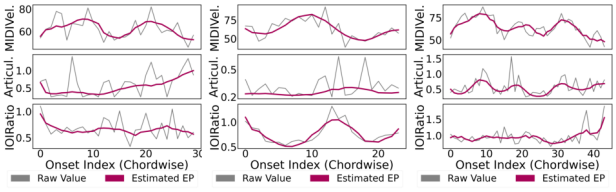
4.2 Disentangling Latent Representations

We verify whether the latent representations are well-disentangled by appropriate information [24]. To this



(a) (b) (c)

Figure 2: Qualitative samples for the proposed system. Light-blue, blue and gray lines denote the reconstructed results, sampled results from the inferred $z^{(\text{pln})}$, and their ground truths, respectively; black and orange lines denote the controlled results that are generated from different random $\tilde{z}^{(\text{str})}$; and green lines denote the "sketch" values, or α, that are inserted to $z^{(\text{pln})}$. The samples demonstrate three excerpts that are: (a) Haydn's Keyboard Sonata, Hob. XVI:39, 3rd movement, mm. 53-56; (b) Schubert's Impromptu, Op. 90, No. 4, mm. 149-152; and (c) Balakirev's Islamey, Op. 18, mm. 29-32.



(a) (b) (c)

Figure 3: Qualitative results for estimating the explicit planning from raw piano performances. Pink and gray lines denote the estimated contours and raw performance parameters, respectively. The results in (a), (b), and (c) are from the same excerpts for (a), (b), and (c) in Figure 2, respectively.

Dataset	Internal			External		
Metric	R_{recon}	$\text{R}_{x\|\text{pln}}$	$\text{R}_{x\|\text{pln}_0}$	R_{recon}	$\text{R}_{x\|\text{pln}}$	$\text{R}_{x\|\text{pln}_0}$
Notewise	**0.870**	0.392	0.203	**0.875**	0.479	0.177
CVAE	0.730	0.338	0.223	0.741	0.399	0.216
$\mathcal{L}_{\text{pln}}$	0.627	0.357	0.229	0.687	0.414	**0.220**
$\mathcal{L}_{\text{pln}} + \mathcal{L}_{\text{str}}$	0.770	0.325	0.181	0.837	0.398	0.195
w/o $\mathcal{L}_{\text{fac}}$	0.774	0.289	0.176	0.838	0.354	0.173
w/o $\mathcal{L}_{\text{reg}}$	0.737	**0.437**	0.224	0.793	**0.502**	0.216
Ours	0.737	0.427	**0.231**	0.789	0.498	0.203

Table 1: Evaluation results for the generation quality. The higher score is the better.

end, each model infers the latent representations $z^{(\text{pln})}$ and $z^{(\text{str})}$ from the test sets. Each model also randomly samples $\tilde{z}^{(\text{str})}$ and infers $z_0^{(\text{pln})} \sim q_\phi(z^{(\text{pln})}|x_0)$. We use $z_0^{(\text{pln})}$ to measure the structural attribute, since $z_0^{(\text{pln})}$ represents a flat expression where the structural attribute can be solely exposed. Each model generates new outputs as $x^{(\text{pln})} \sim p_\theta(x^{(\text{pln})}|z^{(\text{pln})}, \tilde{z}^{(\text{str})}, y)$ and $x^{(\text{str})} \sim p_\theta(x^{(\text{str})}|z_0^{(\text{pln})}, z^{(\text{str})}, y)$. Then, we compute a new signal $\tilde{I}^{(\text{pln})}$ from $x^{(\text{pln})}$ using the polynomial regression. The MSE values are calculated as $\text{MSE}_\text{p} = \text{MSE}(\tilde{I}^{(\text{pln})}, I^{(\text{pln})})$ and $\text{MSE}_\text{s} = \text{MSE}(x^{(\text{str})}, k - I^{(\text{pln})})$.

Dataset	Internal		External	
Metric	MSE_p	MSE_s	MSE_p	MSE_s
Notewise	0.003	0.006	0.022	0.028
CVAE	0.034	0.045	0.085	0.092
$\mathcal{L}_{\text{pln}}$	0.028	0.036	0.074	0.077
$\mathcal{L}_{\text{pln}} + \mathcal{L}_{\text{str}}$	0.012	0.015	0.022	0.027
w/o $\mathcal{L}_{\text{fac}}$	0.018	0.023	0.021	0.025
w/o $\mathcal{L}_{\text{reg}}$	0.002	0.004	0.014	0.022
Ours	**0.001**	**0.002**	**0.012**	**0.020**

Table 2: Evaluation results for the disentanglement of the latent representations.

Dataset	Internal			External		
Metric	C	R	L	C	R	L
Notewise	0.782	0.916	0.632	0.775	0.914	0.656
CVAE	0.798	0.812	0.620	0.773	0.802	0.649
$\mathcal{L}_{\text{pln}}$	0.693	0.852	0.323	0.694	0.834	0.324
$\mathcal{L}_{\text{pln}} + \mathcal{L}_{\text{str}}$	0.633	0.882	0.253	0.639	0.865	0.277
w/o $\mathcal{L}_{\text{fac}}$	0.831	0.846	0.789	0.832	0.831	0.847
w/o $\mathcal{L}_{\text{reg}}$	0.804	**0.955**	0.653	0.808	**0.946**	0.657
Ours	**0.942**	0.953	**0.976**	**0.944**	0.945	**0.977**

Table 3: Evaluation results for the controllability of the expressive attributes. C, R, and L denotes consistency, restrictiveness, and linearity, respectively. Each score is the average score for the expressive attributes.

Table 2 shows that our method achieves the best scores in all metrics for both datasets. This confirms that our proposed system can learn the latent representations that reflect the intended attributes. Notewise and the model without $\mathcal{L}_{\text{reg}}$ also show the robust scores compared to other baseline models. It indicates that using the notewise modeling alone is still relevant for achieving appropriate representations. It also implies that $\mathcal{L}_{\text{reg}}$ may not contribute to the disentanglement as much as other loss terms.

4.3 Controllability of Expressive Attributes

We sample a new input $\bar{x}$ where entries of each feature are constant across time. Then, each model infers

Metric	Winning Rate (Human-likeness)		
Group	T	UT	Overall
Notewise	0.317(±0.223)	0.541(±0.316)	0.493(±0.309)
CVAE	**0.467(±0.356)**	0.477(±0.342)	0.475(±0.338)
Ours	0.417(±0.256)	**0.555(±0.256)**	**0.525(±0.258)**

Table 4: Evaluation results for the winning rate in terms of human-likeness. T, UT, and Overall denote musically trained, untrained, and all groups, respectively.

$\bar{z}^{(\text{pln})} \sim q_\phi(\bar{z}^{(\text{pln})}|\bar{x})$. We control each attribute by varying dimension values of $\bar{z}^{(\text{pln})}$ following Tan *et al.* [28] and examine the new samples generated from $\bar{z}^{(\text{pln})}$. We leverage the existing metrics to measure the controllability of each model [28]: *Consistency* ("C") measures consistency across samples in terms of their controlled attributes; *restrictiveness* ("R") measures how much the uncontrolled attributes maintain their flatness over time; and *linearity* ("L") measures how much the controlled attributes are correlated with the corresponding latent dimensions. We average over the three expressive attributes–dynamics, articulation, and tempo–into one score for each metric.

Table 3 demonstrates that our system shows the best scores in consistency and linearity in both internal and external datasets. This indicates that our proposed method can robustly control the latent representation $z^{(\text{pln})}$ in intended way. The model without $\mathcal{L}_{\text{reg}}$ outperforms our method in restrictiveness. It indicates that the uncontrolled attributes by this model are the least interfered by the controlled attribute. However, its scores on consistency and linearity are lower than ours. It confirms that $\mathcal{L}_{\text{reg}}$ promotes linear control of the target attributes.

4.4 Subjective Evaluation

We conduct a listening test to compare the proposed model architecture to Notewise and CVAE. We qualitatively evaluate the base quality of the samples that have flat expressions, so that quality judgments are independent of any preference of arbitrary explicit planning. We generate each sample using $z_0^{(\text{pln})}$. A listening test is composed of 30 trials where each participant chooses a more "human-like" sample out of the generated sample and its plain MIDI [9]. Both samples have the same length which is a maximum of 15 seconds, rendered with TiMidity++ [3] without any pedal effect. *Human-likeness* denotes how similar the sample is to an actual piano performance that commonly appears in popular music. A total of 28 participants are involved, and 6 participants are professionally trained in music.

The results are demonstrated in Table 4. We measure a *winning rate*, a rate of winning over the plain MIDI, and a *top-ranking rate*, a rate of being the highest rank among the three models in terms of winning rate. These metrics are further explained in the supplementary material. The results show that musically *trained* ("T") and *untrained* ("UT") groups show the different tendency of each other: in the trained group, CVAE shows the best winning rate, and our method gets the best top-ranking rate; in the untrained group, our method shows the highest winning rate, whereas Notewise is top-ranked most frequently. We note that our system reveals smaller variances than those of CVAE and Notewise of the musically trained and untrained groups in the winning rate, respectively. Moreover, our system receives the highest overall scores for both metrics. It indicates that our system can be stably perceived more human-like than the plain MIDI compared to other baseline models.

[3] https://sourceforge.net/projects/timidity/

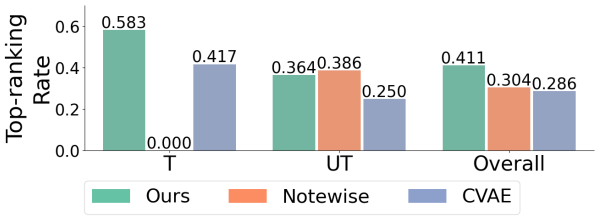


Figure 4: Evaluation results for the top-ranking rate. T, UT, and Overall denote musically trained, untrained, and all groups, respectively.

4.5 Qualitative Examples

Our system can render new piano performances from the scratch given a musical score. It can directly generate expressive parameters from the randomly sampled $\tilde{z}^{(\text{pln})} \sim p_\theta(z^{(\text{pln})})$ and $\tilde{z}^{(\text{str})} \sim p_\theta(z^{(\text{str})})$. We note that $\tilde{z}^{(\text{pln})}$ does not have temporal dependency: each $\tilde{z}_c^{(\text{pln})}$ is sampled independently of $\tilde{z}_{c-1}^{(\text{pln})}$. Hence, we need to insert specific values $\{\alpha^{(c)}\}_{c=1}^{C}$, which we call as "smooth sketches", into the target dimensions of $z^{(\text{pln})}$ if any temporal dependency of explicit planning is necessary. Figure 2 shows that the controlled parameters are greatly correlated with α, while their local characteristics follow those of the ground truth. In addition, the black and orange lines together demonstrate granular variety in the parameters induced by different $\tilde{z}^{(\text{str})}$ for the same musical structure. Moreover, Figure 3 shows that our system can estimate explicit planning from arbitrary human performances, indicating that our system can derive relevant information on explicit planning from the unseen data.

5. CONCLUSION

We propose a system that can render expressive piano performance with flexible control of musical expression. We attempt to achieve representations for the explicit planning and structural attribute through self-supervised learning objectives. We also leverage the two-step modeling of two hierarchical units for an intuitive generation. Experimental results confirm that our system shows stable generation quality, disentangles the target representations, and controls all expressive attributes independently of each other. Future work can be improving our system using a larger dataset for various genres and composers. We can also further compare our system with recent piano-rendering models [8] to investigate any connections between a performer's explicit planning and a composer's intent.

6. ACKNOWLEDGMENTS

We deeply appreciate Dasaem Jeong, Taegyun Kwon, and Juhan Nam for giving technical support to initiate this research. We also especially appreciate Hyeong-Seok Choi for providing critical feedback on the model architecture and evaluation. We greatly appreciate You Jin Choi and all of my colleagues who gave great help with respect to the listening test.

7. REFERENCES

[1] G. Widmer and W. Goebl, "Computational models of expressive music performance: The state of the art," *Journal of New Music Research*, vol. 33, no. 3, pp. 203–216, 2004.

[2] C. E. Cancino-Chacón, M. Grachten, W. Goebl, and G. Widmer, "Computational models of expressive music performance: A comprehensive and critical review," *Frontiers in Digital Humanities*, vol. 5, no. 25, pp. 1–23, 2018.

[3] G. Widmer, S. Flossmann, and M. Grachten, "YQX plays Chopin," *AI Magazine*, vol. 30, no. 3, pp. 35–48, 2009.

[4] T. H. Kim, S. Fukayama, T. Nishimoto, and S. Sagayama, "Statistical approach to automatic expressive rendition of polyphonic piano music," in *Guide to Computing for Expressive Music Performance*. Springer, 2013, pp. 145–179.

[5] C. E. Cancino-Chacón and M. Grachten, "An evaluation of score descriptors combined with non-linear models of expressive dynamics in music," in *Proceedings of the International Conference on Discovery Science*, 2015.

[6] C. E. Cancino-Chacón, T. Gadermaier, G. Widmer, and M. Grachten, "An evaluation of linear and non-linear models of expressive dynamics in classical piano and symphonic music," *Machine Learning*, vol. 106, no. 6, pp. 887–909, 2017.

[7] A. Maezawa, "Deep piano performance rendering with conditional VAE," in *Late-Breaking Demo, the 19th International Society for Music Information Retrieval Conference*, 2018.

[8] D. Jeong, T. Kwon, Y. Kim, K. Lee, and J. Nam, "VirtuosoNet: A hierarchical RNN-based system for modeling expressive piano performance," in *Proceedings of the 20th International Society for Music Information Retrieval*, 2019.

[9] D. Jeong, T. Kwon, Y. Kim, and J. Nam, "Graph neural network for music score data and modeling expressive piano performance," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[10] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, 2013.

[11] A. Maezawa, K. Yamamoto, and T. Fujishima, "Rendering music performance with interpretation variations using conditional variational RNN," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 2019.

[12] H. H. Tan, Y.-J. Luo, and D. Herremans, "Generative modeling for controllable audio synthesis of expressive piano performance," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[13] R. Bresin and A. Friberg, "Emotional coloring of computer-controlled music performances," *Computer Music Journal*, vol. 24, no. 4, 2000.

[14] S. R. Livingstone, R. Muhlberger, A. R. Brown, and W. F. Thompson, "Changing musical emotion: A computational rule system for modifying score and performance," *Computer Music Journal*, vol. 34, no. 1, 2010.

[15] M. Bernays and C. Traube, "Investigating pianists' individuality in the performance of five timbral nuances through patterns of articulation, touch, dynamics, and pedaling," *Frontiers in Psychology*, vol. 5, no. 157, pp. 1–19, 2014.

[16] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, vol. 32, pp. 955–967, 2020.

[17] A. Bhatara, A. K. Tirovolas, L. M. Duan, B. Levy, and D. J. Levitin, "Perception of emotional expression in musical performance," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 37, no. 3, pp. 921–934, 2011.

[18] A. Friberg, R. Bresin, and J. Sundberg, "Overview of the KTH rule system for musical performance," *Advances in Cognitive Psychology*, vol. 2, no. 2-3, pp. 145–161, 2006.

[19] S. Flossmann, M. Grachten, and G. Widmer, "Expressive performance rendering with probabilistic models," in *Guide to Computing for Expressive Music Performance*. Springer, 2013, pp. 75–98.

[20] R. H. Woody, "The relationship between explicit planning and expressive performance of dynamic variations in an aural modeling task," *Journal of Research in Music Education*, vol. 47, no. 4, pp. 331–342, 1999.

[21] A. Lerch, C. Arthur, A. Pati, and S. Gururani, "Music performance analysis: A survey," in *Proceedings of the 20st International Society for Music Information Retrieval Conference*, 2019.

[22] B. H. Repp, "A microcosm of musical expression: II. quantitative analysis of pianists' dynamics in the initial measures of Chopin's Etude in E major," *The Journal of the Acoustical Society of America*, vol. 105, no. 3, pp. 1972–1988, 1999.

[23] H. Honing, "From time to time: The representation of timing and tempo," *Computer Music Journal*, vol. 25, no. 3, 2001.

[24] Y. Zhu, M. R. Min, A. Kadav, and H. P. Graf, "S3VAE: Self-supervised sequential VAE for representation disentanglement and data generation," in *Proceedings of Computer Vision and Pattern Recognition*, 2020.

[25] F. Locatello, S. Bauer, M. Lucic, G. Rätsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," in *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[26] D. Hendrycks, M. Mazeika, S. Kadavath, and D. Song, "Using self-supervised learning can improve model robustness and uncertainty," in *Proceedings of the 33rd Conference on Neural Information Processing Systems*, 2019.

[27] A. Pati and A. Lerch, "Attribute-based regularization of latent spaces for variational auto-encoders," in *Neural Computing and Applications*, 2020.

[28] H. H. Tan and D. Herremans, "Music FaderNets: Controllable music generation based on high-level features via low-level feature modeling," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 2020.

[29] D. Jeong, T. Kwon, Y. Kim, and J. Nam, "Score and performance features for rendering expressive music performances," in *Proceedings of the Music Encoding Conference*, 2019.

[30] A. Roberts, J. Engel, and D. Eck, "Hierarchical variational autoencoders for music," in *Proceedings of the 31st Conference on Neural Information Processing Systems*, 2017.

[31] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.

[32] S.-L. Wu and Y.-H. Yang, "MuseMorphose: Full-song and fine-grained music style transfer with just one Transformer VAE," *arXiv preprint arXiv:2105.04090*, 2021.

[33] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, J. Z. (Jake), and G. Xia, "Pianotree VAE: Structured representation learning for polyphonic music," in *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[34] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[35] K. Sohn, X. Yan, and H. Lee, "Learning structured output representation using deep conditional generative models," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015.

[36] Y. Li and S. Mandt, "Disentangled sequential autoencoder," in *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[37] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015.

[38] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, 1989.

[39] Z. Hu, Z. Yang, X. Liang, R. Salakhutdinov, and E. P. Xing, "Toward controlled generation of text," in *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[40] W. Goebl, "Melody lead in piano performance: Expressive device or artifact?" *The Journal of the Acoustical Society of America*, vol. 110, no. 1, 2001.

[41] M. Grachten and G. Widmer, "Linear basis models for prediction and analysis of musical expression," *Journal of New Music Research*, vol. 41, no. 4, pp. 311–322, 2012.

[42] Z. Shi, "Computational analysis and modeling of expressive timing in Chopin Mazurkas," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, 2021.

[43] F. Foscarin, A. McLeod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: A dataset of aligned scores and performances for piano transcription," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 2020.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[45] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "β-VAE: Learning basic visual concepts with a constrained variational framework," in *Proceedings of the 5th International Conference on Learning Representations*, 2017.

EXPLOITING DEVICE AND AUDIO DATA TO TAG MUSIC WITH USER-AWARE LISTENING CONTEXTS

Karim M. Ibrahim[1,2] **Elena V. Epure**[2] **Geoffroy Peeters**[1] **Gaël Richard**[1]
[1] LTCI, Télécom Paris, Institut Polytechnique de Paris
[2] Deezer Research
karim.ibrahim@telecom-paris.fr

ABSTRACT

As music has become more available especially on music streaming platforms, people have started to have distinct preferences to fit to their varying listening situations, also known as context. Hence, there has been a growing interest in considering the user's situation when recommending music to users. Previous works have proposed user-aware autotaggers to infer situation-related tags from music content and user's global listening preferences. However, in a practical music retrieval system, the autotagger could be only used by assuming that the context class is explicitly provided by the user. In this work, for designing a fully automatised music retrieval system, we propose to disambiguate the user's listening information from their stream data. Namely, we propose a system which can generate a situational playlist for a user at a certain time 1) by leveraging user-aware music autotaggers, and 2) by automatically inferring the user's situation from stream data (e.g. device, network) and user's general profile information (e.g. age). Experiments show that such a context-aware personalized music retrieval system is feasible, but the performance decreases in the case of new users, new tracks or when the number of context classes increases.

1. INTRODUCTION

Since the invention of recorded music, people have been shifting from consuming music as a main activity in a live setting, to using music as a background activity as they go through the day. With the growing availability of music on streaming platforms, people developed distinct preferences for the varying listening situations, also known as context [1]. Consequently, there has been a growing interest in considering the user's situation when automatically recommending music to users.

Previous works have proposed user-aware autotaggers to infer situation-related tags from music content and user's global listening preferences [2]. However, in a practical music retrieval system, the autotagger could be only used by assuming that the context class is explicitly provided by the user. In this work, we perform a study to evaluate the feasibility of inferring the listening situation. The listening situation for our system is an activity, location, or time that is influencing the listener's preferences.

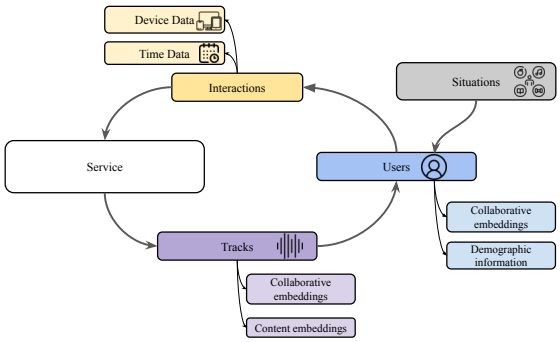


Figure 1. The available data to online music streaming services.

The process of music streaming from the perspective of our proposed approach can be found in Figure 1. We find that the music service is informed of the users, their track history, plus their past and current interactions with the service, i.e. the device and time data sent during an active session. However, the service is unaware of the influencing listening situation. Our goal is to utilize the available information for the service to infer the listening situation and the suitable tracks for the inferred situation. We propose an approach that infers the potential context from the user interactions in near real time, while the tagging of tracks with their potential listening situation happens in the background using autotaggers. Both systems are user-aware.

Our contributions in this paper are: 1) a large dataset of tracks, device data, and user embeddings labeled with their situational use through a rigorous labelling pipeline; 2) an extended evaluation of music autotaggers in predicting personalized situational tags in various scenarios; 3) a simple, yet effective model that ranks the potential listening situations for a given user based on the transmitted data from the device to the service.

2. RELATED WORK

Our proposed approach is related to two different problems: music autotagging with contextual tags, and instant

© Karim M. Ibrahim, Elena V. Epure, Geoffroy Peeters, Gaël Richard. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Karim M. Ibrahim, Elena V. Epure, Geoffroy Peeters, Gaël Richard, "Exploiting device and audio data to tag music with user-aware listening contexts", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

prediction of the user's listening situation. Previous work has already showed that listening situation (i.e. context) has a strong influence on the user's preferences [1, 3–5]. Hence, context has become an important factor for reaching a personalized user experience [6].

On one hand, music content is highly complex and is often challenging to be analyzed and described in human readable terms. This missing link between the content of the music and a set of semantic descriptors is referred to as the "semantic gap" [7]. One common way, which is often used when searching for or organising music, is the intended listening situation [8]. Unlike most tags that depend solely on the music content, certain tags depend also on the user [9, 10]. There has been a recent work on predicting personalized situation-related tags from music content and user embeddings [2], which we adopt here too.

On the other hand, the listening situation, e.g. activity or location, can change frequently, which leads to changes in user preferences. Explicitly inferring the listening situation is a challenging task that has only been studied on a small scale [11]. We aim at addressing this missing link by performing an extensive study on predicting the listening situation using available device data. In order to employ the personalized autotagging approach in an actual real-world setting, it is also important to be able to predict when a specific listening situation is being experienced.

3. OBJECTIVE AND PROPOSED APPROACH

A *session* consists of a sequence of audio-tracks a a given user u is listening to over time t on a music streaming service in a continuous time span [1]. A session is therefore defined as a sequence of *streams* which are each a tuple (audio-track a, user u, device data $\mathbf{d}_u^{(t)}$).

A *situational (or contextual) session* is a session resulting from listening to tracks in a certain situation (or context) c such as "gym". However, in our case, in order to gather a ground-truth dataset, we consider that a situational session can also result from listening to a playlist that contains a context-related keyword in the title [2]. A situational (or contextual) session is defined as a sequence of tuples (audio-track a, user u, device data $\mathbf{d}_u^{(t)}$, situation c).

Our objectives is to propose for a given user u a session (sequence of audio-tracks a) that fits their current situation c. However, since we don't know its current situation c we estimate it based on its device data $\mathbf{d}_u^{(t)}$ (such as the time of the day, day of the week, or type of network connections).

3.1 Proposed approach

To do so, we first estimate for each pair audio-track/user (a, u) its situation $\hat{c}_{a,u}$. In other words, we estimate in which situation c the user u would intend to use the track a. This is done using an Audio+User based Situation Prediction (`auSP`) trained to estimate situation tags c given as input a pair (audio-track a, user u). This is done offline on the server side and stored in a database.

We then estimate in real-time (with a lightweight model on the client side) for a given user u and the transmitted data from its device to the service $\mathbf{d}_u^{(t)}$, its potential current situation $\hat{c}_{d,u}$. This is done using a Device+User based Situation Prediction (`duSP`) trained to estimate situation tags c given as input a pair (device-data $\mathbf{d}_u^{(t)}$, user u). `duSP` provides us with a list of the most likely situations $\hat{c}_{d,u}$ (ranked from the most to the less likely).

Finally, to create the situational playlist, we simply select the audio tracks a for which situation $\hat{c}_{a,u}$ matches the most-likely current situations of the user $\hat{c}_{d,u}$.

Figure 2 indicates the overall architecture.

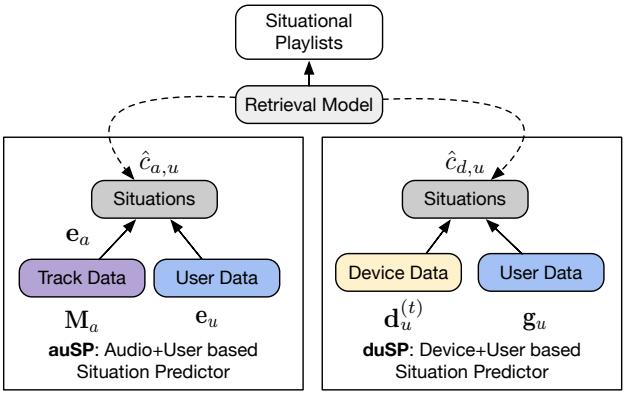


Figure 2. Overview of the system to generate a situational playlist. The left side (`auSP`) tags each track/user pair with a situational tag. The right side (`duSP`) ranks the potential situations for a device/user pair to be presented to the user.

3.2 Data description

We first describe what are exactly the data for the tracks a, the users u and the devices.

Track data $\mathbf{M}_a$. For each audio-track a, we retrieve its 30 s. snippet from the Deezer API. We represent a by its 96 Mel-bands $\times$ 646 frames matrix $\mathbf{M}_a$.

User data $\mathbf{e_u}$ and $\mathbf{g}_u$. Representing the users can be achieved through various versatile techniques. Consistent with our requirements (lightweight model and preserving privacy), we choose to represent the users using the basic data available during streaming. We use two different representations of the user that will be used for estimating $\hat{c}_{a,u}$ and $\hat{c}_{d,u}$ respectively.

For the `auSP` (estimation of $\hat{c}_{a,u}$) we use a user embedding $\mathbf{e_u}$. Similar to previous works on `auSP`, we used the users' listening history to derive user embeddings that encode their listening preferences. We compute these embeddings through matrix factorization of the user/track interactions matrix, leading to a 128-d embedding vector per user, which in commonly used to generate representations [13]. The constructed matrix uses all the tracks available in the catalogue to model the user preferences, i.e. it

[1] without any break longer than a pre-defined gap, defined here as 20 minutes as proposed by [5]

[2] The underlying assumption is that if the user started streaming a playlist with a certain title related to a situation (or context), most likely the intention of the user was to play something suitable for that situation (or context) [12].

Table 1. Summary of the notations

Symbol	Definition	Dimension
a	an audio track	
$\mathbf{M}_a$	Mel-spectrogram of a	$\mathbb{R}^{96\times646}$
$\mathbf{e}_a$	Embedding of a	$\mathbb{R}^{256}$
u	a user	
$\mathbf{e}_u$	Embedding of u	$\mathbb{R}^{128}$
$\mathbf{g}_u$	Demographics data of u	$\mathbb{R}^3$
$\mathbf{d}_u^{(t)}$	Device data of u at time t	$\mathbb{R}^8$
c	a situation (or context)	
s	a stream, a tuple $(a, u, \mathbf{d}_u^{(t)}, c)$	

is not computed exclusively with the tracks included in our dataset. The computed embeddings will be published with the dataset for reproducibility.

For the `duSP` (estimation of $\hat{c}_{d,u}$), we use the basic demographic data $\mathbf{g}_u$ of the user recorded during registration. This data is composed of: `|age,country,gender|`. While this data selection is prone to errors and short of fully representing the users, it is consistent with our requirements of using basic always-available data.

Device data $\mathbf{d}_u^{(t)}$. We collect only basic data sent by the device to the service and selected those that deemed relevant to the situation prediction. The data are: the time stamp (in local time), day of the week, device used and network used. Additionally, we extend the time/day data with circular representation of the time and day similar to the one used in [14]. The final feature vector representing device data is made of 8 features: `linear-time, linear-day, circular-time-X, circular-time-Y, circular-day-X, circular-day-Y, device-type, network-type`. The `device-type` can be: `mobile`, `desktop` (e.g. a laptop), or `tablet`. The `network-type` can be: `mobile` (a connection through cellular data), `wifi` (a WiFi connection), `LAN` (a connection through wired Ethernet), or `plane` (an offline stream from a device without a connection).

4. COLLECTING THE DATA

Pichl et al. and Ibrahim et al. proposed methods for labelling streaming sessions with a situational tag by leveraging playlist titles [2, 12]. Although sometimes prone to errors and false positives, playlist titles provide an appropriate proxy for labelling streams with tags [2, 12]. Users create playlists with a common theme according to their use [12]. One common theme of these playlists is the listening situation.

First, we collected a set of situational keywords used previously in the literature [1, 11, 15]. We extended these keywords by adding synonyms and hashtags that are frequently used on Twitter to refer to music listening. Afterwards, we retrieve from all public playlists from Deezer [3], an online music streaming service we were given access to, those playlists that include any of the keywords in their "stemmed" title. We then filtered out playlists that contained more than 100 tracks [4] or where a single artist or album represented more than 25% of the playlist, similar to [2].

From the extensive list of situational keywords and their corresponding playlists, we settled on three different subsets with an increasing number C of situational tags (4, 8, and 12): `work, gym, party, sleep | morning, run, night, dance | car, train, relax, club`.

These tags were selected by popularity [5]. We used these situations as independent tags without attempting to merge potentially similar activities and places (e.g. "party" and "dance"). Working with three situational tag sets (of increasing C) allowed us to observe how the system performs as the complexity of the problem increases.

We then focused on the users who actively listened to these playlists and retrieve the device data of those users while they were actively listening to the playlist. This resulted in a set of streams each described by an audio-track a, a user u, a playlist with a situational keyword c in the title, along with the device data $\mathbf{d}_u^{(t)}$ sent during this stream. Note that an audio-track/user/device triplet have a joint tag, none of them are tagged individually.

To ensure high quality data, we selected the month of August 2019 for inspection, because this period had more stable use patterns, before the Covid-19 pandemic. We had access to data from two locations: France and Brazil. These two locations were provided because they have the most active users in Deezer, while being in two distinctive time zones and seasons. This allowed us to perform our study on diverse data with different sources and patterns. We release the dataset [6] along with the code [7].

4.1 Dataset Analysis

As a sanity check on the collected data, we plot the distribution of the situations c across the different device-data. Figure 3 shows the ratio of the used `network-type` to connect to the service across situations c. We observe variations that correspond to what is expected from each situation, i.e. outdoors vs. indoors. However, we also find certain networks that do not match the expectations, e.g. LAN connections in a car situation, which represents noise in the dataset that can be a continuation of already existing sessions that moved indoors. Figure 4 represents the used `device-type` across situations c. We notice that most users overwhelmingly use mobile device in most cases, with small variations that also match expectations of indoor and outdoor situations. Finally, Figure 5 shows the distribution of all situations for each hour of the day. Similarly, we find predictable patterns for each situation ranging from night-related situation in the early hours that gradually progress as the time passes. These patterns support

[3] `www.deezer.com`

[4] to increase the chance that playlists reflect a selection of situation-related tracks, and not randomly added ones

[5] Estimated as the number of corresponding playlists in the service catalogue.

[6] `https://zenodo.org/record/5552288`

[7] `https://github.com/Karimmibrahim/Situational_Session_Generator`

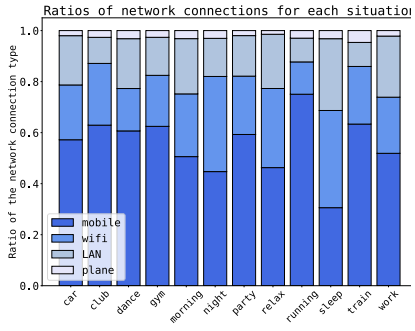


Figure 3. Network across situations c

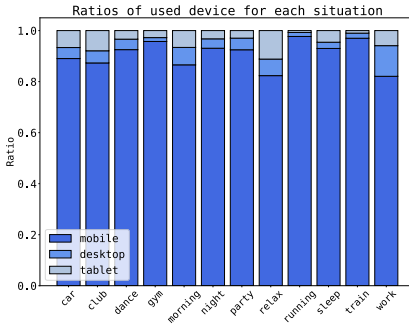


Figure 4. Device across situations c

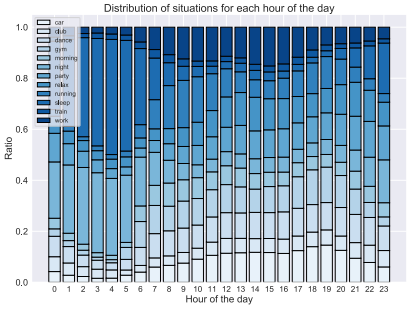


Figure 5. Distributions of situations c over hours of the day

the hypothesis of using playlist titles as proxy for inferring the actual listening situation.

5. DETAILED MODELS DESCRIPTION

5.1 Audio+User based Situation Prediction (`auSP`)

The `auSP` estimates the probability of each situation $c \in \{1 \dots C\}$ given a pair (track a represented by $\mathbf{M}_a$, user u represented by e_u): $P(c|\mathbf{M}_a, \mathbf{e_u})$. It is implemented as a Deep Neural Network $\hat{c}_{a,u} = f_\theta(\mathbf{M}_a, \mathbf{e}_u)$ with softmax output and trainable parameters θ. To train it we use the set of training streams represented as tuples $(\mathbf{M}_a, \mathbf{e}_u, c)$ We train it by minimizing the categorical cross-entropy $\mathcal{L}(\hat{\mathbf{c}}_{a,u}, \mathbf{c}, \theta)$ where $\hat{\mathbf{c}}_{a,u}$ is the estimated probability and $\mathbf{c}$ the one-hot-encoded ground-truth.

Practical implementation. The **audio input**, $\mathbf{M}_a$, is passed to a batch normalization layer then to 4 layers each made of a convolutional (CNN) and a Max-pooling operation. The CNNs have various numbers of filters (32, 64, 128, 256) but each with the same size (3x3). They are each followed by a ReLU. All Max-poolings are (2x2). The flattened output of the last CNN layer is passed to a fully connected (FC) layer with 256 units followed by a ReLU. The output of the audio branch $\mathbf{e}_a$ is a 256-d audio embedding vector $\mathbf{e_a}$. The **user input**, $\mathbf{e}_u$, is processed through 2 FC layers each with a ReLU. This output is then concatenated with $\mathbf{e}_a$ and passed to a FC layer with ReLU activation, and a dropout (with 0.3 ratio) for regularization. The final layer is made of C output units with a Softmax activation function, where C is the number of situations to be predicted. We train the model until convergence by minimizing the categorical cross entropy, optimized with Adam [16] and a learning rate initialized to 0.1 with an exponential decay every 1000 iterations.

5.2 Device+User based Situation Prediction (`duSP`)

The `duSP` estimates the probability of each situation $c \in \{1 \dots C\}$ given a pair (device-data d represented by $\mathbf{d}_u^{(t)}$, user u represented by $\mathbf{g}_u$): $P(c|\mathbf{d}_u^{(t)}, \mathbf{g}_u)$. It is implemented as a function $f_\gamma(\mathbf{d}_u^{(t)}, \mathbf{g_u})$ with Softmax output and trainable parameters γ. To train it we use the set of training streams represented as tuples $(\mathbf{g}_u, \mathbf{d}_u^{(t)}, c)$

Practical implementation. In choosing a real-time "light" `duSP` model, we prioritize the computational complexity requirements over accuracy. The low dimensional input features (11-d = 8 device features + 3 demographic features) already provide a strong case for the investigated models. For our implementation, we experimented with different classifiers: Decision Trees, K-Nearest Neighbors, and eXtreme Gradient Boosting (XGBoost) [17]. While all gave comparable results, we chose XGBoost for its consistent performance across splits and different evaluation scenarios. Similar to the autotagger model, the output predictions depend on the number C of situations in the dataset.

6. EVALUATION

We evaluate here the performance of our system which aims at proposing for a given user u a session (sequence of audio-tracks a) that fits their current situation c. For this, we first evaluate the performance of the two branches of our system (`auSP` and `duSP`) to correctly estimate the situation c. We evaluate this using various numbers of situations: $C \in \{4, 8, 12\}$. To evaluate the `auSP`, we use the common AUC (Area Under Roc Curve) and Accuracy performance measures. To evaluate the `duSP`, we use the Accuracy but also the Accuracy@K. This measures the capability of `duSP` to include the correct situation in the top K predictions. We then evaluate the global system by measuring the overlap of correct predictions between the `auSP` and `duSP` branches. This accuracy can be interpreted as the ratio of existing streams that would have occurred in these sessions if the playlists were generated with this system instead.

6.1 Scenario

We approach the evaluation of this system from two different perspectives: 1) evaluating the system on its capability of learning and generalizing, 2) evaluating the proposed system in a stable use-case with frequent users/tracks.

We simulate these scenarios through a different split criteria for the test-set. Let the full set of streams in our collected dataset S, where each stream s has a user u and a track a. We will be referring to the training-set as S_{train}, the test-set as S_{test}, the set of unique users in training and

testing as U_{train} and U_{test} respectively, and similarly the unique audio-tracks in the splits as A_{train} and A_{test}.

To evaluate the system intelligence and fit to the data, we restrict the evaluation splits to include either: 1) new users (cold-user case): $S_{test} = \{s|u \notin U_{train}, a \in A_{train}\}$, 2) new tracks (cold-track case): $S_{test} = \{s|u \in U_{train}, a \notin A_{train}\}$. We exclude the specific case of both new tracks and new users because splitting the data with only new user/track pairs in the testset is difficult and rare to find. Additionally, recommending a new track to a new user is not a common nor practical scenario to use for evaluating a system.

To evaluate how the system would perform in a regular use-case (warm case): $S_{test} = \{s|u \in U_{train}, a \in A_{train}, s \notin S_{train}\}$. The regular use-case does not restrict the system to neither new users nor tracks. However, the test-set contains exclusively new streams, i.e. (user/track) pairs, not present in the training-set. The evaluation of this regular use-case is relatively complex and includes several entwined evaluation criteria. The goal is to compare the overlap of generated sessions with groundtruth sessions.

6.1.1 auSP Evaluation

The results for the auSP can be found in Table 2.

As shown, the model can reach satisfying performance relative to the evaluation scenario. In terms of AUC, the model's fit for both new users and tracks in the cold user/track splits is not significantly impaired compared to the warm case. The performance decreases evidently as the problem gets harder with more situations C to tag, though in some cases it increases given the increase of dataset size from additional situations. In terms of accuracy, the model's performance in the intended use-case, i.e. warm case, is satisfying (Accuracy above 70). That is to say, the system can correctly tag around two thirds of the user/track listen streams with their correct situational use, when it has seen the user or the track before, but not jointly.

Note that this accuracy was computed by selecting the most probable situation from the predictions. While the high values of AUC (above 0.94) suggest a threshold optimization is needed for each class, in real use-case we do not necessarily need a threshold. The prediction probability could be used directly to retrieve tracks, e.g. by ranking tracks with the prediction probabilities and include top ranked tracks in the generated sessions. However, this max-probability threshold is needed for further evaluations with the situation predictor and with the sequential retrieval model.

Additionally, Figure 6 represents the confusion matrix obtained in the $C = 12$ and warm case.

6.1.2 duSP Evaluation

The results for the duSP can be found in Table 3. We find that predicting the situation for new users becomes noticeably harder. In the case of C=12 situations, the system was able to correctly predict the situation for only 25% of the streams. However, when the system is allowed to make multiple guesses (Accuracy@3), the accuracy evidently increases. In the case where the user is to make the last decision, the system is able to include the correct situation in the top 3 suggestions 96%, 80%, and 68% in the cases of C =4, 8, and 12 situations respectively. The choice of K, when evaluating with accuracy@K, can be obviously changed, and the performance will increase as K increases. We choose to display the results for K=3 since 3 is around the number of visible items in the carousels displayed by most streaming services on the suggestions screen on mobile devices.

Additionally, Figure 7 shows the confusion matrix obtained in the C=12 situations and warm case. We observe that the confusion is mostly coherent with the statistic shown earlier of the distribution of situations with the device data. Situations that are likely to originate with similar device data are harder to discriminate than the rest. For example, we observe a cluster of night-related situations including night, sleep, and relax situations. Similarly, outdoors situation are also often confused together. Discriminating those situations is hindered by the limited data available. However, the convenience of recommending top k situations provides as easy solution to further discriminate between these similar situations.

Table 2. Results of the auSP evaluated with AUC and Accuracy in the three evaluation protocol splits (cold-user, cold-track, and warm case) and the three subsets of situations (4, 8, and 12). The results are shown as mean(std.).

C	AUC		
	Cold User	Cold Track	Warm
4	0.889 (.009)	0.873 (.013)	0.959 (.013)
8	0.815 (.005)	0.866 (.007)	0.945 (.007)
12	0.852 (.004)	0.824 (.012)	0.941 (.012)
C	Accuracy		
	Cold User	Cold Track	Warm
4	69.72 (1.07)	63.77 (2.33)	83.75 (2.33)
8	47.56 (0.53)	52.44 (2.31)	70.81 (1.45)
12	52.68 (1.25)	37.61 (3.47)	69.14 (3.79)

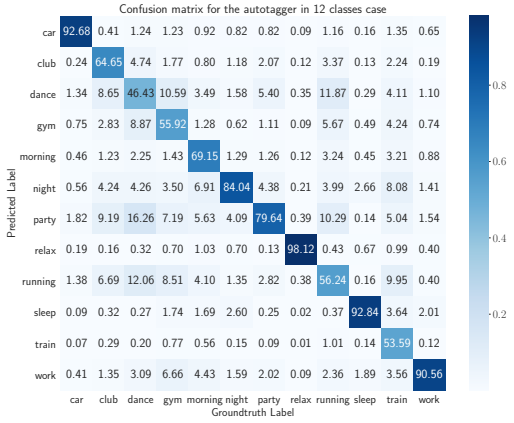


Figure 6. Confusion Matrix of the auSP in the case C=12 and warm case

Table 3. Results of the duSP evaluated with Accuracy and Accuracy@3 in the three evaluation protocol splits (cold-user, cold-track, and warm case) and the three subsets of situations (4, 8, and 12). The results are shown as mean(std.).

C	Accuracy		Accuracy @3	
	Cold User	Warm	Cold User	Warm
4	47.46 (0.98)	66.96 (0.39)	90.51 (0.31)	96.3 (0.1)
8	30.95 (0.89)	49.23 (0.16)	64.11 (1.42)	79.62 (0.13)
12	25.00 (0.29)	39.92 (0.13)	52.04 (0.61)	67.62 (0.21)

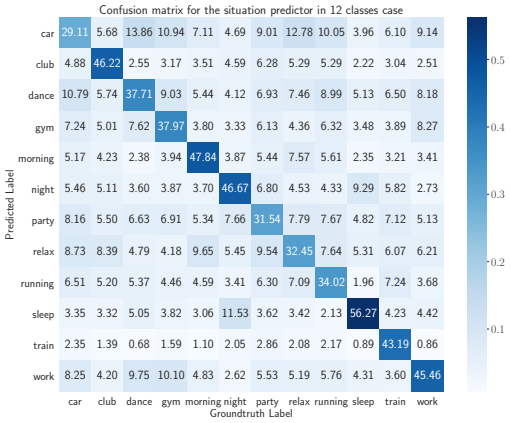


Figure 7. Confusion Matrix of the duSP with C=12 and warm case

Finally, to evaluate the challenge in classifying situations from multiple sources, we compare between the evaluation results in each location (France, Brazil) separately. We compare between two different cases: 1) a model trained globally on the data from both locations but tested locally, 2) a model trained locally on each location independently and tested on the corresponding location. Table 4 shows the results for this evaluation setting. We find that training the models locally slightly improves the results, but not significantly. This suggests that using a single unique model for all locations gives comparable results to using multiple local models. We also observe a clear distinction in the accuracy between the two locations, where Brazil scores higher than France in all cases. This is due to the larger number of users in our dataset who are in France, i.e. there are more users with more distinct patterns in the France case.

Table 4. Evaluation results of the globally and locally trained models for each of the two locations in our dataset, France and Brazil, evaluated with accuracy at each subset of situations in the warm case. The results are shown as mean(std.).

C	France		Brazil	
	Global	Local	Global	Local
4	53.4 (0.2)	55.1 (0.2)	59.2 (0.2)	61.7 (0.2)
8	35.8 (0.1)	36.8 (0.1)	45.9 (0.1)	48.2 (0.1)
12	27.6 (0.1)	28.2 (0.1)	39.6 (0.2)	41.8 (0.1)

Table 5. The joint evaluation results of the auSP and duSP and their overlapping predictions evaluated with Accuracy in the three evaluation protocol splits (cold-user, cold-track, and warm case) and the three subsets of situations (4, 8, and 12). The results are shown as mean(std.).

Model	Cold Users	Cold Tracks	Warm Case
	4 Situations		
auSP	69.73 (1.07)	63.78 (2.33)	83.75 (2.33)
duSP	47.46 (0.98)	66.81 (0.35)	67.20 (0.26)
Overlap	36.22 (1.27)	44.60 (1.01)	**58.92** (1.71)
	8 Situations		
auSP	47.56 (0.53)	52.44 (2.31)	70.81 (1.45)
duSP	30.95 (0.89)	49.13 (0.24)	49.35 (0.19)
Overlap	17.77 (0.49)	28.94 (1.24)	39.52 (1.27)
	12 Situations		
auSP	52.68 (1.25)	37.61 (3.47)	69.14 (3.79)
duSP	25.00 (0.29)	39.05 (0.31)	39.19 (0.14)
Overlap	16.19 (0.32)	18.75 (1.63)	31.26 (1.30)

6.1.3 Joint Evaluation

The results for the joint system can be found in Table 5. As we can see, each variable in our evaluation influences the performance of the system. The most influential parameter is the number C of potential situations. As the complexity increases, we find the accuracy of the model decreasing: from 58% in the case C=4 with no new users or tracks to 16% in the case C=12 with cold scenarios. Additionally, we find the expected variation in performance between the cold cases and the warm case of intended use. We observe how the drop in the performance of the auSP and duSP, on new users/tracks, negatively affects the joint system performance.

However, in the harder evaluation case of generating a situational playlists with only 1 guess allowed out of C=12, the proposed system would have been able to include at least a third of the actual listened tracks (31.26%) in those playlists, while pushing them to the user at the exact listened time.

7. CONCLUSION

In this study, we address the problem of the unobserved listening situation which influences the users' preferences. We proposed a two-branch framework to predict when a situation is being experienced based on the device data, while simultaneously autotagging the music tracks with their intended listening situation in a personalized manner. Through the proposed approach, users could access a set of predicted potential situations. These situations are also associated with a set of tracks "likely" to be listened to by the user. This likelihood is estimated using an autotagger trained on predicting the situational use of tracks, given a specific user and his/her listening history. We evaluated each of our system's blocks individually and combined. The evaluation results indicated that the system is capable of learning personalized patterns for users, which can be employed to provide contextual music recommendation.

8. REFERENCES

[1] A. C. North and D. J. Hargreaves, "Situational influences on reported musical preference." *Psychomusicology: A Journal of Research in Music Cognition*, vol. 15, no. 1-2, p. 30, 1996.

[2] K. Ibrahim, E. Epure, G. Peeters, and G. Richard, "Should we consider the users in contextual music auto-tagging models?" in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[3] A. E. Greasley and A. Lamont, "Exploring engagement with music in everyday life using experience sampling methodology," *Musicae Scientiae*, vol. 15, no. 1, pp. 45–71, 2011.

[4] M. Gorgoglione, U. Panniello, and A. Tuzhilin, "The effect of context-aware recommendations on customer purchasing behavior and trust," in *Proceedings of the fifth ACM conference on Recommender systems*, 2011, pp. 85–92.

[5] C. Hansen, C. Hansen, L. Maystre, R. Mehrotra, B. Brost, F. Tomasi, and M. Lalmas, "Contextual and sequential user embeddings for large-scale music recommendation," in *Fourteenth ACM Conference on Recommender Systems*, 2020, pp. 53–62.

[6] M. Kaminskas and F. Ricci, "Contextual music information retrieval and recommendation: State of the art and challenges," *Computer Science Review*, vol. 6, no. 2-3, pp. 89–119, 2012.

[7] Ò. Celma and X. Serra, "Foafing the music: Bridging the semantic gap in music recommendation," *Journal of Web Semantics*, vol. 6, no. 4, pp. 250–256, 2008.

[8] K. Ibrahim, J. Royo-Letelier, E. Epure, G. Peeters, and G. Richard, "Audio-based auto-tagging with contextual tags for music," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.

[9] M. Schedl, A. Flexer, and J. Urbano, "The neglected user in music information retrieval research," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 523–539, 2013.

[10] F. Korzeniowski, O. Nieto, M. McCallum, M. Won, S. Oramas, and E. Schmidt, "Mood classification using listening data," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR*, 2020.

[11] X. Wang, D. Rosenblum, and Y. Wang, "Context-aware mobile music recommendation for daily activities," in *Proceedings of the 20th ACM international conference on Multimedia*, 2012, pp. 99–108.

[12] M. Pichl, E. Zangerle, and G. Specht, "Towards a context-aware music recommendation approach: What is hidden in the playlist name?" in *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*. IEEE, 2015, pp. 1360–1365.

[13] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[14] P. Herrera, Z. Resa, and M. Sordo, "Rocking around the clock eight days a week: an exploration of temporal patterns of music listening," in *Proceedings of the 1st Workshop On Music Recommendation And Discovery (WOMRAD), ACM RecSys, 2010, Barcelona, Spain*, 2010.

[15] M. Gillhofer and M. Schedl, "Iron maiden while jogging, debussy for dinner?" in *MultiMedia Modeling*, X. He, S. Luo, D. Tao, C. Xu, J. Yang, and M. A. Hasan, Eds. Cham: Springer International Publishing, 2015, pp. 380–391.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[17] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, 2016, pp. 785–794.

JUKEDRUMMER: CONDITIONAL BEAT-AWARE AUDIO-DOMAIN DRUM ACCOMPANIMENT GENERATION VIA TRANSFORMER VQ-VAE

Yueh-Kao Wu
Academia Sinica
yk.lego09@gmail.com

Ching-Yu Chiu
National Cheng Kung University
x2009971@gmail.com

Yi-Hsuan Yang
Taiwan AI Labs
yhyang@ailabs.tw

ABSTRACT

This paper proposes a model that generates a drum track in the audio domain to play along to a user-provided drum-free recording. Specifically, using paired data of drumless tracks and the corresponding human-made drum tracks, we train a Transformer model to improvise the drum part of an unseen drumless recording. We combine two approaches to encode the input audio. First, we train a vector-quantized variational autoencoder (VQ-VAE) to represent the input audio with discrete codes, which can then be readily used in a Transformer. Second, using an audio-domain beat tracking model, we compute beat-related features of the input audio and use them as embeddings in the Transformer. Instead of generating the drum track directly as waveforms, we use a separate VQ-VAE to encode the mel-spectrogram of a drum track into another set of discrete codes, and train the Transformer to predict the sequence of drum-related discrete codes. The output codes are then converted to a mel-spectrogram with a decoder, and then to the waveform with a vocoder. We report both objective and subjective evaluations of variants of the proposed model, demonstrating that the model with beat information generates drum accompaniment that is rhythmically and stylistically consistent with the input audio.

1. INTRODUCTION

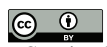
Deep generative models for musical audio generation have witnessed great progress in recent years [1–8]. While models for generating symbolic music such as MIDI [9–12] or musical scores [13] focus primarily on the *composition* of musical content, an audio-domain music generation model deals with *sounds* and thereby has extra complexities related to timbre and audio quality. For example, while a model for generating symbolic guitar tabs can simply consider a guitar tab as a sequence of notes [11], a model that generates audio recordings of guitar needs to determine not only the underlying sequence of notes but also the way to render (synthesize) the notes into sounds. Due to the complexities involved, research on deep generative models for musical audio begins with the simpler task of synthesizing individual musical notes [1–3], dispensing the need to consider the composition of notes. Follow-up research [4–6] extends the capability to generating musical passages of a single instrument. The Jukebox model [7] proposed by OpenAI greatly advances the state-of-the-art by being able to, quoting their sentence, "generate high-fidelity and diverse songs with coherence up to multiple minutes." Being trained on a massive collection of audio recordings with the corresponding lyrics but not the symbolic transcriptions of music, Jukebox generates multi-instrument music as raw waveforms directly without an explicit model of the underlying sequence of notes.

This work aims to improve upon Jukebox in two aspects. First, the backbone of Jukebox is a hundred-layer Transformer [14, 15] with billions of parameters that are trained with 1.2 million songs on hundreds of NVIDIA V100 GPUs for weeks at OpenAI, which is hard to reproduce elsewhere. Inspired by a recent Jukebox-like model for singing voice generation called KaraSinger [8], we instead build a light-weight model with only 25 million parameters by working on Mel-spectrograms instead of raw waveforms. Our model is trained with only 457 recordings on a single GeForce GTX 1080 Ti GPU for 2 days.

Second, and more importantly, instead of a fully autonomous model that makes a song from scratch with various instruments, we aim to build a model that can work cooperatively with human, allowing the human partner to come up with the musical audio of *some* instruments as input to the model, and generating in return the musical audio of *some other* instruments to accompany and to complement the user input, completing the song together. Such a model can potentially contribute to human-AI co-creation in songwriting [16] and enable new applications.

In technical terms, our work enhances the controllability of the model by allowing its generation to be steered on a user-provided audio track. It can be viewed as an interesting sequence-to-sequence problem where the model creates a "target sequence" of music that is to be played along to the input "source sequence." Besides requirement on audio quality, the coordination between the source and target sequences in terms of musical aspects such as style, rhythm, and harmony is also of central importance.

We note that, for controllability and the intelligibility of the generated singing, both Jukebox [7] and KaraSinger [8] have a lyrics encoder that allows their generation to be steered on textual lyrics. While being technically similar,



© Y.-K. Wu, C.-Y. Chiu, and Y.-H. Yang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Y.-K. Wu, C.-Y. Chiu, and Y.-H. Yang, "JukeDrummer: Conditional Beat-aware Audio-Domain Drum Accompaniment Generation via Transformer VQ-VAE", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

our *accompaniment generation* task ("audio-to-audio") is different from the lyric-conditioned generation task ("text-to-audio") in that the latter does not need to deal with the coordination between two audio recordings.

Specifically, we consider a *drum accompaniment generation* problem in our implementation, using a "drumless" recording as the input and generating as the output a drum track that involves the use of an entire drum kit. We use this as an example task to investigate the audio-domain accompaniment generation problem out of the following reasons. First, datasets used in musical source separation [17] usually consist of an isolated drum stem along with stems corresponding to other instruments. We can therefore easily merge the other stems to create paired data of drumless tracks and drum tracks as training data of our model. (In musical terms, drumless, or "Minus Drums" songs are recordings where the drum part has been taken out, which corresponds nicely to our scenario.) Second, we suppose a drum accompaniment generation model can easily find applications in songwriting [18], as it allows a user (who may not be familiar with drum playing or beat making) to focus on the other non-drum tracks. Third, audio-domain drum accompaniment generation poses interesting challenges as the model needs to determine not only the *drum patterns* but also the *drum sounds* that are supposed to be, respectively, rhythmically and stylistically consistent with the input. Moreover, the generated drum track is expected to follow a steady tempo, which is a basic requirement for a human drummer. We call our model the "JukeDrummer."

As depicted in Figure 1, the proposed model architecture contains an "audio encoder" (instead of the original text encoder [7, 8]) named the *drumless VQ encoder* that takes a drum-free audio as input. Besides, we experiment with different ways to capitalize an audio-domain beat and downbeat tracking model proposed recently [19] in a novel *beat-aware module* that extracts *beat-related information* from the input audio, so that the language model for generation (i.e., the Transformer) is better informed of the rhythmic properties of the input. The specific model [19] was trained on drumless recordings as well, befitting our task. We extract features from different levels, including low-level tracker embeddings, mid-level activation peaks, and high-level beat/downbeat positions, and investigate which one benefits the generation model the most.

Our contribution is four-fold. First, to our best knowledge, this work represents the first attempt to drum accompaniment generation of a full drum kit given drum-free mixed audio. Second, we develop a light-weight audio-to-audio Jukebox variant that takes an input audio of up to 24 seconds as conditioning and generates accompanying music in the domain of Mel-spectrograms (Section 3). Third, we experiment with different beat-related conditions in the context of audio generation (Section 4). Finally, we report objective and subjective evaluations demonstrating the effectiveness of the proposed model (Sections 6 & 7). [1] .

[1] We share our code and checkpoint at: `https://github.com/legoodmanner/jukedrummer`. Moreover, we provide audio examples at the following demo page: `https://legoodmanner.github.io/jukedrummer-demo/`

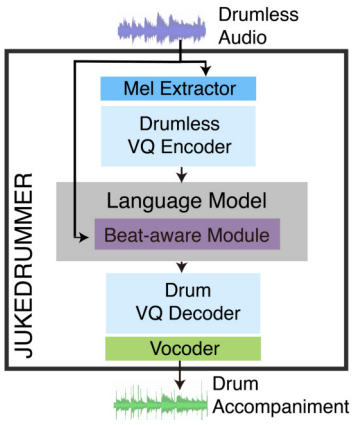


Figure 1: Diagram of the proposed JukeDrummer model for the inference stage. The training stage involves learning additional Drum VQ Encoder and Drumless VQ Decoder (see Figure 2) that are not used at inference time.

2. BACKGROUND

2.1 Related Work on Drum Generation

Conditional drum accompaniment generation has been studied in the literature, but only in the symbolic domain [20, 21], to the best of our knowledge. Dahale *et al.* [20] used a Transformer encoder to generate an accompanying symbolic drum pattern of 12 bars given a four-track, melodic MIDI passage. Makris *et al.* [21] adopted instead a sequence-to-sequence architecture with a bi-directional long short-term memory (BLSTM) encoder extracting information from the melodic input and a Transformer decoder generating the drum track for up to 16 bars in MIDI format. While symbolic-domain music generation has its own challenges, it differs greatly from the audio-domain counterpart studied in this paper, for it is not about generating sounds that can be readily listened to by human.

Related tasks that have been attempted in the literature with deep learning include symbolic-domain generation of a monophonic drum track (i.e., kick drum only) of multiple bars [4], symbolic-domain drum pattern generation [22–25], symbolic-domain drum track generation as part of a multi-track MIDI [26–29], audio-domain one-shot drum hit generation [30–34], audio-domain generation of drum sounds of an entire drum kit of a single bar [35], and audio-domain drum loop generation [36]. Jukebox [7] generates a mixture of sounds that include drums, but not an isolated drum track. By design, Jukebox does not take any input audio as a condition and generate accompaniments.

2.2 The Original Jukebox model

The main architecture of Jukebox [7] is composed of two components: a multi-scale vector-quantized variational autoencoder (VQ-VAE) [37–41] and an autoregressive Transformer decoder [14, 15]. The **VQ-VAE** is for converting a continuous-valued raw audio waveform into a sequence of so-called discrete *VQ codes*, while the **Transformer** establishes a language model (LM) of the VQ codes capable of generating new code sequences.

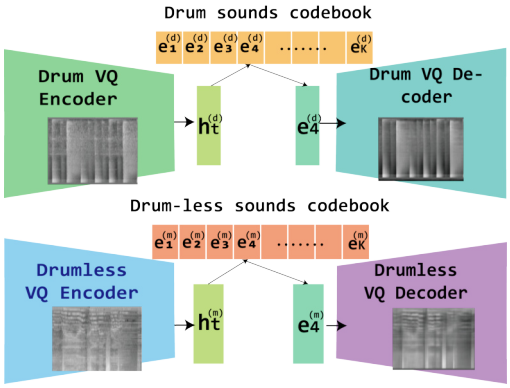


Figure 2: We use separate VQ-VAEs for the drumless and drum tracks, both operating on the Mel-spectrograms.

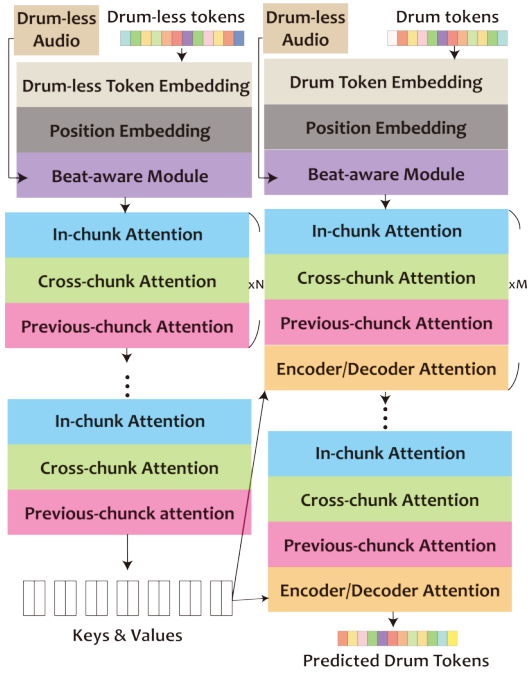


Figure 3: Details of our Transformer encoder/decoder.

Specifically, VQ-VAE consists of an encoder and a decoder, referred to as the VQ-VAE encoder $\mathcal{E}$ and VQ-VAE decoder $\mathcal{D}$ below, respectively. Given an audio waveform $\mathbf{x} \in \mathbb{R}^{1\times N}$, where N denotes the number of audio samples (e.g., N = 1,058,400 for a 24-second audio clip sampled at 44.1 kHz), the VQ-VAE encoder would convert $\mathbf{x}$ into a sequence of latent vectors $\{\mathbf{h}_t \in \mathbb{R}^D, t = 1, ..., T\}$, where the sequence length T is proportional to N, and the VQ-VAE decoder would have to reconstruct $\mathbf{x}$ from the "vector-quantized" version of the latent vectors, denoting as $\{\mathbf{h}'_t \in \mathbb{R}^D, t = 1, ..., T\}$, that is obtained by finding the nearest prototype vector $\mathbf{h}'_t = \mathbf{e}_z$ of each $\mathbf{h}_t$ in a codebook of prototype vectors $\{\mathbf{e}_k \in \mathbb{R}^D, k = 1, ..., K\}$, where K is the size of the codebook. Each prototype vector can be regarded as a cluster centroid in the latent space as a result of K-means clustering. The VQVAE encoder/decoder and the codebook are jointed learned by minimizing the reconstruction loss $\|\mathbf{x}_t - \mathcal{D}(\mathbf{h}'_t)\|_2^2$, and the commitment loss of the clustering $\|\mathcal{E}(\mathbf{x}_t) - sg(\mathbf{e}_z)\|_2^2$, where $\mathbf{x}_t$ denotes a slice of $\mathbf{x}$, $sg(.)$ the stop-gradient operation, and $\mathbf{h}_t = \mathcal{E}(\mathbf{x}_t)$.

Once the VQ-VAE is trained, the $\mathbf{x}$ can be viewed as a sequence of "IDs" $\{z_t \in \mathbb{Z}_{1:K}, t = 1, ..., T\}$, each corresponding to the index of the element of the codebook that is used to represent each slice of $\mathbf{x}$. The Transformer can then be trained on such sequences of IDs to learn the underlying language, or "composition rules," of the audio codes. Once trained, the transformer can be used to generate a novel sequence of codes, which can then be converted into an audio waveform by the VQVAE decoder.

As the waveforms are extremely long, Jukebox actually uses a "multi-scale" VQ-VAE that converts a waveform into three levels of codes, and accordingly three Transformers for building the LM at each level [7]. KaraSinger works on Mel-spectrograms but also uses multi-scale VQ-VAE [8]. We simplify their architecture by working on Mel-spectrograms with only one level of codes instead of multiple levels, as introduced below.

3. PROPOSED METHODS

In our task, we are given pairs of audio waveforms, namely a drumless stem $\mathbf{x}^{\mathrm{m}}$ and a drum stem $\mathbf{x}^{\mathrm{d}}$. Our goal is to train a model that can generate $\mathbf{x}^{\mathrm{d}}_*$ given an unseen $\mathbf{x}^{\mathrm{m}}_*$ from the test set. To achieve so, we propose several extensions of the Jukebox model. While we focus on drum accompaniment generation only, the same methodology may apply equally well to other conditional generation tasks.

Two VQ-VAEs. As illustrated in Figure 2, we build separate VQ-VAEs for $\mathbf{x}^{\mathrm{m}}$ and $\mathbf{x}^{\mathrm{d}}$ using drumless stems and drum stems respectively. Once trained, the drumless VQ-VAE encoder and the drum VQ-VAE encoder would convert $\mathbf{x}^{\mathrm{m}}$ and $\mathbf{x}^{\mathrm{d}}$ into $\{z^{\mathrm{m}}_t \in \mathbb{Z}_{1:K^{\mathrm{m}}}, t = 1, ..., T\}$ and $\{z^{\mathrm{d}}_t \in \mathbb{Z}_{1:K^{\mathrm{d}}}, t = 1, ..., T\}$ separately. Assuming that the drumless tracks are more diverse, we suggest use a larger codebook size for the drumless sounds codebook than the drum sounds codebook, namely $K^{\mathrm{m}} \geq K^{\mathrm{d}}$.

Mel VQ-VAE & Vocoder. To reduce computational cost, we build VQ-VAEs that take Mel-spectrograms as the input and target output. Specifically, we use the convolutional blocks of UNAGAN [5] to build a pair of VQ-VAE encoder and VQ-VAE decoder that has symmetric architectures (one downsampling and the other upsampling). We omit the details of UNAGAN due to space limit. Our Transformer accordingly learns the LM for codes of the Mel-spectrograms. Once a novel sequence of (drum) codes is generated by the Transformer, we use our (drum) VQ-VAE decoder to convert the codes into a Mel-spectrogram, and then use a neural *vocoder* [42] to convert the Mel-spectrogram into the corresponding waveform. While the latent vector $\mathbf{h}_t$ (and accordingly z_t) corresponds to a slice of waveform in the original Jukebox, here the latent vectors $\mathbf{h}^{\mathrm{m}}_t$ and $\mathbf{h}^{\mathrm{d}}_t$ (and z^{m}_t, z^{d}_t) both correspond to a fixed number of L frames of the short-time Fourier Transform (STFT) while computing the Mel-spectrograms.

Seq2seq Transformer. While the Jukebox model uses a Transformer decoder to model the sequences $\{z_t, t = 1, ..., T\}$ corresponding to mixtures of sounds, in our case we need a dedicated Transformer encoder to take the se-

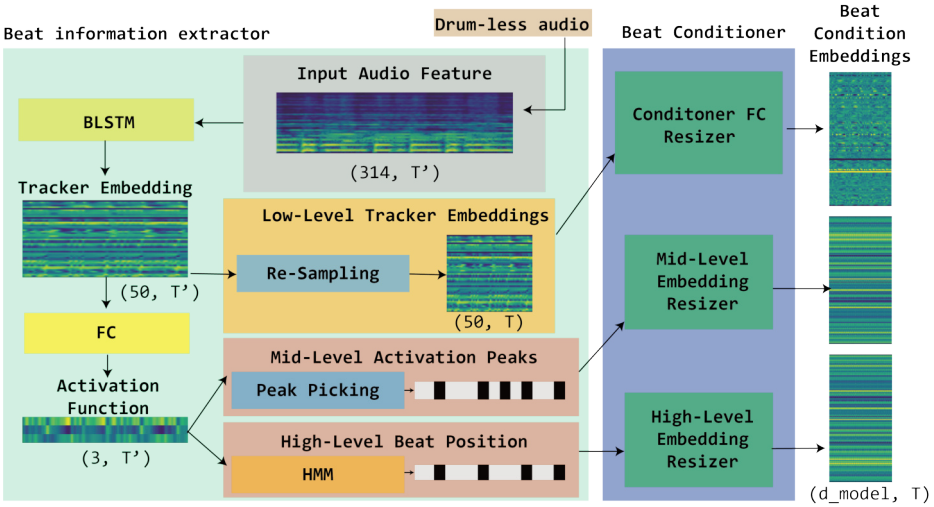


Figure 4: Details of the proposed beat-aware module that extracts beat condition embeddings from the drumless audio.

quence $\{z_t^{\mathrm{m}}, t = 1, ..., T\}$ corresponding to a drumless audio as the input, and a Transformer decoder to generate the sequence $\{z_t^{\mathrm{d}}, t = 1, ..., T\}$ corresponding to the accompanying drum audio as the output, leading to a sequence-to-sequence (seq2seq) architecture. Following Jukebox, we use the attention mechanism of the Scalable Transformer [15] in our Transformer encoder/decoder. As depicted in Figure 3, to determine its output z_i^{d} at each i, our Transformer decoder uses factorized "in-chunk," "cross-chunk" and "previous-chunk" attention layers to attend to the drum codes it generates previously, namely $z_{<i}^{\mathrm{d}} = \{z_t^{\mathrm{d}}, t = 1, ..., i-1\}$, to maintain the coherence of its output. A "chunk" here is a slice of the corresponding sequence. Moreover, the Transformer decoder uses "encoder/decoder attention" to attend to the final layer of the Transformer encoder to get contextual information from the drumless code sequence $\{z_t^{\mathrm{m}}, t = 1, ..., T\}$. To better synchronize the input and output, position embeddings are used. We refer readers to [7, 15] for details.

Beat-Aware Module. Figure 3 also shows that we use a novel "beat-aware module" (after the position embedding) in both Transformer encoder and decoder. We note that the code z_t^{m} is trained to provide essential information needed to reconstruct the drumless audio $\mathbf{x}_t^{\mathrm{m}}$, so the information carried by z_t might be *mixed*, covering different musical aspects including timbre, style, rhythm, etc. To supply the Transformer with *clear* rhythm-related information of $\mathbf{x}^{\mathrm{m}}$, we might need such a dedicated beat-aware module.

As depicted in Figure 4, the beat-aware module consists two sub-modules, the "beat information extractor" and the "beat conditioner." The former extracts beat-related information per frame from $\mathbf{x}^{\mathrm{m}}$ using an existing beat/downbeat tracker [19], while the latter incorporates that beat-related information for each t as a beat condition embedding $\mathbf{c}_t^{\mathrm{m}} \in \mathbb{R}^{d_{\mathrm{model}}}$ that has the same length d_{model} as the token embedding of the Transformers, and adds together the beat condition embedding and token embedding per t to serve as the input to the subsequent attention layers.

Despite that deep learning-based beat/downbeat trackers such as those proposed by Böck *et al.* [43–45] have achieved excellent performance for mainstream pop, rock or dance music, their performance and behaviors are influenced by the sound source composition (i.e., drum/non-drum sounds) of their training data [19, 46]. Considering that our input music is drumless, which is quite different from the music that existing common beat/downbeat trackers [44, 45] are trained with and trained for, we use in our beat information extractor the "non-drum tracker" developed by Chiu *et al.* [19] exclusively for drumless stems. In Section 4, we describe three ways to extract different levels of beat information feature from the non-drum tracker.

4. BEAT CONDITION EMBEDDING

As shown on the left of Figure 4, following the common design [44], the beat/downbeat tracker [19] uses BLSTM layers to get 50-dimensional "tracker embeddings" from $\mathbf{x}^{\mathrm{m}}$, and then uses fully-connected (FC) layers to compute 3-dimensional "activation functions" indicating the likelihood of observing a beat, downbeat, or non-beat at each frame. Finally, either a simple peak-picking or a hidden Markov model (HMM)-based algorithm [44] can be used to finalize the beat and downbeat time positions from the activation functions. We accordingly investigate extracting features from different stages of this processing pipeline.

Low-level (tracker) embeddings. We simply use the high-dimensional tracker embeddings, resampling it temporally, pooling them every L frames, and converting each of them to the desired length d_{model} with an FC.

High-level beat/downbeat positions. From the beat and downbeat time positions estimated by HMM, a frame can be labeled as either a beat, downbeat, or non-beat. We learn three d_{model}-dimensional embedding vectors corresponding to each case, and represent every L frames with one of the three vectors according to the frame labels.

Mid-level activation peaks. As the beat/downbeat positions are sparse over time, we consider a "denser" version by simply picking the peak positions of the activation func-

tions by `scipy.signal.find_peaks` [47] and similarly learning embedding vectors that are assigned to the frames according to whether a frame corresponds to a peak or not. Such a peak might represent a musical onset.

5. EXPERIMENT SETUP

Multi-track datasets for research on musical source separation [17] usually host recordings that each consists of an isolated drum stem along with stems corresponding to other instruments. We can simply take the drum stem as the target drum track, and the summation of the remaining stems as the input drumless track. In our implementation, we used the multi-track recordings from three datasets. MUSDB18 [17] contains 150 recordings, each of which has four stems corresponding to vocal, drums, bass, and "others." It is commonly used in source separation. MedleyDB [48] contains 196 multi-track recordings, each of which includes a drum stem along with many other stems. MixingSecret [49] has 257 multi-track recordings with various instruments, all including drums. We removed the 46 duplicate recordings between MUSDB18 and MedleyDB and the 100 duplicate recordings between MUSDB18 and MixingSerect, leading to 457 recordings to be used in our work. We randomly split the recordings into 80%, 10%, 10% as the training set, validation set (for parameter tuning), and testing set (for objective and subjective evaluation) at the "recording-level," ensuring that a recording does not appear in different splits.

All the recordings are sampled at 44.1 kHz and the stems are all monaural. Following Jukebox [7], we used 24-second audio clips in our work. We sliced each recording (and accordingly the stems) to 23.8-second audio clips with 50% temporal overlaps (as $2^{20}/44100 = 23.8$), and discarded the clips that do not contain any drum sounds. We then computed the Mel-spectrograms of each clip with PyTorch v1.7.1 with a Hann window of 1,024 samples for STFT, a hop size of 256 samples and 80 Mel-filter banks.

To evaluate the performance of the proposed JukeDrummer and validate the effectiveness of model components, we adopted the following variants in our experiments. [2]

- `seq2seq+beat (low)`: given a drumless clip $\mathbf{x}^{\mathrm{m}}$, we computed the drumless codes $\{z_t^{\mathrm{m}}\}$ and the low-level beat/downbeat tracker embeddings $\{\mathbf{c}_t^{\mathrm{m}}\}$ as the model input, to predict the drum codes $\{z_t^{\mathrm{d}}\}$ that eventually lead to the generated drum clip $\mathbf{x}^{\mathrm{d}}$, using the proposed seq2seq Transformer.
- `seq2seq+beat (mid)`: using the beat condition embeddings computed from the mid-level activation peaks (see Section 4) for $\{\mathbf{c}_t^{\mathrm{m}}\}$ instead.
- `seq2seq+beat (high)`: using the beat condition embeddings from the high-level beat/downbeat positions as $\{\mathbf{c}_t^{\mathrm{m}}\}$ instead.
- `seq2seq w/o beat`: to study the usefulness of the beat conditioning, we predicted $\{z_t^{\mathrm{d}}\}$ from $\{z_t^{\mathrm{m}}\}$, not using any beat-related conditions at all.
- `decoder+beat (low)`: to study the usefulness of the drumless codes $\{z_t^{\mathrm{m}}\}$, we used only the low-level beat condition embeddings $\{\mathbf{c}_t^{\mathrm{m}}\}$ as input to predict $\{z_t^{\mathrm{d}}\}$, via a Transformer decoder-only architecture (i.e., not seq2seq Transformer).
- `decoder w/o beat`: this is the baseline drummer that "plays its own," generating $\{z_t^{\mathrm{d}}\}$ autoregressively via a Transformer decoder without taking any information (neither $\{z_t^{\mathrm{m}}\}$ nor $\{\mathbf{c}_t^{\mathrm{m}}\}$) from the drumless clip $\mathbf{x}^{\mathrm{m}}$ it is supposed to play along to.

While the Mel-spectrogram for a clip in our case has 4,096 frames, the VQ-VAE encoder downsamples it to a sequence of $T = 1{,}024$ latent vectors, namely each corresponding to $L = 4$ frames. For VQ-VAE, we set the codebook size of drumless sounds $K^{\mathrm{m}} = 1{,}024$ and that of drums $K^{\mathrm{d}} = 32$ (so $K^{\mathrm{m}} \gg K^{\mathrm{d}}$), and the dimension of the latent vectors $D = 64$. The VQ-VAE encoders/decoders for both drumless and drums all have two layers. For those models employing a seq2seq LM, the Transformer encoder has 9 layers and the Transformer decoder has 20 layers. [3]

At inference time, we used the drum VQ decoder to convert the drum codes $\{z_t^{\mathrm{d}}\}$ to a Mel-spectrogram, which is then turned into the waveform of the drum clip $\mathbf{x}^{\mathrm{d}}$ by a HiFi-GAN V1 vocoder [42]. We trained the vocoder from scratch with audio of drum sounds from our dataset for 2.5 days, and then, inspired by [50, 51], fine-tuned it on the reconstructed Mel-spectrograms of the Drum VQ decoder.

6. OBJECTIVE EVALUATION

As audio-domain drum accompaniment generation is new, we propose customized metrics. Specifically, we use the "drum tracker" trained exclusively for drum stems by Chiu *et al.* [19], which follows exactly the same pipeline as the non-drum tracker (cf. Figure 4), to extract rhythmic features at three different levels for the ground-truth and generated drum sounds for the test split. We then compare the rhythmic features of the ground-truth and the generated in such three levels, using the mean square error for the low-level tracker embeddings (**TrackEmb-MSE**), cross entropy for the mid-level activation functions (**Act-Entropy**), and the F-measure for beat/downbeat estimation (**B/DB-F1**). The last one, computed by `mir_eval` [52], uses the beat positions of the ground-truth drums as the reference and those of the generated drums as the estimated beats. These metrics evaluates only the *rhythmic consistency* between the generated drums and the drumless audio (using its human-made drum track as a proxy), not other aspects such as stylistic consistency and audio quality, which will be evaluated subjectively in Section 7.

[2] A reviewer suggested that we should have compared our model with other accompaniment generation models that operate in the symbolic domain such as [20] and [21], saying that we can use existing audio samples or drum synthesis models to render their output to audio. However, the problem is that such a symbolic-domain accompaniment generation model also requires its input to be a MIDI file rather than audio.

[3] We used Adam as our optimizer with a learning rate of 0.0003 for both VQ-VAE and Transformer LM. We trained our LM with a batch size of 16, input feature dimension $d_{\text{model}} = 512$, two heads for multi-head attention, and the chunk size for factorized attention being 16. For those employing a decoder-only architecture, the number of layers of the Transformer decoder reduces to 15 because 5 of the layers corresponding to the "encoder/decoder attention" are removed.

Model	Objective metrics			Subjective MOS ($\in [1, 5]$)				
	TrackEmd-MSE↓	Act-Entropy↓	B/DB-F1↑	$\mathbf{R}^{m/d}$	$\mathbf{S}^{m/d}$	$\mathbf{Q}^{d}$	$\mathbf{R}^{d}$	$\mathbf{O}^{d}$
`seq2seq+beat (low)`	**.068**±.023	**.928**±.189	**.340**	**3.27**	**3.44**	**3.39**	**3.37**	**3.20**
`seq2seq+beat (mid)`	.077±.022	.963±.200	.212	—	—	—	—	—
`seq2seq+beat (high)`	.080±.024	.938±.160	.132	—	—	—	—	—
`seq2seq w/o beat`	.081±.022	.968±.233	.110	1.78	2.34	2.95	2.58	2.05
`decoder+beat (low)`	**.068**±.023	**.931**±.200	**.339**	**3.05**	**3.34**	**3.33**	**3.07**	**3.17**
`decoder w/o beat`	.087±.025	.987±.240	.114	1.59	1.83	2.56	1.88	1.73
Real data (not vocoded)	—	—	—	4.39	3.95	3.85	4.61	4.17

Table 1: Results of objective and subjective evaluation of variants of the proposed JukeDrummer model. The metrics are (from left to right): beat/downbeat tracking embedding MSE, beat/downbeat activation entropy, beat/downbeat F1, $\mathbf{R}^{m/d}$ (rhythmic consistency), $\mathbf{S}^{m/d}$ (stylistic consistency), $\mathbf{Q}^{d}$ (audio quality), $\mathbf{R}^{d}$ (rhythmic stability), $\mathbf{O}^{d}$ (overall). ↓/↑: the lower/higher the better; best two results (among the six model variants) per column highlighted in bold.

Result shown in the middle of Table 1 clearly shows that the models with low-level beat information achieve lower MSE, cross entropy, and higher F1, suggesting the usefulness of the low-level beat information for rhythmic consistency. The mid-level and high-level ones seem less effective (possibly because they are relatively monotonic), so we did not evaluate them further in Section 7. The objective scores also suggest that the Transformer encoder does not contribute much to rhythmic consistency.

7. SUBJECTIVE EVALUATION

With an online study, we solicited 22 anonymous volunteers to rate the result for 3 out of 15 random drumless tracks (each 23.8 seconds) from the test split. Each time, a volunteer listened to a drumless tracks ($\mathbf{x}_*^{m}$) first, and then (in random orders) the mixture (i.e., $\mathbf{x}_*^{m} + \mathbf{x}_*^{d}$) containing drum samples generated by four different models, plus the real human-made one (to set a high anchor). The volunteer then rated them in the following aspects on a 5-point Likert scale: **rhythmic consistency** between the drumless input and generated drums; **stylistic consistency** concerning the timbre and arrangement of the drumless input and generated drums; **audio quality** and **rhythmic stability** (whether the drummer follows a steady tempo) of the generated drum; and **overall** perceptual impression.

The mean opinion scores (MOS) in Table 1 show that `seq2seq+beat (low)` consistently outperforms the others, validating the effectiveness of using both the drumless codes and beat conditions. `decoder+beat (low)` performs consistently the second best, outperforming the two models without beat information significantly in three aspects according to paired t-test (p-value < 0.05), validating again the importance of the beat-aware module. Complementing Section 6, the MOS result suggests that the beat conditions seem more important than the drumless codes, though the best result is obtained with both.

Figure 5 further demonstrates that, given the same input, our model can generate multiple accompaniments with diversity in both beat and timbre. Diversity is an interesting aspect that is hard to evaluate, but it is desirable as there is no single golden drum accompaniment for a song. This may also explain why the F1 scores in Table 1 seem low.

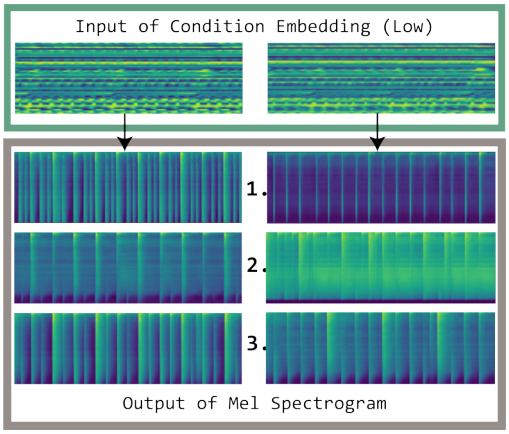


Figure 5: Two sets of three different generated samples by the same model given the same beat condition embedding.

Verbal feedbacks from the subjects confirm that our best model generates drum accompaniment that is rhythmically and stylistically consistent with the input, especially for band music or music with heavy use of bass. However, the model still has limits. At times the model generates total silence, though it can be avoided by sampling the LM again. The model may struggle to change its tempo going through different sections of a song. Moreover, the generation might be out-of-sync with the input in the beginning few seconds, until the model gets sufficient context. Please visit the demo page for various examples.

8. CONCLUSION

We have presented JukeDrummer, a novel audio-to-audio extension of OpenAI's JukeBox model capable of adding the drum part of a drumfree recording in the audio domain. To our knowledge, this represents the first attempt to audio-domain generation conditioned on drumless mixed audio. With objective and subjective evaluations, we validated the effectiveness of the customized VQ-VAE plus the seq2seq Transformer design, and the proposed beat-aware module. Among the beat conditions, we found that the low-level embeddings work the best. Future work can be done to further improve the language model (LM), and to extend our work to other audio-to-audio generation tasks.

9. ACKNOWLEDGEMENT

We are grateful to the anonymous reviewers for their valuable comments. This research is funded by grant NSTC 109-2628-E-001-002-MY2 from the National Science and Technology Council of Taiwan.

10. REFERENCES

[1] J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *Proc. Int. Conf. Machine Learning*, 2017.

[2] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial neural audio synthesis," in *Proc. Int. Conf. Learning Representations*, 2019.

[3] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *Proc. Int. Conf. Learning Representations*, 2019.

[4] S. Lattner and M. Grachten, "High-level control of drum track generation using learned patterns of rhythmic interaction," in *Proc. IEEE Work. Applications of Signal Processing to Audio and Acoustics*, 2019.

[5] J.-Y. Liu, Y.-H. Chen, Y.-C. Yeh, and Y.-H. Yang, "Unconditional audio generation with generative adversarial networks and cycle regularization," in *Proc. INTERSPEECH*, 2020.

[6] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *Int. Conf. Learning Representations*, 2021.

[7] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[8] C.-F. Liao, J.-Y. Liu, and Y.-H. Yang, "KaraSinger: Score-free singing voice synthesis with VQ-VAE using Mel-spectrograms," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2022, pp. 956–960.

[9] C. M. Payne, "MuseNet," *OpenAI Blog*, 2019.

[10] C. Donahue *et al.*, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2019.

[11] Y.-H. Chen, Y.-H. Huang, W.-Y. Hsiao, and Y.-H. Yang, "Automatic composition of guitar tabs by Transformers and groove modeling," in *Proc. Int. Soc. Music Inf. Retr. Conf.*, 2020.

[12] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, J. Zhao, and G. Xia, "PianoTree VAE: Structured representation learning for polyphonic music," in *Proc. Int. Soc. Music Inf. Retr. Conf.*, 2020.

[13] M. Suzuki, "Score Transformer: Transcribing quantized MIDI into comprehensive musical score," in *Proc. Int. Soc. Music Inf. Retr. Conf., Late-breaking demo paper*, 2021.

[14] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[15] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse Transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[16] C. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, and C. J. Cai, "AI song contest: Human-AI co-creation in songwriting," in *Proc. Int. Soc. Music Inf. Retr. Conf.*, 2020, p. 708–716.

[17] Z. Rafii *et al.*, "The MUSDB18 corpus for music separation," 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[18] E. Deruty, M. Grachten, S. Lattner, J. Nistal, and C. Aouameur, "On the development and practice of AI technology for contemporary popular music production," *Transactions of the International Society for Music Information Retrieval*, vol. 5, no. 1, pp. 35–49, 2022.

[19] C.-Y. Chiu, W.-Y. Su, and Y.-H. Yang, "Drum-aware ensemble architecture for improved joint musical beat and downbeat tracking," *IEEE Signal Processing Letters*, vol. 28, pp. 1100–1104, 2021.

[20] R. Dahale, V. Talwadker, P. Verma, and P. Rao, "Neural drum accompaniment generation," in *Proc. Int. Soc. Music Inf. Retr. Conf., Late-breaking demo paper*, 2021.

[21] D. Makris, G. Zixun, M. Kaliakatsos-Papakostas, and D. Herremans, "Conditional drums generation using compound word representations," in *Proc. Artificial Intelligence in Music, Sound, Art and Design*, 2022, p. 179–194.

[22] J. Gillick, A. Roberts, J. Engel, D. Eck, and D. Bamman, "Learning to groove with inverse sequence transformations," in *Proc. Int. Conf. Machine Learning*, 2019.

[23] V. Thio, H.-M. Liu, Y.-C. Yeh, and Y.-H. Yang, "A minimal template for interactive web-based demonstrations of musical machine learning," in *Proc. Workshop on Intelligent Music Interfaces for Listening and Creation*, 2019.

[24] G. Alain, M. Chevalier-Boisvert, F. Osterrath, and R. Piche-Taillefer, "DeepDrummer: Generating drum loops using deep learning and a human in the loop," *arXiv preprint arXiv:2008.04391*, 2020.

[25] N. Tokui, "Can GAN originate new electronic dance music genres? – Generating novel rhythm patterns using GAN with genre ambiguity loss," *arXiv preprin: 2011.13062*, 2020.

[26] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "MuseGAN: Symbolic-domain music generation and accompaniment with multi-track sequential generative adversarial networks," in *Proc. AAAI Conf. Artificial Intelligence*, 2018.

[27] I. Simon, A. Roberts, C. Raffel, J. Engel, C. Hawthorne, and D. Eck, "Learning a latent space of multitrack measures," *arXiv preprint arXiv:1806.00195*, 2018.

[28] H.-M. Liu and Y.-H. Yang, "Lead sheet generation and arrangement by conditional generative adversarial network," in *Proc. IEEE. Conf. Machine Learning and Applications*, 2018, pp. 722–727.

[29] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "PopMAG: Pop music accompaniment generation," in *Proc. ACM Multimedia Conf.*, 2020.

[30] J. Nistal, S. Lattner, and G. Richard, "DrumGAN: Synthesis of drum sounds with timbral feature conditioning using generative adversarial networks," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2020.

[31] A. Ramires, P. Chandna, X. Favory, E. Gómez, and X. Serra, "Neural percussive synthesis parameterised by high-level timbral features," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, 2020.

[32] C. Aouameur, P. Esling, and G. Hadjeres, "Neural drum machine: An interactive system for real-time synthesis of drum sounds," *arXiv preprint arXiv:1907.02637*, 2019.

[33] J. Drysdale, M. Tomczak, and J. Hockman, "Adversarial synthesis of drum sounds," in *Proc. Int. Conf. Digital Audio Effects*, 2020.

[34] ——, "Style-based drum synthesis with GAN inversion," in *Proc. Int. Soc. Music Inf. Retr. Conf., Late-breaking demo paper*, 2021.

[35] M. Tomczak, M. Goto, and J. Hockman, "Drum synthesis and rhythmic transformation with adversarial autoencoders," in *Proc. ACM Int. Conf. Multimedia*, 2020, p. 2427–2435.

[36] T. Hung, B. Chen, Y. Yeh, and Y. Yang, "A benchmarking initiative for audio-domain music generation using the Freesound Loop Dataset," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2021.

[37] A. v. d. Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proc. Advances in Neural Information Processing Systems*, 2017.

[38] L. Kaiser *et al.*, "Fast decoding in sequence models using discrete latent variables," in *Proc. Int. Conf. Machine Learning*, 2018, pp. 2390–2399.

[39] A. Razavi *et al.*, "Generating diverse high-fidelity images with VQ-VAE-2," in *Proc. Advances in Neural Information Processing Systems*, 2019.

[40] A. Polyak *et al.*, "Speech resynthesis from discrete disentangled self-supervised representations," in *Proc. Interspeech*, 2021, pp. 3615–3619.

[41] J. Walker, A. Razavi, and A. v. d. Oord, "Predicting video with VQVAE," *arXiv preprint arXiv:2103.01950*, 2021.

[42] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Proc. Advances in Neural Information Processing Systems*, 2020.

[43] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: A new Python audio and music signal processing library," *Proc. ACM Multimed. Conf.*, pp. 1174–1178, 2016.

[44] S. Böck *et al.*, "Joint beat and downbeat tracking with recurrent neural networks," in *Proc. Int. Soc. Music Information Retrieval Conf.*, 2016.

[45] S. Böck and M. E. P. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," in *Proc. Int. Soc. Music Inf. Retr. Conf.*, 2020, p. 574–582.

[46] C.-Y. Chiu, J. Ching, W.-Y. Hsiao, Y.-H. Chen, W.-Y. Su, and Y.-H. Yang, "Source separation-based data augmentation for improved joint beat and downbeat tracking," in *Proc. Eur. Signal Process. Conf.*, 2021.

[47] P. Virtanen *et al.*, "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[48] R. Bittner *et al.*, "MedleyDB: A multitrack dataset for annotation-intensive MIR research," in *Proc. Int. Soc. Music Inf. Retr. Conf.*, 2014, [Online] http://medleydb.weebly.com/.

[49] "The 'Mixing Secrets' free multitrack download library," [Online] https://www.cambridge-mt.com/ms/mtk/.

[50] K.-W. Kim, S.-W. Park, J. Lee, and M.-C. Joe, "ASSEM-VC: Realistic voice conversion by assembling modern speech synthesis techniques," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, 2022.

[51] X. Tan *et al.*, "NaturalSpeech: End-to-end text to speech synthesis with human-level quality," *arXiv preprint arXiv:2205.04421*, 2022.

[52] C. Raffel *et al.*, "mir_eval: A transparent implementation of common MIR metrics," in *Proc. Int. Soc. Music Inf. Retr. Conf.*, 2014, pp. 367–372.

LEARNING HIERARCHICAL METRICAL STRUCTURE BEYOND MEASURES

Junyan Jiang[1,2] **Daniel Chin**[1,2] **Yixiao Zhang**[1,3] **Gus Xia**[1,2]
[1] Music X Lab, NYU Shanghai [2] MBZUAI [3] Centre for Digital Music, QMUL
jj2731@nyu.edu, daniel.chin@nyu.edu, yixiao.zhang@qmul.ac.uk, gxia@nyu.edu

ABSTRACT

Music contains hierarchical structures beyond beats and measures. While hierarchical structure annotations are helpful for music information retrieval and computer musicology, such annotations are scarce in current digital music databases. In this paper, we explore a data-driven approach to automatically extract hierarchical metrical structures from scores. We propose a new model with a Temporal Convolutional Network-Conditional Random Field (TCN-CRF) architecture. Given a symbolic music score, our model takes in an arbitrary number of voices in a beat-quantized form, and predicts a 4-level hierarchical metrical structure from downbeat-level to section-level. We also annotate a dataset using RWC-POP MIDI files to facilitate training and evaluation. We show by experiments that the proposed method performs better than the rule-based approach under different orchestration settings. We also perform some simple musicological analysis on the model predictions. All demos, datasets and pre-trained models are publicly available on Github [1] .

1. INTRODUCTION

Music contains rich structures at different levels, and the structure annotations play an important role in music understanding [1, 2] and generation [3, 4]. Progress has been made in music structure analysis on certain levels, like beat tracking [5–7], downbeat detection [8–11] and part segmentation [12–15]. These levels of structures are often inter-connected with each other, and can be described in a hierarchical way. For example, a part may contain several sections, each containing several measures. Measures can be further decomposed into beats.

Several views of hierarchical music structures are formally discussed in the Generative Theory of Tonal Music (GTTM) [16], including the grouping structure, the metrical structure, the time-span tree, and the prolongational tree, each focusing on different music properties (i.e., the grouping structure focuses more on melodic grouping while the metrical structure focuses on rhythmic patterns). Among these structures, we choose the metrical structure as the topic for two reasons: (1) For polyphonic music, the metrical structures of different voices are usually compatible, building up a common song-level metrical structure. This property makes data annotation easier and provides opportunities for self-supervision; (2) Some low-level metrical structures (e.g., beats and downbeats) are already annotated in music scores and most MIDI datasets, which can be a helpful source of supervision. In this paper, we will focus our analysis on pop songs, which often contain a well-defined hierarchy of metrical structures [17].

Figure 1. The hierarchical metrical structure of the beginning of RWC-POP No. 001. Only the main melody is shown. Despite the pick-up bar, these 4 measures are grouped into 2 hypermeasures of length 2, which are further grouped into 1 hypermeasure of length 4.

Our main goal is to infer the high-level metrical structures given low-level ones like beats and downbeats. The hierarchy of the metrical structure is created by recursively grouping lower metrical units into upper ones (see Figure 1 for an example). We call the grouping of measures (or other larger metrical units) a *hypermeasure*, and the number of units that form the group as its *hypermeter* [18, 19]. While some properties of beats and measures can be generalized to upper-level metrical structures, there are still many differences to take into account. Similar to the meter, the hypermeter of each layer tends to stay the same to maintain a regular rhythmic pulse, but it is not uncommon to see hypermeter changes in a piece, as shown in Figure 4. Such changes occur more often than low-level meter changes since listeners are less sensitive to long-term rhythmic regularity. Another major difference is that the decision of upper-level metrical structures requires a longer context in the time domain compared to downbeat or beat tracking.

To resolve these issues, we design a new model that contains a Temporal Convolutional Network (TCN) front-end for metrical level prediction and a Conditional Random Field (CRF) decoder for joint metrical structure decoding. We design the transition of CRF hidden states to allow hypermeter changes with some penalties. To han-

[1] https://github.com/music-x-lab/Hierarchical-Metrical-Structure

© J. Jiang, D. Chin, Y. Zhang and G. Xia. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J. Jiang, D. Chin, Y. Zhang and G. Xia, "Learning Hierarchical Metrical Structure Beyond Measures", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

dle polyphony, the model takes an arbitrary number of voices/tracks as input, and predicts a confidence score for each track. The final prediction is a weighted average of the results from all tracks. We annotated 70 songs from the RWC-POP dataset and used them for model training and evaluation. We conduct experiments under different orchestration setups with results shown in section 4.4.

2. RELATED WORK

2.1 Downbeat and Meter Tracking

Downbeat tracking for raw audio is a well-studied task with many promising results. Krebs et al. [8] use particle filters to find the downbeats, yielding a 2-level metrical structure. Durand et al. [9] use an ensemble of convolutional networks to locate downbeats robustly. Fuentes et al. [10] use skip-chain CRF and deep learning to track downbeats, noting that longer-term musical contexts can better inform downbeat tracking. The reader is referred to Gouyon and Dixon's review [11] for non-symbolic low-level metrical analyzers before 2005.

Notably, locating the beats and downbeats for symbolic music is not a trivial task. Kostek et al. [20] apply neural networks and rough sets to a polyphonic symbolic piece and classify whether each note is accented or not, yielding a 2-level metrical structure (beat and downbeat). Chuang and Su [21] use various RNNs to classify each timestep in a piano roll into non-beat, beat, or downbeat.

Another related task is time signature detection. Benoit [22] uses symbolic-level auto-correlation to obtain a 4-level metrical structure, and ultimately extracts the meter. More auto-correlative methods [23, 24] share a similar logic, since note onsets usually display periodicity in every measure. More recently, inner metric analysis was also used to infer the time signature by Haas et al. [25].

2.2 Music Segmentation

The task of music segmentation is to infer musically meaningful section or part boundaries from the music content. Audio-based music segmentation is usually achieved by detecting similarity or repetition of the audio spectral features. McFee and Ellis [13] evaluate inter-time frame similarity and use spectral clustering to obtain a multi-level segmentation of music. Salamon et al. [14] and McCallum [15] replace traditional features with pre-trained deep embeddings to estimate timbre and harmonic similarity. Tralie and McFee [26] fuse multiple similarity metrics for better prediction results. Ullrich et al. [27] train fully supervised CNN on a segment-annotated dataset to detect segment boundaries. See Dannenberg and Goto's review [28] and the 2010 SOTA report by Paulus et al. [29] to learn more about audio-based music segmentations.

Segmentation of symbolic music relies more on domain knowledge of music composition. Van der Werf and Hendriks [30] restate GTTM grouping rules in terms of Optimality Theory (OT) and design a Prolog program to find an optimal parse for short monophonic pieces. Dai et al. [31] identify phrases as units of melodic repetition by minimizing the Structural Description Length (SDL) for the entire piece, and then extract a 2-layer hierarchy.

2.3 Hierarchical Structure Analysis

The Generative Theory of Tonal Music (GTTM) [16] discusses several views of the hierarchical music structures, but GTTM does not describe how to realize an analyzer computationally. Various efforts have been made to mechanize GTTM. For example, Jones et al. [32] use a rule-based expert system to obtain a 6-level metrical structure for monophonic pieces, satisfying all GTTM's well-formedness rules while following Povel's grid theory [33]. Rosenthal's Machine Rhythm [34] ventures into the polyphonic domain and uses rule-based methods to extract a 3-level rhythmic annotation for MIDI input. Temperley and Sleator [35] use a preference-rule approach and dynamic programming to obtain the optimal parse. Hamanaka et al. [36] propose the Automatic Time-span Tree Analyzer (ATTA) for structural analysis of 8-bar monophonic scores. Temperley [37] use Bayesian reasoning to jointly analyze metrical, harmonic, and stream structures. Wojcik and Kostek [18] use rule-based methods to retrieve hypermetric rhythm from only the melody.

Machine learning models are also used for hierarchical structure analysis. Hamanaka et al. propose Deep-GTTM [38, 39] which is a fully automatic analyzer that uses a neural network to predict the applicability of GTTM rules for each note, yielding a 5-level metrical structure for 8-bar monophonic scores.

There are some issues when applying previous works to large polyphonic MIDI databases. Most systems work only for monophonic music or music with limited polyphony. Also, they often work in a short context, usually up to 8 bars.

3. METHODOLOGY

3.1 Problem Setting

We first formally define the metrical structure prediction task for this paper. Assume we have a music score with T voices (or tracks, in the sense of MIDI files) $\mathbf{m}_1, ..., \mathbf{m}_T$. We also have a list of pre-annotated downbeats $d_1, ..., d_N$ for the music. The aim is to assign hierarchical metrical labels $l_i \in \{0, 1, ..., L\}$ to each downbeat d_i where L is the total number of layers we want to build beyond measures. Each label serves as the level of the metrical boundary at d_i. $l_i = l$ means d_i serves as a metrical boundary of all levels for the first l levels beyond measures. Specially, $l_i = 0$ means d_i serves only as a measure boundary but not any metrical boundary beyond measures.

If we use GTTM's metrical structure notation, we can use (l_i+1) dots to represent a metrical boundary level of l_i. For the example in figure 1, the first 4 measures (excluding the pickup measure) would have labels $l_1 = 2, l_2 = 0, l_3 = 1, l_4 = 0$.

We can also introduce the following notations.

Hypermeasures: A hypermeasure of level l is an interval

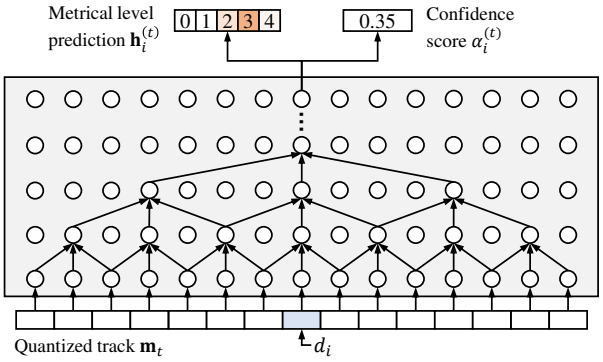


Figure 2. The architecture of the neural network.

between two downbeats d_i and d_j where $l_i \geq l, l_j \geq l$ and $l_k < l$ for all $k = i+1...j-1$. In other words, any $l_k \geq l$ serves as a separator of a level-l hypermeasure. Specially, level-0 hypermeasures are just measures.

Hypermeters: A hypermeter is the generalization of meters by counting how many level-$(l-1)$ hypermeasures are in a level-l hypermeasure. Since a binary structure is the most commonly used [19, 40], we assume a general hypermeter of 2 in all levels, with a few exceptions that cause *binary irregularity*.

3.2 Temporal Convolutional Network

Temporal Convolutional Networks (TCNs) have been proven an effective model for beat, downbeat, and tempo tracking [7, 41, 42]. We believe it is useful for general metrical structure analysis for its unique property we will mention below. We mainly reference [43] for the design of the TCN, but we made it non-causal similar to [7]. Each TCN block contains 8 sequential layers. The first 4 layers are a dilated convolutional layer with kernel size 3 and 256 channels, a batch normalization layer, a Rectified Linear Unit (ReLU) activation layer, and a dropout layer. The next 4 layers repeat this configuration. There is also a residual component that adds a linear transformation of the input to the block output, allowing shortcut connections.

Our model uses 6 TCN blocks sequentially. Each block multiplies the dilation by 2, starting from 1 at block 1, resulting in an exponentially growing context range for each layer. This allows the model to capture long-term context and more importantly, integrate prior knowledge about binary metrical structure into the network. The model input contains the piano roll and the onset roll of a track quantized into a 16th note level. Under a 4/4 meter, the dilations of the convolutional layers in each block are therefore 1/4 beat, 1/2 beat, 1 beat, 2 beats, 1 measure and 2 measures, respectively. This encourages the convolutional layers to capture more musically meaningful context for binary metrical structures.

For each track $\mathbf{m}_t$ in a song, we first feed them into the TCN blocks to get the features for each time step, and then discard the time steps that do not correspond to any downbeat. We use linear layers to project the features into a metrical level prediction $\mathbf{h}_i^{(t)}$ and a confidence score $\alpha_i^{(t)}$.

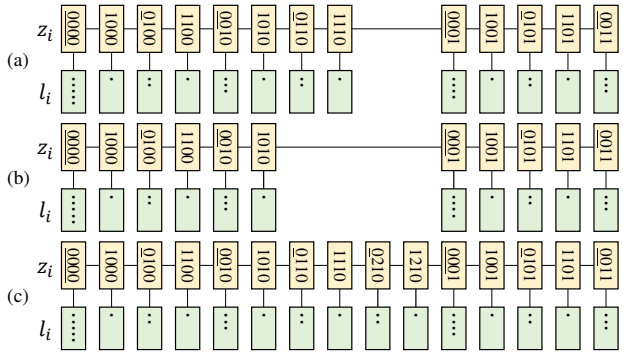


Figure 3. Examples of the CRF hidden variables z_i (shown as a L-digit number $z_i^{(1)}...z_i^{(L)}$) and the corresponding metrical boundary levels l_i when $L = 4$. (a) Binary regularity is satisfied. (b) A level-1 hypermeasure is deleted from (a). (c) A level-1 hypermeasure is inserted into (a). Both (b) and (c) are examples of binary irregularity.

Each prediction $\mathbf{h}_i^{(t)}$ is a vector of size $(L+1)$ for the labels $0...L$. The final prediction of l_i is the weighted average of all predictions $\mathbf{h}_i^{(t)}$ on the same time step weighted by their confidence scores:

$$a_i^{(t)} := \exp \alpha_i^{(t)} / \sum_{t'} \exp \alpha_i^{(t')} \quad (1)$$

$$\mathbf{p}_i := \sum_t a_i^{(t)} \text{Softmax}(\mathbf{h}_i^{(t)}) \quad (2)$$

3.3 Conditional Random Fields

Conditional Random Fields (CRFs) have been widely used in downbeat tracking [10, 44] to enforce the regularity of the decoded downbeat patterns. Inspired by this, we also use a structured prediction method for decoding. The major differences are that this model needs to predict a hierarchy of $L = 4$ layers of metrical structures jointly.

The state space of hierarchical metrical structure can be complex and ambiguous. To make it simple, we restrict our model to accept a hypermeter of 1, 2 or 3 on any level. In the sense of transformational grammar, a level-l hypermeter of 1 can be constructed by deleting some level-$(l-1)$ hypermeasures from a deep structure with binary regularity, and a hypermeter of 3 can be constructed by inserting some level-$(l-1)$ hypermeasures (see figure 3 for an example). A hypermeter of 4 or more is not allowed and needs to be decomposed into multiple metrical levels (e.g., $4 = 2 + 2$).

We design the linear CRF with a joint state space $z_i = (z_i^{(1)}, ..., z_i^{(L)})$ where each $z_i^{(l)} \in \{0, 1, 2\}$ corresponds to the current hypermeasure position at level l, i.e., the number of complete level-$(l-1)$ hypermeasures in this level-l hypermeasure up to the current time step. It can be seen as a generalization of the beat position in [10]. A state $z_i^{(1...l)} = 0 \wedge z_i^{(l+1)} \neq 0$ denotes a metrical boundary level $l_i = l$. Specially, the highest-level metrical boundary $l_i = L$ is associated and only associated with the state $(0, ..., 0)$, and the lowest-level metrical boundary $l_i = 0$ is associated with any z_i where $z_i^{(1)} \neq 0$. See figure 3 for some concrete examples.

We manually design the transition potential function to encode the belief of binary regularity on each level. We define the transition potential matrix of a single level as

$$\mathbf{A}^{(l)} = \begin{bmatrix} \exp\big(-w_{\text{del}}^{(l)}\big) & 1 & 0 \\ 1 & 0 & \exp\big(-w_{\text{ins}}^{(l)}\big) \\ 1 & 0 & 0 \end{bmatrix} \quad (3)$$

where $A_{ij}^{(l)}$ denotes the potential of transition from hypermeasure position i to position j on level l. $w_{\text{del}}^{(l)} > 0, w_{\text{ins}}^{(l)} > 0$ are hyperparameters that controls the penalty of a level-l hypermeasure deletion and insertion respectively. Intuitively, an alternating state sequence like $0, 1, 0, 1$ on a single level satisfies binary regularity and will not be penalized. Binary irregularity by inserting (e.g., $0, 1, 2, 0, 1$) or deleting (e.g., $0, 0, 1$) states are penalized.

In a hierarchical metrical transition to a level-l metrical boundary, the hypermeasure positions of level $1...(l+1)$ are updated, and the ones above level-$(l+1)$ keep the same. Therefore, the joint transition potential is defined as

$$\phi(z_{i-1}, z_i) = \prod_{l=1}^{L} \begin{cases} A_{z_{i-1}^{(l)} z_i^{(l)}} & l \leq l_i + 1 \\ \mathbb{I}[z_{i-1}^{(l)} = z_i^{(l)}] & l > l_i + 1 \end{cases} \quad (4)$$

where l_i denotes the corresponding metrical boundary level of z_i, and $\mathbb{I}[b]$ is the indicator function that returns 1 if b is true and 0 if b is false.

The emission potential function is designed as $\psi(z_i, \mathbf{p}_i) = p_{il_i}$ where p_{il_i} is the l_i-th entry of $\mathbf{p}_i$. We use Viterbi decoding to decode the optimal hidden states $z_{1..d}$ given the observations $\mathbf{p}_{1...N}$:

$$\hat{\mathbf{z}} = \arg\max_{\mathbf{z}} \psi(z_1, \mathbf{p}_1) \prod_{i=2}^{N} \phi(z_{i-1}, z_i) \psi(z_i, \mathbf{p}_i) \quad (5)$$

4. EXPERIMENTS

4.1 Datasets

The main dataset we use in model training and evaluation is the RWC-POP dataset. It contains 100 songs with aligned MIDI files. The MIDI files have beat and downbeat annotations that are mostly correct [2]. We manually annotated the 4-layer song-level metrical structure for 70 songs. We use 50 songs for training, 10 for validation, and 10 for testing. We acknowledge the ambiguity in data annotation which potentially causes bias by annotator subjectivity [45], so we publicized the annotation data and methods to facilitate discussion and future research.

All samples are quantized into 16th-note units. The binary piano roll and onset roll are extracted for each track. During training, a random 512-unit (32 measures) segment is chosen from each song, resulting in a 512×256 feature matrix (128 MIDI pitches for piano roll and 128 MIDI pitches for onset roll) for each track. To prevent overfitting, we perform label-preserving data augmentation including random pitch shift augmentation of -12 to +12 semitones and a microtiming shift up to an 8th note on the training set. The drum track does not use pitch shift augmentation and does not share the same parameter with pitched instruments for the first convolutional layer.

[2] 2 out of 70 songs have minor beat/downbeat annotation issues.

4.2 Model Training

For model training, we use a mini-batch of 16. We use the Adam optimizer [46] on the cross-entropy classification loss with a learning rate 0.0001 for 100 epochs. In one epoch, we go through each augmented version of each song for 5 times. For each song, we randomly select 2 tracks and try to predict the song-level metrical structure given the weighted average of their predictions. We also apply dropout with a probability 0.5 after each convolutional layer to further suppress overfitting.

4.3 Baseline Models

While there are some existing rule-based hierarchical metrical structure analyzers [32,39,47] in previous works, they mostly focus on low-level (e.g., beat & downbeat) boundary features like note transitions, durations and local rhythmic patterns. Those features are not effective enough for metrical structures above the measure level. We here build our baseline model using the methodology from [48]. For each metrical level, we calculate the similarity matrix of piano rolls on different granularity and estimate their novelty score as observations. We use a CRF decoder as mentioned above with a different set of hyper-parameters tuned for the baseline model.

To assess the necessity of introducing metrical irregularity, We also introduce another hypothetical baseline model called the *oracle* model. The oracle model is not allowed to predict any hypermeter changes (i.e., it always assumes binary regularity) but it always performs the best possible prediction (i.e., maximal F1 score) for each level. This hypothetical model serves as an upper bound for any model that does not allow binary irregularity.

4.4 Results

Table 1 shows the performance of each model on the test split of RWC-POP songs. To perform a more systematic evaluation, we also created two difficult versions of each test song: (1) **no drums**: the drum track(s) are removed from the original song and each model is required to predict the same metrical structure without referring to any drum clues; (2) **mel. only**: all tracks except the melody track are removed. Each model is required to predict the structure by purely referring to the main melody track.

From the results, we can see that our proposed model performs better than the rule-based counterpart on all metrical levels. Since most test songs have more than 10 MIDI tracks, they provide sufficient metrical hints to both the proposed model and the rule-based model even if the drum track is removed. When we only have the melody track, both the proposed model and the rule-based model's performances are not satisfactory even on the first level beyond measure. Still, the data-driven approach shows improved performance compared to the rule-based system.

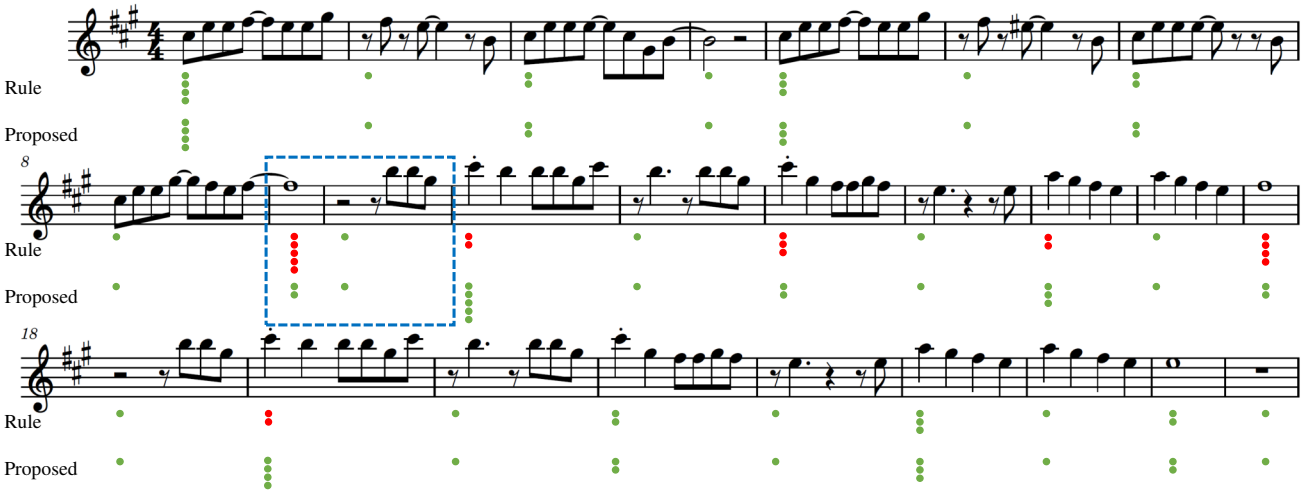


Figure 4. A case study with song RWC-POP No. 008 from the test split. The song is multi-track but we only show the main melody here. The metrical structure of the song does not satisfy binary regularity because of the 2-bar extensions in the pre-chorus (marked by a dashed blue box), causing hypermeter changes. The prediction of the proposed method aligns well with the reference. The errors in the rule-based prediction are marked in red (better viewed in color).

Model	Level 1	Level 2	Level 3	Level 4
Proposed	0.9848 ±0.0215	**0.9559 ±0.0386**	**0.8880 ±0.0889**	**0.6849 ±0.1900**
Proposed w/o CRF	0.9338 ±0.0390	0.8528 ±0.0937	0.7971 ±0.1276	0.6646 ±0.0844
Rule	0.9228 ±0.0698	0.8425 ±0.1195	0.7485 ±0.1536	0.5185 ±0.2656
Oracle	0.9427 ±0.1120	0.7782 ±0.2076	0.5188 ±0.1751	0.4225 ±0.1234
Proposed (no drums)	**0.9868 ±0.0174**	0.9519 ±0.0346	0.8803 ±0.1023	0.6611 ±0.2170
Rule (no drums)	0.9312 ±0.0660	0.8107 ±0.1568	0.7055 ±0.2008	0.4823 ±0.2239
Proposed (mel. only)	0.7413 ±0.2139	0.6253 ±0.2448	0.5551 ±0.2536	0.3808 ±0.2399
Rule (mel. only)	0.6606 ±0.1451	0.4395 ±0.1522	0.3142 ±0.1211	0.1863 ±0.1310

Table 1. Evaluated F1 scores on the test split of the RWC-POP dataset.

We observe that the proposed model is better at capturing irregular metrical structures than the rule-based approach. Figure 4 shows a cherry-picked example where binary irregularity can be found. Both the proposed model and the rule-based baseline can detect such irregularity but only the proposed model correctly tells the exact position of the hypermetrical change.

There is also a tendency for the performance to drop rapidly from lower to higher levels. We believe there are 2 main reasons. First, the higher levels have fewer positive samples, making it hard for the model to learn its semantic characteristics. Second, metrical structures on higher levels are often more ambiguous than lower ones even for human listeners. Sometimes, the highest level (level 4) needs to decide how to group parts together (e.g., verse + pre-chorus or pre-chorus + chorus). Different decisions are sometimes all acceptable.

4.4.1 Out of Distribution Evaluation

We also want to know whether the proposed model can be applied to MIDI files with very different orchestration setups. Such experiments are hard to perform because of the lack of ground truth annotations. We here perform a small-scale experiment on the POP909 [49] dataset. We select the first 5 songs in the dataset ordered by index and manually annotate the metrical structure [3]. POP909 is a dataset of Chinese pop songs rearranged for piano performance. Each song only has 3 tracks, i.e., a vocal track and two piano tracks, making it harder compared to RWC-POP. The results are shown in table 2.

From the results, we can see the performance degrades even when all 3 tracks are present. By case inspection, we find that the proposed model has generally satisfactory performance on 3 out of 5 songs on lower layers. However, there is one complex song [4] with multiple metrical and hypermeter changes that make all the approaches fail. Also, due to the lack of rhythmic clues (e.g., drums), a deeper understanding of the syntax and semantics of melody and chords might be required to perform musically meaningful segmentation, which we assume our model can hardly acquire on a small training set of 50 pop songs.

4.5 Confidence Score Analysis

To perform a statistical analysis of the model behavior and provide a musicological view of the model prediction, we perform model inferences on a large selection of the Lakh MIDI dataset [50]. To ensure enough accuracy of model prediction, we only select a part of the Lakh MIDI dataset

[3] We are aware that POP909's downbeat annotations are sometimes inaccurate and we manually fixed them.

[4] POP909 No. 005: *I Believe* by Van Fan.

Model	Level 1	Level 2	Level 3	Level 4
Proposed	**0.9084 ±0.0896**	**0.8742 ±0.1101**	**0.6470 ±0.2174**	0.4930 ±0.3132
Rule	0.6818 ±0.1855	0.6625 ±0.1550	0.5163 ±0.1195	0.3446 ±0.1856
Oracle	0.8527 ±0.1735	0.7348 ±0.2763	0.5883 ±0.2493	**0.5767 ±0.2474**
Proposed (mel. only)	0.6742 ±0.2962	0.6542 ±0.2737	0.5685 ±0.2403	0.4797 ±0.2053
Rule (mel. only)	0.6062 ±0.1034	0.3933 ±0.0275	0.2642 ±0.0546	0.1551 ±0.0305

Table 2. Evaluated F1 scores on the first 5 songs in the POP909 dataset. Mel. only denotes that the melody track is used. Otherwise, all 3 tracks (melody, bridge and piano) are used.

that has a similar orchestration compared to RWC-POP. We filter the MIDI files according to the following criteria: (1) it contains at least 6 MIDI tracks, including 1 drum track and 1 track whose name contains strings "melody" or "vocal"; (2) if multiple MIDI files' identified audio sources are the same, at most one MIDI file is kept. A filtered dataset of 3,739 MIDI files is collected.

We here evaluate the relevance of instruments and the model's predicted confidence score. Notice that the instrument program number is not a part of the model input, so the only difference comes from the rhythmic properties of their scores. We collect the unnormalized confidence scores $\alpha^{(t)}$ for each track $\mathbf{m}_t$ of different instruments, and calculate their means and standard derivations. Specially, we regard all melody tracks (identified by their names) as a new instrument and ignore its original MIDI program number. Also, we remove all tracks with too many measure-level rests (more than 1/3 of the whole song) since they trivially result in low confidence scores.

Table 3 shows the results of confidence score analysis. We can see that drums are the strongest clue for metrical structures. The melody track and many melodic instruments (e.g., guitars) also serve as useful clues. On the other hand, instruments that produce slow accompaniments (e.g., string ensemble and pads) are less preferred.

4.5.1 Drum Track Analysis

As another experiment of musicological analysis, we perform an experiment on the relation between drum notes and the metrical structure level. For simplicity, we only collect samples that a certain drum event happens exactly on the downbeat, and we collect the corresponding metrical boundary level of that downbeat. The results are shown in Table 4. While the occurrence of many drum events does not significantly change the distribution of the metrical boundary level, the crash cymbal and splash cymbal are certainly useful clues to a high-level metrical boundary. This aligns with people's perception of them since these cymbals are usually associated with a strong burst of energy, serving as an important rhythmic hint.

Track/Instrument	Confidence
Melody	1.78 ± 1.22
Drum	3.61 ± 2.58
Acoustic Grand Piano	0.35 ± 1.69
Electric Guitar (jazz)	0.75 ± 1.55
Acoustic Bass	0.33 ± 1.66
String Ensemble	0.11± 1.70
Pad (warm)	-0.86 ± 1.78

Table 3. A selected view of the means and standard derivations of the confidence score for different tracks/instruments. The melody track is identified by its name instead of the MIDI instrument. The drum track is identified by its MIDI channel number (No. 10).

Drums (%)	L-0	L-1	L-2	L-3	L-4
Any	50.00	24.71	12.06	6.21	7.02
Bass Drum	48.66	25.10	12.46	6.47	7.30
Acoustic Snare	52.27	23.28	11.19	6.27	7.00
Closed Hi Hat	51.32	25.14	11.91	5.54	6.11
Open Hi Hat	51.90	25.07	11.33	5.38	6.32
Crash Cymbal	20.97	19.13	18.58	18.94	22.38
Ride Cymbal	51.41	25.03	11.57	5.61	6.39
Splash Cymbal	34.80	22.30	16.00	12.90	14.01

Table 4. A selected view of the frequency of different drum instruments (on a downbeat) associated with different levels of metrical boundaries. L-n means level-n metrical boundary.

5. CONCLUSION AND FUTURE WORK

In this paper, we propose a data-driven approach for hierarchical metrical structure analysis of symbolic music. Our model adopts a TCN-CRF architecture and accepts an arbitrary number of voices as input. Experiments on MIDI datasets show that our model performs better than rule-based methods under different orchestration settings.

The model performance is still not satisfactory, especially for high-level metrical structures and music with very different orchestration. We assume the performance would be better if more data were annotated, but there are also other possible directions for data-driven methods. First, self-supervised or semi-supervised methods might be a helpful complement to the lack of labeled datasets. For example, a consistency loss can be used to evaluate the prediction between different voices in the same music piece. Different data augmentation strategies might also be helpful. Second, it might be useful to utilize datasets of related tasks (e.g., section labels) as a source of weak supervision. Related tasks can also be used for multi-task learning.

Other potential future works include improving the automatic analysis system of other hierarchical structures, e.g., the grouping structures. The application of hierarchical structure analysis in the audio domain is also worth exploring.

6. REFERENCES

[1] N. C. Maddage, C. Xu, M. S. Kankanhalli, and X. Shao, "Content-based music structure analysis with applications to music semantics understanding," in *Proceedings of the 12th annual ACM international conference on Multimedia*, 2004, pp. 112–119.

[2] D. P. Ellis and G. E. Poliner, "Identifying 'cover songs' with chroma features and dynamic programming beat tracking," in *2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 4, 2007, pp. IV–1429–IV–1432.

[3] S. Dieleman, A. van den Oord, and K. Simonyan, "The challenge of realistic music generation: modelling raw audio at scale," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[4] S. Lattner, M. Grachten, and G. Widmer, "Imposing higher-level structure in polyphonic music generation using convolutional restricted boltzmann machines and constraints," *Journal of Creative Music Systems*, vol. 2, pp. 1–31, 2018.

[5] D. P. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[6] S. Dixon, "Evaluation of the audio beat tracking system beatroot," *Journal of New Music Research*, vol. 36, no. 1, pp. 39–50, 2007.

[7] E. MatthewDavies and S. Böck, "Temporal convolutional networks for musical audio beat tracking," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.

[8] F. Krebs, A. Holzapfel, A. T. Cemgil, and G. Widmer, "Inferring metrical structure in music using particle filters," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 5, pp. 817–827, 2015.

[9] S. Durand, J. P. Bello, B. David, and G. Richard, "Robust downbeat tracking using an ensemble of convolutional networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 76–89, 2016.

[10] M. Fuentes, B. McFee, H. C. Crayencour, S. Essid, and J. P. Bello, "A music structure informed downbeat tracking system using skip-chain conditional random fields and deep learning," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 481–485.

[11] F. Gouyon and S. Dixon, "A review of automatic rhythm description systems," *Computer music journal*, vol. 29, no. 1, pp. 34–54, 2005.

[12] T. Grill and J. Schlüter, "Music boundary detection using neural networks on combined features and two-level annotations." in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015, pp. 531–537.

[13] B. McFee and D. Ellis, "Analyzing song structure with spectral clustering." in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. Citeseer, 2014, pp. 405–410.

[14] J. Salamon, O. Nieto, and N. J. Bryan, "Deep embeddings and section fusion improve music segmentation," in *Proceedings of the 22th International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[15] M. C. McCallum, "Unsupervised learning of deep features for music segmentation," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 346–350.

[16] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music, reissue, with a new preface*. MIT press, 1996.

[17] J. M. Robins, "Phrase structure, hypermeter, and closure in popular music," Ph.D. dissertation, The Florida State University, 2017.

[18] J. Wojcik and B. Kostek, "Representations of music in ranking rhythmic hypotheses," in *Advances in Music Information Retrieval*. Springer, 2010, pp. 39–64.

[19] D. Temperley, "Hypermetrical transitions," *Music Theory Spectrum*, vol. 30, no. 2, pp. 305–325, 2008.

[20] B. Kostek, J. Wojcik, and P. Szczuko, "Searching for metric structure of musical files," in *International Conference on Rough Sets and Intelligent Systems Paradigms*. Springer, 2007, pp. 774–783.

[21] Y.-C. Chuang and L. Su, "Beat and downbeat tracking of symbolic music data using deep recurrent neural networks," in *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2020, pp. 346–352.

[22] B. Meudic, "Automatic meter extraction from MIDI files," in *Journées d'Informatique Musicale*, Marseille, France, Jun. 2002, pp. 1–1.

[23] J. C. Brown, "Determination of the meter of musical scores by autocorrelation," *The Journal of the Acoustical Society of America*, vol. 94, no. 4, pp. 1953–1957, 1993.

[24] D. Eck and N. Casagrande, "Finding meter in music using an autocorrelation phase matrix and shannon entropy." in *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, 2005, pp. 504–509.

[25] W. B. De Haas and A. Volk, "Meter detection in symbolic music using inner metric analysis," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, p. 441.

[26] C. J. Tralie and B. McFee, "Enhanced hierarchical music structure annotations via feature level similarity fusion," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 201–205.

[27] K. Ullrich, J. Schlüter, and T. Grill, "Boundary detection in music structure analysis using convolutional neural networks." in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, pp. 417–422.

[28] R. B. Dannenberg and M. Goto, "Music structure analysis from acoustic signals," in *Handbook of signal processing in acoustics*. Springer, 2008, pp. 305–331.

[29] J. Paulus, M. Müller, and A. Klapuri, "State of the art report: Audio-based music structure analysis." in *Proceedings of the 11st International Society for Music Information Retrieval Conference (ISMIR)*. Utrecht, 2010, pp. 625–636.

[30] S. Werf and P. Hendriks, "A constraint-based approach to grouping in language and music," in *Proceedings of the Conference on Interdisciplinary Musicology*. Department of Musicology, University of Graz, 2004.

[31] S. Dai, H. Zhang, and R. B. Dannenberg, "Automatic analysis and influence of hierarchical structure on melody, rhythm and harmony in popular music," *arXiv preprint arXiv:2010.07518*, 2020.

[32] J. A. Jones, B. O. Miller, and D. L. Scarborough, "A rule-based expert system for music perception," *Behavior Research Methods, Instruments, & Computers*, vol. 20, no. 2, pp. 255–262, 1988.

[33] D.-J. Povel, "A theoretical framework for rhythm perception," *Psychological research*, vol. 45, no. 4, pp. 315–337, 1984.

[34] D. F. Rosenthal, "Machine rhythm–computer emulation of human rhythm perception," Ph.D. dissertation, Massachusetts Institute of Technology, 1992.

[35] D. Temperley and D. Sleator, "Modeling meter and harmony: A preference-rule approach," *Computer Music Journal*, vol. 23, no. 1, pp. 10–27, 1999.

[36] M. Hamanaka, K. Hirata, and S. Tojo, "Implementing "a generative theory of tonal music," *Journal of New Music Research*, vol. 35, no. 4, pp. 249–277, 2006.

[37] D. Temperley, "A unified probabilistic model for polyphonic music analysis," *Journal of New Music Research*, vol. 38, no. 1, pp. 3–18, 2009.

[38] M. Hamanaka, K. Hirata, and S. Tojo, "deepgttm-i&ii: Local boundary and metrical structure analyzer based on deep learning technique," in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2016, pp. 3–21.

[39] ——, "deepgttm-iii: simultaneous learning of grouping and metrical structures," in *the 13th International Symposium on Computer Music Multidisciplinary Research (CMMR2017)*, 2017, pp. 161–172.

[40] M. Rohrmeier, "Towards a formalization of musical rhythm," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 621–629.

[41] S. Böck, M. E. Davies, and P. Knees, "Multi-task learning of tempo and beat: Learning one to improve the other." in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 486–493.

[42] S. Böck and M. E. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 12–16.

[43] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[44] S. Durand and S. Essid, "Downbeat detection with conditional random fields and deep learned features." in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016, pp. 386–392.

[45] H. V. Koops, W. B. de Haas, J. Bransen, and A. Volk, "Chord label personalization through deep learning of integrated harmonic interval-based representations," *arXiv preprint arXiv:1706.09552*, 2017.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[47] S. Tojo, K. Hirata, and M. Hamanaka, "Computational reconstruction of cognitive music theory," *New Generation Computing*, vol. 31, no. 2, pp. 89–113, 2013.

[48] J. Serrà, M. Müller, P. Grosche, and J. L. Arcos, "Unsupervised detection of music boundaries by time series structure features," in *AAAI*, 2012.

[49] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, and G. Xia, "POP909: A pop-song dataset for music arrangement generation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*. Montreal, Canada: ISMIR, Oct. 2020, pp. 38–45.

[50] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, 2016.

MID-LEVEL HARMONIC AUDIO FEATURES FOR MUSICAL STYLE CLASSIFICATION

Francisco Almeida, Gilberto Bernardes
Univ. Porto, Faculty of Engineering & INESC TEC
{up201909574,gba}@fe.up.pt

Christof Weiß
International Audio Laboratories Erlangen
christof.weiss@audiolabs-erlangen.de

ABSTRACT

The extraction of harmonic information from musical audio is fundamental for several music information retrieval tasks. In this paper, we propose novel harmonic audio features based on the perceptually-inspired tonal interval vector space, computed as the Fourier transform of chroma vectors. Our contribution includes mid-level features for musical dissonance, chromaticity, dyadicity, triadicity, diminished quality, diatonicity, and wholetoneness. Moreover, we quantify the perceptual relationship between short- and long-term harmonic structures, tonal dispersion, harmonic changes, and complexity. Beyond the computation on fixed-size windows, we propose a context-sensitive harmonic segmentation approach. We assess the robustness of the new harmonic features in style classification tasks regarding classical music periods and composers. Our results align with, slightly outperforming, existing features and suggest that other musical properties than those in state-of-the-art literature are partially captured. We discuss the features regarding their musical interpretation and compare the different feature groups regarding their effectiveness for discriminating classical music periods and composers.

1. INTRODUCTION

Over the last decades, music consumption has shifted from physical media to streaming services comprising large digital collections [1]. Methods for organizing these collections are fundamental for user navigation, browsing, and retrieval. In this context, a significant effort has been devoted to the automatic classification of musical audio signals into style or genre categories within Musical Information Retrieval (MIR) [2–4]. While the traditional approach to such tasks is based on hand-crafted features and classical machine learning, end-to-end deep-learning approaches have led to major improvements [4]. Nevertheless, strategies based on hand-crafted *mid-level* features are still of relevance since they allow interpretable and controllable systems that focus on specific aspects of the music.

© F. Almeida, G. Bernardes, and C. Weiß. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** F. Almeida, G. Bernardes, and C. Weiß, "Mid-level Harmonic Audio Features for Musical Style Classification", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

Existing approaches for style classification mostly rely on timbral or rhythmic mid-level features, which appear suitable for discriminating top-level genres such as pop, rock, jazz, and classical music [2, 5, 6] or sub-genres of popular music [7, 8]. However, such features are less suitable for discriminating sub-genres or historical periods within Western classical music—consider, e. g., the co-existence of solo piano music composed over several centuries [9]. To address this challenge, harmonic features have shown promising results [10–13]. Yet, existing harmonic audio features exhibit two main limitations. First, many of these features focus on low-level and short-term properties, which do not explicitly capture the horizontal or long-term structure of harmony, known to be relevant to style classification. Second, these features do not explicitly consider perceptual qualities, such as the degree of dissonance or the perceptual relationship of sonorities.

To account for the two limitations identified above, we consider in this paper a set of harmonic features based on the Tonal Interval Vector (TIV) space proposed in [14]. Multi-level pitch is mapped into a 6-dimensional complex space whose distances capture perceptual relationships between sonorities. We make the following four main contributions. (1) We consider the TIVs proposed in [14] for style classification. (2) On the TIV space, we advance a set of novel harmonic features for capturing long-term hierarchical harmonic relationships. (3) Moreover, we propose a structural audio segmentation based on harmonic changes and compare this approach to a fixed-window segmentation. (4) To assess the newly proposed harmonic features, we perform experiments for style classification of Western classical music, considering historical periods (eras) and composers as sub-genre taxonomies.

The paper is structured as follows. Section 2 discusses related work on the harmonic description of musical audio and style classification. Section 3 presents novel harmonic features based on the TIV space. Section 4 presents our experimental results using TIV features for style classification compared to the state-of-the-art. Finally, Section 5 presents the conclusions and future work.

2. RELATED WORK

The harmonic description of musical audio typically adopts chroma vectors to represent the energy of pitch-class content and to design higher-level harmonic features. Several methods have been proposed for the extraction of

chroma vectors [15–18]. The Non-Negative Least Squares (NNLS) chroma reduces the impact of overtones and has shown to be one of the most robust chroma vector representations for audio transcription [17].

For describing harmonic properties based on chroma vectors, Weiß et al. proposed a set of template-based chord and interval features [10] as well as tonal complexity features [11] for style period and composer classification within Western classical music. Most of these features are transposition-invariant and therefore do not depend on the key of the piece. These two sets of features are detailed in Sections 2.1 and 2.2. Furthermore, mid-level features for capturing chord transitions over time using Hidden Markov Models were proposed in [13].

2.1 Template-based Features

Motivated by the study on stylistic features from pitch-class sets by Honingh and Bod [19, 20], Weiß et al. [10] propose a set of template-based features (denoted as $\mathbf{F}$) studying the likelihood of a given complementary interval or triad type in a chroma vector $\mathbf{c} = (c_0, c_1, ..., c_{11}) \in \mathbb{R}^{12}$. For example, the likelihood of a perfect fourth/fifth interval, F_{IC5}, results from multiplying c_0 by c_5. To make these features transposition-invariant, the authors sum all cyclic shifts of the same interval in a given chroma vector $\mathbf{c}$. This calculation is simplified by applying a binary template $\mathbf{I}$ according to the desired type of interval or triad. A perfect fourth/fifth interval template is $\mathbf{I} = (1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0)^\top$.

2.2 Tonal Complexity Features

Tonal Complexity features (denoted as $\mathbf{G}$) aim to capture musical attributes such as the amount of tonal variation in musical content. In [11], a total of seven complexity features are defined. For example, G_{CompEntr} represents the Shannon entropy of a chroma vector while $G_{\mathrm{CompFifth}}$ describes the spread of a chroma vector over the circle of fifths. Please refer to [11] for a thorough mathematical definition and musical interpretation of these features.

2.3 Tonal Interval Vectors

TIVs represent multi-level pitch in a geometrical space where vector distances relate to their perceived proximity [14]. The perceptual basis of the TIV space addresses three common limitations in preceding tonal pitch spaces [21–24]. First, it allows the representation and comparison of pitch at multiple time scales, namely individual pitches, chords, and keys. Second, prior knowledge of the key center is not required when measuring pitch distances. Third, it provides an indicator of consonance, lacking in related spaces. Moreover, distances between TIVs capture musical properties such as voice leading and shared interval content. Similar to the approach used in [25], the 6-dimensional complex TIV $\mathbf{T} \in \mathbb{C}^6$ is computed as the Discrete Fourier Transform (DFT) to a chroma vector $\mathbf{c} \in \mathbb{R}^N$

k	IC	Harmonic Quality	Intervals	w_k
1	IC1	Chromaticity	m2/M7	3
2	IC6	Dyadicity	Tritone	8
3	IC4	Triadicity	M3/m6	11.5
4	IC3	Diminished Quality	m3/M6	15
5	IC5	Diatonicity	P4/P5	14.5
6	IC2	Wholetoneness	M2/m7	7.5

Table 1. Harmonic quality and interval category associated with each TIV coefficient magnitude $\frac{\|T_k\|}{w_k}$.

as follows:

$$T_k = w_k \sum_{n=0}^{N-1} \bar{c}_n e^{-\frac{j2\pi kn}{N}}, \quad 1 \leq k \leq 6, \\ \text{with} \quad \bar{c}_n = \frac{c_n}{\sum_{n=0}^{N-1} c_n} \quad , \tag{1}$$

where $N = 12$ is the dimension of the chroma vector. We set $1 \leq k \leq 6$ due to the properties of the DFT by which the remaining coefficients are symmetric. $\mathbf{w} = (3, 8, 11.5, 15, 14.5, 7.5)$ are weights adjusting the contribution of each dimension T_k to improve the perceptual basis of the space in representing musical audio. They adjust the contribution of each coefficient T_k and promote the importance of the most relevant intervals within tonal music, such as fourths/fifths and major and minor thirds/sixths. These weights are derived from empirical dissonance ratings of complementary intervals and triads (major/minor, sus4, augmented, diminished). The adoption of the weights $\mathbf{w}$ have shown to capture perceptual distances between musical audio sonorities irrespective of timbral differences to a greater degree than chroma vectors or an unweighted DFT space [26].

2.4 TIV Basic Features

We now describe a group of features directly computed from TIVs, referred to as TIV Basic (denoted as $\mathbf{B}$). Musical interpretations are attributed to the magnitude $\|T_k\|/w_k$ of each of the six coefficients, evaluating the intervallic content of pitch configurations and its associated harmonic quality (see Table 1). For example, $k = 5$ corresponds to the $B_{\mathrm{Diatonicity}}$ feature. We establish a correspondence between each coefficient magnitude and the six complementary interval categories defined in Western music theory [20].

A TIV's indicator of consonance is computed as its magnitude (vector length) normalized to the norm of the weight vector $\mathbf{w}$, such that:

$$B_{\mathrm{Dissonance}} = 1 - \|\mathbf{T}\|/\|\mathbf{w}\| \quad . \tag{2}$$

3. FEATURE DESIGN IN THE TONAL INTERVAL SPACE

Aiming to take full advantage of the properties of TIVs, we now propose novel mid-level harmonic audio features that consider the harmonic structure and long-term hierarchical dependencies between audio segments.

3.1 TIV Complexity Features

Our first feature group, denoted as **Q**, captures aspects of tonal complexity and consonance, somehow mirroring the features presented in Section 2.2.

3.1.1 Distance Between Audio Segments

Distances between TIVs relate to their perceived similarity such that small distances equate to perceptually similar sonorities. To this end, Euclidean and cosine distance metrics between TIVs, which measure different perceptual characteristics of the signal, have been considered in the literature [14, 27].

The cosine distance θ measures the phase difference between two TIVs, capturing pitch class distances that roughly correspond to voice leading parsimony. Smaller values indicate a higher number of shared pitch classes.

The Euclidean distance metric d captures the phase φ_k and magnitude $\|T_k\|$ differences between TIVs. In addition to measuring the number of shared pitch classes between TIVs, it also accounts for shared interval content.

To exploit the musical properties of these two distance metrics, we propose new features that measure the perceptual distance between consecutive musical audio segments. Given the TIV $\mathbf{T}_n = (T_{0,n}, \ldots, T_{5,n})$ of the n-th segment, the Euclidean (d) and cosine (θ) distances between two consecutive segments is defined as:

$$\begin{aligned} Q_{\text{EucDist}} &= s_n^{\text{euc}} = d\{\mathbf{T}_n, \mathbf{T}_{n+1}\}, \\ Q_{\text{CosDist}} &= s_n^{\text{cos}} = \theta\{\mathbf{T}_n, \mathbf{T}_{n+1}\} \end{aligned} \tag{3}$$

The metrics can be applied at multiple hierarchies (or time scales) by adopting musical audio segments of different sizes. While short segments capture chord changes, larger segments can capture the overall tonal structure (e.g., key modulations).

3.1.2 Tonal Dispersion

The tonal dispersion of the n-th audio segment n is defined as the distance of its respective TIV to the tonal center $\overline{\mathbf{T}}$:

$$\begin{aligned} Q_{\text{EucTonalDisp}} &= u_n^{\text{euc}} = d\{\mathbf{T}_n, \overline{\mathbf{T}}\}, \\ Q_{\text{CosTonalDisp}} &= u_n^{\text{cos}} = \theta\{\mathbf{T}_n, \overline{\mathbf{T}}\} \end{aligned} \tag{4}$$

The tonal center of a piece corresponds to the TIV of the pitch-class distribution averaged across its entire duration. The tonal dispersion indicates how much the harmonic content of a given segment deviates from the tonal center $\overline{\mathbf{T}}$ and can indicate the degree of modulations across the piece or the segments with non-diatonic pitch classes (i.e., further away from the tonal center). This feature can help discriminate later classical musical periods that typically have larger tonal dispersion [28]. To compute the distance of a given segment n from the tonal center, we adopt both the Euclidean and cosine distance metrics to capture the different harmonic aspects mentioned in Section 3.1.1.

3.1.3 TIV Entropy

Related literature has been adopting Shannon entropy to capture the harmonic complexity within musical style [11, 29]. Amiot [30] recently proposed the computation of harmonic complexity from symbolic pitch-class distributions as the Shannon entropy of its Fourier coefficients. Following [30], we propose a TIV entropy feature to capture harmonic complexity from musical audio. The TIV entropy is high for random pitch-class distributions and low for pitch-class distributions that exhibit some degree of organization (i.e., periodicity or sparseness). We define the TIV entropy feature as the entropy of the normalized coefficient magnitudes of a TIV $\mathbf{T}$:

$$Q_{\text{TIVEntropy}} = \sum_{k=1}^{6} -p_k \log p_k, \quad p_k = \frac{\|T_k\|}{\sum_{j=1}^{6} \|T_j\|} \tag{5}$$

While entropy has been shown to capture different stylistic attributes depending on the time scale under analysis [29], we demonstrate in Table 2 the TIV entropy of various pitch-class sets. Lower TIV entropy (i.e., denoting greater organization) includes fairly common pitch-class distributions such as diatonic scales, triads, and tetrads. Higher TIV entropy denotes musical structures with less common adoption in the tonal music lexica, such as the chromatic set and its many subsets.

Pitch class set	TIV Entropy
Octatonic scale	0.3551
Diminished seventh chord	0.3552
Whole tone scale	0.5268
Diatonic scale	1.2444
Pentatonic scale	1.2972
Harmonic minor scale	1.4927
Minor seventh chord	1.5542
Single note	1.6767
Chromatic scale	1.6906

Table 2. TIV entropy of several pitch-class sets. For details on the elements of each set please refer to [30].

3.2 Harmonic Rhythm Features

Harmonic rhythm is the rate at which the chords change in a musical piece. It can provide a prominent indicator to distinguish style periods. For example, a fast harmonic rhythm is a characteristic of the Baroque period [31] in comparison with the large-scale structures of a Romantic symphony. To this end, we advance new features (denoted as **R**) that study the rate and magnitude of harmonic changes based on the Harmonic Change Detection Function (HCDF) proposed in [32]. The value ξ_n of the HCDF at frame n is defined as the rate of change between the TIVs $\mathbf{T}_{n-1}$ and $\mathbf{T}_{n+1}$ after Gaussian smoothing and is computed using the Euclidean distance.

Let us now consider Ξ_m as the set of frame indices of the peaks of the HCDF, considering all local maxima as peaks. We define the inter-peak interval feature as the difference between the frame numbers of consecutive peaks of the HCDF:

$$R_{\text{HCDFPeakInterval}} = \Delta_m = \Xi_{m+1} - \Xi_m \tag{6}$$

As an additional feature, we consider the magnitude of the peaks, $R_{\text{HCDFPeakMag}}$, given by ξ_Y, with $Y = \Xi_m$, which captures the degree of the harmonic changes.

3.3 Temporal Resolution of Tonal Interval Vectors

We detail two segmentation strategies for the feature extraction process: multiple fixed-size segment resolutions (Section 3.3.1) and variable-size segments resulting from the analysis of harmonic changes (Section 3.3.2).

3.3.1 Fixed-size Segmentation with Multiple Resolutions

Fixed-size segments with equal duration are adopted at multiple time scales or resolutions to capture the piece's different harmonic or tonal dimensions. Following [10–12], we consider four time resolutions in our work: 100ms, 500ms, 10s, and global (i.e., the entire piece or excerpt under analysis). Smaller time scales (e.g., 100ms and 500ms) capture finer musical elements such as individual notes, intervals, and chords. Larger time scales (e.g., 10s and global) capture higher harmonic structures at the level of structural parts or the overall piece, enhancing the harmonic changes in the large-scale tonal structure of the composition (such as key and modulations) and the harmonic trajectories of the horizontal structure (such as chord progressions).

3.3.2 Harmonic Structural Segmentation

We define a context-sensitive segmentation at the peaks of the HCDF (see Section 3.2). This strategy considers prominent harmonic changes as segment boundaries, therefore producing segments of varying duration according to the specific harmonic changes in the audio content. The use of context-sensitive segmentation in the feature extraction process of musical analysis systems has been shown to improve their accuracy compared to fixed segmentation approaches, which are less aware of musical structure [33, 34].

4. EXPERIMENTS AND RESULTS

To assess the degree to which the harmonic information conveyed by TIV harmonic audio features in Section 3 discriminate musical style, we evaluate the proposed features in two classical music style classification tasks from musical audio: historical periods (e.g., Baroque or Romantic) and composers. Furthermore, we compare the accuracy of the above features to the state-of-the-art template-based and complexity features described in Sections 2.1 and 2.2. To study the accuracy of different groups of features, we consider the two following sets: (template-based $\mathbf{F}$ and tonal complexity $\mathbf{G}$) and (TIV basic $\mathbf{B}$, TIV complexity $\mathbf{Q}$, harmonic rhythm $\mathbf{R}$). Finally, we inspect the impact of different context-sensitive vs. fixed-window segmentation strategies.

4.1 Experimental Procedure

For our experiments, we consider the *Cross-Era* [1] and *Cross-Composer* [2] datasets, which include 1600 and 1100 pieces, respectively. In detail, the *Cross-Era* dataset has 400 pieces per classical style period and features the following four periods: Baroque, Classical, Romantic, and Modern. The *Cross-Composer* dataset includes 100 pieces for each of the 11 featured classical music composers across all style periods. These datasets are further divided into the subsets presented in Table 3. The datasets provide pre-extracted NNLS chroma features at a resolution of 100ms (10Hz) for each piece.

Dataset	No. Classes (Z)	Items per Class
Cross-Era-Full	4	400
Cross-Era-Piano	4	200
Cross-Era-Orchestra	4	200
Cross-Comp-11	11	100
Cross-Comp-5	5	100

Table 3. Balanced subsets obtained from the *Cross-Era* and *Cross-Composer* datasets [12].

Based on [12], we employ the following classification procedure. Using the subsets listed in Table 3, we compute the features and calculate their piece-wise mean (μ) and standard deviation (σ). Then, we perform a stratified split into three cross-validation (CV) folds, one for testing and two others for training the model. Next, we use the two training folds to compute a Linear Discriminant Analysis (LDA) matrix to reduce the dimensionality of all three folds to $L = Z - 1$ with Z being the number of classes per task. We then perform a five-fold grid search on the two training folds to train an SVM classifier while optimizing its hyperparameters. We conduct this procedure three times, with each fold serving for testing once. To evaluate the robustness of the model concerning the random distribution into folds, we repeat the procedure ten times with re-initialized folds to calculate the accuracy deviation between the different classes (i.e., inter-class deviation).

To counteract problems stemming from adopting tracks from the same CD recording on the training and test folds (known as the album effect [35]), we take additional measures to prevent the model from adapting to the acoustic conditions of specific recordings. To this end, inspired by [11], we apply a composer filter (for period classification) or a performer filter (for composer classification) during the CV procedure that forces pieces from the same composer/performer to be placed within the same fold, thus avoiding overfitting while making the task more challenging and closer to real-world application scenarios.

4.2 Influence of Different Types of Segmentation

We analyze the impact of different audio segmentation strategies on the classification of classical music periods. Table 4 displays the classification accuracy for the TIV Basic ($\mathbf{B}$), TIV Complexity ($\mathbf{Q}$), and Harmonic Rhythm ($\mathbf{R}$) feature groups, as well as the combination of all the above features (through concatenation), on the *Cross-Era* dataset. As segmentation strategies, we consider fixed-size segmentation with a single resolution of 100ms (FS), fixed-size segmentation with multiple resolutions (MR)—100ms, 500ms, 10s, and global—and harmonic structural

[1] https://www.audiolabs-erlangen.de/resources/MIR/cross-era
[2] https://www.audiolabs-erlangen.de/resources/MIR/cross-comp

	FS	MR	HS	MR + HS
Cross-Era-Piano				
TIV Basic (B)	57.54%	**66.84%**	51.65%	65.64%
TIV Complexity (Q)	56.03%	**57.99%**	56.07%	57.67%
Harm. Rhythm (R)	-	-	21.26%	-
Combined	62.61%	**65.47%**	48.97%	63.80%
Cross-Era-Orchestra				
TIV Basic (B)	66.84%	74.80%	64.22%	**75.34%**
TIV Complexity (Q)	67.35%	69.87%	64.80%	**70.00%**
Harm. Rhythm (R)	-	-	28.31%	-
Combined	73.80%	77.19%	68.89%	**77.78%**
Cross-Era-Full				
TIV Basic (B)	60.41%	**70.13%**	57.24%	70.08%
TIV Complexity (Q)	55.97%	62.18%	57.35%	**62.86%**
Harm. Rhythm (R)	-	-	21.65%	-
Combined	64.94%	**71.63%**	62.30%	70.99%

Table 4. Classification accuracy for different types of segmentation: fixed-size (FS), fixed-window with multiple temporal resolutions (MR), harmonic structural segmentation (HS), and a combination of the last two approaches.

segmentation (HS). Additionally, we consider combining the two approaches presented in Section 3.3 (MR + HS).

For the *Cross-Era-Piano* and *Cross-Era-Full* datasets, the MR strategy shows slightly higher accuracy than other segmentation strategies. For other scenarios such as the *Cross-Era-Orchestra* dataset, combining the MR and HS approaches leads to similar or slightly better results. Overall, the MR strategy obtains the highest accuracy values in most cases and is less computationally expensive than the HS approach. Therefore, we opt for the MR strategy in all subsequent experiments, using the HS strategy only for the harmonic rhythm feature group $\mathbf{R}$.

4.3 Style Period and Composer Classification

Using the MR strategy, we perform style period and composer classification experiments on a larger set of feature combinations and show the results in Table 5. We first compare the template-based ($\mathbf{F}$) and TIV basic ($\mathbf{B}$) feature groups, which capture similar harmonic aspects. Indeed, we observe that these groups perform similarly on most datasets, with accuracy differences of 2.98% and 4.74% on the *Cross-Era-Piano* and *Cross-Comp-5* datasets, respectively. The tonal complexity ($\mathbf{G}$) and TIV complexity ($\mathbf{Q}$) groups show a greater performance accuracy difference in the *Cross-Era-Piano* dataset (7.85%) and a smaller difference in the *Cross-Era-Full* dataset (3.33%). Concerning the two largest datasets (*Cross-Era-Full* and *Cross-Comp-11*), the best feature groups result from combining TIV features with those proposed in [10, 11], achieving 74.04% and 38.25% accuracy, which correspond to an improvement of 2.88% and 4.74%, respectively, compared to using only the features proposed in previous work.

From these findings, we draw two main conclusions. First, conceptually similar feature groups lead to quite similar classification accuracies—an encouraging finding, which proves that our approach is valid (a kind of "sanity check" against [10, 11]) and indicates that the features exhibit the intended mid-level semantic properties (feature groups describing related harmonic properties behave similarly for classification). Second, the novel features at least partially capture complementary information such that their combination improves upon individual groups. Overall, we do not reach the state-of-the-art results on the *Cross-Era-Orchestra* and *Cross-Era-Full* subsets, which were obtained in [13] using chord transition features in combination with template-based and complexity features, surpassing our results by 6.01% and 4.16%. However, since TIV-based features improve upon the ones of [10,11], we hypothesize that, in combination with the chord transition features of [13], they might be able to reach even better classification accuracies.

4.4 Harmonic Features Correlation

We adopt hierarchical clustering to assess the degree of information (and redundancy) of harmonic audio features. Figure 1 shows the hierarchical clustering of all harmonic features detailed in Sections 2.2, 2.1, and 3, which we adopt in our experiments. Features are computed at a fixed-size 100ms resolution with average and standard deviation statistics.

Template-based ($\mathbf{F}$) and TIV basic ($\mathbf{B}$) features appear strongly correlated, often ending up in the same cluster. This suggests they may be capturing similar musical properties. For example, the two top-most features $F_{\mathrm{IC5},\sigma}$ and $B_{\mathrm{Diatonicity},\sigma}$ are direct neighbours and are both based on perfect-fifth relationships. Tonal Complexity ($\mathbf{G}$) and TIV Complexity ($\mathbf{Q}$) features also appear correlated, albeit to a slightly lower degree. For example, the green cluster is mainly composed of features from these groups. On the other hand, the red cluster contains several TIV Complexity features and none from the Tonal Complexity group, suggesting that those, in particular, describe different harmonic characteristics. The adopted global statistics, mean and standard deviation, tend to be grouped under the same cluster, suggesting to capture complementary information.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel set of mid-level and perceptually-inspired harmonic audio features based on TIVs, which provide indicators for musical dissonance, chromaticity, dyadicity, triadicity, diminished quality, diatonicity, and whole-toneness, as well as quantify perceptual relations of short- and long-term harmonic structures, tonal dispersion, harmonic changes, and complexity. Features are computed using context-sensitive harmonic structure segmentation and a fixed-size segmentation with multiple temporal resolutions.

Proposed TIV harmonic features were assessed in two classical music style period and composer classification tasks using five different datasets. The novel TIV harmonic features have been compared to state-of-the-art harmonic features proposed in [10–12] and showed similar results, slightly outperforming in four out of the five datasets, with an improvement of up to 4.74%. A high degree of re-

		Cross-Era-Piano	Cross-Era-Orch	Cross-Era-Full	Cross-Comp-11	Cross-Comp-5
Tonal [10, 11]	**Template-based (F)**	**69.82 ±13.45**	75.25 ±12.16	70.51 ±11.57	37.41 ±19.96	50.01 ±19.77
	Tonal Complexity (G)	65.84 ±12.80	71.68 ±14.09	65.51 ±13.26	29.74 ±21.33	43.32 ±25.68
	Combined (F, G)	67.77 ±15.36	75.23 ±12.17	71.16 ±11.42	36.97 ±21.59	48.86 ±22.47
TIV	**TIV Basic (B)**	66.84 ±15.31	74.80 ±11.22	70.13 ±12.52	37.84 ±20.66	**54.75 ±17.96**
	TIV Complexity (Q)	57.99 ±17.67	69.87 ±12.87	62.18 ±13.71	29.61 ±19.77	43.16 ±19.06
	Harm. Rhythm (R)	21.26 ±28.06	28.31 ±21.51	21.65 ±26.44	7.77 ±17.64	16.47 ±19.35
	Basic + Comp. (B, Q)	65.59 ±16.77	76.68 ±10.35	71.50 ±12.03	37.82 ±20.30	53.40 ±16.85
	Combined (B, Q, R)	65.47 ±16.63	**77.19 ±9.89**	71.63 ±12.03	37.83 ±20.01	53.75 ±16.65
Combined	**F, G, B, Q, R**	64.39 ±16.27	76.70 ±11.41	73.78 ±10.80	**38.25 ±21.01**	49.72 ±19.25
Combined, no R	**F, G, B, Q**	64.78 ±15.80	76.56 ±11.29	**74.04 ±10.7**	37.89 ±21.57	50.44 ±18.72
Tonal+Transitions [13]		73.2	83.2	78.2	–	–

Table 5. Classification results for several feature groups across the *Cross-Era* and *Cross-Composer* datasets. The values on the table represent the mean classification accuracy and inter-class deviation, both expressed as percentages. For comparison, the state-of-the-art results for the *Cross-Era* dataset reported in [13] were also included.

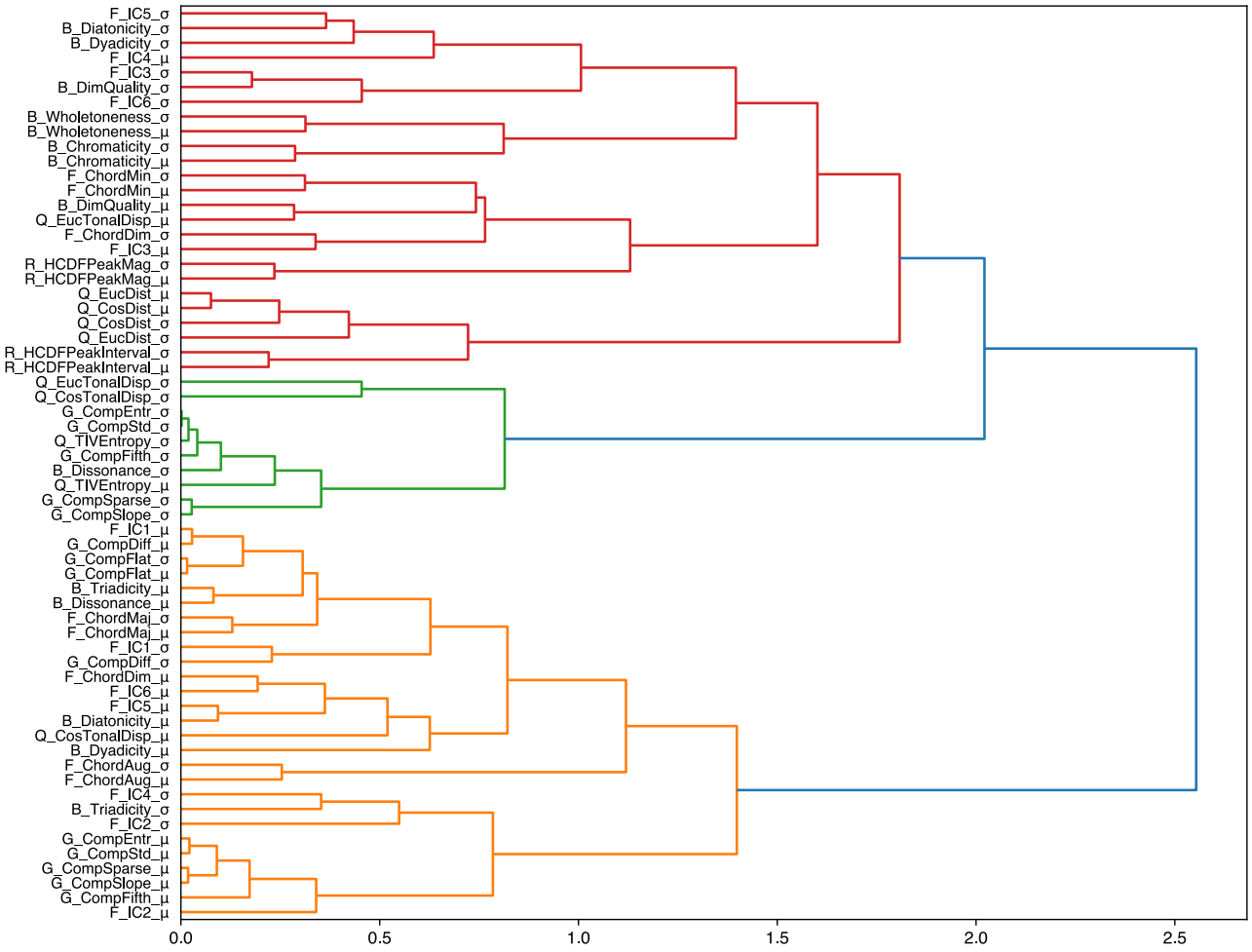


Figure 1. Hierarchical clustering of harmonic features computed at a 100ms resolution. The clustering is based on Spearman correlation distances using Ward's linkage method.

dundancy with existing state-of-the-art features has been found. However, the results suggest that new information is captured in the proposed TIV harmonic features, namely in what concerns the horizontal or long-term harmonic structure, e.g., tonal dispersion and distance between audio segments, and harmonic rhythm. While the context-sensitive segmentation strategy introduced slight improvements to classification accuracy, these do not seem to outweigh its higher computational cost.

Finally, for research reproducibility, we made the implementation of the proposed system publicly available online at `github.com/fcfalmeida/style-ident`. In future work, we may consider optimizing the segmentation strategy, expanding it to account for multiple time scales, conducting classification experiments on other musical genres, and experimenting with different machine learning approaches such as deep learning.

Acknowledgements: The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and

Fraunhofer Institut für Integrierte Schaltungen IIS. "Experimentation in music in Portuguese culture: History, contexts and practices in the 20th and 21st centuries" (POCI-01-0145-FEDER-031380) co-funded by the European Union through the Operational Program Competitiveness and Internationalization, in its ERDF component, and by national funds, through the Portuguese Foundation for Science and Technology.

6. REFERENCES

[1] K. Negus, "From creator to data: the post-record music industry and the digital conglomerates," *Media, Culture and Society*, vol. 41, no. 3, pp. 367–384, 2019.

[2] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[3] S. Argamon, K. Burns, and S. Dubnov, *The structure of style: Algorithmic approaches to understanding manner and meaning*. Springer Berlin Heidelberg, 2010.

[4] J. Nam, K. Choi, J. Lee, S. Chou, and Y. Yang, "Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 41–51, 2019.

[5] C. Weihs, U. Ligges, F. Mörchen, and D. Müllensiefen, "Classification in music research," *Advances in Data Analysis and Classification*, vol. 1, no. 3, pp. 255–291, 2007.

[6] T. Li, M. Ogihara, and Q. Li, "A comparative study on content-based music genre classification," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Toronto, Canada, 2003, pp. 282–289.

[7] V. Tsatsishvili, "Automatic subgenre classification of heavy metal music," Master's Thesis, University of Jyväskylä, Jyväskylä, Finland, 2011.

[8] D. Gärtner, C. Zipperle, and C. Dittmar, "Classification of electronic club-music," in *Proceedings of the Deutsche Jahrestagung für Akustik (DAGA)*, Berlin, Germany, 2010.

[9] B. van 't Spijker, "Classifying classical piano music into time period using machine learning," in *Proceedings of the Twente Student Conference on IT*, Enschede, The Netherlands, 2020.

[10] C. Weiß, M. Mauch, and S. Dixon, "Timbre-invariant audio features for style analysis of classical music," in *Proceedings of the Joint Conference 40th International Computer Music Conference (ICMC) and 11th Sound and Music Computing Conference (SMC)*, Athens, Greece, 2014, pp. 1461–1468.

[11] C. Weiß and M. Müller, "Tonal complexity features for style classification of classical music," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brisbane, Australia, 2015, pp. 688–692.

[12] C. Weiß, "Computational methods for tonality-based style analysis of classical music audio recordings," Ph.D. dissertation, Ilmenau University of Technology, Ilmenau, Germany, 2017.

[13] C. Weiß, F. Brand, and M. Müller, "Mid-level chord transition features for musical style analysis," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Brighton, UK, 2019, pp. 341–345.

[14] G. Bernardes, D. Cocharro, M. Caetano, C. Guedes, and M. E. Davies, "A multi-level tonal interval space for modelling pitch relatedness and musical consonance," *Journal of New Music Research*, vol. 45, no. 4, pp. 281–294, oct 2016.

[15] E. Gómez, "Tonal description of music audio signals," PhD Thesis, Universitat Pompeu Fabra, Barcelona, Spain, 2006.

[16] M. Stein, B. Schubert, M. Gruhne, G. Gatzsche, and M. Mehnert, "Evaluation and comparison of audio chroma feature extraction methods," vol. 1, pp. 324–332, 01 2009.

[17] M. Mauch and S. Dixon, "Approximate note transcription for the improved identification of difficult chords," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, 2010, pp. 135–140.

[18] Y. Wu and W. Li, "Automatic audio chord recognition with MIDI-trained deep feature and BLSTM-CRF sequence decoding model," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 2, pp. 355–366, 2019.

[19] A. Honingh and R. Bod, "Pitch class set categories as analysis tools for degrees of tonality," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, 2010, pp. 459–464.

[20] ——, "Clustering and classification of music by interval categories," in *Proceedings of the International Conference on Mathematics and Computation in Music*. Berlin and Heidelberg, Germany: Springer, 2011, pp. 346–349.

[21] F. Lerdahl, "Tonal Pitch Space," *Music Perception*, vol. 5, no. 3, pp. 315–349, 1988.

[22] G. Gatzsche, M. Mehnert, D. Gatzsche, and K. Brandenburg, "A symmetry based approach for musical tonality analysis," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Vienna, Austria, 2007, pp. 207–210.

[23] E. Chew, "Towards a mathematical model of tonality," PhD Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2000.

[24] R. L. Cohn, "Neo-riemannian operations, parsimonious trichords and their tonnetz representations," *Journal of Music Theory*, vol. 41, no. 1, pp. 1–66, 1997.

[25] C. Viaccoz, D. Harasim, F. C. Moss, and M. Rohrmeier, "Wavescapes: A visual hierarchical analysis of tonality using the discrete fourier transform," *Musicae Scientiae*, 2022.

[26] G. Bernardes, M. E. Davies, and C. Guedes, "A hierarchical harmonic mixing method," in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2017, pp. 151–170.

[27] A. Ramires, G. Bernardes, M. Davies, and X. Serra, "TIV.lib: an open-source library for the tonal description of musical audio," *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20), Vienna, Austria*, no. September, pp. 304–309, 2020.

[28] E. Wold and J. Tenney, "A history of consonance and dissonance," *Computer Music Journal*, vol. 13, p. 94, 1989.

[29] J. L. Snyder, "Entropy as a measure of musical style: The influence of a priori assumptions," *Music Theory Spectrum*, vol. 12, no. 1, pp. 121–160, 1990.

[30] E. Amiot, "Entropy of fourier coefficients of periodic musical objects," *Journal of Mathematics and Music*, vol. 15, no. 3, pp. 235–246, 2021.

[31] G. Haydon and M. F. Bukofzer, "Music in the Baroque Era from Monteverdi to Bach," *The Journal of Aesthetics and Art Criticism*, vol. 7, no. 3, p. 262, mar 1949.

[32] P. Ramoneda and G. Bernardes, "Revisiting harmonic change detection," in *149th Audio Engineering Society Convention 2020, AES 2020*, oct 2020. [Online]. Available: https://www.researchgate.net/publication/342563614

[33] D. P. Ellis, C. V. Cotton, and M. I. Mandel, "Cross-correlation of beat-synchronous representations for music similarity," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, 2008, pp. 57–60.

[34] J. P. Bello and J. Pickens, "A robust mid-level representation for harmonic content in music signals," in *ISMIR 2005 - 6th International Conference on Music Information Retrieval*, 2005, pp. 304–311.

[35] E. Pampalk, A. Flexer, and G. Widmer, "Improvements of audio-based music similarity and genre classification," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, London, UK, 2005, pp. 628–633.

DISTORTION AUDIO EFFECTS: LEARNING HOW TO RECOVER THE CLEAN SIGNAL

Johannes Imort[♮] **Giorgio Fabbro**[♯] **Marco A. Martínez-Ramírez**[♭]
Stefan Uhlich[♯] **Yuichiro Koyama**[♭] **Yuki Mitsufuji**[♭]
[♮]RWTH Aachen University, Germany [♯]Sony Europe B.V., Stuttgart, Germany
[♭]Sony Group Corporation, Tokyo, Japan
johannes.imort@rwth-aachen.de, giorgio.fabbro@sony.com

ABSTRACT

Given the recent advances in music source separation and automatic mixing, removing audio effects in music tracks is a meaningful step toward developing an automated remixing system. This paper focuses on removing distortion audio effects applied to guitar tracks in music production. We explore whether effect removal can be solved by neural networks designed for source separation and audio effect modeling.

Our approach proves particularly effective for effects that mix the processed and clean signals. The models achieve better quality and significantly faster inference compared to state-of-the-art solutions based on sparse optimization. We demonstrate that the models are suitable not only for declipping but also for other types of distortion effects. By discussing the results, we stress the usefulness of multiple evaluation metrics to assess different aspects of reconstruction in distortion effect removal.

1. INTRODUCTION

With the emergence of musical recordings, audio effects have become indispensable in the music production process. They are used by musicians as a creative tool to alter the sound of their instruments, and by sound engineers to craft a balanced mix from multiple recording tracks [1].

For the task of mixing and automatic remixing [2], the *dry* (i.e., unprocessed) source tracks are required. Given the recent advances in automatic mixing [3, 4] and music source separation (MSS) [5], a system could facilitate the adjustment of a stereo mixture to the taste and preferences of the user similar to [6]. However, when separating sources with a system trained on music stems (e.g., `MUSDB18` [7]) the mixing process is not considered, and, hence, the output of such a system contains the *wet* (i.e., processed) signal. As nonlinear distortion is one of the most commonly used effects for electric instruments, this work focuses on musical distortion effects that are used for aesthetic means (e.g., guitar overdrive/distortion pedals) and applied in the process of mixing (e.g., tape saturation). These distortion effects result in added harmonics, intermodulation distortion, and a compressed sound [8].

This paper investigates different deep neural network (DNN) approaches regarding their applicability to the audio effect removal problem. Our contributions can be summarized as follows:

- We show that recovering the clean signal from clipped or overdriven guitar signals can be efficiently solved with neural networks designed for source separation. The models achieve high quality and fast inference in contrast to solutions based on sparse optimization.
- We found that the superior performance of the models evaluated on the de-overdrive task can be traced back to superimposing the overdriven signal with the clean signal. We show that the architectures are suitable not only for declipping but also for other types of distortion effects.
- By discussing the results, we highlight that the metrics under evaluation prove beneficial in measuring different aspects of the reconstruction and can be advised for further investigations.

This work is organized as follows: Sec. 2 gives a formal introduction to audio effect removal and introduces the types of distortions that were used throughout this study. Sec. 3 briefly discusses previous work on iterative and DNN-based declipping approaches. The methods under evaluation are outlined in Sec. 4. Sec. 5 describes the data that were used for training, reports details about the experimental setup, and presents the chosen objective evaluation metrics. In Sec. 6, we evaluate the results of the comparative study of four different neural network architectures on the task of distortion removal in guitar signals for the SoX overdrive implementation. Then, we compare the same architectures on the declipping task to one state-of-the-art declipping algorithm using guitar signals as well as generic music signals. Lastly, Sec. 7 gives a conclusion and presents an outlook for future work. Audio examples are available online at *joimort.github.io/distortionremoval/*.

© J. Imort, G. Fabbro, M. A. Martínez-Ramírez, S. Uhlich, Y. Koyama, and Y. Mitsufuji. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J. Imort, G. Fabbro, M. A. Martínez-Ramírez, S. Uhlich, Y. Koyama, and Y. Mitsufuji, "Distortion Audio Effects: Learning How to Recover the Clean Signal", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

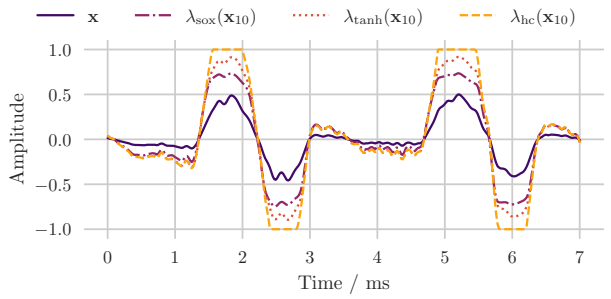


Figure 1. Example of different distortion types applied to a guitar signal. The input signal $x(k)$ is amplified before being modified by a wave-shaper function.

2. PROBLEM FORMULATION

We introduce audio effect removal as the task of recovering the original discrete audio signal $\mathbf{x} \in \mathbb{R}^n$ from the processed discrete signal $\mathbf{y} \in \mathbb{R}^n$, which is obtained by applying the possibly nonlinear and time-varying function f to the signal $\mathbf{x}$:

$$\mathbf{y} = g(\mathbf{x}) = \alpha f(\mathbf{x}) + (1-\alpha)\mathbf{x}, \tag{1}$$

with $\alpha \in [0, 1]$ denoting the weight of the wet signal, and g the summation function of the dry and wet signal. The goal is to find an estimation of the original signal $\hat{\mathbf{x}}$ by estimating the inverse function $\hat{\mathbf{x}} = \hat{g}^{-1}(\mathbf{y})$.

Generally, distortion effects clip the input signal and can be divided into systems that apply hard-clipping or soft-clipping. As this work focuses on both types of distortion, we introduce the following generic formulation of a wave-shaper that maps the amplified signal $\mathbf{x}_\gamma = 10^{\frac{\gamma}{20}}\mathbf{x}$ to a fixed range:

$$f(\mathbf{x}) = \lambda(\mathbf{x}_\gamma) \quad \text{with} \quad \lambda : \mathbb{R} \to [-\theta_\text{c}, \theta_\text{c}]. \tag{2}$$

Here, γ denotes the gain in decibels, λ the arbitrary wave-shaper function, and θ_c the fixed clipping threshold. For the case of hard-clipping, λ is defined as:

$$\lambda_{\text{hc}}(x_{\gamma,k}) = \begin{cases} x_{\gamma,k}, & \text{if } |x_{\gamma,k}| \leq \theta_\text{c} \\ \theta_\text{c}\text{sgn}(x_{\gamma,k}) & \text{otherwise,} \end{cases} \tag{3}$$

with sgn denoting the sign function, and $x_{\gamma,k}$ the k-th time sample of the amplified signal. Fig. 1 highlights the difference between different distortion types. Hard-clipping cuts off the amplitude when it exceeds a defined threshold (as typical for saturation in digital signal processing). Soft-clipping (e.g., $\lambda_{\tanh}(x_{\gamma,k}) = \tanh(x_{\gamma,k})$) gradually applies a smooth transition before reaching a fully saturated state (as typical for saturation in analog amplifiers).

Modeling the characteristics of distortion pedals in reality is more complex: [9] provides an overview on different methods and discusses DNN-based approaches. In order to simplify the problem for our investigation, we focus on wave-shaping. The overdrive algorithm of the audio editing software SoX [10] serves as an example of soft-clipping (λ_{sox}) but, in contrast to (2), it is dependent on previous samples. Furthermore, it blends the wet signal with the dry one ($\alpha < 1$).

3. RELATED WORK

To the best of our knowledge, there has been no previous research on distortion audio effect removal. Therefore, this section outlines the most relevant iterative and DNN-based declipping approaches since declipping is a special case of distortion audio effect removal.

3.1 Iterative Declipping Methods

Previous research on approaches to declipping has focused mainly on unsupervised algorithms that recover the signal under the assumption of a generic regularization such as signal sparsity [11]. Usually, these approaches target the hard-clipping case only (see (3)). While early approaches were based on auto-regressive models, (e.g., [12]) recent state-of-the-art methods evolved by combining ideas from inverse problems and sparse regularization [13].

Recently, [11] and [13] discussed popular declipping algorithms. One of the current state-of-the-art methods is A-SPADE, which will serve as a baseline for this study. The algorithm is briefly introduced in Sec. 4.

3.2 DNN-Based Declipping

In contrast to iterative algorithms, to date, there are only few contributions comprising supervised DNNs.

Kashani *et al.* [14] introduced a declipping method based on the U-Net architecture [15]. It operates on magnitude spectrograms while the waveform of the output is obtained by reusing the phase information from the distorted input signal. The system is trained and evaluated on pairs of hard-clipped and clean speech samples.

Mack and Habets [16] proposed an architecture comprising a BLSTM-based deep filtering method that works on complex spectrograms and hence also considers phase information. Similar to [14], they train the system on speech data only. Unlike any other approach, they not only investigate the system on the hard-clipping case but also on the soft-clipping case.

Tanaka *et al.* recently proposed APPLADE [17], a declipping method that takes advantage of the sparse optimization techniques described above together with deep learning. Accordingly, they embed a DNN in the iterative algorithm to enhance the thresholding operation. They report a slightly higher performance than previous algorithms, better robustness to mismatches between training and test data, and faster inference.

4. METHODS

This section describes four neural network architectures that we selected from the literature and evaluated on the distortion removal task. We approach the distortion effect removal problem from the perspective of filtering the added harmonics and intermodulation distortion, similar to [16]. Therefore, we include one model from the domain

of audio effect modeling and three architectures originally proposed for music source separation. [1] For the latter models, instead of multiple output channels for different sources, we use only one output channel and consider them as general audio-to-audio transformation architectures.

CRAFx was proposed as a system for modeling time-varying audio effects with a neural network [9, 18]. The end-to-end model operates on the signal in the time domain and is divided into an adaptive front-end (encoder), a bi-directional long-short-term-memory (BLSTM)-based structure that applies the modeled effect in the latent space, and a synthesis back-end (decoder). In contrast to the other DNN-based models presented in this section, this architecture employs architectural priors (e.g., learnable nonlinear activations) in the context of audio effects.

Open-Unmix (UMX) was introduced as a reference implementation for music source separation [19]. The architecture is based on the BLSTM model from [20] and uses magnitude spectrograms as input features. The essential element of Open-Unmix is its three-layer BLSTM network that enables to learn both long- and short-time dependencies [21].

An element-wise multiplication of the input spectrograms with the estimated masks yields the final output. Commonly, spectrogram-based source separation models are compared with the oracle performance of an ideal ratio mask (IRM) that is defined as the ratio between the reference and the test spectrogram [22] in decibels. For reconstruction, the phase of the input signal is used. The model was adapted for a sampling rate of $f_s = 16\,\mathrm{kHz}$. We include this model in our evaluation as a standard frequency domain MSS model that relates to the BLSTM-based declipping model from [16] (cf. Sec. 3.2).

Wave-U-Net was proposed as one of the first end-to-end approaches for music source separation based on time domain signals [23]. Hence, it incorporates not only the magnitude but also the phase of music signals. It adapts the U-Net architecture [15] to one-dimensional audio signals. We decreased the models' number of learnable parameters from 17M to approximately 1M by reducing the number of layers from 12 to 8 resulting in a reduced receptive field, as we experienced overfitting for our datasets. Since the model was successfully employed not only for source separation but automatic mixing as well [3], we assume a general suitability for audio-to-audio transformation tasks. Moreover, the model is highly related to the U-Net-based declipping model from [14] (cf. Sec. 3.2).

Demucs was initially designed to be an end-to-end model for music source separation in the time domain [24]. While it builds on the Wave-U-Net model, it introduces several improvements to the architecture. The model comprises a convolutional encoder, a BLSTM structure, and a convolutional decoder. As for Wave-U-Net, we decreased the number of trainable parameters from 66M to 1M by reducing the number of blocks from 12 to 6 resulting in a reduced receptive field.

A-SPADE was introduced as a sparsity-based declipping algorithm that outperforms previous similar approaches [25]. For each time frame of the clipped signal $\mathbf{y}$, it approximates a solution of the following problem:

$$\min_{\mathbf{x},\mathbf{z}} ||\mathbf{z}||_0 \text{ s.t. } \mathbf{x} \in \Gamma(\mathbf{y}) \text{ and } ||\mathcal{F}(\mathbf{x}) - \mathbf{z}||_2 \leq \epsilon, \quad (4)$$

where $\mathbf{z}$ denotes the unknown discrete Fourier coefficients of each time frame and $\mathcal{F}$ the Fourier transform operator. Γ is defined as the feasible space of solutions (i.e., clipping consistency constraint). We included the algorithm in the evaluation of the declipping task as a baseline that delivers state-of-the-art performance [11, 13].

5. EXPERIMENTS

Our experiments focus on the following three scenarios: **a)** We conducted experiments on guitar recordings that were processed using the overdrive algorithm of the audio editing software SoX [10] (CEG-OD). **b)** We performed the same experiments as in the previous scenario while processing the same clean guitar recordings with hard-clipping (CEG-HC). **c)** We tested the systems on an extensive dataset comprising various hard-clipped sounds (SignalTrain-HC) to evaluate their performance against a popular declipping algorithm when the models are trained given an increase in the amount and variety of data.

5.1 Data

The models were trained on two different datasets to assess the distortion audio effect removal capabilities. The audio signals from a dataset that contains a single instrument class (e.g., electric guitar) exhibit consistent signal statistics. In order to restrict the statistics that need to be modeled in a first step, we chose to concentrate on dry guitar samples as target data.

Since a large-scale, polyphonic dataset from clean electric guitar sounds is, to the best of our knowledge, not available [2], we used an internal dataset, which we refer to as CEG (Clean Electric Guitar) dataset. The monaural data were gathered from various sources, mainly commercial audio loop packages and recordings of solo guitar, and has a duration of $1.68\,\mathrm{h}$. All signals were re-sampled to a common sampling rate of $16\,\mathrm{kHz}$ in order to speed up convergence during training. To create the input dataset CEG-OD, the overdrive algorithm of SoX was applied to the data using five uniformly sampled gain levels in the range of $\gamma \in [20, 50]\mathrm{dB}$. Likewise, the input dataset for the hard-clipping task, CEG-HC, was created using hard-clipping (see (3)) with gain levels from the same distribution. Both datasets have a total length of $8.4\,\mathrm{h}$.

Although CEG represents a good source of data for our experiments due to its specificity of timbre, it remains a

[1] Two of the methods under evaluation, Demucs and Wave-U-Net, do not explicitly filter the signals in the audio domain, rather they perform a nonlinear mapping. Nevertheless, they were both successfully employed for MSS, which is a filtering problem.

[2] A publicly available guitar dataset for the recognition of audio effects exists (IDMT-SMT-Audio-Effects [26]). However, the dataset contains primarily homogeneous monophonic sounds and therefore, we choose to use the CEG dataset instead.

limited resource in terms of size and variety. Before attempting to train a system to handle recordings in real environments (e.g., a commercial song), we need to investigate how the current models at our disposal handle the availability of more and diverse training data. For this purpose, we also performed experiments on the SignalTrain dataset, which consists of more than 24 h of music and randomly-generated test signals [27]. By applying hard-clipping to these clean data using a uniformly-sampled input SDR value in the range $\mathrm{SDR}_{\mathrm{inp}} \in [1, 20]\mathrm{dB}$, we created SignalTrain-HC. During evaluation, we applied each input SDR in the set $\mathrm{SDR}_{\mathrm{inp}} \in \{1, 3, 5, 7, 10, 15, 20\}\mathrm{dB}$ to each sample in the test set.

Each dataset was split into non-overlapping subsets for training (80%), validation (10%) and testing (10%). We evaluated the models on the test split.

5.2 Experimental Setup

During the supervised training procedure, Adam [28] was used as optimizer with initial learning rates according to the model's original implementations. The learning rate was reduced by a factor of 10 after 150 epochs of no decrease in the validation loss. All models were optimized using the source-to-distortion ratio (SDR) between their full output sequences $\hat{\mathbf{x}}$ and the respective target sequences $\mathbf{x}$ in each batch $\mathcal{B}$ of N elements (not to be confused with the definition in the `BSS_eval` toolkit [29]):

$$\mathcal{L}_{\mathcal{B}}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N} \sum_{i \in \mathcal{B}} 10 \log_{10} \left(||\mathbf{x}_i||^2 / ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 \right). \quad (5)$$

We stopped all trainings after 1000 epochs. All models processed audio sequences that are randomly extracted from each clip in the dataset; the length of the extracted sequences is equal to 2 s (2.3 s for CRAFx due to its architecture). We used a batch size of 16 for all experiments.

5.3 Objective Metrics

In speech enhancement and source separation, the ubiquitous measure to estimate the quality of a system is the SDR. However, applying (2) to a signal does not retain its energy. Therefore, we follow the approach of [30]: the **scale-invariant SDR (SI-SDR)** is obtained by rescaling the target signal s such that the residual $s - \hat{s}$ is orthogonal to s by using the optimal scaling factor $\hat{s}^T s / ||s||^2$.

An evaluation exclusively based on the objective similarity of the signals does not necessarily imply a correlation with human perception [31, 32]. Accordingly, we observed that the SI-SDR scores occasionally disagreed with our qualitative evaluation. Therefore, we also considered three metrics based on human perception.

The **perceptual evaluation of audio quality (PEAQ)** [33] is a widely used perceptual metric [11, 13, 34, 35] that measures the amount of degradation between two audio signals. The output of PEAQ is an Overall Difference Grade (ODG), which can reach values between 0 (*imperceptible impairment*) and -4 (*very annoying impairment*). Even though PEAQ is used in declipping and audio restoration studies [11, 13, 34], it was developed for audio codecs.

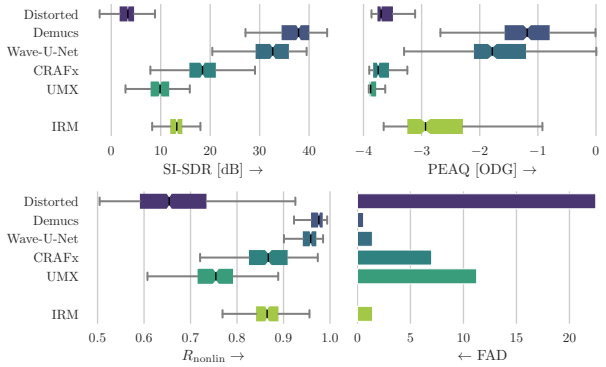


Figure 2. Box plot of scores for the CEG-OD dataset. The boxes show the first and third quartile of the data while the median is indicated with a line in the box. Higher scores indicate superior performance except for FAD. The results suggest that U-Net-based models performing convolutions in the time domain are the most promising approach to solving the task of overdrive removal.

The **R-nonlin metric** [36], in contrast, was developed specifically for detecting nonlinear distortions and, like PEAQ, considers the human auditory system. R_{nonlin} is defined between 0 (*high distortion*) and 1 (*no distortion*).

The **Fréchet audio distance (FAD)** was recently proposed as a reference-free evaluation metric for music enhancement algorithms. It has shown to correlate more with human perception than the SDR [37]. In order to obtain the FAD, the embedding statistics of both the whole clean and distorted test set are generated using a VGGish model [38]. The FAD is calculated based on the Fréchet distance between two multivariate Gaussians computed from both the test and the reference embeddings. [39].

6. RESULTS

In this section, we provide the results of the experiments introduced in the previous section.

6.1 De-Overdrive (CEG-OD)

Fig. 2 shows the results of the models that remove overdrive from guitar tracks.

Firstly, in SI-SDR, both Demucs and Wave-U-Net perform exceptionally well and even outperform the ideal-ratio-mask by more than 24 dB. While CRAFx yields considerably worse performance, it also surpasses the IRM oracle. UMX is the least performing of all the models in our comparison. It should be noted that for UMX, a model that operates on magnitude spectrograms, the IRM represents its upper limit in performance.

Similar results are obtained with PEAQ, R_{nonlin} and FAD: while Demucs and Wave-U-Net yield the best scores and surpass the IRM, the ones for CRAFx and UMX are considerably worse. It seems that directly processing the signals in the time domain using a U-Net-based architecture represents the most promising approach for the removal of the overdrive effect.

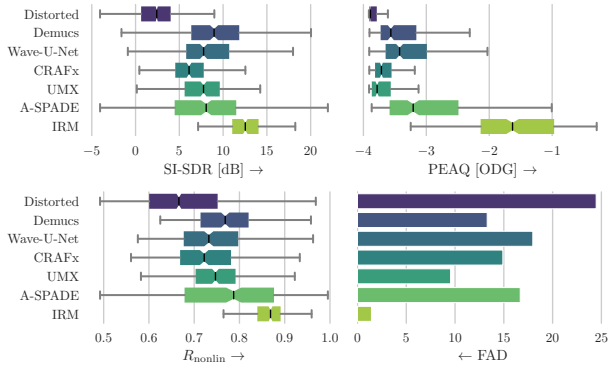


Figure 3. Box plot of scores for the CEG-HC dataset. It is difficult to clearly determine the best model, although Demucs represents a good compromise for all the metrics.

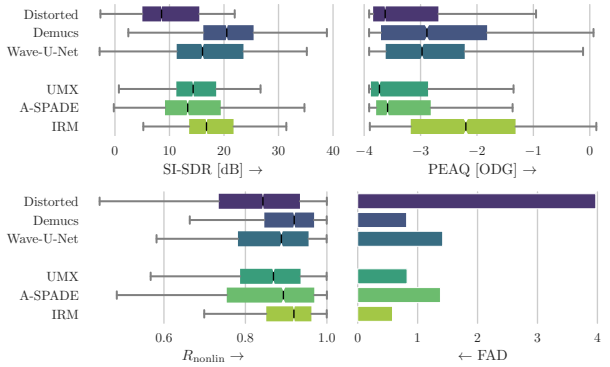


Figure 4. Box plot of scores for the SignalTrain-HC dataset. Demucs can be regarded as the best model regarding the median score across all metrics.

6.2 Declipping (CEG-HC)

Fig. 3 shows the results on the task of declipping on guitar recordings. Generally, we experienced a drop in the scores: now the models have been trained on declipping, which is an ill-posed problem, as missing parts of the signal need to be reconstructed. Additionally, we report scores for our declipping baseline, A-SPADE.

Despite the general performance drop, Demucs surpasses the results of A-SPADE in terms of SI-SDR by almost $1\,\mathrm{dB}$. While UMX and Wave-U-Net yield similar performances, CRAFx is the method with lowest scores.

Regarding PEAQ, no method surpasses the IRM and the A-SPADE algorithm. As before, Demucs and Wave-U-Net have similar performance: while PEAQ slightly favors the latter, SI-SDR favors the former. In contrast to the previous results, CRAFx has no significant advantage over UMX.

While Demucs achieves the best score for R_{nonlin} among the neural models, it does not surpass A-SPADE since the difference in their score is marginal. Interestingly, UMX achieves better results than CRAFx and Wave-U-Net. As the R_{nonlin} metric was explicitly designed to detect nonlinear distortions, we conclude that UMX' outputs contain fewer nonlinear distortions than the outputs of CRAFx and Wave-U-Net.

Surprisingly, when computing the FAD, UMX outperforms all other methods, including A-SPADE. This might be accounted to the fact that the FAD is based on the mel spectrum, whereas UMX optimizes the magnitude spectrogram. While Demucs and CRAFx outperform A-SPADE in FAD as well, the score for Wave-U-Net is slightly worse. The FAD for the IRM is relatively small because the difference between the reference and test embeddings from the VGGish model can be traced back to the masking operation and quantization. Demucs seems to constitute a good compromise regarding performance since it yields first- or second-best results for all metrics.

6.3 Declipping (SignalTrain-HC)

Fig. 4 shows the results for declipping SignalTrain-HC. Due to the lower gains that are used to prepare the data (see Sec. 5.1), the overall performance seems to be superior but cannot be directly compared to the previous results. Because of its considerably worse performance in the previous task, CRAFx was left out of the evaluation.

Demucs surpasses all other models in SI-SDR, including IRM and A-SPADE. Moreover, Wave-U-Net and UMX both surpass A-SPADE but not the IRM. PEAQ gives a similar ranking of the methods: only UMX cannot reach the A-SPADE baseline. None of the neural methods surpasses the IRM. In terms of R_{nonlin}, the same ranking is obtained, with Demucs surpassing, Wave-U-Net reaching, and UMX just missing A-SPADE. The FAD highlights that UMX and Demucs deliver comparable performance, outperforming the baseline, but not surpassing the IRM.

When only looking at SI-SDR or PEAQ, we notice the superiority of the time domain models. Future research should investigate whether the waveform in the time domain is the best input representation for the task, compared to, e.g., the real and imaginary part of a spectrogram [40] or both the waveform and the spectrogram [41]. Ultimately, Demucs can be considered the best model in our experiments for the task of declipping on SignalTrain.

6.4 Discussion

Qualitative Evaluation Although the abundance of evaluation metrics in the literature has the potential to analyze the results in very detailed ways, it does not always aid the judgement of which method is best given the individual context. Often, different applications have different requirements concerning the sound quality and can afford some types of distortions or artifacts to be left in the signal. Therefore, we report some qualitative considerations that need to be taken into account concerning our results, with the aim of finding some descriptive patterns among them. Fig. 5 shows spectrograms of a hard-clipped guitar signal and the outputs of all models under evaluation.

We found that the characteristics of the artifacts that each model produces are consistent, independent from the task it is applied to. Nevertheless, for the time domain-based models, the artifacts are less prominent in the de-overdrive task. Here, Demucs often produces outputs that

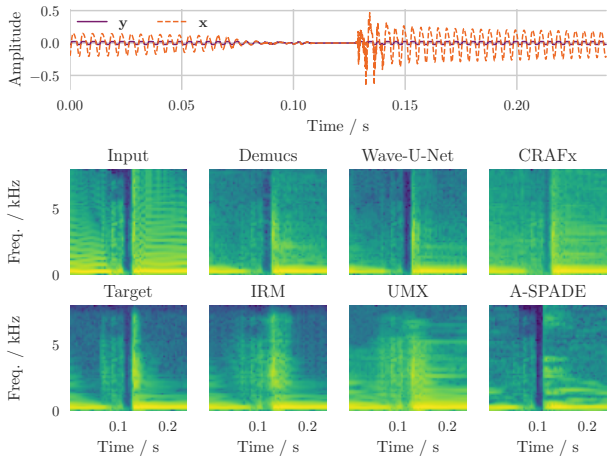


Figure 5. Spectrograms of a clipped guitar signal ($\gamma = 45\,\mathrm{dB}$), its target signal, and the output of each method. The respective time domain input and target signal are shown at the top. While all methods significantly reduce the harmonics, it can be observed that IRM and UMX smear the transients. A-SPADE relies on the strongest bins in the clipped spectrum, which leads to tonal artifacts.

are virtually indistinguishable from the original signal.

Generally, Demucs is the model that most often produces high quality results. Especially for inputs with a low amount of distortion, it can reconstruct the original sound without any perceptual artifact. Wave-U-Net behaves similarly, although it often cannot reach the same quality.

UMX generally removes the distortion characteristics very well at the cost of strong phasing artifacts: Despite the absence of input distortions, the spectral features of the output are not necessarily consistent with the ones of the target. This is most likely due to UMX re-using the phase of the distorted input to reconstruct the signal in the time domain and the frame-wise processing. Moreover, Fig. 5 highlights that the transients are smeared by re-using the phase of the degraded signal.

CRAFx does not suffer from phasing artifacts, but occasionally leaves part of the distortion features in the output. In some cases, the model fails to reconstruct the onset of some notes, penalizing the listening experience.

Finally, A-SPADE is the model that exhibits the strongest and most frequent artifacts, especially for strongly clipped signals. Although it considers phase information by working in the complex frequency domain, it leads to non-optimal solutions. Nevertheless, distortion features (like those left by Demucs) or transient smearing (left by UMX) do not occur.

Influence of the Dry Signal We have observed a substantial difference in performance between models that need to remove overdrive (CEG-OD dataset) and models that need to remove hard-clipping (CEG-HC dataset). The superior performance of the models trained and tested on CEG-OD can be mainly traced back to the presence of the non-distorted signal in the overdrive output and not to the soft-clipping character of the specific overdrive implementation. We verified this hypothesis by training Wave-U-Net on hard-clipped data superimposed with the clean signal (even when the amplitude of the clean signal is considerably low). We obtained results similar to those in the de-overdrive task (median results without superposition: SI-SDR $= 7.2\,\mathrm{dB}$, with superposition: SI-SDR $= 34.4\,\mathrm{dB}$). While the performance drop is present for all metrics, it is less pronounced for UMX which seems not to utilize the additional information in the signal.

Inference Speed The benefits of Demucs in the context of declipping go beyond the quality of its outputs: using a neural approach also has advantages regarding inference speed. Inference with A-SPADE is comparably slow (real-time factor on CPU $\times\mathrm{RT} \in [4.2, 27.3]$ depending on $\mathrm{SDR}_{\mathrm{inp}}$ [13]), being an iterative approach that requires a computation of the Fourier transform and its inverse at each iteration. Demucs ($\times\mathrm{RT} = 0.072$), Wave-U-Net ($\times\mathrm{RT} = 0.113$) and UMX ($\times\mathrm{RT} = 0.026$) instead, allow for fast inference on the CPU and even surpass real-time constraints independently on $\mathrm{SDR}_{\mathrm{inp}}$ without sacrificing the quality of the results.

Evaluation Metrics The results highlight how our evaluation metrics focus on different aspects of the reconstructions: While SI-SDR measures differences between two audio signals in the time domain, PEAQ focuses on the perceptual quality without differentiating between degradation related to nonlinear distortions and artifacts/quality. In contrast, R_{nonlin} specifically highlights nonlinear distortions that have not been removed. Finally, FAD focuses primarily on the degradation that is observable in the mel spectrum. Hence, each metric proves beneficial in measuring specific aspects in the analysis of audio effect removal systems and can be advised for further investigations.

7. CONCLUSION

We showed that recovering the clean signal from clipped or overdriven audio signals can efficiently solved with neural networks designed for source separation. We found that Demucs achieves high quality according to the chosen evaluation metrics, especially when the distortion algorithm to be removed blends the distorted sound with the original one. This outcome highlights the potential of the proposed approach for other audio effects that mix dry and wet signals (e.g., parallel compression, reverberation, delay, modulation effects).

Moreover, we showed that Demucs, Wave-U-Net, and UMX outperform one state-of-the-art declipping method on our test data. This outcome is promising, considering that the dataset to train such a system is potentially much larger than the size of the dataset we used. By discussing the results, we stressed the usefulness of multiple evaluation metrics suitable to assess distortion removal systems.

Future work should include gathering more clean electric guitar data and generating a dataset using high-quality distortion emulations, which is required to improve generalization on real-world data. Furthermore, the knowledge from sparsity-based declipping algorithms could yield a valuable prior for declipping through DNNs.

8. REFERENCES

[1] U. Zölzer, *DAFX: digital audio effects*, 2nd ed. Chichester, West Sussex, England: Wiley, 2011.

[2] R. Stables, B. De Man, and J. D. Reiss, "Ten years of automatic mixing," in *Proceedings of the 3rd Workshop on Intelligent Music Production, Salford, UK, 15 September 2017*, 2017.

[3] M. A. Martínez-Ramírez, D. Stoller, and D. Moffat, "A Deep Learning Approach to Intelligent Drum Mixing With the Wave-U-Net," *Journal of the Audio Engineering Society*, vol. 69, no. 3, pp. 142–151, Mar. 2021.

[4] C. J. Steinmetz, J. Pons, S. Pascual, and J. Serra, "Automatic Multitrack Mixing With A Differentiable Mixing Console Of Neural Audio Effects," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Toronto, ON, Canada: IEEE, Jun. 2021, pp. 71–75.

[5] Y. Mitsufuji, G. Fabbro, S. Uhlich, F.-R. Stöter, A. Défossez, M. Kim, W. Choi, C.-Y. Yu, and K.-W. Cheuk, "Music Demixing Challenge 2021," *Frontiers in Signal Processing*, vol. 1, Jan. 2022.

[6] W. Choi, M. Kim, M. A. Martínez-Ramírez, J. Chung, and S. Jung, "AMSS-Net: Audio Manipulation on User-Specified Sources with Textual Queries," in *Proceedings of the 29th ACM International Conference on Multimedia*, Apr. 2021, pp. 1775–1783.

[7] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017.

[8] T. Wilmering, D. Moffat, A. Milo, and M. B. Sandler, "A History of Audio Effects," *Applied Sciences*, vol. 10, no. 3, p. 791, Jan. 2020.

[9] M. A. Martínez-Ramírez, E. Benetos, and J. Reiss, "Deep Learning for Black-Box Modeling of Audio Effects," *Applied Sciences*, vol. 10, p. 638, Jan. 2020.

[10] C. Bagwell, et al., "Sound eXchange (SoX)," Feb. 2015.

[11] P. Záviška, P. Rajmic, A. Ozerov, and L. Rencker, "A Survey and an Extensive Evaluation of Popular Audio Declipping Methods," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 1, pp. 5–24, Jan. 2021.

[12] A. Janssen, R. Veldhuis, and L. Vries, "Adaptive interpolation of discrete-time signals that can be modeled as autoregressive processes," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 2, pp. 317–330, Apr. 1986.

[13] C. Gaultier, S. Kitić, R. Gribonval, and N. Bertin, "Sparsity-Based Audio Declipping Methods: Selected Overview, New Algorithms, and Large-Scale Evaluation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1174–1187, 2021.

[14] H. B. Kashani, M. M. Goodarzi, A. Jodeiri, and S. G. Firooz, "Image to Image Translation based on Convolutional Neural Network Approach for Speech Declipping," *4th Conference on Technology In Electrical and Computer Engineering (ETECH 2019)*, 2019.

[15] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham: Springer International Publishing, 2015, pp. 234–241.

[16] W. Mack and E. A. P. Habets, "Declipping Speech Using Deep Filtering," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, Oct. 2019, pp. 200–204.

[17] T. Tanaka, K. Yatabe, M. Yasuda, and Y. Oikawa, "APPLADE: Adjustable Plug-and-Play Audio Declipper Combining DNN with Sparse Optimization," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Singapore, Singapore: IEEE, May 2022, pp. 1011–1015.

[18] M. A. Martínez-Ramírez, E. Benetos, and J. D. Reiss, "A general-purpose deep learning approach to model time-varying audio effects," in *22nd International Conference on Digital Audio Effects (DAFx-19)*, Jun. 2019.

[19] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-Unmix - A Reference Implementation for Music Source Separation," *Journal of Open Source Software*, vol. 4, no. 41, p. 1667, Sep. 2019.

[20] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA: IEEE, Mar. 2017, pp. 261–265.

[21] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[22] A. Narayanan and D. Wang, "Ideal ratio mask estimation using deep neural networks for robust speech recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 7092–7096.

[23] D. Stoller, S. Ewert, and S. Dixon, "Wave-U-Net: A Multi-Scale Neural Network for End-to-End Audio Source Separation," in *Proceedings of the 19th ISMIR Conference, Paris, France, September 23-27, 2018*, Paris, Jun. 2018.

[24] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music Source Separation in the Waveform Domain," *arXiv:1911.13254 [cs, eess, stat]*, Apr. 2021.

[25] S. Kitić, N. Bertin, and R. Gribonval, "Sparsity and Cosparsity for Audio Declipping: A Flexible Non-convex Approach," in *Latent Variable Analysis and Signal Separation*, E. Vincent, A. Yeredor, Z. Koldovský, and P. Tichavský, Eds. Cham: Springer International Publishing, 2015, pp. 243–250.

[26] M. Stein, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic Detection of Audio Effects in Guitar and Bass Recordings," in *AES 128th Convention*, London, UK, May 2010.

[27] S. H. Hawley, B. Colburn, and S. I. Mimilakis, "SignalTrain: Profiling Audio Compressors with Deep Neural Networks," in *AES 147th Convention*, May 2019.

[28] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, {ICLR} 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[29] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 4, pp. 1462–1469, Jul. 2006.

[30] J. L. Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR – Half-baked or Well Done?" in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Brighton, United Kingdom: IEEE, May 2019, pp. 626–630.

[31] E. Cano, D. FitzGerald, and K. Brandenburg, "Evaluation of quality of sound source separation algorithms: Human perception vs quantitative metrics," in *2016 24th European Signal Processing Conference (EUSIPCO)*. Budapest, Hungary: IEEE, Aug. 2016, pp. 1758–1762.

[32] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann, "Subjective and Objective Quality Assessment of Audio Source Separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2046–2057, Sep. 2011.

[33] T. Thiede, W. C. Treurniet, R. Bitto, C. Schmidmer, T. Sporer, J. G. Beerends, and C. Colomes, "PEAQ - The ITU Standard for Objective Measurement of Perceived Audio Quality," *Journal of the Audio Engineering Society*, vol. 48, no. 1/2, pp. 3 EP – 29, Feb. 2000.

[34] S. Lattner and J. Nistal, "Stochastic Restoration of Heavily Compressed Musical Audio Using Generative Adversarial Networks," *Electronics*, vol. 10, no. 11, p. 1349, Jun. 2021.

[35] J. You, U. Reiter, M. M. Hannuksela, M. Gabbouj, and A. Perkis, "Perceptual-based quality assessment for audio–visual services: A survey," *Signal Processing: Image Communication*, vol. 25, no. 7, pp. 482–501, Aug. 2010.

[36] C.-T. Tan, B. C. J. Moore, N. Zacharov, and V.-V. Mattila, "Predicting the Perceived Quality of Nonlinearly Distorted Music and Speech Signals," *Journal of the Audio Engineering Society*, vol. 52, no. 7/8, pp. 699–711, Jul. 2004.

[37] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms," in *Interspeech 2019*. ISCA, Sep. 2019, pp. 2350–2354.

[38] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN Architectures for Large-Scale Audio Classification," *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Jan. 2017.

[39] D. Dowson and B. Landau, "The Fréchet distance between multivariate normal distributions," *Journal of multivariate analysis*, vol. 12, no. 3, pp. 450–455, 1982.

[40] W. Choi, M. Kim, J. Chung, D. Lee, and S. Jung, "Investigating U-Nets with various Intermediate Blocks for Spectrogram-based Singing Voice Separation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference*, Oct. 2020.

[41] A. Défossez, "Hybrid Spectrogram and Waveform Source Separation," in *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, Nov. 2021.

END-TO-END FULL-PAGE OPTICAL MUSIC RECOGNITION FOR MENSURAL NOTATION

Antonio Ríos-Vila
U.I for Computer Research
University of Alicante, Spain
arios@dlsi.ua.es

José M. Iñesta
U.I for Computer Research
University of Alicante, Spain
iñesta@dlsi.ua.es

Jorge Calvo-Zaragoza
U.I for Computer Research
University of Alicante, Spain
jcalvo@dlsi.ua.es

ABSTRACT

Optical Music Recognition (OMR) systems typically consider workflows that include several steps, such as staff detection, symbol recognition, and semantic reconstruction. However, fine-tuning these systems is costly due to the specific data labeling process that has to be performed to train models for each of these steps. In this paper, we present the first segmentation-free full-page OMR system that receives a page image and directly outputs the transcription in a single step. This model requires only the annotations of full score pages, which greatly alleviates the task of manual labeling. The model has been tested with early music written in mensural notation, for which the presented approach is especially beneficial. Results show that this methodology provides a solution with promising results and establishes a new line of research for holistic transcription of music score pages.

1. INTRODUCTION

The majority of music creations, both historical and modern, are only available through music scores. Unfortunately, most of them have never been stored in a structured digital format, such as MEI [1] or Humdrum **kern [2], that allows their indexing, retrieval, and digital processing. The high cost of performing a manual transcription of these documents demands processes that solve this task automatically.

Optical Music Recognition (OMR) is the research field that studies how to computationally read music scores [3]. Most OMR literature is framed within a multi-stage workflow, where several steps, such as image binarization [4], staff-line removal [5], symbol classification [6, 7], and notation assembly [8], are considered. These pipelines, although effective to some extent, are hard to manage in a production environment, as they imply the development of several models to obtain a complete OMR system.

With the advent of machine learning technologies, namely those related to deep neural networks, OMR systems have evolved towards alternatives that simplify these complex workflows. Specifically, two trends emerged from this advance. On the one hand, complex multi-stage pipelines for symbol isolation and retrieval have been replaced by object detection algorithms [9], where symbols are directly located in the image. On the other hand, holistic strategies—commonly referred to as end-to-end approaches [10, 11]—are also used to transcribe music scores. In this case, the system directly outputs the transcription of a single-staff image.

Although the advances brought about by these new technologies have simplified these pipelines, these systems still rely on other models to complete the transcription of scores. In the case of the object detection approach, models for notation reconstruction and sequence encoding need to be implemented to obtain a readable representation of the music score. End-to-end approaches, in turn, are only able to transcribe single music staves, so staff detection algorithms have to be implemented to retrieve all the information from the page and be recognized individually [12, 13]. The latter pipeline is represented in the upper workflow of Fig. 1.

Following the path of end-to-end approaches, in this paper we present the first segmentation-free strategy for full-page music transcription that unifies the two required steps—see the lower workflow in Fig. 1—based on evolving the current state of the art. For this model to be trained, only page-level annotations—which could be available in any music encoding—are required. This greatly alleviates the labeling of music scores, as additional OMR-specific data preprocessing steps are avoided.

The motivation for implementing this idea is to solve some issues of pipeline-based end-to-end approaches. The main inconvenience is that they are composed of two models that are trained specifically on independent tasks—that is, each document has to be manually labeled twice. First, bounding boxes and region classes have to be inserted to train the staff detection model. Once this information has been retrieved, the corresponding symbolic music representation must be written for each staff present in the dataset to train the staff-level recognition system. Therefore, labeling a corpus to train a production-ready OMR system for full-page transcription is costly—due to the need for manual segmentation and transcription of individual music staves—and time-consuming [14]. Another issue is that there has to be assumed an accumulated er-

© A. Ríos-Vila, J. M. Iñesta, and J. Calvo-Zaragoza. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** A. Ríos-Vila, J. M. Iñesta, and J. Calvo-Zaragoza, "End-to-End Full-Page Optical Music Recognition for Mensural Notation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

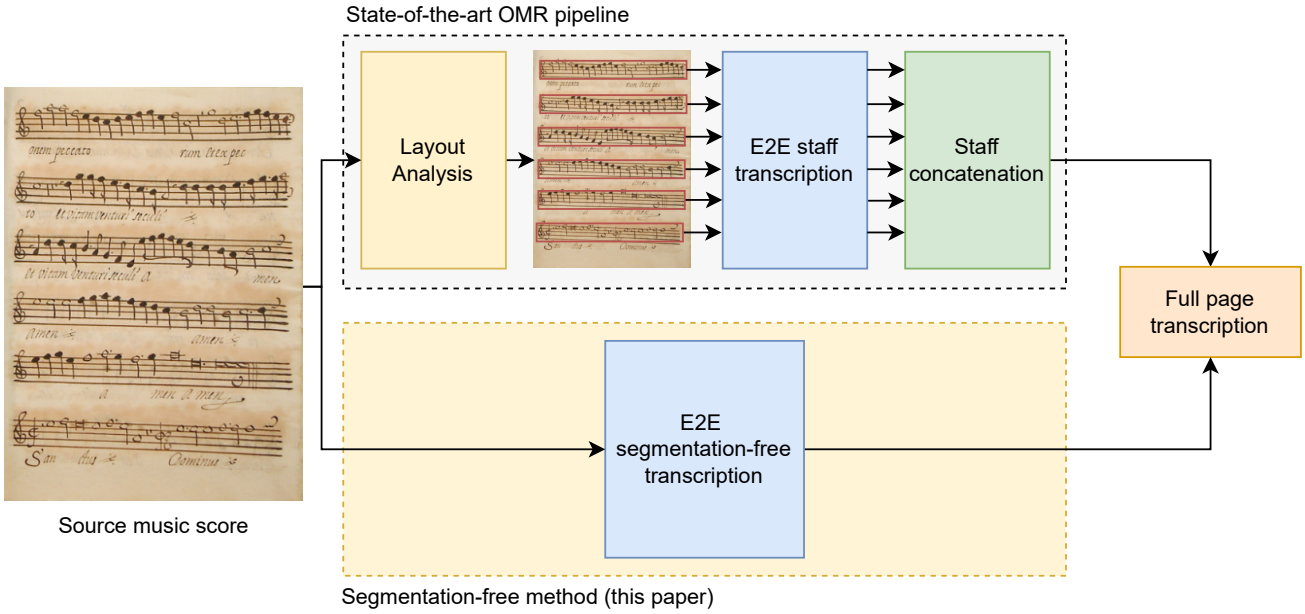


Figure 1: General overview of the current full page OMR pipeline (top) in contrast to this paper proposal (bottom), where a previous layout analysis is not needed to transcribe the music symbols in the score.

ror that is carried over from one step to the next, as some works have studied [12, 15]. That is, the performance of a specific step is highly related to the effectiveness of the previous one, which is also dependent on their predecessors, thus creating a snowball effect.

2. EXTENDING OMR TRANSCRIPTION SYSTEMS

In this section, we explain how state-of-the-art end-to-end OMR models work, what issues they present when transcribing a complete music document, and how they can be adapted to a full-page scenario.

2.1 End-to-end staff transcription systems

The state-of-the-art end-to-end music staff transcription systems are neural models that directly output the music notation sequence of a given single-staff image. These networks are based on two main blocks: first, there is an encoder block, which filters an input image to learn and establish its relevant features. Commonly, this part is approached with Convolutional Neural Networks (CNN). Then, the output of this module is processed by the decoder block, which models temporal dependencies in the obtained features—which are represented as a vector sequence—to enhance results. This second module is commonly implemented with Recurrent Neural Networks (RNN). The whole model, referred to in the literature as a Convolutional Recurrent Neural Network (CRNN), is trained using the Connectionist Temporal Classification (CTC) [16] loss strategy, which maximizes through expectation-maximization the probability of obtaining a music sequence in a given notation alphabet. That is, the CTC method forces the network to align the output sequence to the available information extracted from the image, retrieving the full transcription in a single step. One important step to understand the model is how the output of the encoder is adapted to be processed as a sequence in the decoder. State-of-the-art implementations [10] reshape this obtained feature map—which is a 3D structure of size (h, w, c), where h is the height of the feature map, w is its width, and c is the number of filters (*channels*) obtained from the last convolutional layer—, by concatenating all the consecutive frames on the height axis of the image. That is, we obtain a sequence-like 2D structure that now has a shape of $(w, c \times h)$. This methodology has proven to be effective to transcribe both printed music staves [17] and handwritten ones [10, 11].

2.2 The issues with full-page images

The methodology described in the previous section is ideal for single-staff images, as it solves the transcription problem by reading the obtained feature map from left to right. However, in the case of page images, this formulation cannot be applied. The main issue with this formulation is that a single frame—which is an image column—contains more than one symbol. This is troublesome in two ways. First, transcription and retrieval of the score would result in a costly task, as ground-truth should be written and read from top to bottom and then from left to right, introducing undesirable post-processing methods to obtain a readable score. The second problem is related to the training of the system, as the CTC method should maximize the probability of non-consecutive temporal symbols in a few time steps. That is, the network has to classify symbols from different staves in the same image columns. This implicit interpretation is wrong, as these symbols are located in farther positions of the sequence, as they belong to different staves.

2.3 From one staff to full page

To transcribe full-page music scores, a new interpretation of the obtained features map is considered. It is required to use an alternative reshape method from the output of the encoder block. In this paper, we apply a score unfolding reshape method. Instead of concatenating frame-wise elements along the height axis, as it is done for staff transcription, we reshape the image in the concatenation of all of its rows, obtaining then a $(h \times w, c)$ sequence. From a high-level perspective, this method can be understood as a staff concatenation process, as it is depicted in Fig. 2. This operation has to be done from top to bottom of the page, as it is crucial to correctly transcribe the music score.

Processing the feature maps this way prevents the aforementioned issues, as the score can be labeled as it is naturally read—from top to bottom and left to right—and the CTC method will not face the problem of collision of non-consecutive times, as now the used sequence does not have merged features from different staves.

This methodology, although theoretically a valid solution, presents some points that should be noted. The adaptation of the current music transcription models only needs the transcription of the full page for training, as the model directly outputs the music sequence from the input image. By avoiding the previously required object detection algorithm from the pipeline, now the system has to face two main challenges: (i) identifying all the staves in the score and (ii) transcribing them into an ordered sequence. The second challenge is solved by the reshape method, as we force the model to align and read all the staves in a specific order. In this case, the CTC *blank* token—which is an additional element introduced in the notation vocabulary to indicate time step separations—denotes both music element separations and staff breaks, since the single-staff-like produced sequence identifies the first symbol of the next staff as a consecutive timestep to the last symbol of the previous one. Challenge (i), however, is somewhat relegated to the network learning. The hypothesis is that, by not modifying the sequence structure in the decoder block, this alignment can be learned and mapped by the encoder. This hypothesis needs to be validated by the performance of the proposed models during experimentation.

2.4 Further considerations

This paper evolves the already established OMR staff recognition models by implementing a learned music score unfolding. Theoretically, this method is still a single-staff transcription system. All the staves of the pages can be understood as a single long staff that, due to physical constraints of paper, had to be divided into several staves. This methodology implicitly learns to reconstruct this original interpretation and transcribe the resulting long single-staff image in one step. The alignment process between the different staves is learned by the network during training.

It is important to note that, at this point, the proposed method is suitable only for monophonic staves, as polyphony requires additional information and processing to be transcribed holistically. Vocal music scores—such as those written in mensural notation—can benefit from this approach because polyphony is usually written through a series of independent monophonic voices. For this reason, the experiments will be carried out with early music scores written in mensural notation, as detailed in Section 3.2.

3. EXPERIMENTAL SETUP

In this section, we describe the proposed models to address full-page OMR, present the used corpora during experimentation, and define the metrics used to evaluate the performance of our proposals [1] .

3.1 Models

As mentioned throughout this paper, we are adapting state-of-the-art OMR models to full-page transcription. However, two additional variations have been included in the proposed model to extend the study on this topic. All the presented models have a fully convolutional block, which acts as an encoder of the input image features. This network is composed of stacked convolutional layers, which end up producing a feature map of size $(h/32, w/8, c, b)$, h and w being the height and the width of the input image, c the filters in the last convolutional layer, and b the batch size. The downscale of the original image size is produced by pooling operators. Then, the decoding architectures proposed for processing the sequence obtained after the reshaping procedure are described below.

3.1.1 Recurrent Neural Network

We follow the implementation of the original CRNN-CTC staff transcription model from [10], where the reshaped feature map is fed into a Bidirectional LSTM (BLSTM) and linearly projected onto the music notation dictionary. Specifically, we implemented a BLSTM with 256 units, whose output matches the output space of the fully convolutional network.

3.1.2 The Transformer

The base model of this work uses RNNs to process the reshaped feature map as a sequence. However, a recent recurrent-free model has gained popularity in the Natural Language Processing (NLP) field: the Transformer [18]. This model replaces the RNN architecture by implementing sequence modeling through attention mechanisms and position learning. This model solves some common issues that RNNs have—such as processing long sequences, training time effort, and contextual information retrieval—at the cost of needing more data to converge. As we have observed in the reshape step, the model would have to process significantly long sequences in one step, something that can have a negative impact on the performance of RNNs. For this reason, the first alternative model implemented in this paper replaces the recurrent layer of the CRNN model with a Transformer encoder (referred to as

[1] Source code for the implementation of the presented models can be found in `https://github.com/antoniorv6/e2e_poliphony`

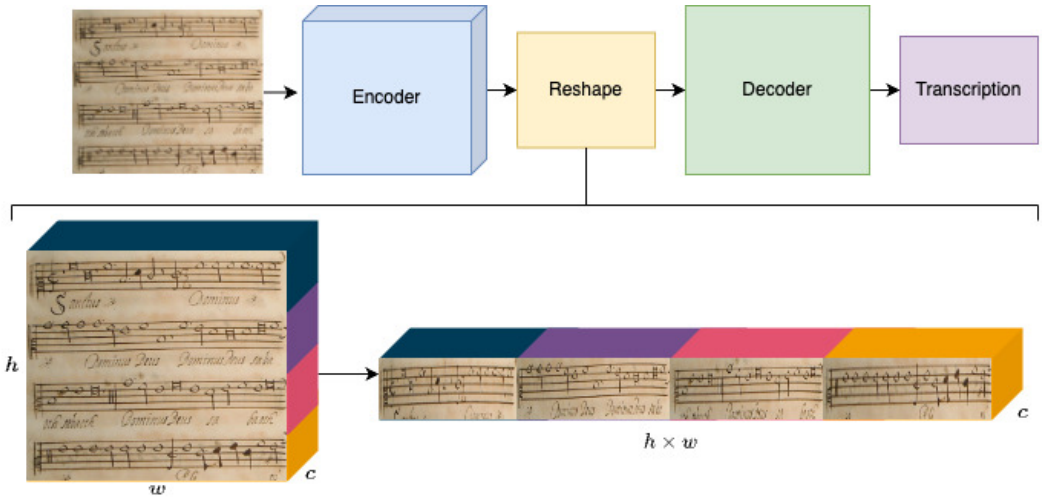


Figure 2: Graphic visualization of the reshape method to adapt current systems to full-page transcription. The reshape module learns how to separate and concatenate the staves on a page in a single line. Note that the original image has been used in the reshape for clarity of explanation, but the alignment is done in the *feature space* extracted by the encoder.

.

CNNT hereafter). In particular, we implemented one encoder layer with an embedding size of 512, a feed-forward dimension of 1024, and 8 attention heads.

3.1.3 Sequence-processing-free module

One of the challenges the model has to face during the score recognition is the alignment of the staves extracted from a 2D image to build a 1D sequence. This leads to the question of how a sequence processing module could impact learning this specific task while performing backpropagation. In analogous text recognition models, like [19], the solution lies in preserving the prediction space in 2 dimensions, applying backpropagation directly to the retrieved feature map before being reshaped. We have implemented this model to analyze the impact of the sequence processing module on the score page transcription. It is also based on fully convolutional layers, so it will be referred to as FCN in the results section.

3.2 Corpora

Two mensural-notation music datasets with different characteristics in engraving style were used, in order to represent the different challenges that the model can face in these real-case scenarios.

The first corpus is "Il Lauro Secco" [20] (denoted as SEILS), which corresponds to an anthology of 150 typeset printed images of the 16th-century Italian madrigals.

The second corpus is the CAPITAN dataset [10], which contains a complete ninety-six pages manuscript from the 17th-century containing a handwritten *missa*.

Specific details about the used corpora can be found in Table 1, and Fig. 3 depicts examples of these documents.

3.3 Data augmentation

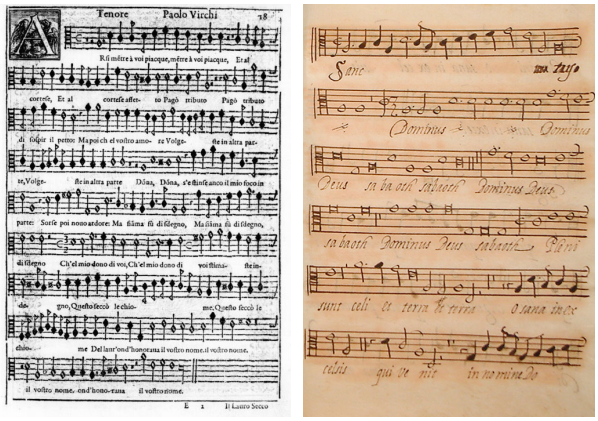
One constraint that our proposed model can find observing Table 1 is the scarcity of data, as these kind of early music documents usually have few pages completely labeled. To address this potential issue, we applied a data augmentation process to increase the number of samples per corpus. This procedure is composed of several image distortion operations, such as reduction, erosion, dilation, or perspective modifications. These distortions are randomly applied for each batch sample, allowing us to obtain massively extended corpora that also have an added variability for the samples. The SEILS corpus is increased up to 29000 pages and the CAPITAN dataset to 24000.

	SEILS	**CAPITAN**
Num pages	150	123
Max page size	1200 × 813	1593 × 2126
Min page size	1200 × 813	1100 × 780
Avg staves per page	4	10
Max staves per page	9	12
Min staves per page	1	2
Avg symbols per page	222	136
Max symbols per page	331	220
Min symbols per page	110	23
Unique symbols	183	321

Table 1: Details of the corpora regarding the pages' features, such as sizes in pixels, number of samples and staves per page, number of symbols present and unique symbols per dataset (*vocabulary*).

3.4 Metrics

Currently, there are no OMR-specific metrics to evaluate the performance of the transcription systems. In this paper, we resort to the Symbol Error Rate (SER), which is the most commonly used metric in the OMR literature for end-



(a) SEILS (b) CAPITAN

Figure 3: Music page examples from the used corpora.

Model	Augmentation	SEILS	CAPITAN
CRNN	-	6.3	26.6
	✓	**4.3**	**15.5**
CNNT	-	12.9	28.4
	✓	7.2	18.2
FCN	-	23.3	89.5
	✓	13.3	22.5
Staff-based [12]	-	3.6	10.8

Table 2: Test SER (%) obtained for the studied models. Castellanos et al.'s work is included in the last row as a reference value to observe how current OMR pipelines work to transcribe the used corpora in this paper.

to-end approaches. This metric is computed as

$$\text{SER}\,(\%) = 100\,\frac{\text{ED}(\hat{S}, S)}{|S|}\,,$$

where ED is the edit distance between the transcription hypothesis $\hat{S}$ and its corresponding ground truth, S. $|S|$ is the length (in tokens) of S. We chose this metric because it represents accurately the recognition performance of the model and correlates with the effort that a user would have to invest to manually correct the output sequence.

The datasets have been split into fixed partitions, where 60% of the samples have been used for training, 20% have been used for validation, and 20% for testing.

4. RESULTS

Table 2 shows the results obtained by the studied methodology on the test set for each corpus. Results reported by Castellanos et al. [12] are included to establish a reference value from a state-of-the-art algorithm based on a standard OMR pipeline (the one depicted in Fig. 1-top). Note that this reference model is trained under more favorable conditions, as it addresses the full-page transcription in two separate tasks, which have specific data for training each. However, it requires much more labeling work to build the training sets than our segmentation-free implementation. Therefore, it should be understood only as a reference and not as a competing approach.

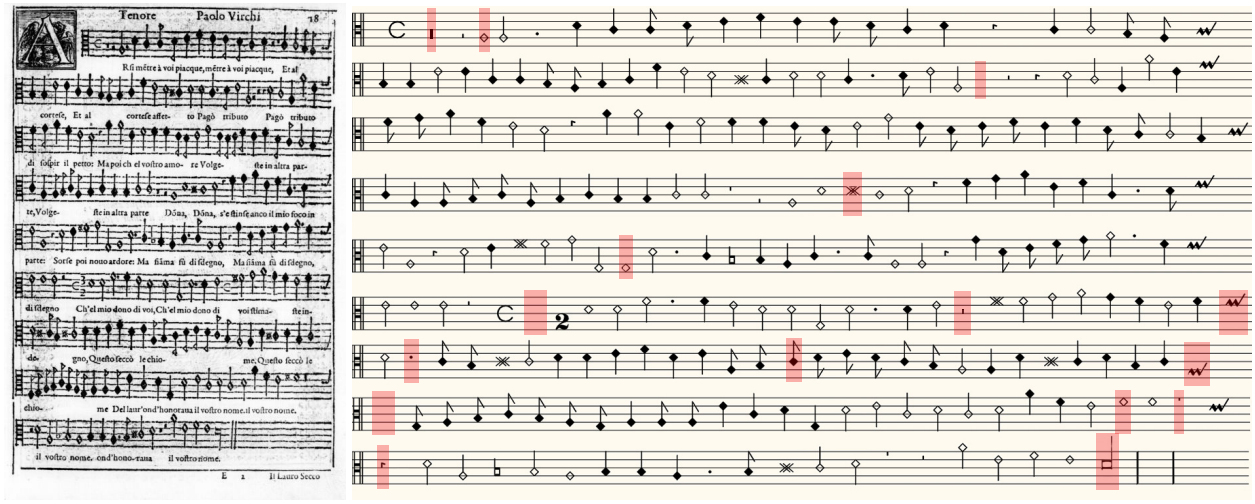
The results obtained show that the extended models were able to transcribe full-page scores with fair SER values, below 30% except for the FCN without data augmentation in the CAPITAN dataset. These error values also scale depending on the engraving complexity of each corpus, being the printed documents (SEILS) easier to transcribe than the handwritten ones (CAPITAN). The model that reported the best results was the combination of the convolutional network with the RNN decoder (CRNN), which obtained an error rate of 4.3% in the SEILS corpus and 15.5% in CAPITAN.

The models that use sequence processing decoders (CRNN and CNNT) performed better than the single FCN network. This was expected to some extent, as these architectures exploit and optimize sequential information to improve their performance, which is a considerable advantage over the FCN approach. In fact, they also seem to bring some robustness to the model against the scarcity of training data, if we compare their results with the high error rate of the FCN when no data augmentation was applied to the CAPITAN corpus.

Continuing the data dependency analysis, it can be observed that applying data augmentation, significantly improved the overall performance of all models. The most notable case of this dependency was found in the FCN network on the CAPITAN corpus, where overfitting issues seemed to be solved with the augmented database. For the other models, improvements were reported as well, reducing the SER by approximately 30%–40%. In other words, the models are able to work with few samples, but they require a considerable amount of data to obtain their best results.

Comparing the two sequence processing decoders, we observe that RNNs performed better than CNN Transformers in all cases. The reason for this is aligned with the results obtained in works that explore the use of these models on document transcription [22]. Looking at the Table 1, we observe that both models contain, on average, short sequences. Transformers, by replacing recurrence with self-attention and position encoding, improved computation time and accuracy at the cost of more data needed to converge. However, the Transformers literature has reported relevant improvements with long sequences, containing approximately 512 tokens, with which the RNNs have convergence problems. In this case, the Transformer model is in a disadvantageous scenario, where few data samples are available to train and the output sequences are relatively short, which is a much better scenario for RNN-based decoders.

For the sake of visualization, Fig. 4 presents the results obtained in a test set page from the SEILS dataset by the best model (CRNN). As can be seen, the system produces a good transcription, in which most of the symbols are correctly labeled and aligned within their corresponding



(a) Original page.

(b) Transcribed prediction with CRNN, rendered with the Verovio toolkit [21].

Figure 4: Visualization of the transcription produced by the CRNN model in a music page from the SEILS dataset. Errors are highlighted. Note that the output is actually produced on a single line, but a multi-line representation of the score has been reconstructed to facilitate comparison with the original document. In this particular case, the obtained SER is 6.1% (higher than the average on this corpus).

staves. If we analyze the produced errors, most of them are subtle—such as vertical position misplacement— and can be easily corrected with score editing software. This happens often with narrow symbols—such as rests—whose manual annotation would also have been difficult to perform.

5. CONCLUSION

In this paper, we present a first segmentation-free end-to-end approach for full-page OMR. This method is trained with weakly-annotated data: it only requires a set of page images with their corresponding transcription, in contrast to current state-of-the-art full-page OMR pipelines that require spatial information—such as bounding boxes or pixel-wise regions. Our methodology extends the current staff-level end-to-end systems to full-page transcription by applying a concatenation step that learns how to process the two-dimensional sequence document.

We evaluated three variants of this model with two early music collections written in mensural notation, which have been used in many other works as a benchmark for OMR. The reported results showed that the proposed system produces competitive results for full-page transcriptions. Although precision is slightly lower than a multi-stage OMR pipeline, the proposed segmentation-free approach stands as an interesting alternative for those models. The model provides a favorable trade-off between the cost of labeling and the system's accuracy.

Several avenues for future research arise from this work. First, it only covers the extent of historic vocal music in mensural notation, where only monophonic staves are found. In fact, the method can handle any page representing monophonic staves but, in modern music, the coverage is more restricted, since polyphony is much more common. This scenario could be an interesting case to analyze alternative architectures for OMR, such as attention-based systems [23] or non-CTC-based image-to-sequence architectures [24], which do not have to be constrained by the specific layout.

Another aspect to consider in the future is the need for data. Although one of the goals of this work is to reduce the labeling effort, the models still require a large number of fully annotated pages. We believe that further research needs to be done to study both transfer learning and self-supervised learning approaches to address this issue.

6. ACKNOWLEDGEMENTS

This paper is part of the project MultiScore (PID2020-118447RA-I00), funded by MCIN/AEI/10.13039/501100011033. The first author is supported by grant ACIF/2021/356 from "Programa I+D+i de la Generalitat Valenciana". Third author was supported with a 2021 Leonardo Grant for Researchers and Cultural Creators, BBVA Foundation. The Foundation takes no responsibility for the opinions, statements and contents of this paper, which are entirely the responsibility of its authors.

7. REFERENCES

[1] A. Hankinson, P. Roland, and I. Fujinaga, "The music encoding initiative as a document-encoding framework." in *Proc. of the 12th Int. Society for Music Information Retrieval Conference*, 2011, pp. 293–298.

[2] D. Huron, "Humdrum and kern: Selective feature encoding," *Beyond MIDI*, 1997.

[3] J. Calvo-Zaragoza, J. Hajič Jr., and A. Pacha, "Understanding optical music recognition," *ACM Computing Surveys*, vol. 53, no. 4, Jul. 2020.

[4] J. A. Burgoyne, L. Pugin, G. Eustace, and I. Fujinaga, "A comparative survey of image binarisation algorithms for optical recognition on degraded musical sources," in *Proc. of the 8th Int. Conf. on Music Information Retrieval*, pp. 509–512.

[5] J. dos Santos Cardoso, A. Capela, A. Rebelo, C. Guedes, and J. P. da Costa, "Staff detection with stable paths," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 1134–1139, 2009.

[6] A. Pacha and H. Eidenberger, "Towards a universal music symbol classifier," in *Proc. of 14th Int. Conf. on Document Analysis and Recognition*, 2017, pp. 35–36.

[7] A. Fornés, J. Lladós, and G. Sánchez, "Old handwritten musical symbol classification by a dynamic time warping based method," in *Graphics Recognition. Recent Advances and New Opportunities*, W. Liu, J. Lladós, and J.-M. Ogier, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 51–60.

[8] F. Rossant and I. Bloch, "Robust and adaptive OMR system including fuzzy modeling, fusion of musical rules, and possible error detection," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, p. 081541, 2006.

[9] A. Pacha, J. Hajič, and J. Calvo-Zaragoza, "A baseline for general music object detection with deep learning," *Applied Sciences*, vol. 8, no. 9, p. 1488, 2018.

[10] J. Calvo-Zaragoza, A. H. Toselli, and E. Vidal, "Handwritten music recognition for mensural notation with convolutional recurrent neural networks," *Pattern Recognition Letters*, vol. 128, pp. 115–121, 2019.

[11] A. Baró, C. Badal, and A. Fornés, "Handwritten historical music recognition by sequence-to-sequence with attention mechanism," in *Proc. of the 17th Int. Conf. on Frontiers in Handwriting Recognition*. IEEE, 2020, pp. 205–210.

[12] F. J. Castellanos, J. Calvo-Zaragoza, and J. M. Iñesta, "A neural approach for full-page optical music recognition of mensural documents," in *Proc. of the 21th Int. Society for Music Information Retrieval Conference*, 2020, pp. 12–16.

[13] M. Kletz and A. Pacha, "Detecting staves and measures in music scores with deep learning," in *Proc. of the 3rd Int. Workshop on Reading Music Systems*, J. Calvo-Zaragoza and A. Pacha, Eds., Alicante, Spain, 2021, pp. 8–12.

[14] M. Alfaro-Contreras, D. Rizo, J. M. Iñesta, and J. Calvo-Zaragoza, "OMR-assisted transcription: A case study with early prints," in *Proc. of the 22nd Int. Society for Music Information Retrieval Conference*, 2021, pp. 35–41.

[15] A. Ríos-Vila, D. Rizo, and J. Calvo-Zaragoza, "Complete optical music recognition via agnostic transcription and machine translation," in *Proc. of the 16th Int. Conf. on Document Analysis and Recognition*, J. Lladós, D. Lopresti, and S. Uchida, Eds., 2021, pp. 661–675.

[16] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proc. of the 23rd Int. Conf. on Machine Learning*, 2006, p. 369–376.

[17] L. Tuggener, I. Elezi, J. Schmidhuber, M. Pelillo, and T. Stadelmann, "DeepScores-A Dataset for Segmentation, Detection and Classification of Tiny Objects," in *Proc. of the 24th Int. Conf. on Pattern Recognition*, 2018, pp. 3704–3709.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008.

[19] D. Coquenet, C. Chatelain, and T. Paquet, "Span: A simple predict & align network for handwritten paragraph recognition," in *Proc. of the 16th Int. Conf. on Document Analysis and Recognition*, 2021, pp. 70–84.

[20] E. Parada-Cabaleiro, A. Batliner, and B. W. Schuller, "A Diplomatic Edition of Il Lauro Secco: Ground Truth for OMR of White Mensural Notation," in *Proc. of the 20th Int. Society for Music Information Retrieval Conference*, 2019, pp. 557–564.

[21] L. Pugin, R. Zitellini, and P. Roland, "Verovio: A library for engraving MEI music notation into SVG," in *Proc. of the 15th Int. Society for Music Information Retrieval Conference*, 2014, pp. 107–112.

[22] L. Kang, P. Riba, M. Rusiñol, A. Fornés, and M. Villegas, "Pay attention to what you read: Non-recurrent handwritten text-line recognition," *Pattern Recognition*, vol. 129, p. 108766, 2022.

[23] D. Coquenet, C. Chatelain, and T. Paquet, "End-to-end handwritten paragraph text recognition using a vertical attention network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[24] S. S. Singh and S. Karayev, "Full page handwriting recognition via image to sequence extraction," in *Proc. of the 16th Int. Conf. on Document Analysis and Recognition*, 2021, pp. 55–69.

MEL SPECTROGRAM INVERSION WITH STABLE PITCH

Bruno Di Giorgi[*]
Apple
bdigiorgi@apple.com

Mark Levy[*]
Apple
mark_levy@apple.com

Richard Sharp
Apple
richard_sharp@apple.com

ABSTRACT

Vocoders are models capable of transforming a low-dimensional spectral representation of an audio signal, typically the mel spectrogram, to a waveform. Modern speech generation pipelines use a vocoder as their final component. Recent vocoder models developed for speech achieve a high degree of realism, such that it is natural to wonder how they would perform on music signals. Compared to speech, the heterogeneity and structure of the musical sound texture offers new challenges. In this work we focus on one specific artifact that some vocoder models designed for speech tend to exhibit when applied to music: the perceived instability of pitch when synthesizing sustained notes. We argue that the characteristic sound of this artifact is due to the lack of horizontal phase coherence, which is often the result of using a time-domain target space with a model that is invariant to time-shifts, such as a convolutional neural network.

We propose a new vocoder model that is specifically designed for music. Key to improving the pitch stability is the choice of a shift-invariant target space that consists of the magnitude spectrum and the phase gradient. We discuss the reasons that inspired us to re-formulate the vocoder task, outline a working example, and evaluate it on musical signals[1]. Our method results in 60% and 10% improved reconstruction of sustained notes and chords with respect to existing models, using a novel harmonic error metric.

1. INTRODUCTION

In modern speech synthesis pipelines a first model generates a low-dimensional audio representation, usually the mel spectrogram, from text; and a second model, named Vocoder, transforms the mel spectrogram to an audio waveform. Theoretically, vocoders designed for speech could be directly applied to musical signals; however closer inspection reveals features and constraints that are exclusive to the music domain. For example, unlike speech, music signals can be polyphonic and contain longer sustained notes whose pitch precision and stability is essential.

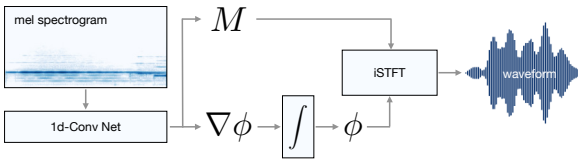


Figure 1: Our proposed model for mel spectrogram inversion. A one dimensional CNN estimates the magnitude and the phase gradient from the mel spectrogram. The phase gradient is then integrated to estimate the phase spectrum and finally audio is obtained via the inverse STFT.

The stability of a sustained pitched note manifests in the time-domain audio signal as the steady repetition of a periodic waveform. Periodic patterns are by definition not shift-invariant, except for shifts of an integer number of periods, therefore they require some form of auto-regression in order to be reproduced accurately. As expected, time-domain vocoders using shift invariant architectures [1, 2], despite other advantages such as generation efficiency, produce jitters that are perceived as pitch and timbre instability. For this reason, other time-domain generative models for audio include an autoregressive mechanism in the neural architecture [3–5]. In practice, time-domain models are required to learn all possible shifts of periodic patterns, a space that increases exponentially for polyphonic music, and how to create smooth sequences of these patterns.

Inspired by a recent generative model for single notes [6], we propose a new vocoder model for music (Fig. 1), where the target of the neural network is an intermediate frequency-domain audio representation that is shift-invariant for sustained notes. This representation is composed of the magnitude spectrum and the phase gradient, and can be later turned to audio via: 1. a phase integration algorithm and 2. the inverse STFT. The proposed design can be used with an efficient shift-invariant neural architecture and still yield stable reconstruction of sustained notes. Specifically, our contributions include:

- a formulation of the mel spectrogram inversion task, matching shift-invariant network and target, in order to improve the perceived stability of sustained notes
- a phase integration algorithm
- an evaluation metric measuring pitch stability for multiple notes

[*]Equal contribution

[1]Reconstruction examples https://machinelearning.apple.com/research/mel-spectrogram

© B. Di Giorgi, M. Levy, and R. Sharp. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** B. Di Giorgi, M. Levy, and R. Sharp, "Mel Spectrogram Inversion with Stable Pitch", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

2. BACKGROUND

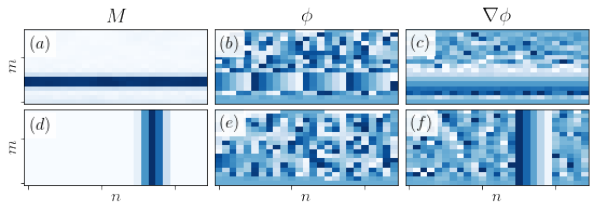


Figure 2: Magnitude M, phase ϕ and phase gradient $\nabla\phi$ patterns for a sinusoidal (top row) and an impulse (bottom row) signal. While the magnitude spectrum is easy to interpret visually, patterns in the phase spectrum are harder to decipher, but become evident in the phase gradient.

A discrete audio signal x can be analyzed in the time-frequency space using the STFT:

$$X[m,n] = \sum_i x[i+nR]w[i]e^{-j\omega_m i}, \tag{1}$$

where m and n are the integer frequency bin and time frame indices, R is the hop size between successive frames, w is a window function defined in the $[-N/2, N/2)$ interval, with N being the frame size and $\omega_m = 2\pi m/N$ the angular frequency. The STFT is a complex-valued matrix, as such it can be represented in the polar form:

$$X[m,n] = M[m,n]e^{-j\phi[m,n]}. \tag{2}$$

The magnitude component M highlights the energy of the signal at various locations in the time-frequency grid: it is easier to interpret and more widely used than the phase component ϕ. However, the phase spectrum is of primary perceptual importance to reconstruct the audio signal precisely, and while harder to interpret at first sight, it does contain patterns that can guide model design choices (Fig. 2).

In Sect. 2.1 we describe two patterns that form in the magnitude and phase spectrum corresponding to the occurrence of ideal sinusoidal and impulsive signal components.

2.1 Sinusoidal and impulsive components

2.1.1 Sinusoidal components

In the magnitude spectrum, sinusoidal components such as any single harmonic of a pitched instrument's sustained note, show up as horizontal lines (Fig. 2(a)). While the magnitude spectrum does not depend on the frame index n, and is therefore shift-invariant, the phase spectrum depends linearly on n (Fig. 2(b)), and the rate of change is given by the frequency of the sinusoidal component. Failing to reconstruct this linear relation between phase and time results in loss of horizontal phase coherence, perceived as unstable pitch, because errors are attributed to sudden changes of the frequency of the sinusoidal component.

2.1.2 Impulsive components

Impulsive components such as the attack of a percussion instrument show up as vertical lines in the magnitude spectrum (Fig. 2(d)). While the magnitude spectrum does not depend on the frequency index m, the phase spectrum depends linearly on m (Fig. 2(e)), and the rate of change depends on the offset between the location of the impulse and the frame center. Failing to reconstruct this linear relation between phase and frequency results in loss of vertical phase coherence, which is perceived as smeared transients, because the errors are attributed to the location of the impulse.

2.2 Phase gradient

The linear patterns that emerge in the phase for sinusoidal and impulsive components are better highlighted in the two components of the phase gradient $\nabla\phi = (\phi'_i, \phi'_m)$.

The partial derivative of phase along the time dimension ϕ'_i is called *instantaneous frequency*. For the bins that belong to sinusoidal components, ϕ'_i is constant and the phase can be propagated horizontally:

$$\phi[m, n+1] = \phi[m,n] + R\phi'_i[m,n]. \tag{3}$$

The partial derivative along the frequency dimension ϕ'_m is called *local group delay*. For the bins that belong to impulsive components, ϕ'_m is constant and the phase can be propagated vertically:

$$\phi[m+1, n] = \phi[m,n] + \phi'_m[m,n]. \tag{4}$$

In the time-frequency reassignment literature (see e.g. [7]), the phase gradient components are used to assign the energy of a spectral bin (m,n) to a nearby point of maximum contribution $(\dot{m}, \dot{n})$:

$$\begin{aligned} \dot{m}[m,n] &= m + \Delta m[m,n] \\ \dot{n}[m,n] &= n + \Delta n[m,n], \end{aligned} \tag{5}$$

where $\Delta n[m,n]$ (Fig. 2(f)) and $\Delta m[m,n]$ (Fig. 2(c)) represent time and frequency bin offsets and are derived from the phase gradient

$$\begin{aligned} \Delta m[m,n] &= \phi'_i[m,n]\tfrac{N}{2\pi} - m \\ \Delta n[m,n] &= -\phi'_m[m,n]\tfrac{N}{2\pi R}. \end{aligned} \tag{6}$$

In the following sections, we discuss how the phase gradient can be used for mel spectrogram inversion.

2.3 Mel spectrogram inversion

The log-amplitude mel spectrogram (simply mel spectrogram from now on) is a low-resolution time-frequency representation that is derived from the power spectrogram M^2, by first warping the frequency axis using the mel scale, then scaling the values to log-amplitude. Estimating the original audio signal x from the mel spectrogram requires recovering the information that has been lost in the direct computation, i.e. the phase information and

the linearly-spaced and higher frequency resolution of the magnitude spectrum.

While the majority of the recent approaches try to learn this inverse transformation end-to-end, this is especially hard for a polyphonic music signal. To precisely reproduce a sustained note, an end-to-end model needs to learn: 1. different patterns for every combination of phase shift and period of a periodic waveform, and 2. how to activate them in the right sequence [6]. Accomplishing both tasks is challenging for speech and arguably even more so for music, which contains generally longer pitched sounds with wider pitch range, possibly multiple concurrent fundamental frequencies (polyphony), and whose absolute precision is essential.

Instead of reconstructing the signal in the time domain, we propose to use as output space an intermediate time-frequency representation consisting of three channels: the magnitude spectrum and the two components of the phase gradient: (M, ϕ'_i, ϕ'_m). The phase gradient is later integrated to estimate the phase spectrum ϕ, and finally audio is computed via the inverse STFT.

A model trained on our proposed output representation does *not* need to learn: 1. the shift variations of periodic waveforms as those are explicitly modeled by the inverse STFT, and 2. how to sequence phase, which is handled via the phase integration algorithm. Differently from the phase spectrum, the phase derivative along time is shift-invariant, thus it is a more suitable target for a shift-invariant architecture, such as the convolutional neural network.

The approaches that have been suggested in the recent years for neural audio synthesis in the time-domain have to use auto-regression to achieve horizontal phase coherence. For example, autoregression is at the core of models like WaveNet [3] and WaveRNN [4], however the fact that it is applied at the rate of audio samples make these model prohibitively expensive for the generation of high-resolution audio signals.

Audio domain shift-invariant convolutional neural vocoders can generate audio samples with much higher efficiency, but are not suited to reconstruct long pitched components precisely. This holds regardless of the training strategy, and includes for example generative adversarial networks (GAN) based models [1, 2] and diffusion based models [8, 9]. A recent neural vocoder for speech mel spectrogram inversion [5] adds an autoregressive loop that works on chunks of audio. The autoregressive nature of this architecture allows performing temporal integration, the operation needed to reconstruct stable sinusoidal components, while advancing by audio chunks rather than samples improves efficiency. However, the poor reconstruction quality observed when applying this model to music signals suggests that it is difficult to learn signal properties such as the rotation of phase from data with sufficient generalization.

In the neural audio synthesis literature, using instantaneous frequency has been considered explicitly in [6], where it is generated alongside the magnitude spectrum in order to reconstruct the audio of single notes, conditioned on the pitch contour and a timbre embedding.

2.4 Phase integration

Recovering the phase spectrum from the phase gradients requires an integration step. Theoretically, perfect integration should be possible under specific constraints, such as continuous phase gradient spectrum and window functions with infinite support. In practice, however there is no closed-form solution for integrating typical discrete phase gradient spectra [10].

Well-known phase gradient integration algorithms have been developed in the Time-Scale Modification (TSM) literature. The standard Phase-Vocoder (PV) algorithm propagates the phase derivative along the time dimension to modify the duration of sinusoidal components [11]. The PV is able to preserve horizontal phase coherence, but struggles with vertical phase coherence, leading to smeared transients. Improvements to the standard phase-vocoder algorithm [12–14] use the magnitude spectrum to identify sinusoidal and/or impulsive components, and can propagate phase in either direction (time or frequency) depending on the local properties of the signal.

The phase gradient integration algorithm that we develop (Sect. 3.2) is inspired by these recent variations, and leads to subjectively improved reconstruction quality, alongside increased computational efficiency. A formal evaluation of the integration algorithm is out of the scope of this manuscript and left as future work.

3. MODEL

In this section we describe our proposed model for mel spectrogram inversion. The model is composed of a time-wise convolutional neural network (Sect. 3.1) that estimates magnitude and phase gradient from the mel spectrogram, and a phase integration algorithm (Sect. 3.2) that estimates the phase spectrum given the phase gradient. The time-domain reconstructed audio is finally obtained via inverse STFT from the magnitude and phase spectra.

3.1 Network architecture

The neural network is a stack of 8 1-d (time) convolutional layers with 1536 hidden channels, a kernel size of 3 frames and ReLU activations (Fig. 3). The input and output have the same number of time frames, but different numbers of frequency bins, and their center frequencies are different, i.e. log-spaced for the mel spectrogram input and linearly spaced for the magnitude and phase gradient outputs. The frequency bins of the input and the magnitude channel of the output are independently standardized using the mean and standard deviation values computed from the training set. We found that a direct path from the input to the magnitude channel of the output leads to significant improvements in the reconstruction of magnitude and the training speed. This direct path consists only of a frequency warping operation from mel- to linear-scale.

The phase gradients (ϕ'_i, ϕ'_m) are computed using the Auger-Flandrin technique [15]. Although we have not ex-

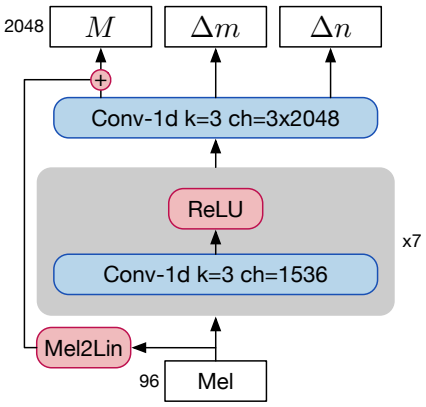


Figure 3: Convolutional network architecture

perimented with other ways to compute the phase gradient, it was argued [16] that using a less precise method, such as finite differences might perform just as well. Instead of using the phase gradients directly, we use the vertical and horizontal bin offsets (Eq. (6)), which are derived from the two components of the phase gradient. In order to remove outliers, the absolute bin offsets are clipped to 4.0 on the frequency dimension and to $N/2R$ on the time dimension.

The model uses linear output activation on all output channels except for the magnitude channel, where we apply a scaled $\tanh$ activation $f_\beta(x) = \beta \tanh(x/\beta)$ with $\beta = 5$ to use mostly the linear regime while also preventing overflow [6]. The three channels are then scaled and offset appropriately to match the statistics of the targets.

3.1.1 Losses

The magnitude channel is trained with a Mean Square Error (MSE) loss term, and a further MSE loss term computed on the first 20 linear-frequency cepstral coefficients (LFCC).

$$\mathcal{L}_1 = (\hat{M} - M)^2 \tag{7}$$

$$\mathcal{L}_2 = (\text{DCT}_{:20}(\hat{M}) - \text{DCT}_{:20}(M))^2, \tag{8}$$

where $\hat{M}$ indicates the estimated magnitude spectrum, M the target magnitude spectrum, and DCT the normalized Discrete Cosine Transform; in these and the following loss formulas the $[m, n]$ indices and the global average operation have been omitted for simplicity. While $\mathcal{L}_1$ acts on point estimates, $\mathcal{L}_2$ pushes the spectral envelope towards its true value, leading to faster convergence and better reconstruction quality.

The phase gradient channels are trained with another MSE loss, weighted by the power spectrum of the target signal M^2. A matrix $\lambda \in [0, 1]$, computed from the phase gradient (Sect. 3.2), is used to distinguish sinusoidal and impulsive components. The idea is that the phase derivative along the time/frequency dimension contributes to the loss only for sinusoidal/impulsive components:

$$\mathcal{L}_3 = \begin{cases} M^2(\hat{\Delta m} - \Delta m)^2 & \lambda > 0.5 \\ M^2(\hat{\Delta n} - \Delta n)^2 & \lambda \leq 0.5, \end{cases} \tag{9}$$

where $(\hat{\Delta m}, \hat{\Delta n})$ are the estimated bin offsets.

Finally, because λ is a function of $\nabla\phi$ and is used for integration, we add a loss term:

$$\mathcal{L}_4 = M^2(\hat{\lambda} - \lambda)^2, \tag{10}$$

where $\hat{\lambda}$ is computed using the phase gradient estimates and λ using the target values. The final loss is a weighted sum of all the loss terms:

$$\mathcal{L} = \sum_l \alpha_l \mathcal{L}_l, \tag{11}$$

where all weights are set to 1 except $\alpha_2 = 0.1$ to balance the contribution of all terms during training.

Differently from time-domain methods for mel spectrogram inversion, we found reconstruction losses to yield satisfying results and did not add any adversarial loss. Evaluating the advantages of including adversarial losses is left for future research.

3.2 Phase integration

The algorithm we use for integrating phase from phase gradients relies on the classification of spectral bins into either sinusoidal, transient or noise components.

The classification uses the phase gradient and relies on the following rationale [7]: around sinusoidal/impulsive components the reassigned frequency/time is approximately constant along frequency/time

$$\lambda[m, n] = e^{-(\frac{d}{dm}\dot{m}[m,n]/\frac{d}{dn}\dot{n}[m,n])^2}, \tag{12}$$

where the derivatives d/dm and d/dn are computed with centered finite differences.

After computing λ, the phase gradients are propagated horizontally (Eq. (3)) if $\lambda > \lambda^{\text{S}}$, vertically (Eq. (4)) if $\lambda < \lambda^{\text{I}}$, and set to a random value otherwise. λ^{I} and λ^{S} are threshold values for impulsive and sinusoidal components, and are used to identify the spectral bins over which respectively vertical or horizontal phase coherence should be enforced. We empirically set $\lambda^{\text{I}} = 0.4$ and $\lambda^{\text{S}} = 0.5$, as these values performed well on early trials.

4. EXPERIMENTS

In this section we discuss how we evaluate the pitch stability of the proposed model, comparing to strong baseline vocoder models from the speech synthesis literature.

4.1 Experimental Setup

We compare the reconstruction of the proposed `phase-gradient` model against state-of-the-art approaches: `melgan` [1][2], `hifigan` [2][3], `cargan` [5][4], and `diffwave` [8][5]. All models have been trained on the same data, containing 13 hours of ambient music loops

[2] https://github.com/descriptinc/melgan-neurips
[3] https://github.com/kan-bayashi/ParallelWaveGAN.git
[4] https://github.com/descriptinc/cargan
[5] https://github.com/lmnt-com/diffwave

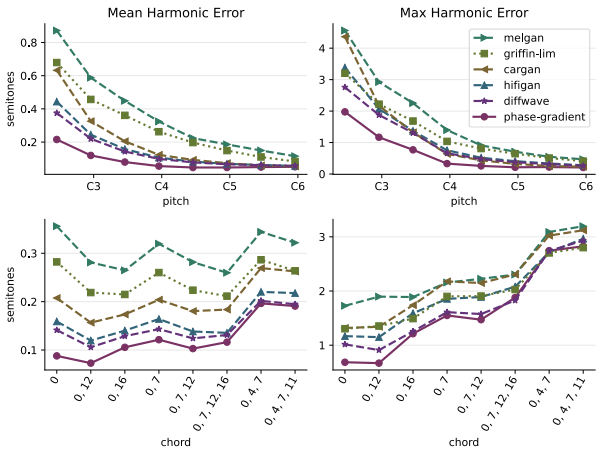


Figure 4: Harmonic error of different mel spectrogram inversion models for synthesized notes in the (C2, C7) range (top row). `phase-gradient` model achieves lower error than the baseline models on the entire range. The values have been smoothed with a moving average with size/stride equal to 12/6 semitones to filter out noise. The bottom row shows the error when adding more notes in different combinations, where the error is averaged over the entire pitch range, and the numbers on the x axis indicate the intervals in semitones that are played simultaneously, e.g. "0, 4, 7" is the major triad.

from commercial libraries[6], split into training, validation and test in the ratio of 80/10/10, which we refer to as the Ambient dataset. The audio from these loop libraries is converted to mono at 44.1kHz, 16-bit, the spectrograms are computed with a frame size of 2048 samples and hop size of 256 samples, and finally 96 bands are used for the mel spectrogram.

All neural networks were trained from scratch. The `phase-gradient` network was trained using 2 Volta GPUs in parallel and batches of 32 examples, with the Adam optimizer and learning rate set to $3e{-}5$. The training was stopped after 1024 epochs, when the validation loss converged, which took approximately 2 days.

As a further non machine-learned baseline, we consider a reconstruction algorithm `griffin-lim`, which generates audio from the mel spectrogram by first warping the frequency axis and the values to obtain a magnitude spectrogram, then applying the Griffin-Lim algorithm [17] for 500 iterations.

4.2 Pitch stability

To evaluate the pitch stability of our model, we used FluidSynth[7] to synthesize a dataset of one second long notes and chords in the (C2, C7) range, using a set of four different sounds: a rhodes piano, a church organ, a string ensemble and a nylon guitar.

[6] we used the following loop packs licensed from Big Fish Audio Ltd.: Ambient Piano, Ambient Skyline 3, Ambient Waves, Eclipse: Ambient Guitars, Ethereal Harp, Zen Ambient Vol. 2

[7] http://www.fluidsynth.org

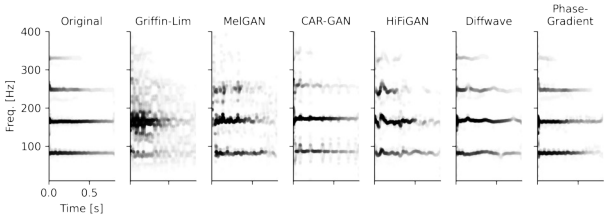


Figure 5: Reconstructions of an E2 nylon guitar note using different mel spectrogram inversion models. The figure shows the reassigned spectrograms [7] which highlight the instability on the fundamental and the harmonics caused by the lack of temporal phase coherence.

Measuring stability with a pitch tracker is only feasible for single notes, and we found that errors computed with a pitch tracker even for monophonic signals did not reflect our perception of reconstruction quality. A possible explanation is that pitch-trackers average out errors in the harmonic frequencies, which is inconvenient in this context because the precise location of the overtones is important for timbre perception [18].

For this reason, we define a harmonic error metric H_{err} as the sum of the frequency errors of the fundamental and the first 4 harmonic frequencies, expressed in the pitch scale:

$$H_{\text{err}}[p,h,n] = 12|\log_2(\frac{\hat{f}_{p,h}[n]}{f_{p,h}[n]})|, \tag{13}$$

where $f_{p,h}[n]$ and $\hat{f}_{p,h}[n]$ are the frequencies of the h-th harmonic of the p-th note, at frame n, for the original and the reconstructed audio respectively; the frequencies are estimated as the closest peaks to the nominal frequency of the partial, using quadratic interpolation, on a magnitude spectrum computed using frame/hop size equal to 4096/256 samples. We look at the mean and maximum values of the harmonic error, as a way to summarize the expected and worst case reconstruction errors.

Results show that our `phase-gradient` model was able to reconstruct the single notes with lower mean and maximum harmonic error over the entire range of notes, especially in the lower pitch range (Fig. 4(a)). A possible explanation for the larger improvement on the low register is that the long waveform period of lower-pitched notes makes them challenging to learn in the time-domain, given the high number of phase variations. A visualization of the different models' reconstruction of the same E2 guitar note is provided in Fig. 5.

We also synthesized different combinations of notes, to test the reconstruction quality on more challenging signals. The combinations include octaves, major 12ths, perfect fifths, an open- and a close-position voicing of the major triad, and a close-position voicing of the major seventh chord. As expected, the harmonic error increases when adding more notes (Fig. 4(b)), as they are harder to recognize in the input mel spectrogram. The `phase-gradient` model is able to yield lower mean and maximum harmonic error on all the considered combi-

	FAD (↓)			H_{err} (↓)		MOS (↑)	RTF (↑)	#Params
	Ambient	NSynth	N+C	Notes	Chords	NSynth		
`griffin-lim` [17]	10.59	6.16	6.79	0.28	0.24	1.42 ± 0.11	7.14	0
`melgan` [1]	2.07	2.80	2.84	0.36	0.30	1.58 ± 0.15	179.90	4M
`cargan` [5]	6.47	8.31	8.45	0.21	0.21	1.74 ± 0.12	4.36	25M
`hifigan` [2]	**0.85**	1.31	**1.81**	0.16	0.17	2.89 ± 0.12	68.12	13M
`diffwave` [8]	2.62	6.84	1.89	0.14	0.15	3.19 ± 0.12	0.32	7M
`phase-gradient`	1.26	**1.26**	1.86	**0.09**	**0.14**	**3.73** ± 0.13	3.58	28M
`oracle`	0.51	0.33	0.00	0.00	0.00	4.34 ± 0.15	—	—

Table 1: Reconstruction results

nations of notes. However, the decrease in the reconstruction quality, particularly on close-position chord voicings, suggests possible connections with the target's frequency resolution.

4.3 Reconstruction quality

To evaluate the overall reconstruction quality of the proposed model, we compute the Frechét Audio Distance (FAD) [19] on the Ambient dataset, the dataset of 1920 one second long notes and chords ("N+C") used in Sect. 4.2, and the NSynth dataset [20]. The results are shown in Table 1 alongside aggregated mean harmonic error results from the experiment discussed in Sect. 4.2.

The FAD metric compares embedding statistics generated on two potentially different sets of audio signals, i.e. evaluation and reference set. On the Ambient and NSynth datasets we compute the FAD between the reconstructed test split (evaluation) and the original training split (reference). On N+C, the reconstructed and original signals from the *entire* dataset are used as evaluation and reference sets. An ideal model (`oracle`) is added to provide reference values, useful when the evaluation and reference sets are different.

The aggregated mean harmonic error results are computed over two meaningful subsets of the N+C dataset: a "Notes" dataset, representing all single notes, and a "Chords" dataset, containing all note combinations with more than one pitch class (see Fig. 4 bottom row).

We conducted a small listening test to evaluate a set of notes reconstructed with different models. The set includes G2 and G3 notes randomly selected from NSynth, for each instrument family. The 5-scale Mean Opinion Score (MOS) values and the 95% confidence intervals are shown in Table 1.

The results show that the `phase-gradient` model is competitive with other state-of-the-art models, despite having simpler neural network and training procedure. The fact that `hifigan` model is able to score lower FAD than the proposed model on the Ambient and N+C datasets suggests it has higher reconstruction accuracy on different sonic characteristics.

Specifically, we noticed that `hifigan` was able to reconstruct impulsive components such as transients and percussive onsets with higher energy and often more accurately than the `phase-gradient` model, while struggling with the stability of pitched notes and chords. The `diffwave` model exhibited high frequency "hissing" noise, but was otherwise surprisingly stable on harmonic components. This quality likely stems from the wide ~7s receptive field, obtained using the entire reverse process at generation time instead of a fast sampling schedule [8]. As reported by its authors in [5], we confirm that the reconstructions made by the `cargan` model contained "boundary artifacts that appear as repeated clicks", compromising their usability. Finally, the reconstructions obtained with the `melgan` model were characterized by heavy phase artifacts such as metallic sounds, and very unstable pitch.

Generation time is also shown in Table 1 in terms of real-time factor (RTF), defined as the number of seconds of audio that can be generated per second, evaluated on a single NVidia Volta GPU. `phase-gradient`'s generation is faster than real time, and close to `cargan`. While `phase-gradient`'s neural network is as fast as `hifigan` and `melgan`, 99% of the generation time is spent during the auto-regressive phase integration stage, suggesting a clear direction for optimization.

5. CONCLUSIONS

In this work we have proposed a new mel spectrogram inversion model designed for music that achieves improved reconstruction of sustained notes and chords, compared to state-of-the-art models from the speech synthesis literature. This improvement is obtained using a frequency-domain target representation that is time shift invariant for harmonic signal components. The proposed model is able to reconstruct single notes and chords respectively 60% and 10% more precisely than existing models, when evaluated with a novel harmonic error metric, while still being competitive on generic loop reconstruction. Potential directions for improvement include using pitch-shift augmentation, investigating log-frequency target representations, and training a separate time-domain model to supply the percussive components.

6. ACKNOWLEDGEMENTS

We would like to thank the following colleagues for their valuable help and feedback: David Varas Gonzalez, Tim O'Brien, Avery Wang, Meghna Ranjit, and André Bergner.

7. REFERENCES

[1] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," *Advances in neural information processing systems*, vol. 32, 2019.

[2] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 022–17 033, 2020.

[3] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *ISCA Speech Synthesis Workshop (SSW)*, 2016.

[4] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. v. d. Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *International Conference on Machine Learning (ICML)*, 2018.

[5] M. Morrison, R. Kumar, K. Kumar, P. Seetharaman, A. Courville, and Y. Bengio, "Chunked autoregressive gan for conditional waveform synthesis," in *International Conference on Learning Representations (ICLR)*, 2022.

[6] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," in *International Conference on Learning Representations (ICLR)*, 2019.

[7] K. R. Fitz and S. A. Fulop, "A unified theory of time-frequency reassignment," *arXiv preprint arXiv:0903.3080*, 2009.

[8] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," in *International Conference on Learning Representations (ICLR)*, 2021.

[9] N. Kandpal, O. Nieto, and Z. Jin, "Music enhancement via image translation and vocoding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 3124–3128.

[10] Z. Pruša and P. L. Søndergaard, "Real-time spectrogram inversion using phase gradient heap integration," in *International Conference on Digital Audio Effects (DAFx)*, 2016, pp. 17–21.

[11] M. Dolson, "The phase vocoder: A tutorial," *Computer Music Journal*, vol. 10, no. 4, pp. 14–27, 1986.

[12] Z. Pruša and N. Holighaus, "Phase vocoder done right," in *IEEE European Signal Processing Conference (EUSIPCO)*, 2017, pp. 976–980.

[13] J. Laroche and M. Dolson, "Improved phase vocoder time-scale modification of audio," *IEEE Transactions on Speech and Audio Processing*, vol. 7, no. 3, pp. 323–332, 1999.

[14] E.-P. Damskägg and V. Välimäki, "Audio time stretching using fuzzy classification of spectral bins," *Applied Sciences*, vol. 7, no. 12, p. 1293, 2017.

[15] F. Auger and P. Flandrin, "Improving the readability of time-frequency and time-scale representations by the reassignment method," *IEEE Transactions on signal processing*, vol. 43, no. 5, pp. 1068–1089, 1995.

[16] S. A. Fulop and K. Fitz, "Algorithms for computing the time-corrected instantaneous frequency (reassigned) spectrogram, with applications," *Journal of the Acoustical Society of America*, vol. 119, no. 1, pp. 360–371, 2006.

[17] N. Perraudin, P. Balazs, and P. L. Søndergaard, "A fast griffin-lim algorithm," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2013, pp. 1–4.

[18] H. Fletcher, E. D. Blackham, and R. A. Stratton, "Quality of piano tones," *Journal of the Acoustical Society of America*, vol. 34, pp. 749–761, 1962.

[19] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms," in *ISCA Interspeech*, 2019, pp. 2350–2354.

[20] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning (ICML)*, 2017.

LATENT FEATURE AUGMENTATION FOR CHORUS DETECTION

Xingjian Du[1] **Huidong Liang**[1] **Yuan Wan**[1] **Yuheng Lin**[1] **Ke Chen**[2] **Bilei Zhu**[1] **Zejun Ma**[1]
[1] ByteDance AI Lab, Shanghai, China
[2] University of California San Diego, San Diego, United States
duxingjian.real@bytedance.com

ABSTRACT

In this paper, we introduce LA-Chorus, a chorus detection model based on **L**atent feature **A**ugmentation and ResNet-FPN architecture. We make three contributions. Firstly, we propose a method for implicitly augmenting chorus data in the latent space during the training stage. Compared to augmentations on audio surfaces such as time stretching and pitch shifting, latent augmentations indicate changes at a higher level in original audio, thereby increasing the diversity and sufficiency in training. Second, we apply Feature Pyramid Network (FPN) to generate additional embeddings from low dimension to high dimension, consequently achieving a multi-scale training paradigm. Lastly, we release Di-Chorus, a new diversified dataset of 13 genres and 14 languages for the community of music structure analysis. In conjunction with other public datasets, we conduct comprehensive experiments to evaluate the performance of our proposed method compared to other state-of-the-art models, where LA-Chorus outperforms other SOTAs by a considerable margin, meanwhile the proposed latent audio augmentation shows dominant advantages over traditional augmentation methods.

1. INTRODUCTION

Chorus detection, aiming for identifying the most "catchy" or "memorable" part of a song, is one of the fundamental tasks in music structure analysis (MSA) [1]. Chorus detection essentially helps better understand music compositions with computational modelling methods and has various applications, such as automatic chorus preview functions in music software that allow users to efficiently select songs from a large library according to their preference [2].

Currently, chorus detection models are based on deep neural network (DNN) architectures with a supervised MSA method, where annotations of different segments are used as target variables during the training stage [3–5]. The common approach is to regard chorus detection as a binary classification task, where each frame (or several frames) is assigned with a class label according to the corresponding segment annotation, and the model is trained to classify these labels [6].

To better perform this classification task, we identify two critical questions: (1) how to locate chorus with high precision as the resolution of feature maps decreases when model goes deeper, and (2) how to learn sufficient variations of chorus characteristics. The first issue requires incorporating chorus positional information into the latent representations learned by models, resembling the task of object detection in computer vision that aims to find the boundaries of target objects [7]. Nevertheless, as the network goes deeper, latent representations begin to lose positional meanings because of the reduced-sized feature maps (i.e. the resolution decreases) [8]. To address the problem of shrinking resolution in feature maps, previous chorus detection methods first used neural network architectures as backbones to generate audio embeddings, and then applied positional modifications on the networks. For example, [3] introduced a multi-task model that jointly detects chorus segments and their boundaries using convolutional neural network (CNN) to increase positioning precision, and [5] proposed a multi-scale CNN model that up-samples/down-samples the original audio features to better capture both global and local information. In this paper, we incorporate Feature Pyramid Networks (FPN) [8], a popular framework for object detection from computer vision, into a standard ResNet [9] as our model's backbone. It is designed specifically to tackle the problem of feature maps' decreasing resolutions by appending a network of reversed size order for each feature map with lateral connection, which will be discussed in Section 3.

The second issue, as learning sufficient chorus characteristics, can be addressed by using a diversified training dataset to feed the model such that it will generalize well at testing time. Nevertheless, despite the promising progress of emerging music annotations such as Isophonics [10], SALAMI [11], and Harmonics [12], the scarcity of labeled data (as they are costly to retrieve) and the deficiency of data diversity have always been challenging for music information retrieval (MIR) and other machine learning fields. To combat this problem, some traditional augmentation techniques have been proposed on the original inputs, such as rotating or flipping the images in computer vision [13], or time stretching [14] and pitch shifting [15] in audio signal processing. Recently, implicit augmentation methods, which focus on the augmentation in latent space, have shown remarkable performance over the pre-

© X. Du, H. Liang, Y. Wan, Y. Lin, K. Chen, B. Zhu, Z. Ma. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** X. Du, H. Liang, Y. Wan, Y. Lin, K. Chen, B. Zhu, Z. Ma, "Latent Feature Augmentation for Chorus Detection", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

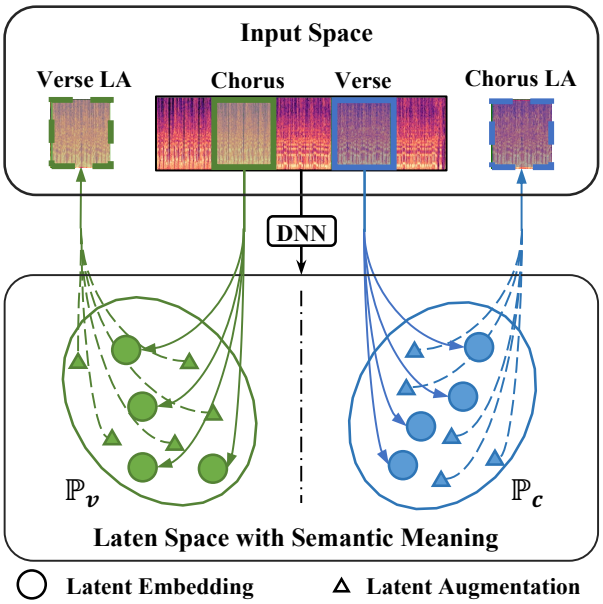


Figure 1. Illustration for implicit audio augmentation in MSA with two annotation types (verse and chorus) from a constant Q transformation (CQT) spectrogram excerpt of "Smooth Criminal" in *Isophonics* [10]. The spectrogram is first encoded into latent embeddings by frame, where chorus embedding and verse embedding follow latent distributions $\mathbb{P}_c$ and $\mathbb{P}_v$ respectively. Then latent augmentations are sampled around embeddings of verse/chorus segments, which correspond to augmented verse/chorus segments in the input space (represented by dashed lines, meaning they are NOT shown explicitly in the input space).

vious "shallow" methods in computer vision [16, 17]. The motivation is that latent representations may carry semantic meanings of original images (e.g. human age, gender, facial expressions) [18]. Such discoveries are also consistent with some works in audio signal processing, where latent audio representations have been found to capture distinctive audio features [19]. For example, [20] investigated the latent spaces of timbre and pitch of various instrument sounds encoded by a GM-VAE model; [21] introduced a Cycle-GAN based model for musical timbre transfer; and [22] disentangled pitch and rhythm representations to produce music analogies. According to these previous works, the latent features of audio correspond to semantic meaning of music and acoustic, such as timbre, rhythm pattern, etc. Therefore, augmentations in the latent space would correspond to changes of semantic features in the input audio segments, which leads to more variations than that of the augmentations in the shallow space. To our best knowledge, there is currently no application of latent augmentations in audio signal processing. In this paper, we illustrate this intuition by an MSA example in Figure 1.

Thus, we propose LA-Chorus, a supervised chorus detection model based on ResNet-FPN architecture that leverages implicit audio augmentations on latent features, which can better locate chorus positions and meanwhile enrich variations in training samples with semantic meanings. Moreover, since songs for most of the public datasets are not easy to retrieve, we further release a diversified collection of songs on YouTube for our MSA community, namely Di-Chorus, which contains 237 songs from 13 genres in 14 languages with annotations by experts. The rest of the paper is structured as follows: the next section introduces related works in chorus detection and latent augmentations, after which we discuss model structure and inference method. The experiment section presents LA-Chorus's performance on public datasets as well as Di-Chorus compared against other state-of-the-art models, followed by an ablation study showing the effectiveness of latent augmentations. Finally, the last section concludes our findings and contributions.

2. RELATED WORK

2.1 Chorus Detection

The origin of chorus detection tightly relates to thumbnailing, which aims to find a short preview (thumbnail) as a meaningful representation of a song [23]. Common approaches for thumbnailing includes evaluating the repeated sections of the audio waveform based on chroma transformation [24], selecting segments with the most repetition [25], and detecting significant change points with self-similarity matrix [26]. On the other hand, MSA assumes that songs contain different types of segments (e.g. chorus, verse, bridge, etc.) with certain structures [27]. Based on the assumption, many chorus detection algorithms in MSA took an unsupervised fashion in the early stage: [28] used heuristics to predict segment labels based on a restricted template for song structures; [29, 30] both applied Hidden Markov Model to derive different song sections; and [31] performed spectral clustering on the co-occurrence matrix generated from k-nearest neighbors.

With the advancement of deep learning in computer vision and natural language processing, DNN gradually makes its presence in MIR, among which ResNet [9], a CNN-based model with residual connections, becomes one of the most popular DNN architectures in recent MIR literature [4, 32]. At the same time, the emergence of labeled databases such as SALAMI [11] and Harmonix [12] make supervised learning gradually attractive for chorus detection, leading to the current paradigm of supervised chorus detection based on deep learning: [33] introduced a hybrid generative model with LSTM to directly predict segment labels; [3] proposed a multi-task method that jointly detects chorus segments and their boundaries; and [5] further proposed a multi-scale network with self-attention convolution to extract latent features of song segments, generating the current state-of-the-art results for chorus detection. In LA-Chorus, we will incorporate Feature Pyramid Network (FPN) [8], a top-down network that dedicates to the object positioning task, into ResNet as our model's backbone. The proposed framework is able to generate latent features with rich positional information and semantic meanings for later augmentation process, which will be discussed in detail in Section 3.

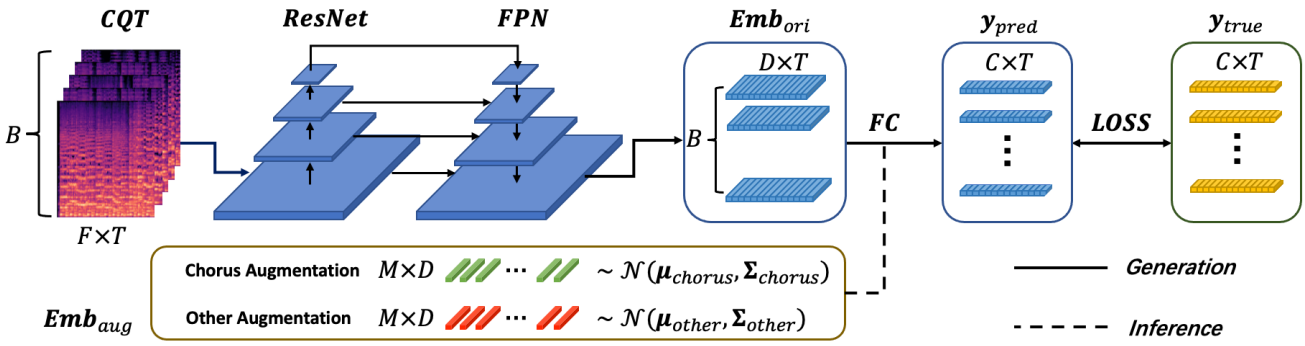


Figure 2. Model structure for LA-Chorus. The solid line represents the model generation processes in the forward pass, while the dashed line represents the inference (i.e. not explicitly generated in the forward pass).

2.2 Audio Data Augmentation

To enhance the diversity and variation in original audio features, traditional audio augmentation techniques such as time stretching, pitch shifting, noise perturbing and SpecAugment [34] are often implemented when transforming audio signals into spectrograms [14, 15]. In this paper, we take a different perspective: augmenting latent representations instead of original audio signals, motivated by the recent advancement of implicit augmentation methods in computer vision, represented by implicit semantic data augmentation (ISDA) and its variants [16, 17]. The ISDA model first used a backbone network to encode input images into latent space with semantic meaning, and then formulated a multi-variate Gaussian distribution for latent features in each class, which was estimated by their mean and covariance within the class via direct calculation. Then augmentations were sampled from the estimated distribution, and the model was later optimized by a novel cross-entropy loss tailored for latent augmentations. In this paper, we will demonstrate how this implicit augmentation method is utilized in audio to improve chorus detection.

3. PROPOSED METHOD

The structure of our proposed model is illustrated in Figure 2. We first use ResNet architecture to encode audio spectrograms into latent embeddings, Then, we apply the latent augmentation by sampling from estimated latent distributions for frames in the "chorus" class and the "non-chorus" (other) class respectively. Finally, the augmented representations are sent to a fully-connected layer to generate probability predictions. At learning and inference stages, we adopt a special cross-entropy loss that handles infinite number of latent augmentations via an upper bound, which saves the cost of sampling procedure.

3.1 ResNet-FPN

We first obtain the constant Q transform (CQT) spectrograms with F frequency bins and T frames in time domain after padding. Then a *ResNet-50* architecture is implemented as the embedding extractor $\mathcal{G}_\theta$ to extract latent embedding. Specifically, the ResNet consists of four stages that contains 3, 4, 6 and 3 residual CNN blocks respectively, where 64 convolution filters of 7×7 kernel size and a max-pooling layer of 3×3 kernel size are designed prior to these residual blocks to process the inputs.

With the size of each feature map reducing as CNN goes deeper, semantic information in deep audio features increases [8]. However, at the same time, the resolution of the feature map decreases and undermines the precision of chorus positioning. To solve this problem, we modify the backbone $\mathcal{G}_\theta$ by incorporating a Feature Pyramid Network (FPN) [8] into our ResNet architecture to construct latent features of high resolution from latent features with semantic information but of low resolution,as shown in Figure 2. The FPN design, different from the bottom-up ResNet part, takes a top-down approach that comprises four 1×1 convolutional layers that corresponds to four residual blocks of ResNet, which maps the low-dimensional outputs of ResNet to high-dimensional latent features with lateral connections.

As a result, the final latent representation $\mathbf{A}$ is of shape $T \times D$, where T is the number of frames and D is the number of latent dimensions. Because of the CNN and FPN designs in our backbone, latent embeddings not only contain both temporal and frequent information of the input audio, but also integrate feature maps of multiple resolutions to better locate chorus segments, further benefiting the latent augmentations later.

3.2 Latent Augmentations on Audio Features

To enrich variations, we apply latent augmentations on each representation $\mathbf{a}_i \in \mathbb{R}^D$ in the song, where $\mathbf{a}_i$ denotes the i_{th} row in $\mathbf{A}$. Similar to other latent variable models such as VAE variants [35] and flow-based models [36], we make a fundamental assumption that latent features within the same class follow the same latent distribution. Specifically, a latent augmentation $\tilde{\mathbf{a}}_i \in \mathbb{R}^D$ for latent feature $\mathbf{a}_i$ follows a multi-variate Gaussian distribution $\mathcal{N}(\mathbf{a}_i, \mathbf{\Sigma}_{y_i})$, where y_i indicates the label class ("chorus" or "other") for frame i, and $\mathbf{\Sigma}_{y_i} \in \mathbb{R}_+^{D \times D}$ represents the covariance matrix for class y_i. Then, we can sample from the distribution regarding to each $\mathbf{a}_i$ to get latent augmentations. In practice, a hyperparameter $\lambda > 0$ is imposed on the covariance matrix $\mathbf{\Sigma}_{y_i}$ to control the deviation of augmentations,

which leads to the final distribution of augmented latent representation $\tilde{\mathbf{a}}_i$:

$$\tilde{\mathbf{a}}_i \sim \mathcal{N}(\mathbf{a}_i\,, \lambda\boldsymbol{\Sigma}_{y_i}). \tag{1}$$

To estimate the covariance matrix $\boldsymbol{\Sigma}$ for different classes, we mathematically calculate the covariance matrix estimate $\hat{\boldsymbol{\Sigma}}_c$ for class c (c is either "chorus" or "other") within the dataset:

$$\hat{\boldsymbol{\Sigma}}_c = \frac{1}{N_c}\sum_{i=1}^{N_c}(\mathbf{a}_i - \bar{\mathbf{a}})(\mathbf{a}_i - \bar{\mathbf{a}})^T, \tag{2}$$

where $\bar{\mathbf{a}} = 1/N_c \sum_{i=1}^{N_c} \mathbf{a}_i$ is the mean of latent features in class c, and N_c is the number of frames belong to class c. This covariance estimate is updated at each iteration after the generation of new latent features during training stage.

Finally, the augmented latent features $\tilde{\mathbf{A}}$ are sent to a fully connected layer with weight $\mathbf{W} \in \mathbb{R}^{D\times C}$ and bias $\mathbf{b} \in \mathbb{R}^C$ to generate the probability predictions, with $\tilde{\mathbf{a}}$ being row-vectors in $\tilde{\mathbf{A}}$:

$$\hat{\mathbf{y}} = \mathcal{F}_\phi(\tilde{\mathbf{A}}) = \tilde{\mathbf{A}}\mathbf{W} + \mathbf{b}. \tag{3}$$

In the next section, we will show an computationally efficient method for learning, which considers infinite augmentations but requires no explicit calculation of the augmented features $\tilde{\mathbf{A}}$.

3.3 Inference and Learning

Given a dataset of size N, a basic approach to formulate a loss function that treats each augmented latent features as a new sample point, and sample M augmentations for each latent feature, which results in an augmented dataset of $N \times (M+1)$ samples. Then we use cross-entropy loss function to train the model. This method is effective when M is relatively large, however, it is computationally expensive as we need to compute extra loss values for $N \times M$ augmentations.

Instead of computing the loss function with discrete augmentations, we adopt the loss function from [16] that incorporates latent augmentations from the continuous domain. For the final fully-connected layer, we denote $\mathbf{w}_c$ as the column vector in $\mathbf{W}$ and b_c as the bias element in b for class c, then the limit of the cross-entropy loss for M augmentations when M approaches to infinity equals:

$$\lim_{M\to\infty} -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{j=1}^{M}\log\Big(\frac{\exp(\mathbf{w}_{y_i}^T\mathbf{a}_i^{(j)} + b_{y_i})}{\sum_{c=1}^{C}\exp(\mathbf{w}_c^T\mathbf{a}_i^{(j)} + b_c)}\Big) \tag{4}$$

which is equivalent to calculating the expectation w.r.t. $\tilde{\mathbf{a}}_i$:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{\tilde{\mathbf{a}}_i}\Big[\log\Big(\frac{\exp(\mathbf{w}_{y_i}^T\tilde{\mathbf{a}}_i + b_{y_i})}{\sum_{c=1}^{C}\exp(\mathbf{w}_c^T\tilde{\mathbf{a}}_i + b_c)}\Big)\Big] \tag{5}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{\tilde{\mathbf{a}}_i}\Big[\log\Big(\sum_{c=1}^{C}\exp(\tilde{\boldsymbol{\xi}})\Big)\Big] \tag{6}$$

where $\tilde{\boldsymbol{\xi}} = (\mathbf{w}_c - \mathbf{w}_{y_i})^T\tilde{\mathbf{a}}_i + (b_c - b_{y_i})$.

Since $\log()$ is a concave function, from Jensen's inequality, we can show:

$$\mathcal{L}_{upper} = \frac{1}{N}\sum_{i=1}^{N}\log\Big(\mathbb{E}_{\tilde{\mathbf{a}}_i}\big[\sum_{c=1}^{C}\exp(\tilde{\boldsymbol{\xi}})\big]\Big) \tag{7}$$

$$\geq \mathcal{L}. \tag{8}$$

As $\tilde{\mathbf{a}}_i$ follows $\mathcal{N}(\mathbf{a}_i\,, \lambda\boldsymbol{\Sigma}_{y_i})$ in Eqn. (1), and $\tilde{\boldsymbol{\xi}}$ is a linear transformation of $\tilde{\mathbf{a}}_i$, then $\tilde{\boldsymbol{\xi}}$ will also follow a Gaussian distribution:

$$\tilde{\boldsymbol{\xi}} \sim \mathcal{N}\big(\boldsymbol{\xi}\,, \Delta\big), \tag{9}$$

where $\boldsymbol{\xi} = (\mathbf{w}_c - \mathbf{w}_{y_i})^T\mathbf{a}_i + (b_c - b_{y_i})$ and $\Delta = \lambda(\mathbf{w}_c - \mathbf{w}_{y_i})^T\boldsymbol{\Sigma}_{y_i}(\mathbf{w}_c - \mathbf{w}_{y_i})$.

Given the moment generating equation $\mathbb{E}[\exp(tx)] = \exp(t\mu + \frac{1}{2}\sigma^2t^2)$ for $x \sim \mathcal{N}(\mu, \sigma^2)$, we can express Eqn. (7) as follows:

$$\mathcal{L}_{upper} = \frac{1}{N}\sum_{i=1}^{N}\log\Big(\sum_{c=1}^{C}\exp(\boldsymbol{\xi} + \frac{1}{2}\Delta)\Big) \tag{10}$$

$$= -\frac{1}{N}\sum_{i=1}^{N}\log\Big(\frac{\exp(\mathbf{w}_{y_i}^T\mathbf{a}_i + b_{y_i})}{\sum_{c=1}^{C}\exp(\mathbf{w}_c^T\mathbf{a}_i + b_c + \frac{1}{2}\Delta)}\Big), \tag{11}$$

which gives us a tractable upper-bound of Eqn. (6)

Therefore, we do not need to explicitly sample augmentations from its distribution in Eqn (1). Instead, only the covariance matrix $\boldsymbol{\Sigma}_c$ of latent features for each class requires update at each iteration, which speeds up the convergence compared to discrete estimation.

The algorithm for learning the embedding extractor $\mathcal{G}_\theta$ and the fully-connected layer $\mathcal{F}_\phi$ is demonstrated in Algorithm 1 below. Because the model is underfitted at beginning epochs, the hyperparameter λ for controlling augmentation deviation is set to be $\lambda_0 \times \frac{\text{epoch}}{\text{total epoch}}$ to alleviate the impact of augmentation at the starting stage of training.

Algorithm 1 Algorithm for training LA-Chorus

Require: Padded CQTs batches; ResNet extractor $\mathcal{G}_\theta$; A fully connected layer $\mathcal{F}_\phi$; Initial covariance matrix $\boldsymbol{\Sigma}_c$ for each class; Initial λ_0 for scaling augmentation

```
for epoch = 1, 2, ..., I do
    for batch = 1, 2, ..., K do
        Encode the CQT batch into latent features
        {a_i}_{i=1}^T via ResNet-FPN extractor G_θ
        Update G_θ and F_φ by computing:
        ∇_{θ,φ} L_upper from Eqn (11)
    end for
    Update Σ_c across all batches with Eqn. (2)
    λ ← λ_0 × epoch/I
end for
return G_θ and F_φ
```

4. EXPERIMENTS

In this section, we conduct comprehensive experiments to evaluate LA-Chorus's performance against other state-of-

Dataset	#Songs	#Genres	#Lang.	#Quality
Di-Chorus	237	13	14	3

Table 1. Key statistics of Di-Chorus.

the-art (SOTA) methods in chorus detection. We first introduce the experimental setups, where details of our newly released dataset Di-Chorus and hyperparameter settings in LA-Chorus are discussed. Then we present the results of chorus detection by LA-Chorus and other methods on popular public datasets, followed by an ablation study that demonstrates the effectiveness of different modules in our proposed method.

4.1 Experimental Setup

For a fair comparison purpose, we conduct our experiment under the same settings in [5] with a cross-dataset paradigm (i.e. testing on different datasets that are not used in training). Specifically, we use 890 songs that contain chorus segments in *Harmonix* [12], along with 38 songs of Michael Jackson and 83 songs of The Beatles in *Isophonics* set [10] for training and validation. At testing time, we use three public datasets and our released dataset Di-Chorus to evaluate our model and other methods. The public datasets used for testing are: 100 "Popular" songs from *RWC* [37, 38], 210 "Popular" songs (denoted as *SP*) and 198 "Live" songs (denoted as *SL*) from *SALAMI* [11], which are chosen in consistence with the testing sets in [5].

Our newly released dataset, Di-Chorus [1] (denoted as DC), contains 237 music annotations of songs on YouTube labeled by experts. Compared to previous datasets mentioned above, songs in Di-Chorus are more easy-to-access from the appended YouTube URLs, and are more diversified since it consists of musics tracks in 14 languages as opposed to other existing datasets that are mostly in English (e.g., Harmonics) or just two or three languages (e.g., RWC and SALAMI). In addition, we also include three different recording qualities to improve the variation within dataset: Studio, Live and Original Sound Track (OST) which contains non-music segments. The key statistics are summarized in Table 1 below.

To demonstrate the performance of our proposed model on the above datasets, we compare LA-Chorus against the following methods:

- **CNMF** [39]: an unsupervised matrix factorization method from *MSAF* [40].
- **SCluster** [31]: a spectral clustering method based on frame co-occurrence matrix from *MSAF* [40].
- **Highlighter** [41]: an CNN model that takes an unsupervised approach to detect emotional highlights as chorus segments.

[1] We provide some demos of Di-Chorus in the supplementary material. Di-Chorus will be made publicly available upon acceptance for retrieval.

Models	AUC on Different Datasets			
	RWC	SP	SL	DC
CNMF	.526	.543	.478	.488
SCluster	.533	.545	.551	.568
Highlighter	.804	.703	.671	.553
Multi2021	.819	.675	.633	-
DeepChorus	.842	.780	.765	.811
LA-Chorus	**.906**	**.887**	**.831**	**.872**

Table 2. AUC results for chorus detection in various models.

Models	F1-score on Different Datasets			
	RWC	SP	SL	DC
CNMF	.403	.422	.340	.332
SCluster	.427	.448	.392	.603
Highlighter	.407	.303	.251	.283
Multi2021	.643	.473	.380	-
DeepChorus	.675	.611	.501	.662
LA-Chorus	**.728**	**.619**	**.526**	**.707**

Table 3. F1-score results for chorus detection in various models.

- **Multi2021** [3]: a CNN model based on a multi-task learning objective that jointly predicts chorus segments and their boundaries.
- **DeepChorus** [5]: a CNN model based on multi-scale networks and self-attention, which is the current state-of-the-art method for chorus detection.

Then, we validate the prediction results by AUC score (Area Under Curve) and F1 score. To evaluate these two metrics, we first create a sequence of the song length from the original annotation, with each element indicating the class of the corresponding segment. Then we can calculate AUC and F1 score for each song independently and take the average over them as the final result.

For the training details, we resample the audio at 22050 Hz and use CQT as our input feature with 12 bins per octave, where Han windowing function is applied with a hop size of 512 for extraction. The model is trained for 100 epochs with a batch size of 32 and a learning rate of 10^{-4} with a cosine decay scheduler. The code is implemented in PyTorch and run at a Tesla-V100-SXM2-32GB GPU.

4.2 Chorus Detection

We retain the experiment results for the chosen SOTAs on RWC, SP and SL from [5], and test them on Di-Chorus with the default settings in their papers, as shown in Table 2 and Table 3 by AUC score and F1-score respectively. Note we do not test Multi2021 [3] on Di-Chorus, since their code is not open-sourced.

For AUC metric, LA-Chorus outperforms other SOTAs

on all datasets by a big margin. Compared to *DeepChorus* [5], which is considered as the current best method for chorus detection, our method improves the performance by over 0.06 across all datasets. The performances on F1-score also exhibit a similar pattern, where LA-Chorus generates better predictions over other models with a considerable improvement on each dataset. In particular, our model performs exceptionally well on the widely used *RWC* dataset that reaches to an AUC score of 0.906 and an F1-score 0.728. We give most of the credits to the implicit augmentation design in our model, and we illustrate this perspective in the ablation study section.

4.3 Ablation Study

To analyze the effectiveness of FPN and latent augmentation, we test our LA-Chorus by separate modules: 1) ResNet backbone only, 2) ResNet with FPN, and 3) ResNet with latent augmentation (denoted as + *LA*). To demonstrate the efficacy of applying latent augmentations over traditional audio augmentation techniques in the input space, we further show the results of applying 4) time stretching (denoted as + *TS*) and 5) pitch shifting (denoted as + *PS*) to the ResNet backbone, with the results summarized in Table 4 for AUC and Table 5 for F1-score below (Note we do not apply TS or PS in our proposed method).

From the results, we can observe that by incorporating FPN, we improve the vanilla ResNet backbone by a remarkable increase of over 0.05 on most of the datasets under both AUC and F1-score metrics, expect for *SALAMI-Live* where the result remains the same. Such findings indicate that FPN is an effective method to locate music segments by increasing the resolution of feature maps, which, without any augmentation, can already generate predictions that are comparative to *DeepChorus*.

On the other hand, when we apply implicit augmentations to latent features generated by ResNet (without FPN), significant improvements of over 0.10 are witnessed for both AUC and F1 scores on most datasets. The results are even notably better than that of the *ResNet+FPN* combination who contains important positional information, implying the dominant role of latent augmentation in the strong performance of LA-Chorus. The results from *ResNet+TS* and *ResNet+PS* further corroborate the benefit of leveraging latent augmentations for chorus detection. Although there are some effects after adopting these two traditional augmentation methods, their improvements on the original model seem incremental compared to that of *ResNet+LA*. Instead, the implicit augmentation method outperforms traditional augmentation methods by a significant margin for both metrics on each dataset, which implies a clear advantage for adopting latent augmentations.

4.4 Discussion of Limitation

Despite of the prominent performance of LA-Chorus, we believe there are still some potential limitations for future explorations. First, further investigation is needed to verify that the latent augmentations are realistic to human when transforming them back to input domain. One possible way is to train a reversed model (such as a decoder or flow-based model) that reconstructs latent features to the original inputs. Second, the AUC and F1 metrics might not measure whether the output is overfragmented or underfragmented. It's needed to design metrics that are more perceptually relevant for chorus detection task. Finally, we only focus on detecting chorus segments in this paper, whereas in MSA, there are other annotation types (e.g. verse, bridge, etc.) to be modeled [4, 33]. We believe LA-Chorus only requires minor modifications in the class dimension of latent augmentations (i.e. augmenting chorus, verse, bridge, etc.) before being applied to predict other label types in music structure analysis.

Ablations	AUC on Different Datasets			
	RWC	SP	SL	DC
ResNet	.801	.773	.767	.751
ResNet + FPN	.865	.830	.767	.807
ResNet + LA	.882	.854	.824	.847
ResNet + TS	.818	.787	.765	.762
ResNet + PS	.822	.777	.789	.766
LA-Chorus	.906	.887	.831	.872

Table 4. AUC results for ablation study.

Ablations	F1-score on Different Datasets			
	RWC	SP	SL	DC
ResNet	.592	.415	.418	.588
ResNet + FPN	.648	.473	.478	.608
ResNet + LA	.692	.540	.516	.687
ResNet + TS	.576	.394	.365	.553
ResNet + PS	.590	.378	.395	.545
LA-Chorus	.728	.619	.526	.707

Table 5. F1-score results for ablation study.

5. CONCLUSION

In this paper, we introduced a novel chorus detection model based on ResNet-FPN architecture with latent augmentations on audio features. The proposed method, different from traditional augmentation algorithms focusing on the input space, augments audio features in the latent space to explore semantic changes in audio data. Besides, we released a new diversified dataset, Di-Chorus, with expert annotations, which contains songs with 13 genres in 14 languages and 3 qualities. Comprehensive experiments have been conducted on public datasets and Di-Chorus, where LA-Chorus shows superior performance against other methods. Lastly, the effectiveness of different modules in LA-Chorus are validated by an ablation study. In the future, we plan to investigate more details on the semantic changes of audio data via latent augmentations and the extensibility of LA-Chorus to other MIR tasks.

6. REFERENCES

[1] J. van Balen, J. A. Burgoyne, F. Wiering, R. C. Veltkamp *et al.*, "An analysis of chorus features in popular song," in *Proceedings of the 14th Society of Music Information Retrieval Conference (ISMIR)*, 2013.

[2] M. Goto, "A chorus-section detecting method for musical audio signals," in *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).*, vol. 5. IEEE, 2003, pp. V–437.

[3] J.-C. Wang, J. B. Smith, J. Chen, X. Song, and Y. Wang, "Supervised chorus detection for popular music using convolutional neural network and multi-task learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 566–570.

[4] J.-C. Wang, J. B. Smith, W.-T. Lu, and X. Song, "Supervised metric learning for music structure feature," in *International Society for Music Information Retrieval Conference*, 2021.

[5] Q. He, X. Sun, Y. Yu, and W. Li, "Deepchorus: A hybrid model of multi-scale convolution and self-attention for chorus detection," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[6] K. Ullrich, J. Schlüter, and T. Grill, "Boundary detection in music structure analysis using convolutional neural networks," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[8] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, and D. Tidhar, "Omras2 metadata project 2009," in *In Late-breaking session at the 10th International Conference on Music Information Retrieval (ISMIR*, 2009.

[11] J. B. L. Smith, J. A. Burgoyne, I. Fujinaga, D. De Roure, and J. S. Downie, "Design and creation of a large-scale database of structural annotations." in *ISMIR*, vol. 11. Miami, FL, 2011, pp. 555–560.

[12] O. Nieto, M. McCallum, M. E. Davies, A. Robertson, A. M. Stark, and E. Egozy, "The harmonix set: Beats, downbeats, and functional segment annotations of western popular music." in *ISMIR*, 2019, pp. 565–572.

[13] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of big data*, vol. 6, no. 1, pp. 1–48, 2019.

[14] T. Ko, V. Peddinti, D. Povey, and S. Khudanpur, "Audio augmentation for speech recognition," in *Sixteenth annual conference of the international speech communication association*, 2015.

[15] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal processing letters*, vol. 24, no. 3, pp. 279–283, 2017.

[16] Y. Wang, X. Pan, S. Song, H. Zhang, G. Huang, and C. Wu, "Implicit semantic data augmentation for deep networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[17] S. Li, K. Gong, C. H. Liu, Y. Wang, F. Qiao, and X. Cheng, "Metasaug: Meta semantic augmentation for long-tailed visual recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 5212–5221.

[18] W. Liu, R. Li, M. Zheng, S. Karanam, Z. Wu, B. Bhanu, R. J. Radke, and O. Camps, "Towards visually explaining variational autoencoders," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[19] P. Agrawal and S. Ganapathy, "Interpretable representation learning for speech and audio signals based on relevance weighting," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2823–2836, 2020.

[20] Y.-J. Luo, K. Agres, and D. Herremans, "Learning disentangled representations of timbre and pitch for musical instrument sounds using gaussian mixture variational autoencoders," in *Proceedings of the 20th Society of Music Information Retrieval Conference (ISMIR)*, 2019, pp. 405–410.

[21] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse, "Timbretron: A wavenet (cyclegan (cqt (audio))) pipeline for musical timbre transfer," in *International Conference on Learning Representations*, 2018.

[22] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[23] W. Chai and B. Vercoe, "Music thumbnailing via structural analysis," in *Proceedings of the eleventh ACM international conference on Multimedia*, 2003, pp. 223–226.

[24] M. A. Bartsch and G. H. Wakefield, "To catch a chorus: Using chroma-based representations for audio thumbnailing," in *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No. 01TH8575)*. IEEE, 2001, pp. 15–18.

[25] M. Muller, N. Jiang, and P. Grosche, "A robust fitness measure for capturing repetitions in music recordings with applications to audio thumbnailing," *IEEE Transactions on audio, speech, and language processing*, vol. 21, no. 3, pp. 531–543, 2012.

[26] M. Cooper and J. Foote, "Automatic music summarization via similarity analysis." in *ISMIR*. Citeseer, 2002.

[27] J. Paulus, M. Müller, and A. Klapuri, "State of the art report: Audio-based music structure analysis." in *Ismir*. Utrecht, 2010, pp. 625–636.

[28] N. C. Maddage, C. Xu, M. S. Kankanhalli, and X. Shao, "Content-based music structure analysis with applications to music semantics understanding," in *Proceedings of the 12th annual ACM international conference on Multimedia*, 2004, pp. 112–119.

[29] G. Peeters, A. La Burthe, and X. Rodet, "Toward automatic music audio summary generation from signal analysis," in *ISMIR*, 2002, pp. 1–1.

[30] J. Paulus, "Improving markov model based music piece structure labelling with acoustic information." in *ISMIR*, 2010, pp. 303–308.

[31] B. McFee and D. P. Ellis, "Analyzing song structure with spectral clustering," in *Proceedings of the 15th Society of Music Information Retrieval Conference (ISMIR)*, 2014, pp. 405–410.

[32] X. Du, Z. Yu, B. Zhu, X. Chen, and Z. Ma, "Bytecover: Cover song identification via multi-loss training," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 551–555.

[33] G. Shibata, R. Nishikimi, and K. Yoshii, "Music structure analysis based on an lstm-hsmm hybrid model," in *International Society for Music Information Retrieval Conference*, 2020, pp. 15–22.

[34] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[35] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," in *International Conference on Learning Representations*, 2014.

[36] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International conference on machine learning*. PMLR, 2015, pp. 1530–1538.

[37] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "Rwc music database: Popular, classical and jazz music databases." in *Ismir*, vol. 2, 2002, pp. 287–288.

[38] M. Goto *et al.*, "Aist annotation for the rwc music database." in *ISMIR*, 2006, pp. 359–360.

[39] O. Nieto and T. Jehan, "Convex non-negative matrix factorization for automatic music structure identification," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 236–240.

[40] O. Nieto and J. Bello, "Systematic exploration of computational music structure research," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[41] Y.-S. Huang, S.-Y. Chou, and Y.-H. Yang, "Pop music highlighter: Marking the emotion keypoints," *Transactions of the International Society for Music Information Retrieval*, vol. 1, no. 1, 2018.

ACCOMONTAGE2: A COMPLETE HARMONIZATION AND ACCOMPANIMENT ARRANGEMENT SYSTEM

Li Yi[1,2] **Haochen Hu**[1,2] **Jingwei Zhao**[3] **Gus Xia**[1,2]
[1] Music X Lab, NYU Shanghai [2] MBZUAI
[3] Institute of Data Science, NUS
ly1387@nyu.edu, hh1933@nyu.edu, jzhao@u.nus.edu, gxia@nyu.edu

ABSTRACT

We propose AccoMontage2, a system capable of doing full-length song harmonization and accompaniment arrangement based on a lead melody. [1] Following AccoMontage, this study focuses on generating piano arrangements for popular/folk songs and it carries on the generalized template-based retrieval method. The novelties of this study are twofold. First, we invent a harmonization module (which AccoMontage does not have). This module generates structured and coherent full-length chord progression by optimizing and balancing three loss terms: a micro-level loss for note-wise dissonance, a meso-level loss for phrase-template matching, and a macro-level loss for full piece coherency. Second, we develop a graphical user interface which allows users to select different styles of chord progression and piano texture. Currently, chord progression styles include Pop, R&B, and Dark, while piano texture styles include several levels of voicing density and rhythmic complexity. Experimental results show that both our harmonization and arrangement results significantly outperform the baselines. Lastly, we release AccoMontage2 as an online application as well as the organized chord progression templates as a public dataset.

1. INTRODUCTION

Accompaniment arrangement is a difficult music generation task involving structured constraints of melody, harmony, and accompaniment texture. A high-quality arrangement could help with various downstream tasks and applications, such as compositional style transfer [1], automatic accompaniment [2], and score-informed source separation and music synthesis [3].

As one of the most promising arrangement systems, AccoMontage [4] uses a generalized template-based approach to first search for roughly-matched accompaniment phrases as the reference and then re-harmonize the selected reference via style transfer. It generates much more coherent results than purely learning-based algorithms, especially for full-length song arrangements.

However, AccoMontage is not yet a "complete" accompaniment generation system in the strict sense, as it still calls for chord input from users and cannot harmonize a melody. To this end, we develop AccoMontage2, a system capable of full-length song harmonization and accompaniment arrangement based on a lead melody by equipping AccoMontage with two extra components: 1) a novel harmonization module, and 2) a graphical user interface.

The main novelty of our system lies in the harmonization module. We first collect a high-quality chord progression dataset and re-organize the phrases with respect to different styles to serve as reference templates. Then, we use dynamic programming (DP) to generate structured and coherent chord progressions given a query lead melody with phrase annotation. Specifically, the DP algorithm optimizes a multi-level loss function consisting of three terms: 1) a micro-level loss for note-wise melody-chord matching, 2) a meso-level loss for phrase-template matching, and 3) a macro-level loss for the whole-piece coherency. The first term evaluates the dissonance between the melody and the candidate chords. The second term prefers chord progressions with the same length as the target melody phrases. The third term computes how well the candidate phrases connect with each other to form an organic whole. Experimental results show that both our harmonization and arrangement results significantly outperform the baselines.

In addition, we develop a graphical user interface which allows the user to select different styles of chord progression and piano texture. Currently, chord progression styles include R&B, Dark, Pop-standard, and Pop-complex. Piano texture styles include several levels of voicing density and rhythmic complexity.

We release the AccoMontage2 as an online application [2] as well as the organized chord progression templates as an open-source dataset.

In brief, the contributions of our paper are as follows:

- A complete system for full-length song harmonization and accompaniment arrangement;
- An effective harmonization algorithm with state-of-the-art performance;

[1] Codes and dataset at https://github.com/billyblu2000/accomontage2.

[2] Online GUI link at https://billyyi.top/accomontage2.

© L. Yi, H. Hu, J. Zhao, and G. Xia. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** L. Yi, H. Hu, J. Zhao, and G. Xia, "AccoMontage2: A Complete Harmonization and Accompaniment Arrangement System", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

- A graphical user interface for controllable piano accompaniment generation.

2. RELATED WORKS

2.1 Melody Harmonization

Melody harmonization refers to the task of generating a harmonic chordal accompaniment for a given melody [5, 6]. It has been typically formulated as a prediction task, *i.e.*, to predict a sequence of chord labels conditioned on the lead melody. Recent mainstream methods range from hidden Markov models [7, 8] to deep neural networks [9–11]. Such models are typically trained to fit a groundtruth melody-chord mapping, but do not account for the fact that one melody can be harmonized with various styles in terms of genre, chord complexity, *etc*. In fact, the current state-of-the-art models [10, 11] only support simple triads and up to a few common seventh chords. Also, predictions are made locally, where neither phrase-level progression nor inter-phrase structures are explicitly considered.

In this paper, we re-formulate melody harmonization with a novel template matching approach. The usage of existing templates for music generation has been a popular idea. Existing template-based methodologies include learning based unit selection [2,12], rule-based score matching [13,14], and genetic algorithms [15]. In our case, we match the lead melody with chord templates from a library based on rule-based criterion and subject to user control. Such an idea is inspired by the fact that music producers tend to pick up off-the-shelf chord templates instead of harmonizing from scratch. In addition, they also have control on what style of the chords to use.

Existing template-matching attempts for harmonization typically focus on half-bar level [16]. In contrast, our model deals with phrase-level matching. We design three loss terms that measure melody-chord fitness at note-wise, intra-phrase, and inter-phrase levels, respectively. Our chord library is finely organized, supporting up to ninth chords with voice leading and various genres. Our model can therefore generate structured and coherent full-length chord progressions with different styles.

2.2 Accompaniment Arrangement

The task of accompaniment arrangement aims to generate an accompaniment conditioned on a given lead sheet (*i.e.*, a lead melody with chord progression). The quality of arrangement is related to chordal harmony, texture richness, and long-term structure. For this task, existing learning-based models often do well in harmony and texture, but are less capable of long-term generation [1, 17–21]. Previous template-matching models can easily maintain long-term structures, but suffer from fixed elementary textures and often fail to generalize [13–15].

To break such a dilemma, the AccoMontage system [4] introduces a generalized template-matching methodology, where phrase-level accompaniment templates are first searched by rule-based criterion, and then re-harmonized via deep learning-based style transfer. The search stage and the style transfer stage each optimize high-level structure and local coherency, thus guaranteeing the arrangement of high-quality accompaniment.

In this paper, we integrate our harmonization module with AccoMontage and upgrade it to a complete accompaniment generation model. An input melody is first harmonized with stylistic chord progression and then arranged with piano textures. With an additional GUI design, our model offers a flexible degree of control, including harmony styles and texture complexity.

3. METHODOLOGY

The system diagram of our AccoMontage2 system is shown in Figure 1. It can achieve full-length song harmonization and accompaniment arrangement. The input of the system is a query lead melody with phrase annotation. The harmonization module will first harmonize the melody. The generated chord progression will be sent into AccoMontage together with the original melody to arrange accompaniment. Lastly, a GUI is provided for users to adjust chord and accompaniment styles.

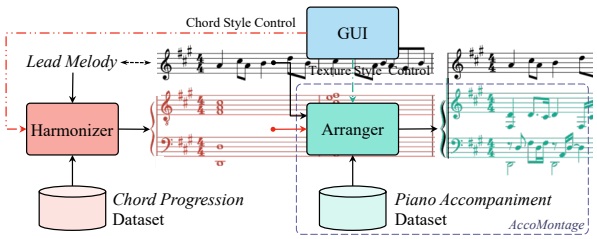


Figure 1: Diagram of AccoMontage2 system.

For the rest of this section, we first introduce the structure of the dataset and how we re-organize it in Section 3.1. Then we describe the harmonization module in Section 3.2. After that, we provide an overview of the AccoMontage system in Section 3.3. Finally, we show how the GUI is constructed to enable style controllability in Section 3.4.

3.1 Dataset Curation

A self-collected chord progression dataset is used as the reference templates for our harmonization algorithm. We create the dataset based on an existing chord progression collection [22] that contains 64,524 MIDI files, most of which are chord progression tracks with different style labels. The original dataset [22] cannot be adapted to our model directly for several reasons. First, some tracks are not pure chord progression, containing mixed melody segments. Second, many style labels are unnecessary. For example, two different styles may have similar musical elements that are hard to differentiate. Third, there are redundant progressions only different in their keys.

To solve the problems above, we process and reorganize the dataset as follows. First, we remove the MIDI files that contain melody segments and complex rhythmic textures. Second, based on our subjective assessment, we

manually define a style mapping function to map the original style labels to newly defined ones. [3] The newly defined styles are: "Pop-standard", "Pop-complex", "Dark", and "R&B". While Pop-standard contains only triad chords, Pop-complex contains seventh, ninth, and sometimes even more chromatic ones. Note that some templates do not have an original style label and are thus labeled as "Unknown". Third, we remove all the redundant progressions.

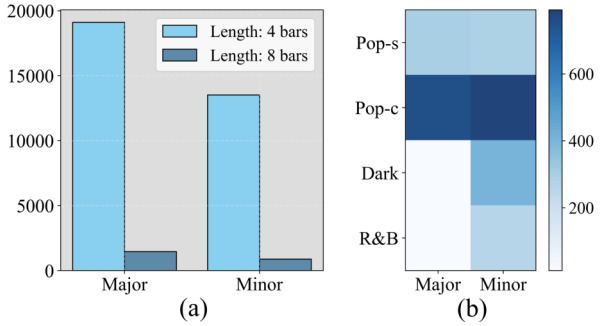


Figure 2: Statistics of the curated dataset.

The final curated dataset contains 5762 pieces of chord progression templates. Each template has 3 additional labels: 1) The length label. Templates are either of 4 bars or 8 bars in length in our dataset. Figure 2 (a) shows a distribution of the length of the templates. 2) The mode label. We currently only label the mode of the templates as either major or minor. 3) The style label. As mentioned above, four styles in total are acquired. The distribution of different styles among modes are shown in Figure 2 (b).

Finally, we represent the reference template space as a collection of tuples:

$$r = \{(r_m^{\text{chord}}, r_m^{\text{label}})\}_{m=1}^{N}, \tag{1}$$

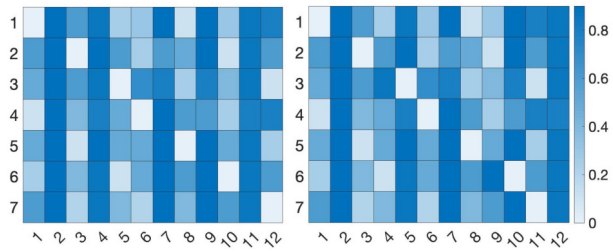
where r_m^{chord} and r_m^{label} are the chord progression and the labels of the i^{th} reference template; N is the volume of the reference space. r_m^{chord} can be seen as a sequence of chords. We quantize and sample the chords at 8th notes from the original chord progression track. Each chord is represented as its root note and a label to indicate whether the corresponding triad is a major triad or a minor triad.

3.2 Harmonization Module

We design a harmonization model that generates structured and coherent full-length chord progression for a given lead melody with phrase annotation. The model takes a multi-phrase melody as input and outputs a list of optimal chord progression identities. Each identity contains a group of progressions that have the same Roman numeral sequence (*e.g.*, I-vi-ii-V) but different styles (*e.g.*, Pop-standard or R&B) and are up for the user to choose. The model considers three levels of losses that involve melody-chord correspondence and is optimized by a dynamic programming (DP) algorithm. Specifically, the DP algorithm optimizes a multi-level loss function consisting of three terms: 1) a micro-level loss for note-wise melody-chord matching, 2) a meso-level loss for phrase-template matching, and 3) a macro-level loss for the full piece coherency.

3.2.1 *Micro-level loss*

The micro-level loss L_{mic} computes the level of dissonance between a melody phrase and candidate progressions note by note. We mainly consider the interval between a melody note and the root of the chord in the corresponding position. The more dissonant the interval is (tritone, minor seconds, etc.), the higher the loss. On the other hand, the more harmonious the interval is (unison, perfect fourth, perfect fifth, etc.), the lower the loss. In addition, the mode of the piece and the harmonic function of the chords will also affect the dissonance level. We refer to the "rank order of consonances and their degree of recurrence" [23] and design the micro-level loss shown in Figure 3, where matrix (a) is for major mode and matrix (b) is for minor. For both matrices, each row represents the degree of the chord (we only consider diatonic chords, *e.g.*, 1 for tonic and 5 for dominant) and each column represents the interval between the melody note and the key center. In Figure 3, the darker the shade, the more dissonant we consider the pair of the melody note and chord is, in which case we set a higher micro-level loss.



(a) Major (b) Minor

Figure 3: Micro-level loss indicating note-wise melody-chord dissonance.

For the i^{th} phrase, L_{mic} is computed by summing up the micro-level loss note by note and then dividing it by the length of the phrase. We further normalize L_{mic} to the range of $[0, 1]$.

3.2.2 *Meso-level loss*

The meso-level loss L_{mes} considers the integrity of a candidate. It encourages chord progressions with the same length as the target melody phrases and penalizes the rest. For a given phrase of melody, we consider chord progressions of the same length and we also concatenate shorter chord progressions to make them candidates. Here, we introduce a transition score to penalize the concatenated candidates. We concatenate two progressions r_m^{chord} and r_n^{chord} from the reference template space, and the transition score considers the very two bars connecting them (*i.e.*, the last bar of r_m^{chord} and the first bar of r_n^{chord}). In specific, we count the occurrence of this two-bar chord progression in the dataset and then normalize it by applying a logarithmic function with N, the size of the reference template space

[3] A detailed description of style mapping is provided in our dataset.

as the base. Suppose the occurrence is $c \in \mathbb{N}$, the transition loss of this $m \to n$ concatenated candidate is defined as:

$$T_{m\to n} := 1 + \log_N \frac{1}{c} \tag{2}$$

Moreover, as we do not wish to penalize the non-concatenated candidates, we introduce δ_1 such that $\delta_1 = 1$ if the candidate is concatenated and $\delta_1 = 0$ otherwise. On the other hand, we do not wish to consider candidates with length not equal to the length of the phrase. Hence we introduce δ_2 such that $\delta_2 = 1$ if the candidate has equal length as the melody phrase, and $\delta_2 = e^{-10}$ otherwise. The meso-level loss of the i^{th} phrase is explicitly:

$$L^i_{\text{mes}} := \delta_1 T_{m\to n} + (\frac{1}{\delta_2} - 1) \tag{3}$$

3.2.3 Macro-level loss

The macro-level loss L_{mac} computes how well the candidate phrases connect with each other. We consider how smooth the transition is from the previous progression to the current progression using the same transition loss as in Section 3.2.2. For the i^{th} phrase and its corresponding progression candidate, L^i_{mac} denotes the macro-level loss of the i^{th} phrase

$$L^i_{\text{mac}} := \begin{cases} 0 & i = 1 \\ T_{m'\to n'} & i = 2, \cdots, p \end{cases} \tag{4}$$

Here, $T_{m'\to n'}$ is the transition loss defined the same way as in Section 3.2.2. m' and n' index progression candidates to be connected. p is the total number of phrases.

Finally, we define the total loss of the s^{th} candidate of the i^{th} phrase by a weighted sum

$$\begin{aligned} L^{i,s}_{\text{total}} = (\beta(1 - L^{i,s}_{\text{mic}}) + (1-\beta)(1 - L^{i,s}_{\text{mes}})) \\ + \max_t \{L^{i-1,t}_{\text{total}} + \alpha(1 - L^{i,s}_{\text{mac}})\}, \end{aligned} \tag{5}$$

where α, β are parameters that we can tune between 0 and 1. Then we use DP to integrate the three levels of losses and search for the optimal chord progressions which minimize the total loss L^i_{total} at $i = p$.

3.3 An Overview of AccoMontage

Based on the harmonization results from the last section, we apply AccoMontage [4] to generate complete piano accompaniments in full length. The AccoMontage system offers a hybrid search-style transfer methodology for accompaniment arrangement: given a lead melody together with a chord progression inferred from Section 3.2, it first searches for reference pieces of accompaniments by dynamic programming. It then re-harmonizes the reference accompaniment to the given chord progression by deep learning-based style transfer. Such a pipeline is inspired by common practice that delicate music textures can often be applied in bulk, instead of composing from scratch.

Specifically, reference accompaniment pieces are searched phrase by phrase based on: 1) phrase-level fitness to the melody, and 2) transition smoothness between consecutive phrases. The overall searching process is optimized by the Viterbi algorithm [24]. The re-harmonization is implemented with a VAE framework which is capable of chord-texture disentanglement [1]. By varying the chord representation, we can re-harmonize the accompaniment while keeping its texture. The whole pipeline of AccoMontage secures a delicate accompaniment arrangement with coherent and structured texture. We refer readers to the original work [4] for more technical details.

3.4 Graphical User Interface for Controllability

AccoMontage2 provides a GUI for users to select styles of harmonized chords and accompaniment textures. The process begins with uploading a MIDI file with a melody track and setting the original chord progression styles and piano texture styles. Three labels have to be assigned manually, including phrase boundaries, key, and mode (major or minor in the current version of the system).

The system will then proceed to generation. When the generation is finished, users are able to: 1) listen to and download the generated audio; 2) select a new harmonization style for each individual phrase or the whole; 3) reset the texture style and re-generate the accompaniment. Figure 4 shows the GUI interface of AccoMontage2.

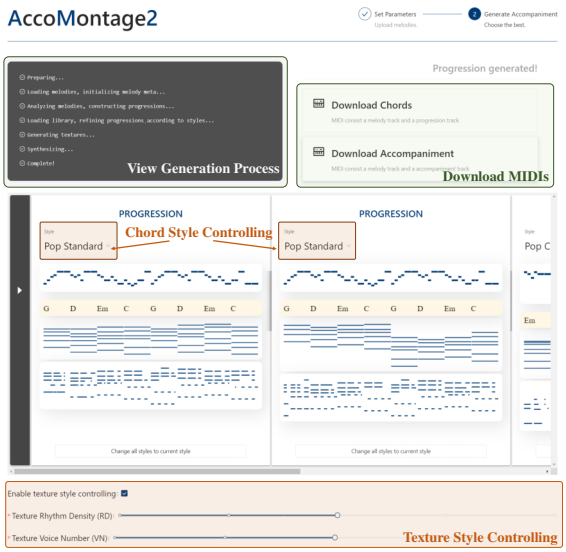


Figure 4: A screenshot of the Graphical User Interface.

4. GENERATION RESULTS

In this section, we showcase one long-term generation result of the AccoMontage2 system. We set $\alpha = 0.1$ and $\beta = 0.5$ in the harmonization algorithm and test our model on a 24-bar melody piece with phrase label `A8A8B8`. The first track in Figure 5 shows the original melody piece and the other two tracks are two different versions of accompaniment generation.

The result shows that AccoMontage2 is able to achieve high controllability in chord style and texture style. On the top of track two (I) and track three (II) in Figure 5, we present using chord notations the harmonization results

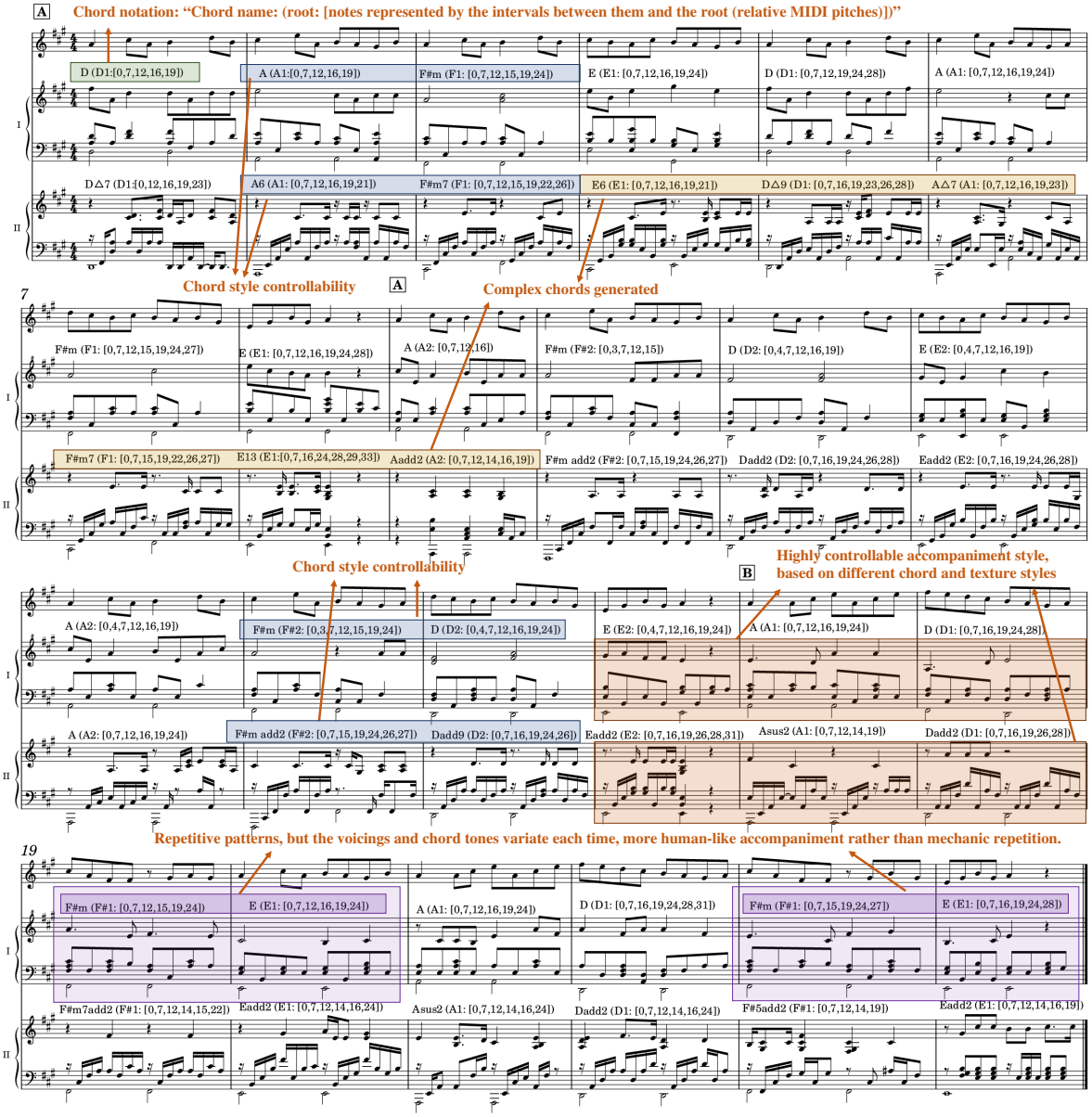


Figure 5: Harmonization and accompaniment arrangement results for *Dinners 1* from the Nottingham Dataset. The 24-bar melody has an `A8A8B8` phrase structure, and the AccoMontage2 system achieves a high degree of style controllability.

of chord style "Pop-standard" and "Pop-complex" respectively. Track two and track three are the whole accompaniment results. While track two (I) is based on "Pop-standard" and a sparse texture style, track three (II) is based on "Pop-complex" and a dense texture style. We see that both results match with the melody in harmonicity and are able to provide chord and texture variations.

5. EVALUATION

We conduct two comparative experiments to validate our AccoMontage2 system, one for harmonization and the other for accompaniment arrangement. We first show the dataset and the baseline model in Section 5.1 and 5.2. Then we present the experimental results in Section 5.3 and 5.4. Audio examples of our proposed system and ablation studies are available via our GUI link.

5.1 Dataset

For the harmonization experiment, we use Nottingham Dataset [25], which provides around 1000 pairs of the query lead melody and the ground-truth chords. For arrangement generation experiments, we use the POP909 Dataset [26], in which each piece contains a lead melody, annotated chords, and a piano accompaniment. For both data sources, we first select the pieces of meter $\frac{2}{4}$ and $\frac{4}{4}$ and then randomly sample 6 pieces for our experiment. We manually annotate their phrase segmentation and only allow the phrase length of 4 bars and 8 bars.

5.2 Baseline Method

We apply the chord generation model [9] based on bidirectional long short-term memory networks (BLSTM) [27] as our baseline model. This model takes a symbolic melody

with the information of time signature, measure, and key as input, and outputs a harmonization result of a sequence of major and minor triads. The BLSTM model considers temporal dependencies by storing both past and future information, reflecting musical context in both forward and backward directions, It is trained on a lead sheet database provided by Wikifonia.org and has achieved reasonable results quantitatively and qualitatively.

5.3 Harmonization Results

We conduct a survey to evaluate the harmonization performance of our model. Our survey has 6 groups of harmonization results and each subject is required to listen to 3 (chosen randomly). Within each group, the subject first listens to a single melody. Each melody is a full-length song randomly selected from the Nottingham Dataset, with an average length of 32 bars (64 seconds). We harmonize the melody using our model and the BLSTM baseline, and additionally acquire an original harmonization using the ground truth chord labels. The subjects are then required to evaluate all three versions of harmonization. The rating is based on a five-point scale from 1 (very poor) to 5 (very high) according to three criteria:

1. **Harmonicity**: How well do the melody and chords stay in harmony with each other;
2. **Creativity**: How creative the harmonization is;
3. **Musicality**: The overall musicality.

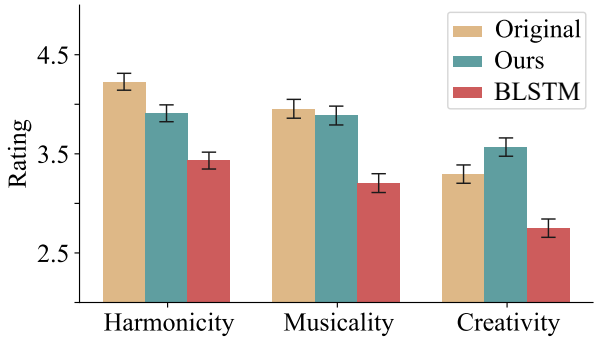


Figure 6: Subjective evaluation for melody harmonization.

A total of 15 subjects with diverse musical backgrounds participated in our survey and we obtain 44 effective ratings for each criterion. As shown in Figure 6, the height of the bars denotes the mean values of the ratings. The error bars stand for the mean square errors (MSEs) computed via within-subject ANOVA [28]. For harmonicity, the original receives the best rating, while our model performs significantly better than the BLSTM baseline. As for the overall musicality, our model is comparable with the original harmonization. For creativity, our model reaches the best. Note that our model supports complex harmonization with voice leading and ninth chords, while the original and the baseline support up to seventh chords in plain root position. Such evaluation results demonstrate that our model introduces *tension* to "flavor" the music while the overall musicality is not affected. For all three criteria, the rating results are statistically significant (p-value $p < 0.05$).

5.4 Accompaniment Generation Results

We conduct another survey to evaluate our model in terms of overall accompaniment generation. In our survey, each subject still listens to 3 groups of generation results (randomly chosen from 6 groups). Within each group, the subjects first listen to a 32-bar melody randomly selected from the POP909 Dataset. Our model generates piano accompaniment through the complete AccoMontage2 pipeline. For the BLSTM baseline, we feed its harmonization results to AccoMontage and obtain the baseline accompaniment. As POP909 contains piano arrangements created by professional musicians for each song, we also have the original accompaniment. The subjects are then required to evaluate all three versions of accompaniment based on the same scale and criteria as in Section 5.3.

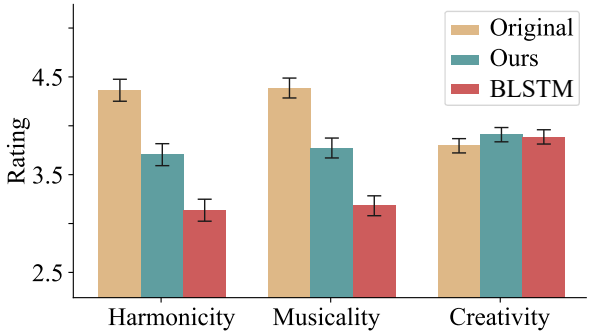


Figure 7: Subjective evaluation for accompaniment arrangement.

We collect a total of 44 effective ratings for each criterion. Figure 7 shows the evaluation results in the same format as in Section 5.3. We report a significantly better performance of our model compared with the BLSTM baseline in harmonicity and musicality ($p < 0.05$), and a marginally better performance than both the baseline and the original in creativity.

6. CONCLUSION AND FUTURE WORK

In conclusion, we contribute a pipeline of algorithms to automatically harmonize and arrange piano accompaniments for melodies of whole-piece popular and folk songs. The system is named after AccoMontage2, built upon its original version which uses a hybrid approach to select accompaniment candidates, edit the candidates using style transfer, and then concatenate them into an organic whole. AccoMontage2 contains two novel modules: a state-of-the-art harmonizer and a GUI for controllable arrangement via interfering in the harmonization and arrangement styles.

In the future, we plan to further optimize the arrangement pipeline by: 1) automatically labeling the melody phrase, 2) extending the model capability to deal with triple meters, and 3) exploring full-band arrangement.

7. REFERENCES

[1] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020, pp. 662–669.

[2] G. Xia, "Expressive collaborative music performance via machine learning," Ph.D. dissertation, Carnegie Mellon University, USA, 2016. [Online]. Available: https://doi.org/10.1184/r1/6716609.v1

[3] L. Lin, G. Xia, Q. Kong, and J. Jiang, "A unified model for zero-shot music source separation, transcription and synthesis," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 381–388.

[4] J. Zhao and G. Xia, "Accomontage: Accompaniment arrangement via phrase selection and style transfer," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 833–840.

[5] C.-H. Chuan, E. Chew *et al.*, "A hybrid system for automatic generation of style-specific accompaniment," in *Proceedings of the 4th international joint workshop on computational creativity*. Goldsmiths, University of London London, 2007, pp. 57–64.

[6] I. Simon, D. Morris, and S. Basu, "Mysong: automatic accompaniment generation for vocal melodies," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2008, pp. 725–734.

[7] H. Tsushima, E. Nakamura, K. Itoyama, and K. Yoshii, "Function- and rhythm-aware melody harmonization based on tree-structured parsing and split-merge sampling of chord sequences," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR*, 2017, pp. 502–508.

[8] ——, "Generative statistical models with self-emergent grammar of chord sequences," *Journal of New Music Research*, vol. 47, no. 3, pp. 226–248, 2018.

[9] H. Lim, S. Rhyu, and K. Lee, "Chord generation from symbolic melody using BLSTM networks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR*, 2017, pp. 621–627.

[10] Y.-C. Yeh, W.-Y. Hsiao, S. Fukayama, T. Kitahara, B. Genchel, H.-M. Liu, H.-W. Dong, Y. Chen, T. Leong, and Y.-H. Yang, "Automatic melody harmonization with triad chords: A comparative study," *Journal of New Music Research*, vol. 50, no. 1, pp. 37–51, 2021.

[11] C.-E. Sun, Y.-W. Chen, H.-S. Lee, Y.-H. Chen, and H.-M. Wang, "Melody harmonization using orderless nade, chord balancing, and blocked gibbs sampling," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 4145–4149.

[12] M. Bretan, G. Weinberg, and L. Heck, "A unit selection methodology for music generation using deep neural networks," *arXiv preprint arXiv:1612.03789*, 2016.

[13] P.-C. Chen, K.-S. Lin, and H. H. Chen, "Automatic accompaniment generation to evoke specific emotion," in *2013 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2013, pp. 1–6.

[14] Y.-C. Wu and H. H. Chen, "Emotion-flow guided music accompaniment generation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 574–578.

[15] C.-H. Liu and C.-K. Ting, "Polyphonic accompaniment using genetic algorithm with music theory," in *2012 IEEE Congress on Evolutionary Computation*. IEEE, 2012, pp. 1–7.

[16] T. Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music," in *Proceedings of the 1999 International Computer Music Conference, ICMC*. Michigan Publishing, 1999.

[17] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[18] H.-M. Liu and Y.-H. Yang, "Lead sheet generation and arrangement by conditional generative adversarial network," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2018, pp. 722–727.

[19] B. Jia, J. Lv, Y. Pu, and X. Yang, "Impromptu accompaniment of pop music using coupled latent variable model with binary regularizer," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–6.

[20] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "Popmag: Pop music accompaniment generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1198–1206.

[21] H. Zhu, Q. Liu, N. J. Yuan, C. Qin, J. Li, K. Zhang, G. Zhou, F. Wei, Y. Xu, and E. Chen, "Xiaoice band: A melody and arrangement generation framework for pop music," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 2837–2846.

[22] N. Kotoulas, "Supersize your midi collection." [Online]. Available: https://www.pianoforproducers.com/nikos-ultimate-midi-pack/

[23] L. Trulla, N. Di Stefano, and A. Giuliani, "Computational approach to musical consonance and dissonance," *Frontiers in Psychology*, vol. 9, p. 381, 04 2018.

[24] G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[25] E. Foxley, "Nottingham database," [EB/OL], https://ifdo.ca/~seymour/nottingham/nottingham.html Accessed May 25, 2021.

[26] Z. Wang*, K. Chen*, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proceedings of 21st International Conference on Music Information Retrieval, ISMIR*, 2020.

[27] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[28] H. Scheffe, *The analysis of variance*. John Wiley & Sons, 1999, vol. 72.

SUPERVISED AND UNSUPERVISED LEARNING OF AUDIO REPRESENTATIONS FOR MUSIC UNDERSTANDING

Matthew C. McCallum **Filip Korzeniowski** **Sergio Oramas**
Fabien Gouyon **Andreas F. Ehmann**
SiriusXM, USA

ABSTRACT

In this work, we provide a broad comparative analysis of strategies for pre-training audio understanding models for several tasks in the music domain, including labelling of genre, era, origin, mood, instrumentation, key, pitch, vocal characteristics, tempo and sonority. Specifically, we explore how the domain of pre-training datasets (music or generic audio) and the pre-training methodology (supervised or unsupervised) affects the adequacy of the resulting audio embeddings for downstream tasks.

We show that models trained via supervised learning on large-scale expert-annotated music datasets achieve state-of-the-art performance in a wide range of music labelling tasks, each with novel content and vocabularies. This can be done in an efficient manner with models containing less than 100 million parameters that require no fine-tuning or reparameterization for downstream tasks, making this approach practical for industry-scale audio catalogs.

Within the class of unsupervised learning strategies, we show that the domain of the training dataset can significantly impact the performance of representations learned by the model. We find that restricting the domain of the pre-training dataset to music allows for training with smaller batch sizes while achieving state-of-the-art in unsupervised learning—and in some cases, supervised learning—for music understanding.

We also corroborate that, while achieving state-of-the-art performance on many tasks, supervised learning can cause models to specialize to the supervised information provided, somewhat compromising a model's generality.

1. INTRODUCTION

In this work, we consider a broad array of classification and labelling tasks under the umbrella of music understanding. Such tasks include the labelling of genre, origin, mood, musical key, instruments, era, emotion and pitch present in music. These tasks have many applications in industry, particularly in music streaming and recommendation services where automated understanding of audio can assist in a range of tasks such as organizing, filtering and personalizing content to a listener's taste and context.

Recent research in automated audio understanding has focused on training convolutional [1–12] and / or attention based [13–21] networks on moderately large collections of frequency-domain audio [2,5–8,12–14,18,20,21], time-domain audio [3, 10, 11, 19] or multi-format / multi-modal [4, 9, 21, 22] data. Such models are often trained on tags encompassing some of the musical labels listed above, achieving promising results [1,3,6,11,13–15,22]. More recent works propose unsupervised strategies for music understanding such as contrastive learning [2, 4, 5, 8–10, 21] or predictive / generative approaches [7, 20, 21, 23]. Unsupervised strategies are appealing because they require no annotated data and generalize well to new tasks [2, 21], but lag the performance of supervised learning at a similar scale [2, 10]. Generative learning strategies [7, 23] have been shown to achieve competitive, and sometimes state-of-the-art (SOTA), performance in several music understanding tasks [24], although, currently there is no evaluation demonstrating the effectiveness of this approach to any of the aforementioned approaches, at comparable scale.

Modern music streaming services have very large music catalogs that amount to many petabytes of audio data if uncompressed. Due to the scale of this data it is desirable to build models that are efficiently scalable, and understand audio in a general enough way that, as needs or requirements change, they may be used to solve novel problems without reprocessing such data. Models in the order of 10M or 100M parameters are currently relatively cost-effective to both train and apply inference to industry-scale catalogs, whilst models consisting of billions of parameters, e.g. that evaluated in [24], are typically impractical, or very expensive, for both training and inference.

More recently, research has adopted approaches producing generalized audio embeddings [2, 4–10, 12, 18, 19, 25] in a supervised or unsupervised way, by training models on large amounts of labelled or unlabelled audio. When such models are applied to novel audio, the internal state of the models has been found to contain much of the information necessary for previously unseen tasks. This is demonstrated by training shallow classifiers (probes) on embeddings consisting of the activations of a given model layer, that map these values to a downstream task. Such an approach achieves competitive results using either unsuper-

© Matthew C. McCallum, Filip Korzeniowski, Sergio Oramas, Fabien Gouyon, Andreas F. Ehmann. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Matthew C. McCallum, Filip Korzeniowski, Sergio Oramas, Fabien Gouyon, Andreas F. Ehmann, "Supervised and Unsupervised Learning of Audio Representations for Music Understanding", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

vised [25] or supervised [6] learning. Most importantly, the embeddings on which the probes are trained are many orders of magnitude smaller than the audio itself and only need to be computed once per audio file. Such embeddings can be stored efficiently, and downstream classifiers can be trained with significantly less resources. The excellent performance, generality, and scalability of this approach are crucial factors for its utility in industry.

This approach to audio understanding has been highlighted in recent benchmarks such as HARES [2] and HEAR [26], where embeddings are evaluated across a number of audio understanding tasks pertaining to a range of content types. Any score aggregation across these benchmarks to determine a "best" embedding is difficult due to the disparate range of metrics employed and furthermore, may obfuscate the strengths and weakness of any given approach. However, evaluating across a common range of tasks can be useful in comparing such strengths and weaknesses. We find that the current tasks evaluated in the HEAR and HARES benchmarks are lacking in evaluation on music content. Wrt. polyphonic music, the HARES benchmark includes only the Magnatagatune dataset, and the HEAR benchmark includes only GTZAN genre and music / speech datasets. While other public music datasets exist, such benchmarks are somewhat limited by the requirement to provide access to the audio of all datasets.

Here, we do not intend to establish a new open benchmark, but investigate the effectiveness of supervised and unsupervised learning for audio embeddings employed specifically for music understanding, across as broad an array of tasks as is available within time and resource constraints. For supervised learning we train models on large scale datasets of annotated magnitude log-mel spectrograms both in the music domain and in the general audio domain. For unsupervised learning we train contrastive models using SimCLR loss [27, 28] on the same sets of magnitude log-mel spectrograms, excluding annotations.

The contributions of this work are as follows: we provide a broad analysis of supervised and unsupervised learning strategies for pre-training audio models for music understanding; we show that for multilabel / multiclass classification of music, large-scale supervised learning on music data achieves SOTA performance, in many cases outperforming both prior SOTA and unsupervised learning by significant margins; we show that supervised learning on labelled music data does not generalize as well as unsupervised learning to novel tasks not covered in those labels; finally, we show that the domain of pre-training audio datasets has a significant impact on the performance of embeddings, particularly for unsupervised learning.

2. PRE-TRAINING METHODOLOGY

To achieve the objectives outlined in Section 1, we follow a familiar transfer learning paradigm (Figure 1) where models are pre-trained using supervised or unsupervised learning. Thereafter, the frozen activations from a layer of that model, forming embeddings, $\mathbf{z}$, are mapped to a downstream task using a simple network $p(\mathbf{z})$.

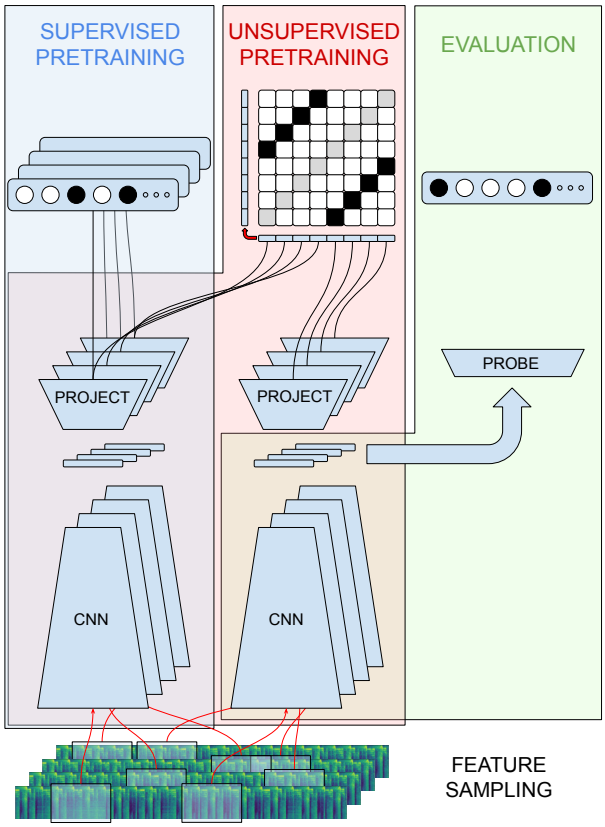


Figure 1. System diagram of both pre-training approaches employed in this paper, and evaluation.

2.1 Supervised and Unsupervised Training

In the supervised setting we learn a function $f(\mathbf{X}) \Rightarrow \hat{\mathbf{y}}$ mapping features $\mathbf{X}$ (log-mel spectrograms) to binary labels, $\mathbf{y}$, by applying Adam optimization [29] to the binary cross-entropy loss function,

$$L_s(\mathbf{y}, \hat{\mathbf{y}}) = \frac{-1}{NK} \sum_{i=0}^{N-1} \mathbf{y}_i \log(\hat{\mathbf{y}}_i) + (1 - \mathbf{y}_i) \log(1 - \hat{\mathbf{y}}_i),$$

where batch size $N = 512$, and K is the number of labels.

In the unsupervised setting, we employ the SimCLR objective [27, 28], which has been shown to provide promising results for both music and audio understanding [2, 10]. The SimCLR objective employs correlated (positive) pairs of samples by mapping each feature to an embedding space, $f(\mathbf{X}) \Rightarrow \mathbf{z} \in \mathbb{R}^m$, with embedding dimensionality $m = 1728$. A projector, then maps the embedding space to a loss space $h(\mathbf{z}) \Rightarrow \mathbf{v} \in \mathbb{R}^n$, with dimensionality $n = 1024$. Here, each element is then compared to all other elements in a batch via distance function, $d(\mathbf{v}_i, \mathbf{v}_j) = \mathbf{v}_i \cdot \mathbf{v}_j / \|\mathbf{v}_i\| \|\mathbf{v}_j\|$. The loss is then computed as the normalized temperature-scaled cross entropy,

$$L_u(\mathbf{v}_i, \mathbf{v}_j) = -\log \frac{\exp\left(d(\mathbf{v}_i, \mathbf{v}_j)/\tau\right)}{\sum_{k=0}^{2N-1} \mathbb{1}_{[k \neq i]} \exp\left(d(\mathbf{v}_i, \mathbf{v}_k)/\tau\right)},$$

which is summed across $2N$ examples in $N = 1920$ positive pairs, where $i = j$, for all values of both $i \in [0, N-1]$

and $j \in [0, N-1]$. Here, $\mathbb{1}_{[k \neq i]}$ is an indicator function evaluating to 1 for $k = i$, otherwise evaluating to 0. $\tau = 0.1$ denotes a temperature hyper-parameter.

SimCLR loss can be interpreted as a one hot classification problem—for each example, identifying its positive pair amongst all other (negative) examples in the batch. Because the batch-size determines the number of negative examples in the batch, and hence, the likelihood of non-trivial negatives, a large batch size of N pairs is crucial to this learning strategy, with larger batch sizes resulting in notable performance improvements [9]. However, batch sizes are limited to the memory of the available compute resources (e.g., of GPUs / TPUs). Hence, when considering compute costs, it is desirable to reduce batch size.

The primary objective of this work is to provide a comparative analysis of the utility of supervised and unsupervised learning strategies for a range of music understanding tasks under different data sources. As such we do not propose to investigate or innovate on the model architecture defining $f(\mathbf{X})$ itself, and employ the Short-Fast Normalizer-Free Net F0 (SF-NFNet-F0) of [2]. This architecture was chosen due to its demonstrated excellent performance in audio understanding, its design intent specifically for audio, and its use of efficient operations such as grouped convolutions. Furthermore, this architecture does not employ batch-normalization, improving training time and resource requirements, particularly when using SimCLR loss, by removing the need to globally synchronize data between GPU devices at each batch-normalization layer. Our intent was to reproduce this model as accurately as possible. However, due to the information available to us at the time of publication, there may be some discrepancies between our implementation and that in [2]. To ensure reproducibility, we release a Tensorflow implementation of the model used in this paper with SCOOCH configurations [1] for each of the models trained [2]. Our SF-NFNet-F0 implementation contains 62.4M parameters.

In both supervised and unsupervised settings we employ mixup [30] directly on the sampled log-mel spectrograms in real-time during training for data augmentation. We acknowledge that additively combined audio sources do not result in additively combined magnitude spectrograms due to both constructive and destructive interference. However, such an augmentation is an efficient operation to perform in real-time data sampling pipelines, mitigating data bandwidth bottlenecks that may be caused by more complex operations. We employ mixup by shuffling features within each batch and additively combining the shuffled features (and labels for supervised learning) to the original batch. Mixup gains are sampled from a beta distribution with parameters $\alpha = 5.0$ and $\beta = 2.0$, for each feature in a batch.

In all pre-training contexts, we use an Adam optimizer with a learning rate following a warm-up cosine decay schedule, first increasing to 0.0002 over 5k steps, then decreasing to 0.0 over 195k steps. While dense projectors, $h(\mathbf{z})$ are an inherent part of the unsupervised SimCLR learning approach, we find projectors to also be useful in the supervised setting, adding a non-linear transformation between the learned embeddings, $\mathbf{z}$, and the supervised labels, $\hat{\mathbf{y}} = h(f(\mathbf{X}))$. We notice such an approach has also been useful in computer vision [31]. Hence, in all contexts we employ a projector consisting of 3×4096 node hidden layers with ReLU activation. Supervised models were trained on 8 v100 GPUs taking approximately 30 hours, while unsupervised models were trained on 16 A100 GPUs taking approximately 80 hours.

[1] https://github.com/PandoraMedia/scooch

[2] https://github.com/PandoraMedia/music-audio-representations

2.2 Datasets

Pre-training datasets can have a significant impact on supervised and unsupervised model performance. In the supervised context, there has been evidence that models are less generalizable to unseen tasks [2], perhaps due to the information present in the supervised labels. In the unsupervised context, less investigation has been conducted into the effect of the content of pre-training datasets. In this context, the problem of sampling positive and non-trivial negative examples within a dataset is closely tied to the diversity of content in that dataset. Furthermore, when employing mixup as a data augmentation strategy, the content of the pre-training dataset also defines the additive noise that is applied to features during training.

We compare pre-training on two large datasets, namely **Musicset** and a version of **Audioset** [32]. The Audioset training set contains $\approx$ 1.7M distinct labelled content pieces, and Musicset $\approx$ 1.8M. Audioset consists of 10 second snippets of various audio sources: music, speech and environmental audio (4,791 hours, $\approx$ 602 GB of audio feature data). Musicset, in contrast, focuses solely on music; it consists only of labelled complete songs, each up to several minutes in length (117,497 hours, $\approx$ 14.7 TB of audio feature data). Musicset is, to our knowledge, the largest dataset of expert-annotated audio ever trained on. While it is not publicly available, we believe it valuable to report the results of models trained on this dataset to communicate the effectiveness of supervised learning at this scale.

For supervised learning, in addition to the features themselves, the labels differ. Each dataset's vocabulary is distinct but similar in size (527 labels for Audioset and 500 labels for Musicset), however, Figure 2 shows the label density and distribution of both datasets differs.

2.3 Feature Sampling

The features, $\mathbf{X}$, are log-magnitude log-mel spectrograms produced from waveforms sampled at 16 kHz. We use 96 HTK-log-mel spaced, power normalized, frequency bins with center frequences from 0 Hz to 8 kHz, analyzed with a window size of 25 ms, a Fourier transform size of 2048 bins and a hop size of 10 ms. We sample 3 s snippets of each content piece in real-time (during training) across the pre-training dataset, forming features, $\mathbf{X} \in \mathbb{R}^{96 \times 300}$. Sampling features in real-time from full-length tracks has the benefit of a high ratio of distinct features per static dataset size by reducing redundant (overlapping) data storage. For

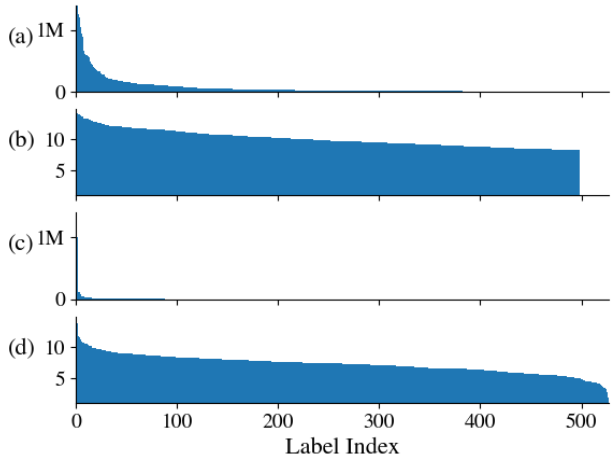


Figure 2. Label density distributions for Musicset and Audioset. (a), (b) plots the sorted Musicset label counts on a linear (in millions) and logarithmic scale, respectively. (c), (d) plots the same data for Audioset, respectively.

example, a 183 second song (6.8MB) results in 18k distinct features (each 112kB) with the above parameters.

To build a batch from a dataset, we first select a uniform random sample of tracks (with replacement); then, from each selected track, we choose a random 3-second spectrogram snippet, without padding. In the unsupervised context, we need to select positive and negative pairs for each selected snippet. Positive pairs are sampled from the same 10 second "track" for Audioset; for Musicset, positive examples are also sampled from the same track, although we require positive pairs to be centered on frames less than 5 seconds apart in the track's timeline. As negative examples, the SimCLR objective implicitly uses all other samples in a batch. We also tried forcing examples from the same track, but further away from the anchor than the positive, into every batch as hard negatives. Such an approach has shown promising results in music segmentation [33], however, we found that for music labelling at the averaged track-level this did not change our results significantly. As such, we omit this approach from our evaluation.

3. EVALUATION

To evaluate embedding performance, we take the global average pooled activations of the final convolutional layer in the SF-NFNet-F0 architecture for 3 second feature snippets sampled along the length of each audio timeline at a frequency of 0.5 Hz, then average these sampled embeddings along the length of the timeline (with the exception of NSynth datasets as described later in this section). We found similar results training probes on both the timeline-averaged embeddings, and individual embeddings sampled directly from track timelines, with slight improvements when using timeline-averaged embeddings in some cases.

Each of the embeddings derived from pre-trained models have different levels of predictive power and will underfit / overfit with different parameters for each downstream-dataset / pre-trained model combination. To demonstrate the potential of embeddings, and to investigate whether a given embedding can achieve SOTA performance on a given dataset, it is important to optimize the probe parameters for each dataset. However, in cases where such a transfer learning methodology is SOTA, it is important to constrain the probe to the limitations imposed in previous works so that it is the quality of embeddings that are evaluated, and not that of the probes. With these considerations, we adopt a hybrid strategy based on the prior SOTA for each dataset. We optimize the parameters of probes in all cases except where the prior SOTA adopts a similar transfer learning methodology, in which case, we constrain the probes to the parameter ranges investigated in those prior works. Regardless of prior works, we restrict all probes to simple multi-layer perceptron (MLP) classifiers which can be trained on a single 32 core CPU in less than 1 hour to maintain the aforementioned benefits of efficiency and scalability that such a transfer learning approach provides.

In all cases, we train probes using Adam optimization with a cosine learning rate schedule with 1,000 steps of warmup followed by a decay to zero over the remainder of the steps. We optimize the learning rate, number of training steps, and l2-regularization of each probe to achieve best performance / prevent overfitting.

In total we evaluate the performance over annotations of 7 distinct collections of audio, comprising 15 distinct datasets in total. An overview of datasets is shown in Table 1, and probe configurations in Table 2. We also publish more detailed SCOOCH configurations for probes, tag-level results, and model weights for the Musicset-ULarge model as a baseline for future research. [3] The remainder of this section covers specific notes on each dataset.

Dataset	Task	#Items	#Labels	Avg. secs	Hours
MSDS	tagging	242k	50	46	3.08k
MSD50	tagging	36k	50	46	464
MSD100	tagging	115k	100	47	1.50k
MSD500	tagging	156k	500	47	2.05k
AMM	mood	67k	188	45	840
MuMu	genre	147k	250	47	1.92k
MTT	tagging	26k	50	29	171
NSynth_P	pitch	306k	112	4	340
NSynth_I	instrument	306k	11	4	340
GTZAN	genre	930	10	30	7.8
Emo	emotion	744	N/A	45	9.3
GS_Key	key	2.1k	24	120	116
Jam-50	tagging	54k	50	244	3.69k
Jam-All	tagging	56k	183	244	3.76k
Jam-MT	mood/theme	18k	56	219	1.10k

Table 1. Overview of evaluation datasets. The sizes for MSD50 and MSD500 disagree with [22]. The published MSD50 splits include 35,745 IDs, and we exclude any tracks in MSD500 without any tags in the vocabulary.

Magnatagatune: Magnatagatune (MTT) [34] annotations include genre, instrumentation / vocal, mood, perceptual tempo, origin and sonority features. We adopt the common approach using the published splits and top 50 tags [4]. Some research removes tracks without tags in the most common 50 tags. To make results comparable to the

[3] https://github.com/PandoraMedia/music-audio-representations
[4] https://github.com/jongpillee/music_dataset_split

Dataset	Layers	Dropout	Batch (N)	Constraint
MSDS	2×1024	0.5	256	None
MSD50	1×512	0.5	256	None
MSD100	2×2048	0.5	256	None
MSD500	3×4096	0.5	256	None
AMM	1×1024	0.5	256	None
MuMu	linear	0.3	256	None
MTT	1×512	0.5	256	[24]
NSynth_P	linear	0.0	64	[2]
NSynth_I	linear	0.0	64	[2]
GTZAN	linear	0.0	2560	None
Emo	1×1024	0.5	512	None
GS_Key	1×512	0.8	512	None
Jam-50	1×512	0.75	256	None
Jam-All	2×1024	0.5	256	None
Jam-MT	1×512	0.75	256	None

Table 2. Probe parameters

most recent SOTA on this dataset [24], we keep items without tags in both training and evalution sets.

GTZAN: This dataset addresses the problem of genre classification in a single-label context [35]. We employ the fault-filtered version of this dataset [4].

NSynth: Here we evaluate detecting the pitch (NSynth_P) and instrument (NSynth_I) of individual musical notes. Following [2], in the case of NSynth_P we analyze the average of embeddings from four non-overlapping consecutive snippets of 1 second feature windows, and for NSynth_I we analyze a single embedding for each note produced using a 4 second feature window.

Million Song Dataset: The Million Song Dataset (MSD) contains labels pertaining to era, instrumentation, sonority, genre, mood, origin and activity. Because of the size of this dataset and variation in label quality throughout we adopt several different splits and vocabularies for the dataset. For comparability with the previous SOTA, we adopt the vocabulary of the 50 most common labels, and the splits employed in [15] [4], namely MSDS. We also take advantage of the several cleaned and artist-separated datasets available for this collection [22]—MSD50, MSD100 and MSD500, for which the splits and vocabularies are available publicly [5].

All Music Moods: The All Music Moods dataset (AMM) [36], focuses on mood prediction for the MSD audio data. Because the mood labels are heavily correlated with the artists in the dataset, we find it important to adopt the artist-seperated split [6]. With no previously published result on this split, we additionally provide results for embeddings produced by the MusiCNN model [37].

Jamendo: This dataset contains genre, instrument and mood / theme tags for audio from Jamendo [38]. We evaluate on all tags (Jam-All) and the 50 most common (Jam-50). Because very few of the mood / theme tags are in Jam-50, we also evaluate the mood / theme category (Jam-MT). We use the official splits [7] and full-length audio.

GiantSteps Key: This dataset (GS_Key) concerns major/minor key classification in electronic music—a 24-way classification problem. It combines two datasets: the first, 604 2-minute samples of electronic music tracks collected from beatport.com [39]; the second [8], consists of 1486 tracks from the same source. Consistent with previous work [23, 40], we use the former 604 samples for testing, and the latter for training / validation, selecting high-confidence annotations partitioned into train and validation sets according to [23]. For evaluation we compute a weighted accuracy score common in key classification [9].

EmoMusic: This dataset (Emo) concerns emotion recognition [41], and provides continuous annotations for valence and arousal. Following [23], we consider the static version of this dataset, where target values are averaged for each clip, and use the artist-based train / test partitioning provided in their work. This poses a regression problem—we use the coefficient of determination as the evaluation metric for both valence (Emo_V) and arousal (Emo_A).

MuMu: The Multimodal Music dataset (MuMu) [42], focuses on multilabel genre predictions for the MSD audio data, with genre annotations from the Amazon Reviews dataset. We evaluate using the official splits [10].

4. RESULTS

In order to derive conclusions on the potential performance of supervised models trained using binary labels on music data, we train a supervised model on the complete Musicset dataset, namely *Musicset-Sup*. To compare these results to supervised learning on a large-scale public dataset covering music, environmental, and speech audio, we provide results for a model trained on ≈1.7M items from Audioset, namely *Audioset-Sup*. To compare supervised and unsupervised learning at a similar scale, we provide results for a model trained using the unsupervised methodology discussed in Section 2 on the same Musicset dataset audio data and the Audioset audio data, namely *Musicset-ULarge* and *Audioset-ULarge* using a batch size of 1920 pairs.

In addition, we train two further unsupervised models using a batch-size of 256. One trained on Audioset (*Audioset-USmall*), and one on ≈1.8M randomly sampled 10 second snippets from Musicset, one per content piece (*Musicset-USmall*). The reasoning for this is two-fold—firstly by sampling short snippets from Musicset we mitigate the confounding variable of dataset size when comparing the two, secondly this allows us to investigate the effect of batch size in the unsupervised learning paradigm.

For convenience, we also include results for the recent evaluations in [2, 24], and those for any other previous SOTA, excluding these two evaluations. The results for each of these models is shown in Table 3 for all datasets excluding annotations of MSD, which are shown in Table 4. There, we note that for all multilabel music tagging tasks, the Musicset-Sup model achieves SOTA performance by significant margins. This is encouraging given that the Musicset training dataset was created naively, and the supervised information therein opens the door to improvements such as label-balanced sampling which has been shown to

[5] https://github.com/minzwon/tag-based-music-retrieval
[6] https://github.com/fdlm/listening-moods
[7] https://github.com/MTG/mtg-jamendo-dataset
[8] https://github.com/GiantSteps/giantsteps-mtg-key-dataset
[9] https://www.music-ir.org/mirex/wiki/2021:Audio_Key_Detection
[10] https://zenodo.org/record/1236906#.YoPIAhNBx0s

Model	MTT		GTZAN	$NSynth_P$	$NSynth_I$	Emo_V	Emo_A	GS_{Key}	Jam-50		Jam-All	
	mAP	ROC	Acc	Acc	Acc	r^2	r^2	W. Acc	mAP	ROC	mAP	ROC
Musicset-Sup	**0.413**	**0.917**	**0.835**	0.793	0.731	0.566	**0.726**	0.286	**0.321**	**0.843**	**0.162**	**0.839**
Audioset-Sup	0.386	0.904	0.748	0.819	0.676	0.341	0.545	0.210	0.284	0.822	0.135	0.813
Musicset-ULarge	**0.404**	**0.914**	0.735	**0.892**	0.740	0.577	0.700	0.667	**0.317**	**0.839**	**0.159**	**0.833**
Audioset-ULarge	0.391	0.906	0.672	0.805	0.721	0.438	0.624	0.287	0.285	0.826	0.131	0.816
Musicset-USmall	0.389	0.905	0.686	0.824	0.714	0.389	0.668	0.508	0.292	0.828	0.138	0.817
Audioset-USmall	0.375	0.897	0.648	0.777	0.698	0.386	0.609	0.197	0.268	0.817	0.127	0.809
Jukebox [23]	**0.414**	**0.915**	0.797	-	-	**0.617**	**0.721**	0.667	-	-	-	-
Prev. SF-NFNet-F0 [2]	0.395	-	-	0.880	**0.782**	-	-	-	-	-	-	-
SOTA Excl. [2, 23]	0.384 [37]	**0.92** [12]	0.821 [11]	- -	0.741 [43]	0.556 [44]	0.704 [45]	**0.796*** [46]	0.298 [47]	0.832 [47]	- -	- -

Table 3. Results for models on all datasets, excluding MSD annotations and Jam-MT. Models within 0.01 of SOTA are bold. *The SOTA for GS_{Key} has no publicly available implementation details and is specialized for this dataset, e.g., [48].

Model	MSDS		MSD50		MSD100		MSD500		MuMu		AMM		Jam-MT	
	mAP	ROC	mAP	ROC	mAP	ROC	mAP	ROC	mAP	ROC	mAP	ROC	mAP	ROC
Musicset-Sup	**0.363**	**0.903**	**0.459**	**0.913**	**0.346**	**0.906**	**0.169**	**0.898**	**0.257**	**0.908**	**0.180**	**0.791**	**0.161**	**0.786**
Audioset-Sup	0.308	0.880	0.375	0.883	0.278	0.877	0.128	0.874	0.191	0.867	0.156	0.760	0.137	0.749
Musicset-ULarge	0.351	**0.900**	0.438	**0.908**	0.321	**0.897**	0.152	**0.891**	0.235	0.893	**0.174**	**0.784**	**0.158**	**0.781**
Audioset-ULarge	0.311	0.885	0.377	0.886	0.276	0.878	0.121	0.873	0.162	0.855	0.156	0.763	0.142	0.765
Musicset-USmall	0.319	0.888	0.384	0.892	0.283	0.881	0.129	0.878	0.190	0.871	0.155	0.762	0.138	0.757
Audioset-USmall	0.286	0.876	0.353	0.878	0.251	0.870	0.110	0.868	0.152	0.850	0.151	0.753	0.136	0.753
SOTA	0.348 [15]	**0.897** [15]	0.386 [14]	**0.921** [14]	0.185 [22]	- -	- -	- -	- -	0.888* [42]	0.163 [37]	0.773 [37]	**0.161**† [49]	**0.781**† [49]

Table 4. Results for all models on MSD annotations and Jam-MTT. Models within 0.01 of SOTA are bold. *The previous SOTA for MuMu evaluated artist-level predictions rather than track-level. †The SOTA for Jam-MT is trained on an expanded within-taxonomy set ($\approx$ 16k tracks), here we restrict probe training to Jamendo tracks ($\approx$ 10k).

realize further performance gains [6]. We leave this investigation for future work.

It is also encouraging to see that unsupervised learning on music audio (Musicset-ULarge) achieves SOTA performance compared to previous unsupervised models, in addition to some supervised models (specifically, for models trained on Jam-50 and all MSD datasets).

Comparing to recent transfer learning approaches [2, 24], we see that in all cases either Musicset-Sup or Musicset-ULarge outperform or are on par with these, with the exception of $NSynth_I$. This is a promising result espousing the value of music only audio datasets considering that previously reported results for unsupervised learning on Audioset [2] used more than double the batch size of those in this paper. Wrt. the Jukebox model, models trained for this paper use approximately 1% of the parameters and take less than 1% of the GPU flop hours to train.

We see that the supervised models evaluated demonstrate shortcomings in performance on pitch ($NSynth_P$) and key (GS_{Key}) tasks. This corroborates the findings in [24] that models trained on tags do not perform well in this task, and the findings in [2] that supervised pre-trained models do not generalize as well to novel tasks. We note that there are no pitch or key labels in the Musicset dataset, and in fact, many of the labels employed (e.g., genre, mood, etc.) require the model to be somewhat agnostic to such information. It is yet to be seen whether including such information in the labels of a pre-training dataset would improve the generalizability of such models, though this is outside of the scope of the current work. Interestingly, we see that unsupervised models trained specifically on music data show significant improvements in key estimation.

When comparing the domain of the pre-training dataset features in the models of Musicset-USmall and Audioset-USmall we see that in all cases, in-domain data (music) results in a better performing model. We conjecture that this may be due to (a) increasing the chances of non-trivial negative examples within each batch relative to each positive pair, due to the homogeneity of the dataset, and (b) music is more spectrally dense and correlated than other common noise sources employed in mixup augmentation, such as speech or various environmental noises.

5. CONCLUSIONS

In this work we investigated both unsupervised and supervised learning strategies, to produce compact audio representations that may be deployed across industry-scale audio catalogs for a range of downstream use cases. We observed that supervised training on large scale expert-annotated music data achieves SOTA results in music tagging. We see that unsupervised learning on datasets of the same scale also achieves excellent performance, across a wider range of tasks than supervised learning, particularly on those that involve information not represented in the supervised dataset. Finally, we see that for unsupervised learning, restricting the domain of the pre-training dataset to music results in improvements in model performance.

6. ACKNOWLEDGEMENTS

The authors would like to thank the Pandora music analyst team for their years of dedicated and careful work that enables us to investigate supervised machine learning systems at modern levels of quality and scale.

7. REFERENCES

[1] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen, "Slow-fast auditory streams for audio recognition," in *ICASSP*, 2021, pp. 855–859.

[2] L. Wang, P. Luc, Y. Wu, A. Recasens, L. Smaira, A. Brock, A. Jaegle, J.-B. Alayrac, S. Dieleman, J. Carreira *et al.*, "Towards learning universal audio representations," in *ICASSP*, 2022, pp. 4593–4597.

[3] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," *ISMIR*, 2018.

[4] L. Wang, P. Luc, A. Recasens, J.-B. Alayrac, and A. Oord, "Multimodal self-supervised learning of general audio representations," *arXiv preprint arXiv:2104.12807*, 2021.

[5] A. Jansen, M. Plakal, R. Pandya, D. P. Ellis, S. Hershey, J. Liu, R. C. Moore, and R. A. Saurous, "Unsupervised learning of semantic audio representations," in *ICASSP*, 2018, pp. 126–130.

[6] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "PANNs: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[7] M. Tagliasacchi, B. Gfeller, F. de Chaumont Quitry, and D. Roblek, "Pre-training audio representations with self-supervision," *IEEE Signal Processing Letters*, vol. 27, pp. 600–604, 2020.

[8] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *ICASSP*, 2021, pp. 3875–3879.

[9] L. Wang and A. Oord, "Multi-format contrastive learning of audio representations," *NeurIPS Workshops*, 2020.

[10] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," *ISMIR*, 2021.

[11] J. Lee, J. Park, K. L. Kim, and J. Nam, "SampleCNN: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, no. 1, p. 150, 2018.

[12] Q. Huang, A. Jansen, L. Zhang, D. P. Ellis, R. A. Saurous, and J. Anderson, "Large-scale weakly-supervised content embeddings for music recommendation and tagging," in *ICASSP*, 2020, pp. 8364–8368.

[13] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio spectrogram transformer," *Interspeech*, 2021.

[14] W.-T. Lu, J.-C. Wang, M. Won, K. Choi, and X. Song, "SpecTNT: a time-frequency transformer for music audio," *ISMIR*, 2021.

[15] M. Won, K. Choi, and X. Serra, "Semi-supervised music tagging transformer," *ISMIR*, 2021.

[16] J. Yan, Y. Song, W. Guo, L.-R. Dai, I. McLoughlin, and L. Chen, "A region based attention method for weakly supervised sound event detection and classification," in *ICASSP*, 2019, pp. 755–759.

[17] A. T. Liu, S.-w. Yang, P.-H. Chi, P.-c. Hsu, and H.-y. Lee, "Mockingjay: Unsupervised speech representation learning with deep bidirectional transformer encoders," in *ICASSP*, 2020, pp. 6419–6423.

[18] Y. Gong, Y.-A. Chung, and J. Glass, "PSLA: Improving audio tagging with pretraining, sampling, labeling, and aggregation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3292–3306, 2021.

[19] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 449–12 460, 2020.

[20] K. Koutini, J. Schlüter, H. Eghbal-zadeh, and G. Widmer, "Efficient training of audio transformers with patchout," *arXiv preprint arXiv:2110.05069*, 2021.

[21] I. Manco, E. Benetos, E. Quinton, and G. Fazekas, "Learning music audio representations via weak language supervision," in *ICASSP*, 2022, pp. 456–460.

[22] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, "Multimodal metric learning for tag-based music retrieval," in *ICASSP*, 2021, pp. 591–595.

[23] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[24] R. Castellon, C. Donahue, and P. Liang, "Codified audio language modeling learns useful representations for music information retrieval," *ISMIR*, 2021.

[25] J. Kim, J. Urbano, C. Liem, and A. Hanjalic, "One deep music representation to rule them all? a comparative analysis of different representation learning strategies," *Neural Computing and Applications*, vol. 32, no. 4, pp. 1067–1093, 2020.

[26] HEAR Benchmark, https://hearbenchmark.com/, [Accessed: 2022-08-01].

[27] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, "Big self-supervised models are strong

semi-supervised learners," *Advances in neural information processing systems*, vol. 33, pp. 22 243–22 255, 2020.

[28] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *ICML*, 2020, pp. 1597–1607.

[29] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[30] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *ICLR*, 2018.

[31] M. Bulent Sariyildiz, Y. Kalantidis, K. Alahari, and D. Larlus, "Improving the generalization of supervised models," *arXiv preprint arXiv:2206.15369*, 2022.

[32] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio Set: An ontology and human-labeled dataset for audio events," in *ICASSP*, 2017, pp. 776–780.

[33] M. C. McCallum, "Unsupervised learning of deep features for music segmentation," in *ICASSP*, 2019, pp. 346–350.

[34] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging." in *ISMIR*, 2009.

[35] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 2002.

[36] F. Korzeniowski, O. Nieto, M. McCallum, M. Won, S. Oramas, and E. Schmidt, "Mood classification using listening data," in *ISMIR*, 2021.

[37] J. Pons and X. Serra, "musicnn: Pre-trained convolutional neural networks for music audio tagging," *arXiv preprint arXiv:1909.06654*, 2019.

[38] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, "The MTG-Jamendo dataset for automatic music tagging," in *ICML Workshops*, 2019. [Online]. Available: http://hdl.handle.net/10230/42015

[39] P. Knees, Faraldo, P. Herrera, R. Vogl, S. Böck, F. Hörschläger, and M. Le Goff, "Two data sets for tempo estimation and key detection in electronic dance music annotated from user corrections," in *ISMIR*, 2015.

[40] F. Korzeniowski and G. Widmer, "End-to-end musical key estimation using a convolutional neural network," in *EUSIPCO*, 2017, pp. 966–970.

[41] M. Soleymani, M. N. Caro, E. M. Schmidt, C.-Y. Sha, and Y.-H. Yang, "1000 songs for emotional analysis of music," in *CrowdMM*, 2013, pp. 1–6.

[42] S. Oramas, F. Barbieri, O. Nieto Caballero, and X. Serra, "Multimodal deep learning for music genre classification," *Transactions of the International Society for Music Information Retrieval*, vol. 1, no. 1, pp. 4–21.

[43] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "BYOL for audio: Self-supervised learning for general-purpose audio representation," in *IJCNN*, 2021, pp. 1–8.

[44] E. Koh and S. Dubnov, "Comparison and analysis of deep audio embeddings for music emotion recognition," *arXiv preprint arXiv:2104.06517*, 2021.

[45] F. Weninger, F. Eyben, and B. Schuller, "On-line continuous-time music mood regression with deep recurrent neural networks," in *ICASSP*, 2014, pp. 5412–5416.

[46] Pioneer DJ, https://rekordbox.com, [Accessed: 2016-09-12].

[47] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of CNN-based automatic music tagging models," *arXiv preprint arXiv:2006.00751*, 2020.

[48] G. Bernardes, M. E. Davies, and C. Guedes, "Automatic musical key estimation with adaptive mode bias," in *ICASSP*, 2017, pp. 316–320.

[49] D. Knox, T. Greer, B. Ma, E. Kuo, K. Somandepalli, and S. Narayanan, "Mediaeval 2020 emotion and theme recognition in music task: Loss function approaches for multi-label music tagging." in *MediaEval*, 2020.

GENERATING COHERENT DRUM ACCOMPANIMENT WITH FILLS AND IMPROVISATIONS

Rishabh Dahale [1] **Vaibhav Talwadker**[1] **Preeti Rao**[1] **Prateek Verma** [2]
[1] Department of Electrical Engineering, Indian Institute of Technology Bombay, India
[2] Stanford University
dahalerishabh1@iitb.ac.in, talwadkerv@gmail.com, prao@ee.iitb.ac.in, prateekv@stanford.edu

ABSTRACT

Creating a complex work of art like music necessitates profound creativity. With recent advancements in deep learning and powerful models such as transformers, there has been huge progress in automatic music generation. In an accompaniment generation context, creating a coherent drum pattern with apposite fills and improvisations at proper locations in a song is a challenging task even for an experienced drummer. Drum beats tend to follow a repetitive pattern through stanzas with fills/improvisation at section boundaries. In this work, we tackle the task of drum pattern generation conditioned on the accompanying music played by four melodic instruments – Piano, Guitar, Bass, and Strings. We use the transformer sequence to sequence model to generate a basic drum pattern conditioned on the melodic accompaniment to find that improvisation is largely absent, attributed possibly to its expectedly relatively low representation in the training data. We propose a novelty function to capture the extent of improvisation in a bar relative to its neighbors. We train a model to predict improvisation locations from the melodic accompaniment tracks. Finally, we use a novel BERT-inspired in-filling architecture, to learn the structure of both the drums and melody to in-fill elements of improvised music.

1. INTRODUCTION

Songs in popular music genres like rock are typically split into different sections such as the verse, bridge, and chorus. While the primary task of a drummer is to play in time, it is also important for the drummer to be consistent with the song structure. Traditionally fills, or short groups of notes, are played as the song transitions from one section to another (say, verse to chorus). Thus, it can serve as an indicator to the audience as well as the band, of an upcoming transition in the song. The duration of fills generally tends to be only a few beats long, no more than the length of a bar. Even though they are rare events in a drum track, fills are an important part of the overall aesthetics. Beyond signaling transitions, drum fills can also be played in sections where the accompanying instrumentation is sparse.

Motivated by the above, we improve the quality of the generated drum tracks from seq-to-seq models by incorporating fills/improvisations towards our overall goal of accompaniment generation. This is achieved via the following three distinct stages:

1. **Basic Drum Pattern Generation:** Using a seq-to-seq model to generate a drumbeat as the accompaniment to given melodic instrument tracks of guitar, bass, strings, and piano (i.e. the Melodic Accompaniment - MA).

2. **Improvisation Location Detection:** Detecting explicitly the position of improvisation from the MA using a self-similarity function and mini BERT model.

3. **Generating Improvised Bars:** Generating the fills in the previously detected bars.

Our main contributions are: (i) We show that traditional attention-based transformer architectures fail to capture the "improvisation" due to implicit data imbalance. (ii) We also show that the sampling-based approaches fail to produce a variation of pattern at the right location. To mitigate this, we learn to predict where to improvise directly from the melody tracks using powerful self-attention-based architectures. (iii) We propose a novel in-filling approach, inspired by BERT that can look at the context of drums and the context of melody and use it to generate the improvised bars. (iv) We demonstrate an MLP-based synthesis module for drum improvisation generation from a latent code. For simplicity we have ignored the dynamic (velocities) in the generated drum patterns of this work.

2. RELATED WORKS

Since the introduction of the Transformer architecture [1], there has been a growing interest in this model for sequential task modeling like in NLP and music generation. The architecture uses an attention mechanism to learn long-term patterns and can easily surpass dilated convolutional-based methods such as WaveNet [2]. They achieve state-of-the-art performance in a variety of natural language problems [3], music/audio understanding [4] and generation tasks [5, 6]. However, for generative models, the decoding strategy still remains an open question. Even though high-quality models can be obtained by the use of likelihood as the training objective, likelihood maximiza-

© Rishabh Dahale, Vaibhav Talwadker, Preeti Rao and Prateek Verma. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Rishabh Dahale, Vaibhav Talwadker, Preeti Rao and Prateek Verma, "Generating Coherent Drum Accompaniment with Fills and Improvisations", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

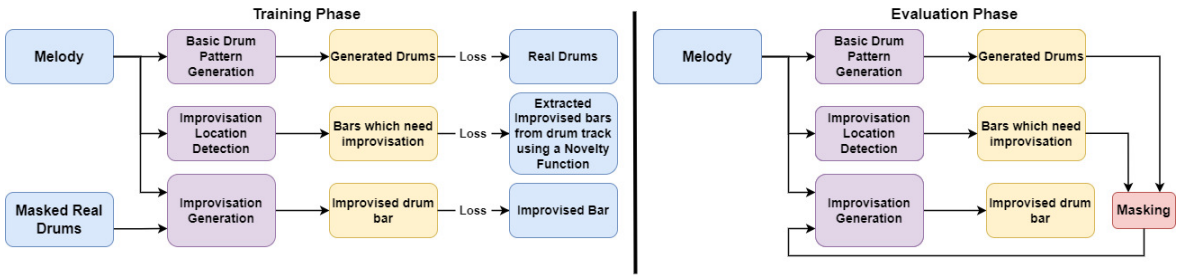


Figure 1. System Overview - Pipeline used for training (left half) of each individual module; combined pipeline (right half) for the evaluation phase.

tion methods like beam search lead to degeneration [7]. To solve this issue, many researchers use sampling-based methods like temperature sampling, top-k sampling, and nucleus sampling [7–9]. Despite all of the success of sequential modeling recently, there still exist many issues that are relevant to our current work such as understanding rare words or sparsely occurring events of interest [10]. Another major problem is the presence of biases in the generated output, as they mimic the distribution present in the training datasets [11]. These issues are ubiquitous across datasets and modalities and are implicit in our task due to the low representation of improvised bars. We here show how these constraints and biases affect the quality of drum generation, and the steps we take to mitigate them.

Conditioned drum beat generation is an important subtask of music generation. Wei et al. [12] observed that the polyphonic melody self-similarity matrix (SSM) is structurally similar to the drum SSM. They used this to predict the drum SSM from the melody SSM and used this intermediate representation to generate the drumbeat. While they used the audio form of the melody as the input, the symbolic domain also offers opportunities for similar research. An example is the work of [13] using symbolic representation of the drum track to predict locations and generate improvisations for the drum track. In this work, we address drum generation conditioned on a melodic accompaniment track, bringing considerably more complexity to the problem.

In the symbolic domain, systems use a discretized representation such as MIDI tag and pianoroll representations that capture the essential information at the semantic level. Pianoroll is a score-like matrix representing a piece of music. Note pitch and time are represented by the vertical and horizontal axes, respectively. The velocities of the notes are represented by the values. The time is generally quantized on a sub-beat level and each instrument track is represented by a separate pianoroll matrix. Dong et al. [14] used this representation along with CNN-based GAN model for music generation. Another popular symbolic domain representation is the MIDI tag representation which we refer to as the "serialized grid representation". In this method, the input is represented by a sequence of MIDI-like tags. Huang et al. [6] used this MIDI tag representation with modified transformer architecture to generate long-duration piano music. Several modifications have also been proposed to this representation. Huang et al. [15] proposed a revamped MIDI (REMI) which introduced `DURATION`, `BAR` and `POSITION` tags to improve the quaity of generated music. Ren et al. [16] further modified this representation to include multi-track representation by introducing `TRACK` tag and used the Transformer-XL model to generate multitrack songs. Nuttall et al. [17] modified the MIDI tags of nine percussion instruments to represent notes being played by a triplet of pitch, velocity, and start time, and used it with the Transformer-XL model to sequentially generate the drum pattern. Thorn et al. [18] demonstrated three experiments with the Transformer-XL model with varying input and output representation and control. In two of the experiments, they tried to control the drum machine while in the third experiment they tried to generate the drum pattern directly.

While the Transformer-XL model facilitates the generation of longer duration (musical) sequences, they still suffer from the same issues of biases in dealing with the implicit data-imbalance that exists in the training dataset [19, 20]. As the specific focus of this work is to find ways to capture the rare events in the generated output, i.e., fills and improvisations, we consider only the necessary context for an improvised bar in the form of 11 bar segments of the songs.

3. DATASET

In this work, we use the Lakh Pianoroll Dataset (LPD-5 cleansed) [14], derived from the Lakh Midi Dataset [21] which is a collection of 21,425 multitrack pianorolls files consisting of following tracks: Piano, Guitar, Bass, Strings and Percussion. All the songs in this dataset are of 4/4-time signature i.e., all the songs contain 4 beats in a bar and the dimensionality of each bar in this dataset is 128 (pitch) x 96 (time steps) i.e. each beat is divided into 24 parts. Our input, which we call melodic accompaniment (MA), consists of notes played by the 4 melodic instruments and output is a percussion instrument pattern conditioned on the input which we call the percussion accompaniment (PA).

Evaluation of generative music systems faces harder challenges than that of image generation systems [22]. We expect our system to replicate the rhythmic consistency and diversity of the dataset. Any drum beat generation system must have correct onset locations in a beat. If the onsets are not properly matched, it appears as if the drums are lagging/leading the melody. We show that our model is able learn this by capturing the onset location distribution.

To compare generated samples with original drums in the dataset, we use the following metric: Instrument Count - Number of distinct percussion instruments used in a bar. To evaluate our outputs in terms of rhythmic consistency, we use the following objective metric: Percussion pattern consistency in consecutive bars.

4. METHOD

The overview of the proposed system is shown in Figure 1. Details of each of the models are provided in subsequent sections. Our models are trained for 300 epochs with Adam optimization [23] starting with a learning rate of 1e-4 and decaying it till 1e-6. All the setup was carried out using the Tensorflow [24] framework. The following two modules are being used in all the models:

1. **Embedding Module:** As the pianoroll matrices are highly sparse, we pass them through 2 layers with 1024 and 128 ReLU [25] activated dense layers to capture the inter instrument and inter pitch dependencies. For the decoder branch of Basic Drum Pattern Generation model, we use 128 dimensional token embedding layer.

2. **Position Encoder:** We concatenate the sinusoidal positional representations [1] with the 128 dimensional vectors and use a dense layer to project them back in 128 dimension space.

4.1 Data Processing & Representation

To compress the input and output representation in our work, we perform the following preprocessing steps: (i) trim the track for start and end silence bars; (ii) resample the beats to 8 parts per beat, making each bar 32 timesteps long; (iii) keep only active MIDI pitches in all melodic instruments, i.e. MIDI 21 to MIDI 83 (notes A0 to B5); (iv) combine similar instruments in the MIDI representation of the percussion track. For example, under the snare drum, MIDI 38, which corresponds to acoustic snare, and MIDI 40, which corresponds to electric snare, are clubbed together; (v) choose only 16 percussion instruments as they capture 85.3% of all the percussion instrument strokes: snare drum, open hi-hat, close hi-hat, kick drum, ride cymbal, crash cymbal, low-floor tom, high-floor tom, high tom, hi-mid tom, low tom, cowbell, pedal hi-hat, tambourine, cabasa, and maracas; (vi) to decrease the amount of training parameters, we binarize the percussion track for seq-2-seq models and exclude velocities; (vii) split the song in non-overlapping contiguous 11 bar samples. The finer grids are superior for representing drum audio and fills [26]. We however opt for a quantized representation, capturing most of the significant musical events, allowing us to model longer duration dependencies by the compressed representation. It will be interesting to compare various representations as a future work.

We use a modified version of pianoroll representation for MA representation. We concatenate the pianorolls (Figure 2a) of different instruments instead of adding them to different channels [14]. As our input is quantized to 8 parts per beat, we represent each $\frac{1}{8}^{th}$ part of the beat by a 256-dimensional vector split amongst the 4 melodic instruments. The first dimension of this 64-dimension vector is the silence state. This is a binary state representing if the instrument under consideration is silent. The rest of the 63 dimensions contain the velocities of the MIDI notes 21-83 being played.

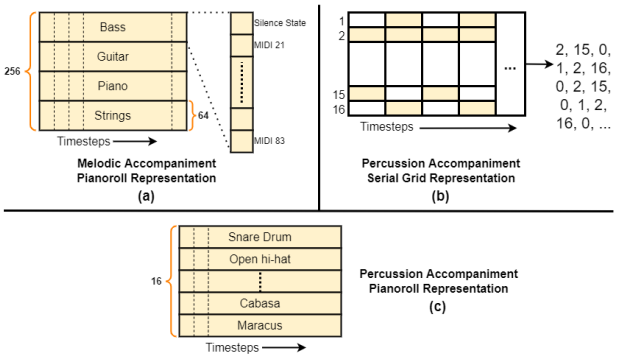


Figure 2. Data representation methods used in this work

For the PA, we adopt a mixed representation. For our Basic Drum Pattern Generation model, which is a transformer seq-2-seq model, we take advantage of the language modeling tasks and use a serialized grid representation (Figure 2b). In this representation, only the active percussion instruments which are being played are unfolded into a sequence of tokens. We add a silence state token and shift by one token making a total of 18 tokens for the percussion track. For the final model, the improvisation generation model, we give the MA and masked basic PA pattern as the inputs. We use the pianoroll representation for the PA representation for this model (Figure 2c) as only a fixed number of timesteps needs to be masked in this representation.

4.2 Train-Test Splits and Data Augmentation

We split the 21,425 songs in LPD-5 into 16,832 songs for training and 4,593 songs for the validation set. As the number of songs is limited, we use the validation set as the test set. We apply the following data augmentation strategies (inspired by sensor dropout methods in robotics [27]) to all of our models' inputs to increase the robustness in the training process:

1. **Random instrument masking:** We randomly mask one of the instruments in MA for 40% of the samples in every epoch. This 40% is equally split between the four melodic instruments. Musically this implies that one of the instruments has stopped playing. This encourages the model to consider all the instruments in the MA while making a prediction.

2. **Random timestep masking:** We randomly mask 20% of the timesteps in every sample. Musically this leads to a small disruption in the rhythm of the song which helps in better generalization of the model.

For the improvised bar generation model, which takes in the MA and masked PA as inputs, (section 4.5), we use the following additional augmentation methods:

1. **Input masking:** We randomly drop one of the inputs (MA or PA) to the model in 20% of the input samples. This

ensures that the model is not dependent on only one input for improvisation generation.

2. **Drum noise:** In the evaluation phase, the drum inputs are taken from the output of the basic drum pattern generation model. As this process could be prone to errors, we simulate this by adding the random noise to the drum samples while training. The random noise can either add new drum strokes or remove some old strokes. Our analysis showed that the PA has a very low density in the pianoroll format (average $\approx$ 5%). Hence we perturb the PA density by a maximum of 1%

4.3 Basic Drum Pattern Generation

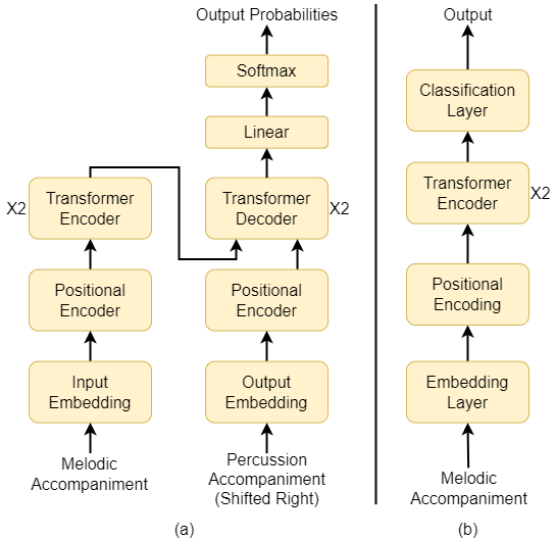


Figure 3. (a) Sequence to sequence model, used for basic drum pattern generation; (b) Improvisation Location Detection Model

We use the Transformer encoder-decoder model [1] to generate a basic drum pattern (Figure 3a). This is done by giving the MA as the input to the encoder and the shifted PA tokens to the decoder branch. Both the inputs are first passed through the embedding module followed by the position encoder. The embedded inputs are passed through 2 layers of encoder/decoder module with 128 dimensional latent space and 8 attention heads. At the output, we have 18 neurons corresponding to the 16 drum instruments, silence token, and shift token.

4.3.1 Sub-module Evaluation

We evaluate the above model with the negative log-likelihood (NLL) values over the train and the validation set. As the model outputs a distribution over the 18 output tokens, there are multiple ways to decode it. We test the greedy method of decoding, where the token with the maximum probability is selected at every step and simple sampling method. The model is trained using categorical cross-entropy loss, achieved a NLL of 0.108 and 0.112 on the train and validation split, respectively.

4.4 Improvisation Location Detection

4.4.1 Novelty Function

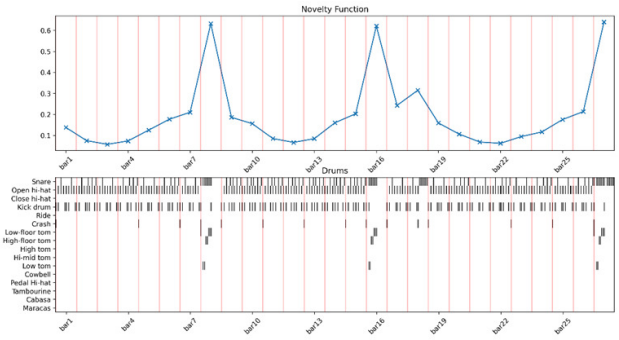


Figure 4. Novelty Function plot for a given drum track

In order to extract locations in the MA that warrant a fill, we propose the following method:

1. We use a 11-bar drum sample to calculate the Novelty value of center bar. The 5 bars on it's left and right are the context bar. The novelty value of a bar is calculated as the average weighted dissimilarity over all the context bars. We use the following equation to calculate the dissimilarity between 2 bars:

$$\frac{||bar_i - bar_j||_1 \times k}{||bar_i||_1 + ||bar_j||_1} \quad (1)$$

We utilize the hanning window to represent the weighting parameter k.

2. This calculation is done for all the bars across a track, except the first and the last 5 bars due to the lack of context bars. From Figure 4 it can be seen that the novelty function peaks at the bar with a drum fill. These bars are then extracted using a peak picking mechanism.

3. To generate the dataset for our task, we pick the bars with a local maxima as the positive samples. To filter out minor deviations, e.g., bar 18 in Figure 4, we put a threshold of 0.1 on the peaks height difference from its neighbours. Maximum of 10% of total bars in a song with these characteristics are selected as the positive samples. Same number of bars from the rest of the non peak regions are selected as negative samples.

4.4.2 Model Architecture

We use a 2 layer BERT [28] (Figure 3b) to detect the location of improvisations. The input to this model is an 11-bar MA and the prediction is done for the middle bar. The MA is passed through an embedding layer followed by the positional encoder. The embedded inputs are then passed through 2 layers of transformer encoder with 64 dimension latent space and 12 attention heads, followed by 1024 neurons. These are finally passed to a 2 dimensional softmax activated dense layer which acts as a classification module. The above model is trained using Huber loss [29], as it is robust to outliers and less sensitive to noise.

4.4.3 Sub-module Evaluation

We monitored the accuracy, precision, and recall of the model in terms of detecting the improvised bars where the

target is the original drum track. The final results can be found in Table 1. As the dataset is split equally between positive and negative samples, we have balanced precision and recall values.

	Precision	Recall	Accuracy	F1 Score
Train	92.3	92.3	92.3	92.3
Val	79.1	79.4	79.3	79.2

Table 1. Performance of the Improvisation Location Detection model. All the values are in %.

4.5 Improvisation Generation

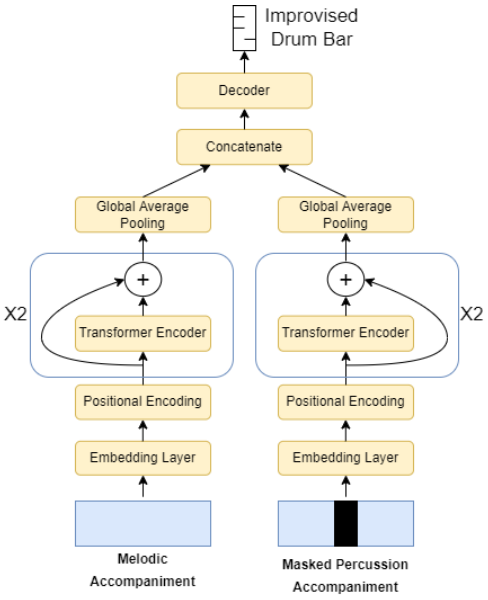


Figure 5. Improvised Bar Generation Model

The final step in our system is the generation of improvised bars. To achieve this we use the architecture shown in Figure 5. We provide 11 bar MA and PA as the input to the model. During the training phase, the PA is the original percussion track, whereas, during the evaluation/generation phase, the percussion track generated by our first stage model (section 4.3) is used. The 6th bar in the percussion sample (middle bar) is the target bar and is masked while giving as the input.

We generate a summary vector of both the MA and the PA inputs. Both are first passed through an embedding layer followed by a position encoder module. This is then passed through 2 layers of transformer encoders with 128 dimensional latent space and 8 attention heads. Even though larger/bigger models could potentially lead to better results, we have used the resources at our disposal. We note that any further improved performance will only apply to in-fill detection and synthesis.

We add skip connections to ease the flow of gradients [30]. To generate the summary vector for each input, we use a global averaging technique. These two vectors are concatenated and passed through a decoder structure which looks at the concatenated vector to generate the improvised drum bar. We test the following decoder architectures with our model:

1. **MLP:** A 3-layer dense network with 2048-2048-512 neurons is used. The final layer is sigmoid activated. The outputs are reshaped to 32 (timesteps) $\times$ 16 (percussion instruments) to get the improvised bar.

2. **MLP mixer:** MLP mixers [31] are simple alternatives to convolution and self-attention. They are based on multi-layered perceptrons applied across either temporal dimension or feature dimension.

3. **Conv1d:** A simple conv1d architecture with blocks of 2 layers of conv1d followed by upsampling.

We did not opt for an auto-regressive architecture, as this work does not assume causality, and we incorporate the right context as well as melody for the fill synthesis. There have been other works for improvisation synthesis, e.g. [32], also using the left and right context, even if only using drums

4.5.1 Sub-module Evaluation

We treat the prediction of the improvised bars as a regression problem. We similarly train it with Huber loss as it is less sensitive to outliers. We do not use cross-entropy loss for generation firstly purely as a design choice, and intuitively each of the time step token in the prediction within a bar lack probabilistic interpretation. We monitor and report the accuracy, precision, and recall of the models. As the distribution of 0s and 1s is not uniform in the predicted sample, F1 score provides a better insight in the performance of the models. From Table 2 it can be seen that a simple 3 layered MLP decoder is able to perform better than the complex MLP mixer and Conv1D architecture.

		Precision	Recall	Accuracy	F1 Score
MLP	Train	98.8	93.2	86.3	95.9
	Val	82.9	**70.3**	79.0	**76.0**
MLP Mixer	Train	97.5	45.0	88.7	61.6
	Val	**83.5**	42.1	**86.0**	56.0
Conv1D	Train	55.2	70.6	86.2	61.7
	Val	53.1	70.3	86.1	60.5

Table 2. Results for various decoders used in the Improvised Bar Generation model (all the values are in %)

5. EVALUATION

To evaluate the quality of the generated PA pattern of the proposed system, we conduct both objective and subjective tests [1] with: **O:** Original MIDI drum patterns from the dataset; **P1:** The basic drum pattern generated by our Basic Drum Pattern Generation model (section 4.3); **P2:** The final drum pattern with fills and improvisations generated by the complete system.

We screen the generated samples to eliminate those with more than 4 silent bars and those where the variation of bar density is high as measured by the standard deviation of the bar density. After applying the mentioned filtering to 8192 P1 drum samples generated by simple sampling method, we are left with 3762 samples for further evaluation.

[1] Note:Additional objective evaluation methods are reported in the supplementary document `https://bit.ly/2022ismirsupp`

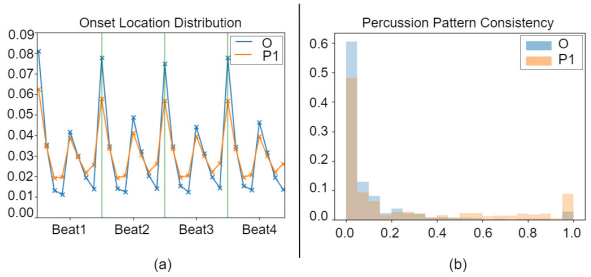


Figure 6. (a) Distribution of onset position in a bar (b) Distribution of Percussion Pattern Consistency

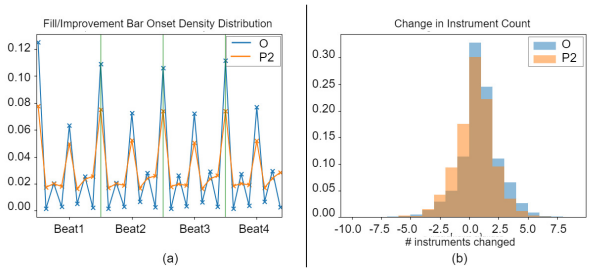


Figure 7. Improvised bars: Distribution of (a) onset location (b) change in instrument count (w.r.t. previous bar)

A basic requirement that any drum pattern generation system must fulfill is the rhythmic positioning of onsets. Generally, in a bar of rock music, the first beat (downbeat) and the third beat have similar instrumentation, featuring a hi-hat and kick drum onset. Beats 2 and 4, commonly referred to as the backbeat, include a hi-hat and snare drum onset. While the quarter notes are accented across the bar, 8^{th} notes are accented within the beat interval. Figure 6a shows the onset distribution present in the bar of both O and P1 samples. We can observe that most of the drum patterns are 8^{th} note patterns and we find a larger proportion of onsets at the accented locations within a beat interval. This is particularly intriguing because the model is given no explicit downbeat or subdivision information yet is still able to emphasis the subdivisions required for an 8^{th} note pattern.

We also do a one-to-one comparison of the P1 outputs against their target drum pattern to understand how closely the patterns match with the original drum sample based on the following metric:

Instrument Count (IC) is defined as the total number of distinct instruments used in a bar. To see whether our model is able to replicate the behavior of multi-instrument dependency, we calculate the deviation of IC in the generated sample with reference to the original target drum track for the same MA. We observe that in 75.8% of the drum bars, we are able to replicate the IC, while in 99.3% of the bars our model was off by at most 1 instrument.

Another important aspect that needs to be considered while generating drum patterns is to have a rhythmic (pattern) consistency across bars. We evaluate this aspects of the generated drum pattern using the following metric:

Pattern Consistency: For consecutive bar pair, we calculate the distance between the drum patterns using (1) keeping $k = 1$. The distribution of the bar distances is shown in Figure 6b. We can see that the generated drum bars are more or less similar to each other with some minor deviations due to the sampling decoding method. The overlapping area of the two distributions is 80.4%.

Next, on the improvised O and P2 bars, we used the following evaluation methods to see how well our models captured the fills/improvisations:

Onset Position: Figure 7a shows the distribution of onset location of percussion instruments across the improvised bars. We observe a slightly higher proportion of 16^{th} note patterns in the improvised O bars as compared to onset distribution across the non-improvised bars seen in Figure 6a. We can see that P2 system is largely able to capture this behavior as well.

Instrument Count (IC) Change: Generally during a fill/improvisation, some additional instrument are being introduced. IC change is measures as the change in IC of improvised bar compared to its previous bar. Figure 7b shows the distribution of IC change. The overlapping area of the two distributions is 87.9%, This shows that our model was able to capture the general trend of IC change.

Additionally, to evaluate the perceptual quality of the generated outputs, we present the generated samples to trained musicians. We created 3 pairs i.e. O & P1; P1 & P2; O & P2 for each MA-PA track and presented them to two guitarists and a multi-instrumentalist with experience ranging from 5 to 10 years. They were asked to provide detailed comments on the drum pattern in terms of timing, appropriateness of fills and coherence of the PA with MA. A common comment from the musicians was regarding the monotonicity in P1 track. As a result when the O & P1 pair was presented, majority of the times O was preferred, but when P1 & P2 were presented, musicians were found to appreciate the fills as it provided a lively feel to the PA.

6. CONCLUSION AND FUTURE WORK

We have successfully shown a method to produce coherent drums accompaniment with improvised bars by conditioning on a given melodic accompaniment. A novel BERT inspired infilling architecture is proposed, along with self-supervised improvisation locator. By learning, where and how to improvise, our evaluations indicate improved generation quality. Thus with a two step approach, we mitigate the biases intrinsic with data-imbalance, and shortcomings that exists with current machine learning architectures. The system can further be improved by learning optimal sampling techniques, which still remains an open problem. As a future work, we could improve the detection performance by employing larger and deeper architectures. This work highlights a serious drawback of traditional language-based generators, which have shown promise in a lot of different fields, yet they fail to capture subtle musical signals, where they are often sparsely occurring in otherwise repetitive and common patterns.

7. REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] P. Verma and C. Chafe, "A generative model for raw audio using transformer architectures," *arXiv preprint arXiv:2106.16036*, 2021.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[4] P. Verma and J. Berger, "Audio transformers: Transformer architectures for large scale audio understanding. adieu convolutions," *arXiv preprint arXiv:2105.00335*, 2021.

[5] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[6] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[7] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2019.

[8] J. Ficler and Y. Goldberg, "Controlling linguistic style aspects in neural language generation," *arXiv preprint arXiv:1707.02633*, 2017.

[9] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," *arXiv preprint arXiv:1805.04833*, 2018.

[10] T. Schick and H. Schütze, "Rare words: A major problem for contextualized embeddings and how to fix it by attentive mimicking," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 8766–8774.

[11] E. Sheng, K.-W. Chang, P. Natarajan, and N. Peng, "The woman worked as a babysitter: On biases in language generation," *arXiv preprint arXiv:1909.01326*, 2019.

[12] I.-C. Wei, C.-W. Wu, and L. Su, "Generating structured drum pattern using variational autoencoder and self-similarity matrix." in *ISMIR*, 2019, pp. 847–854.

[13] F. Tamagnan and Y.-H. Yang, "Drum fills detection and generation," in *International Symposium on Computer Music Multidisciplinary Research*. Springer, 2019, pp. 91–99.

[14] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[15] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[16] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "Popmag: Pop music accompaniment generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1198–1206.

[17] T. Nuttall, B. Haki, and S. Jorda, "Transformer neural networks for automated rhythm generation," 2021.

[18] O. Thörn, "Ai drummer-using learning to enhancearti cial drummer creativity," 2020.

[19] A. Caliskan, J. J. Bryson, and A. Narayanan, "Semantics derived automatically from language corpora contain human-like biases," *Science*, vol. 356, no. 6334, pp. 183–186, 2017.

[20] P.-S. Huang, H. Zhang, R. Jiang, R. Stanforth, J. Welbl, J. Rae, V. Maini, D. Yogatama, and P. Kohli, "Reducing sentiment bias in language models via counterfactual evaluation," *arXiv preprint arXiv:1911.03064*, 2019.

[21] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

[22] J.-P. Briot, G. Hadjeres, and F.-D. Pachet, "Deep learning techniques for music generation–a survey," *arXiv preprint arXiv:1709.01620*, 2017.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[24] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, 2016, pp. 265–283.

[25] A. F. Agarap, "Deep learning using rectified linear units (relu)," *arXiv preprint arXiv:1803.08375*, 2018.

[26] J. Gillick, J. Yang, C.-E. Cella, and D. Bamman, "Drumroll please: Modeling multi-scale rhythmic gestures with flexible grids," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, 2021.

[27] G.-H. Liu, A. Siravuru, S. Prabhakar, M. Veloso, and G. Kantor, "Multi-modal deep reinforcement learning with a novel sensor-based dropout."

[28] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[29] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*. Springer, 1992, pp. 492–518.

[30] L. Pepino, P. Riera, and L. Ferrer, "Emotion recognition from speech using wav2vec 2.0 embeddings," *arXiv preprint arXiv:2104.03502*, 2021.

[31] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[32] J. Gillick, A. Roberts, J. Engel, D. Eck, and D. Bamman, "Learning to groove with inverse sequence transformations," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2269–2279.

[33] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4773–4784, 2020.

BOTTLENECKS AND SOLUTIONS FOR AUDIO TO SCORE ALIGNMENT RESEARCH

Alia Morsi
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain

Xavier Serra
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain

ABSTRACT

Although audio to score alignment is a classic Music Information Retrieval problem, it has not been defined uniquely with the scope of musical scenarios representing its core. The absence of a unified vision makes it difficult to pinpoint its state-of-the-art and determine directions for improvement. To get past this bottleneck, it is necessary to consolidate datasets and evaluation methodologies to allow comprehensive benchmarking. In our review of prior work, we demonstrate the extent of variation in problem scope, datasets, and evaluation practices across audio to score alignment research. To circumvent the high cost of creating large-scale datasets with various instruments, styles, performance conditions, and musician proficiency levels from scratch, the research community could generate ground truth approximations from non-audio to score alignment datasets which include a temporal mapping between a music score and its corresponding audio. We show a methodology for adapting the Aligned Scores and Performances dataset, created originally for beat tracking and music transcription. We filter the dataset semi-automatically by applying a set of Dynamic Time Warping based Audio to Score Alignment methods using out-of-the-box Chroma and Constant-Q Transform extraction algorithms, suitable for the characteristics of the piano performances of the dataset. We use the results to discuss the limitations of the generated ground truths and data adaptation method. While the adapted dataset does not provide the necessary diversity for solving the initial problem, we conclude with ideas for expansion, and identify future directions for curating more comprehensive datasets through data adaptation, or synthesis.

1. INTRODUCTION

Audio to Score Alignment (ASA) is a longstanding Music Information Retrieval (MIR) problem which aims to synchronize a musical score with its audio performance to map between each instant in a recording and a position in the score. When conducted in an online (real-time) fashion, it is often referred to as Score Following, in which the music stream to be aligned (MIDI or audio) must be processed as it is received. When conducted in an offline (non-realtime) fashion, it is often referred to as simply ASA, in which the music stream can be processed after it is fully received. This allows the advantage of looking forward and backward into the input stream [1] before returning the alignment result of each fragment.

Often, ASA problems are solved with methods previously used in audio to audio alignment problems, where the task becomes aligning the performance audio to a synthesized version of the music score [2–5]. In such methods, the alignment is conducted by comparing the same features computed from the synthesized score and the performance audio. Nevertheless, alignment is sometimes performed in the symbolic modality by first transcribing the audio then aligning it with the MIDI score [6–8], and recently alignments were conducted between audio and sheet music images directly through devising intermediate representations allowing both modalities to be compared [9, 10]. The most prevalent approaches through which ASA has been addressed are Dynamic Time Warping (DTW) [1, 2, 4, 7], and Hidden Markov Models (HMM) [6, 8, 11–13], with the former being a popular choice for alignments conducted audio to audio.

ASA research usually begins by defining the scope of the problem and accordingly proposing a system. The characteristics of the target music (i.e. the instruments, recording conditions, musician proficiency, and performance conventions) affect the scope, so ideally, researchers should find annotated data representative of such music. Very often this is not possible, as the datasets available are small and do not cover a variety of musical scenarios. This complicates benchmarking and evaluation because as a result, researchers tend to vary in their choice of data, which we believe hinders the movement of ASA research beyond the typical use-cases. Researchers have sometimes relied on synthetic data as a low-cost and practical way to generate relevant alignment data with properties not found in other datasets. This was done to curate evaluation data [2, 10], to create ground truth for HMM training [8], or to induce temporal mismatches and file corruptions [14].

In our review of prior research (Section 2), we cover several variants of the family of ASA methods comprised of audio representation plus DTW, thus showing how each

© A. Morsi and X. Serra. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** A. Morsi and X. Serra, "Bottlenecks and Solutions for Audio to Score Alignment Research", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

alignment scenario is often addressed in an isolated manner. We believe it would be useful to unify such scenarios under one vision, allowing us to expand the definition of ASA and set clear benchmarking and evaluation strategies representing each of the scenarios formerly addressed in isolation. This would enable researchers to compare between systems and identify directions for improvement, but cannot be achieved without enough varied data to support the development and evaluation of such ASA systems. Since creating datasets is costly, especially at a larger scale, we believe that first we must exhaust the ability to leverage datasets created for tasks other than ASA whenever possible and then synthesize more data as a compliment. This paper demonstrates an example by reusing the Aligned Scores and Performances (ASAP) dataset [15] to generate approximated ground truths for ASA, since it provides 520 classical solo piano performances (audio and MIDI) beat aligned with their symbolic MIDI scores. We thoroughly describe this process in Section 3. In Section 4, we describe the methodology for validating the generated data and discuss their potential problems and aptness for ASA, which involves the application of several DTW-based systems to conduct offline ASA (which, from now onward, we refer to as just ASA). In Section 5, we explain how we use the obtained results to filter data for problems and highlight some limitations of our methods. Although the adapted dataset alone is not a solution to the aforementioned bottleneck, especially since it does not possess the necessary diversity to cover a variety of ASA scenarios, we conclude in Section 6 with ideas for expansion, to support the creation of a unified benchmark and the development of new ideas for ASA research.

2. RELATED WORK

Although this paper concerns ASA, contextualizing its developments requires references to Score Following, which has received more attention over the years. Dixon [16] and Arzt et al. [17] use online versions of DTW suitable for the real-time nature of Score Following. In later years we find the work of Duan et al. [11] and Nakamura et al. [8], both of which propose HMM systems for the same task. Henkel [27] et al. introduce a different paradigm for Score Following, where audio is aligned to score images end-to-end using reinforcement learning. For ASA, recent DTW-based approaches include [2–4, 7]. Table 1 summarizes the datasets used in some of the works above, highlighting the variation among researchers in their choices. The same datasets can be used for the training and evaluation of Score Following and ASA. Although currently inactive, there was a Music Information Retrieval Exchange (MIREX) [1] entry for Score Following which can be used for benchmarking, and it includes several of the datasets shown in Table 1. But the MIREX datasets are small and do not represent a wide variety of scenarios. Moreover, there is no MIREX challenge for ASA.

[1] https://www.music-ir.org/mirex/wiki/2021:Real-time_Audio_to_Score_Alignment_(a.k.a_Score_Following)

2.1 DTW-based Methods

Given two time series $U = u_1, ..., u_n$ and $V = v_1, ..., v_m$, the goal of DTW is to find a minimum cost path $W = w_1, ..., w_n$ where every element in W is an ordered pair (i, j) indicating that the elements u_i and v_j have been aligned. Over the years, researchers have introduced different variants of DTW depending on the specifics of their target problem. For example, FastDTW [28] is a popular, more efficient variant of the algorithm. Moreover, there is the Memory Restricted Multi-Scale DTW (mrmsDTW) [29], which caps the memory requirements of the DTW algorithms for large audio files, for which a python implementation was recently made available in [30]. To the best of our knowledge, there has not been a thorough comparison of all the DTW methods for ASA, although Agrawal et al. [3] compare the results of their proposed system with JumpDTW [31], NWTW [32], and MATCH [16] based on their ability to handle structural variations in the audio compared to the score it is to be aligned with. Moreover, Shan and Tsai [10] compare the alignment results of their proposed Hierarchical DTW with those of JumpDTW and subsequenceDTW [33], where they use intermediate representations [9, 10] allowing the computation of distances between audio and score images. In addition to the variants described above, it is important to note that even within single DTW variant, performance can vary based on system choices such as normalization, the chosen time scales of the feature sequences, and the use of penalties and path constraints [14].

2.2 Audio to Score Alignment Features

Classic features used for ASA are Semitone Energy based features such as Constant-Q Transforms (CQTs) and Pitch Class Profile based features, more commonly known as Chroma representations. In a parameter search by Raffel and Ellis [14] the best alignments were attained with a log-magnitude based CQT. Ewert et al. [21] develop the DLNCO representation, which balances the tradeoff between chroma robustness and time resolution. More recent approaches explore the realm of using learned features [2, 9], or learned distance measures [3]. In [2], the authors explore the use of transposition invariant features learned in an unsupervised way on ASA, thus diverging from pitch based features. They conduct their experiments on piano and orchestral data, and report a result improvement in both. However, in [4], the authors claim that such pitch invariant features underperform in conditions of large tempo variations. So, in their approach, they use features learnt directly from music at the frame level by using a twin Siamese network each containing a Convolutional Neural Network (CNN), and in addition explore the use of salience representations proposed by [34]. In recent work, Automatic Music Transcription (AMT) has been used to first transcribe the target audio before aligning it to the score notes [6, 7].

Source	Datasets	Instruments
[16], [17], [2]	The Vienna 4x22 Piano Corpus [18]	Piano
[11], [8], [1]	Bach10 Dataset [19]	Violin, Clarinet, Tenor Sax, Bassoon
[8], [20], [21]	RWC Database [22]	Polyphonic Multi Instrument
[8]	28 mins of Amateur practice	Clarinet
[7], [2], [4]	MUS Subset of MAPS Database [23]	Piano
[2]	Mozart Sonatas [24], Rachnmaninoff Prelude Op. 23 No. 5 [25]	Piano
[3]	Synthetic dataset based on MSMD [26] and private data	Piano

Table 1: Datasets used across audio to score alignment research

2.3 Evaluation

Cont et al. [35] formalize the quantitative performance metrics of Score Following, forming the basis of the MIREX challenge for the task. Only a subset of their metrics are relevant for ASA since there is no expectation of real-time execution. They define the error as $\mathrm{e}_i = \mathrm{t}_i^e - \mathrm{t}_i^r$, where t_i^e is the estimated time in the score for event i, and t_i^r is the actual time of event i in the reference. The alignment corresponding to an event is considered misaligned only if it exceeds a time threshold θ_e, which we call the misalignment threshold, noting that events could be notes or other time references (See Thickstun et al. [5] for a discussion on the difference between temporal and note based metrics). Accordingly, the following metrics are proposed: the Standard Deviation of the e_i of non misaligned events, the Misalignment Rate (MR) (percentage of events with $|\mathrm{e}_i| \geq \theta_e$, and the Average Imprecision (average absolute error of non misaligned events). For system-wide metrics, they propose the Piece-wise Precision Rate (PPR) over a related subset of scores, calculated as the percentage of non misaligned notes, and the overall precision rate (OPR) calculated similarly to the PPR but over the whole database instead. In practice, researchers slightly vary in their evaluation metrics. They mostly capture the Alignment Rate (AR), according to a set of θ_e usually between 50 ms to 300 ms, sometimes using it as an analogue for PPR.

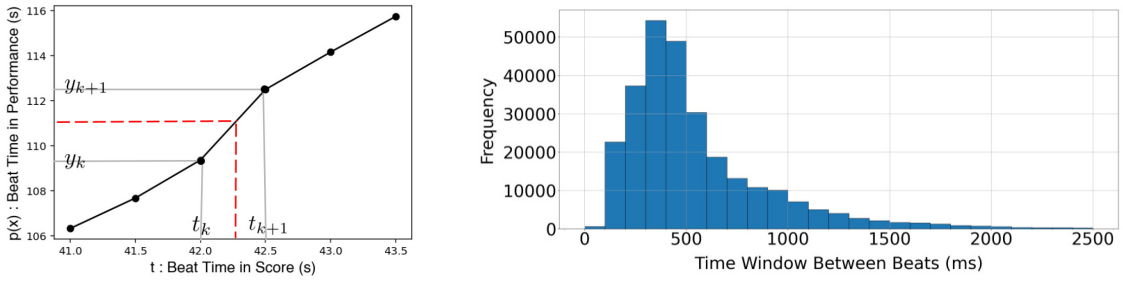
Another commonly used metric is the Average Alignment Error (AAE) defined in [11], which is the average absolute error for each audio frame, distinguishing it from Average Imprecision, which is calculated for non misaligned events only. AAE can be reported in milliseconds or in beats, depending on the end goal in mind [11]. Without using AAE explicitly, Jiang et al. [12] calculate the proportion of misaligned frames by units expressed in beats per measure. A metric with the same essence as AAE is used in [2], where they additionally report the median, 1st, and 3rd quartiles of this difference. In addition to the aforementioned metrics, some authors conduct an extent of qualitative analysis in order to make useful insights about their systems with respect to the scope in which the problem is defined. This has been done to test robustness to performance mistakes [8], for error prone scores [20], to understand the impact of polyphony [11, 19, 20], or the presence of percussion [20], tempo variations [2, 19], or skips [8, 10, 12].

3. GENERATING GROUND TRUTHS FROM ASAP

Reusing the ASAP dataset [15] is a reasonable step to expand the data available for ASA research. First, it offers beat-level aligned audio and music scores for 520 piano performances over various composers and styles. Some pieces are performed by several pianists, and we observe temporal variation in the different interpretations of one piece. In addition, we believe that these alignments could help us create more data by introducing structural variations within a single piece or across different pieces, depending on the scope of the alignment problem we want to consider, which we highlight in Section 6 along with other augmentation ideas to cover a variety of ASA problems. The rest of this section describes how we create ASA ground truth approximations from the beat annotations of ASAP, along with the potential implications and pitfalls of doing so.

3.1 From beat annotations to full alignments

We use the aligned beat annotations of the performance MIDIs and score MIDIs provided by the ASAP dataset to obtain approximated ground truth alignments (performance-aligned scores) for score-performance pairs at a low cost through Piecewise Linear interpolation. Every beat in the score is mapped to a specific time in the performance, yielding an alignment function with which we map each onset time of the MIDI score file to a time in the performance audio. A schematic is shown in Fig 1a. This approach does not give an alignment with a note-to-note resolution. However, we believe it is still usable for evaluating methods outputting temporal alignments that inherently do not provide this level of precision, such as warping paths obtained by DTW alignments, or for training audio to score alignment systems with methods that tolerate weakness in the reference alignments. To understand the extent of the error, we investigate the temporal resolution of the beat annotations (the time distances between consecutive beats) over the chosen subset of the ASAP dataset. As shown in Fig. 1b), the majority of such distances fall between 200 and 1100 ms. Clearly, the faster the tempo of the performance, the less spaced in time consecutive beat annotations are. For context, the distance between two quarter notes in a 120 BPM score is 500 ms.



(a) (b)

Figure 1: (a) Illustrative snippet of the ground truth alignment from the GuoE01M Prelude BMV883 performance. (b) Distributions of the time in between consecutive beats for all files in our chosen subset of the ASAP dataset.

3.2 Error Limits

To understand the limits of the error between the ideal ground truth and the approximated ground truth (ϵ_{gt}) we resort to the definition of Piecewise Linear Interpolation, shown as follows:

$$p(x) = y_k + \frac{y_{k+1} - y_k}{t_{k+1} - t_k}(x - t_k), \text{for } x \in [t_k, t_{k+1}], \quad (1)$$

where x is the time point in the score for which we need to approximate a corresponding time in the performance, t are reference points in the score annotation (in the context of the ASAP dataset, these would be beat times), and y are time reference points in the corresponding performance annotation. This is demonstrated in Fig 1a. In reality, the path between (t_k, y_k) and (t_{k+1}, y_{k+1}) can take any shape as long as it is monotonic (an assumption we can make due to our data). Taking the extreme unrealistic case where $p(x)$ takes on either y_{k+1} or y_k, the ground truth approximation error ϵ_{gt} must be:

$$\epsilon_{gt} = \max(|p(x) - y_k|, |p(x) - y_{k+1}|), x \in [t_k, t_{k+1}]. \quad (2)$$

If x falls on the midpoint of t_k and t_{k+1}, then ϵ_{gt} cannot surpass $\frac{1}{2}(y_k, y_{k+1})$, meaning that even for beats highly spaced apart (1000 ms) the error would be 500ms. Moreover, we would argue that in practice the error would be even less, because of the musical flow. However, the potential of error is a limitation due to which a decision needs to be made on which files to discard. Although not much detail is provided, it is worth noting that Duan and Pardo [19] mention their use of beat annotations to create ground truths, meaning that this process has been accepted in past studies, although it was not discussed elaborately.

4. DATA VALIDATION

The approximated ground truths of our interpolated dataset can have two problem sources: 1) misalignments within the generated annotations due to the low resolution of the score or performance beat annotations (as discussed in Section 3.2), and 2) annotation problems from the original dataset. These need investigation before using the dataset, whether for evaluation or training. We create test audio for every generated annotation file, where the left channel includes the performance-aligned score (the approximated ground truth) and the performance audio on the right channel. Therefore, by listening to all such test audios, it is possible to get a sense of both problems above. However, due to the size of the ASAP subset we reuse, it was not feasible to listen to all the hours of audio. So, we conduct a typical ASA experiment to help give clues as to which files most likely could contain errors and therefore would need to be examined. The rationale of this selective investigation is that files with low misalignment rates have passed an implicit check. Suppose a music score has been aligned with the performance audio using a process different from that with which the ASAP dataset was created. If this result matched the interpolated ground truth, then it is improbable that there is a problem with its beat alignment. Otherwise, there would have been a difference between the alignment result and the ground truth. Moreover, to determine whether we should discard files with large time windows between beats, we listened to a selection of the files with intra-beat annotation time differences of $\approx$ 1500 ms. They sounded correct for several performances, especially for files performed without much temporal variation (such as the Fugue BMV 874 and Prelude BMV 863 performances by Kurz and Shyc, respectively). We decided to keep all such files and only discard them based on the results of the alignment experiment.

4.1 Alignment Experiment

To align a performance and its symbolic score, we sonify the latter using the fluidsynth [2] and conduct DTW-based audio to audio alignment between the synthesized score and the audio performance. We use the librosa [3] [36] DTW implementation, and the distance matrix is computed by applying the Euclidean distance between the feature vectors.

[2] https://www.fluidsynth.org/
[3] https://librosa.org/doc/main/generated/librosa.sequence.dtw.html

4.1.1 Features

We use the CQT and 5 of the chroma representations compared in [37], which are: a Connectionist Temporal Classification loss trained chroma extractor (CTC-Chroma) explained in [38]; Non Negative Least Squares (NNLS) chroma [39]; the Harmonic Pitch Class Profile (HPCP) [40]; the Deep Chroma Extractor (DCE) [41]; and the classic Chroma algorithm implemented in [36]. All these algorithms are easily usable out of the box, and we believe are appropriate to use for piano data. Details on the parameters of each algorithm can be found in the accompanying repository [4] .

4.1.2 Quantitative Results

Since our focus is on filtering files, we report file-based metrics rather than global metrics for each DTW system. Therefore we omit reporting the Overall Alignment Rate (OAR), the system-wide AR considering the notes over all scores (or all temporal units). We use the Average Absolute Error (AAE) for each file in the dataset, shown in the box and whisker plots of Fig. 2 indicating the 1st, median, and 3rd quartiles per each DTW system. We also maintain the AR and the Absolute Errors (AE) for each file. We use these metrics to identify 1) suspects of files with beat alignments that are not highly accurate, which should be discarded, or 2) files for which our ground truth approximation approach yielded a high ϵ_{gt} for annotations within the beats. Those might still be relevant to keep depending on how they will be used. We explain this process in Section 5. However, we must be careful before discarding any files based on performance metrics to avoid cherry-picking only the files for which our ASA systems perform well. This is why we use a variety of audio representations, knowing that some might not be perfectly suitable for audio to score alignment, and no files are discarded unless they performed badly using all DTW systems.

5. RESULT INFORMED DATA FILTERING

Although not the core of our work, we observe that the DTW systems using CQT, HPCP, and Chroma perform better than the rest. This can be seen from Fig. 2 from the lower AAE time windows and the compactness of their distribution, and although we do not show a plot for OAR, these three systems reach very high OARs within the 0 - 60ms error thresholds. In fairness, the CQT system with the best performance is the gold standard obtained by the Bayesian Optimization of [14], suggesting that before making any absolute statements about the superiority of any of the DTW systems, their parameters should be optimized similarly. Besides, comparing such systems or finding the best performing ASA system is not the goal of this paper, and as we argue earlier, ASA is still missing a clear methodology and varied data with which qualitative evaluation can be conducted. This hinders the ability to compare between systems. The ASA results shown are just a means to an end, which is validating the interpolated dataset as described in Section 4, and helping us pinpoint problems and the potential need to filter some files, as shown in Sections 5.1 and 5.2.

5.1 AAE based investigation

Without discarding any generated alignments yet, we start by observing the box-and-whisker plots showing the AAE for all the 520 usable files of ASAP evaluated over the interpolated ground truth references, shown in Fig. 2a. We observe files with very high AAEs, most likely signalling either an annotation or calculation error since they represent alignment error values that are unreasonably high. Drawing a threshold at an AAE of 6000 ms (the red line) allows us to filter those clear outliers, thus arriving at the second plot shown in Fig. 2b. Then, we decide to conduct further filtering at a threshold at an AAE of 1000, arriving at Fig. 2c. We can keep setting lower AAE threshold and filtering more files for as long as needed. But the idea is to listen to the test audio described in Section 4 and to observe the annotations before discarding a file, to make sure that we are not filtering good ground truth approximations.

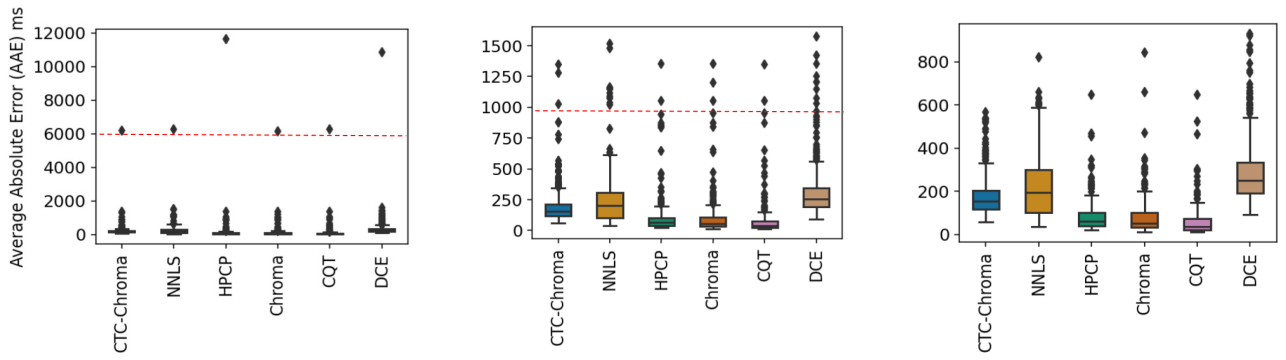
5.2 AR based investigation

Files for which the AR is very low (approx 10%) at θ_e thresholds between 50 and 100 ms signal the need for further investigation. In Section 4 we referred to two problems: 1) the possibility of a temporal offset in the ground truth annotation of the original ASAP dataset, and 2) the possibility of the generated labels being misaligned due to large temporal distances between consecutive beat annotations. If for a file we observe that the 1st quartile, median, and 3rd quartiles do not progress ascendingly as expected a (eg. if the 3 values are nearly equal) and are higher than usual, then this could indicate a temporal offset in the ground truth annotation. Through the listening verification we describe in Section 3, we found that this is the case for at least 6 score-performance pairs. As for the latter problem, if we find that if a file has a low AR, and the 1st, median, and 3rd quartiles of the AE move ascendingly (as expected), then it is a suspect of the low resolution problem. Examples of such files are the 2 performances of Prelude BMV 846, and Prelude BMV 867, where it is clear upon listening that there is a high temporal variation at a phrase level. When coupled with an insufficient resolution of the beat annotations, this would certainly cause ground truth errors. Files of the first category should always be discarded, but files of the second category could be kept, depending on the extent of the misalignment introduced through the ground truth approximation, and the tolerance allowable by the expected use.

5.3 Limitations

A legitimate criticism of our work would be that ground truths generated from the ASAP dataset do not live up to the ambition driving the paper, which is to create a large benchmark for ASA research covering a variety of musical use-cases. Although we do not fully dispute this because

[4] https://github.com/Alia-morsi/asa_benchmarks



(a) Alignment results for all MIDI score-audio pairs. Next cutoff threshold = AAE 6000 ms.

(b) Alignment results for MIDI score-audio pairs with AAE < 6000 ms. Next cutoff threshold = 1000 ms.

(c) Alignment results for MIDI score-audio pairs with AAE < 1000 ms. We do not apply further filtering

Figure 2: Box and Whisker plots showing the Average Alignment Error (AAE) results of the DTW systems using each of the chroma extraction algorithms, calculated using the approximated ground truths. Lower results are better. The red horizontal line of a figure indicates the cutoff threshold to be applied for generating the figure to its right.

all the scores of the ASAP dataset are monotonically increasing classical solo piano performances which highly adhere to their music scores in the performance, our point is that neither the ASAP dataset nor any other single accessible dataset would possess the level of diversity needed to move past the bottleneck. The goal is to start accumulating adapted datasets to eventually arrive at a bigger benchmark. For example, a similar process could be applied to the MazurkaBL dataset [42], and perhaps several others too, although the data preparation methodology and corresponding discussion are coupled with the specifics of the chosen dataset. Moreover, as we better explain in Section 6, even with the generated ground truth approximations from ASAP alone, there is room to create interesting data extensions with the approximated beat alignments. Another drawback of our work could be that the results informed data investigation described in Section 5 is not enough, and there should be a more rigorous manual verification process of the derived ground truths. We agree that manual verification of the whole datatset would be ideal, but we also defend that finding compromises for practical benefit should not be disregarded while being very clear on where these datasets fail. Moreover, perhaps a confidence measure can be created based on comparing the correlation of onsets between the left and right channels of the test audio described in Section 4. Further limitations of this work are that we do not discuss the computational complexity of most ASA methods, and rather constrain the use of the term bottleneck to conceptual hindrances facing ASA.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we argue that ASA research has reached glass ceiling, and a crucial way to get past it is to unify what would be considered core to the problem definition in terms of musical scenarios. For example, what kinds of structural variations between the audio and score should be considered, what kinds of instruments should be supported, what recording quality is expected, etc. We believe this would not be possible without developing benchmarks covering such scenarios, which would support a paradigm shift in how ASA is approached, and would allow us to compare between the performance of ASA systems developed by different researchers. To take a first step towards increasing the size and variety of data, we demonstrate the reuse of the 520 scores of ASAP dataset for which beat aligned scores and performance audio pairs are available. We argue that despite its creation with other MIR research topics in mind, it still can be a very useful resource for researchers interested in ASA for classical piano music. We conduct several data validation steps informed by the AAE and AR from results from a classic DTW pipeline, allowing a selective investigation and filtering of the dataset.

6.1 Future Directions

In addition to adapting more related datasets, we would like to build on this work by artificially extending the data to improve its balance. We need to include cases where the audio performance does not adhere well to the music score, whether through skips, repeats, or performance mistakes. Starting from the generated alignment ground truths (or alignment references from other datasets) we could create semi-artificial data where we shuffle parts of the score, and concatenate the audio from the real performance to match this modified score. To avoid these modifications sounding unnatural, we could try and choose realistic parts of the piece, referring to works on music structure analysis to introduce structural repetitions with more musical sense. Datasets with structural variations would be interesting especially to improve the ability of ASA systems to recover when lost, which is relevant for real-time audio to score alignment. Nevertheless, covering a wider range of instruments still poses a challenge, but this is expected to become easier as synthesis technologies develop further. Finally, we conclude with our hope that ASA approaches would find more inspiration from recent advances in Cover Song Detection and Natural Language Processing.

7. ACKNOWLEDGEMENTS

This research was carried out under the project Musical AI - PID2019-111403GB-I00/AEI/10.13039/501100011033, funded by the Spanish Ministerio de Ciencia e Innovación and the Agencia Estatal de Investigación.

8. REFERENCES

[1] J. Carabias, F. Rodríguez-Serrano, P. Vera-Candeas, N. Ruiz Reyes, and F. Canadas Quesada, "An audio to score alignment framework using spectral factorization and dynamic time warping," in *Proceedings of the 16th International Society for Music information Retrieval Conference (ISMIR)*, October 2015, pp. 742–748.

[2] A. Arzt and S. Lattner, "Audio-to-score alignment using transposition-invariant features," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, September 2018, pp. 592–599.

[3] R. Agrawal, D. Wolff, and S. Dixon, "Structure-aware audio-to-score alignment using progressively dilated convolutional neural networks," in *Proceedings of the 47th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, June 2021, pp. 571–575.

[4] R. Agrawal and S. Dixon, "Learning frame similarity using siamese networks for audio-to-score alignment," in *Proceedings of the 28th European Signal Processing Conference (EUSIPCO)*, January 2020, pp. 141–145.

[5] J. Thickstun, J. Brennan, and H. Verma, "Rethinking evaluation methodology for audio-to-score alignment," *arXiv preprint arXiv:2009.14374*, 2020.

[6] F. Simonetta, S. Ntalampiras, and F. Avanzini, "Audio-to-score alignment using deep automatic music transcription," in *Proceddings of the 23rd IEEE International Workshop on Multimedia Signal Processing (MMSP)*, October 2021.

[7] T. Kwon, D. Jeong, and J. Nam, "Audio-to-score alignment of piano music using rnn-based automatic music transcription," in *Proceedings of the 14th Sound and Music Computing Conference (SMC)*, July 2017, pp. 380–385.

[8] T. Nakamura, E. Nakamura, and S. Sagayama, "Real-time audio-to-score alignment of music performances containing errors and arbitrary repeats and skips," *IEEE/ACM Transactions of Audio, Speech and Language Processing*, vol. 24, no. 2, pp. 329–339, February 2016.

[9] M. Dorfer, A. Arzt, and G. Widmer, "Learning audio-sheet music correspondences for score identification and offline alignment," in *Proceedings of the 18th International Society for Music Information Retreival Conference (ISMIR)*, October 2017, pp. 115–122.

[10] M. Shan and T. J. Tsai, "Automatic generation of piano score following videos," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 4, p. 29–41, March 2021.

[11] Z. Duan and B. Pardo, "A state space model for online polyphonic audio-score alignment," in *Proceedings of the 36th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 197–200.

[12] Y. Jiang, F. Ryan, D. Cartledge, and C. Raphael, "Offline score alignment for realistic music practice," in *Proceedings of the 16th Sound and Music Computing Conference (SMC)*, May 2019, pp. 387–393.

[13] E. Nakamura, K. Yoshi, and H. Katayose, "Performance error detection and post-processing for fast and accurate symbolic music alignment," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, October 2017, pp. 347–353.

[14] C. Raffel and D. P. W. Ellis, "Optimizing DTW-based audio-to-midi alignment and matching," in *41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 81–85.

[15] F. Foscarin, A. McLeod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: a dataset of aligned scores and performances for piano transcription," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, October 2020, pp. 534–541.

[16] S. Dixon, "An on-line time warping algorithm for tracking musical performances," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, August 2005, pp. 1727–1728.

[17] A. Arzt, G. Widmer, and S. Dixon, "Automatic page turning for musicians via real-time machine listening," in *Proceedings of the 18th European Conference on AI (ECAI)*, July 2008, pp. 241–245.

[18] W. Goebl. (1999) The vienna 4x22 piano corpus. http://dx.doi.org/10.21939/4X22.

[19] Z. Duan and B. Pardo, "Soundprism: An online system for score-informed source separation of music audio," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, pp. 1205–1215, October 2011.

[20] C. Joder, S. Essid, and G. Richard, "A comparative study of tonal acoustic features for a symbolic level music-to-score alignment," in *Proceedings of the 35th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2010, pp. 409–412.

[21] S. Ewert, M. Müller, and P. Grosche, "High resolution audio synchronization using chroma onset features,"

in *Proceedings of the 34th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2009, pp. 1869–1872.

[22] M. Goto, "RWC music database: Popular, classical, and jazz music databases," *Proceedings of the 3rd International Society for Music Information Retrieval Conference (ISMIR)*, vol. 1, pp. 287–288, October 2002.

[23] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 6, pp. 1643–1654, August 2010.

[24] G. Widmer, "Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries," *Journal of Artificial Intelligence*, vol. 146, no. 2, pp. 129–148, June 2003.

[25] A. Arzt, "Flexible and robust music tracking," Ph.D. dissertation, Department of Computational Perception, Johaness Kepler University, Linz, December 2016.

[26] M. Dorfer, A. Hajič jr. J.and Arzt, H. Frostel, and G. Widmer, "Learning audio–sheet music correspondences for cross-modal retrieval and piece identification," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 1, p. 22–33, September 2018.

[27] F. Henkel, S. Balke, M. Dorfer, and G. Widmer, "Score following as a multi-modal reinforcement learning problem," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 2, pp. 67–81, November 2019.

[28] S. Salvador and P. K.-F. Chan, "Toward accurate dynamic time warping in linear time and space," in *Journal of Intelligent Data Analysis*, vol. 11, no. 5, October 2007, pp. 561–580.

[29] T. Prätzlich, J. Driedger, and M. Müller, "Memory-restricted multiscale dynamic time warping," in *Proceedings of the 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 569–573.

[30] M. Müller, Y. Özer, M. Krause, T. Prätzlich, and J. Driedger, "Sync toolbox: A python package for efficient, robust, and accurate music synchronization," *Journal of Open Source Software*, vol. 6, no. 64, p. 3434, August 2021.

[31] C. Fremerey, M. Müller, and M. Clausen, "Handling repeats and jumps in score-performance synchronization," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, August 2010, pp. 243–248.

[32] M. Grachten, M. Gasser, A. Arzt, and G. Widmer, "Automatic alignment of music performances with structural differences," in *Proceedings of the 14th International Society for Music Information Retreival Conference (ISMIR)*, November 2013, pp. 607–612.

[33] M. Müller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*, 1st ed. Springer Publishing Company, Incorporated, 2015.

[34] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. Bello, "Deep salience representations for F0 estimation in polyphonic music," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, October 2017, pp. 63–70.

[35] A. Cont, D. Schwarz, N. Schnell, and C. Raphael, "Evaluation of real-time audio-to-score alignment," in *Proceedings of the 8th International Society for Music Information Retreival Conference (ISMIR)*, September 2007, pp. 315–316.

[36] B. McFee, C. Raffel, D. Liang, D. Ellis, M. Mcvicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proc. 14th Python in Science Conference (SciPy)*, July 2015, pp. 18–24.

[37] M. P. Fernández, H. Kirchhoff, and X. Serra", "A comparison of pitch chroma extraction algorithms," in *Proceedings of the Sound and Music Computing Conference*, Saint Etienne, France, June 2022, pp. 222–229.

[38] C. Weiss and G. Peeters, "Training deep pitch-class representations with a multi-label CTC loss," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, Virtual Event, November 2021, pp. 754–761.

[39] M. Mauch and S. Dixon, "Approximate note transcription for the improved identification of difficult chords," in *Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR)*, August 2010, pp. 135–140.

[40] E. Gómez, "Tonal description of polyphonic audio for music content processing," *INFORMS Journal on Computing*, vol. 18, no. 3, pp. 294–304, 08 2006.

[41] F. Korzeniowski and G. Widmer, "Feature learning for chord recognition: The deep chroma extractor," in *Proceedings of the 17th Intternational Society for Music Information Retreival Conference (ISMIR)*, August 2016, pp. 37–43.

[42] K. Kosta, O. F. Bandtlow, and E. Chew, "MazurkaBL: Score-aligned loudness, beat, expressive markings data for 2000 chopin mazurka recordings," in *Proceedings of the 4th International Conference on Technologies for Music Notation and Representation (TENOR)*, May 2018, pp. 85–94.

Papers - Session III - Special Call

RAGA CLASSIFICATION FROM VOCAL PERFORMANCES USING MULTIMODAL ANALYSIS

Martin Clayton[1] **Preeti Rao**[2] **Nithya Shikarpur**[2] **Sujoy Roychowdhury**[2] **Jin Li**[1]
[1]Department of Music, Durham University, United Kingdom
[2]Department of Electrical Engineering, Indian Institute of Technology Bombay, India
martin.clayton@durham.ac.uk, prao@ee.iitb.ac.in

ABSTRACT

Work on musical gesture and embodied cognition suggests a rich complementarity between audio and movement information in musical performance. Pose estimation algorithms now make it possible (in contrast to traditional Motion Capture) to collect rich movement information from unconstrained performances of indefinite length. Vocal performances of Indian art music offer the opportunity to carry out multimodal analysis using this information, combining musician's body movements (i.e. pose and gesture data) with audio features. In this work we investigate raga identification from 12 s excerpts from a dataset of 3 singers and 9 ragas using the combination of audio and visual representations that are each semantically salient on their own. While gesture based classification is relatively weak by itself, we show that combining latent representations from the pre-trained unimodal networks can surpass the already high performance obtained by audio features.

1. INTRODUCTION

In this article we explore the potential for multimodal analysis of Hindustani classical vocal performance. It is well known that Hindustani vocalists use a wide range of manual gestures to accompany their singing: the relationship between their hand movements and the acoustic content of their music has been compared to that between gesture and speech [1–5]. Empirical studies have related gesture to perceived effort and apparent manipulation of imagined objects by singers [6], and have demonstrated the increase of coordinated head movement between soloists and accompanists at cadential moments [7]. The sound-gesture relationship has also been explored in the related Carnatic (South Indian) music tradition [8, 9]. These studies have been based on both empirical analysis and observation of gestures, together with the ethnographic enquiry: empirical study of movement is made possible by various combinations of motion-capture and video-based tracking of individual body parts.

© M. Clayton, P. Rao, N. Shikarpur, S. Roychowdhury, J. Li. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. Clayton, P. Rao, N. Shikarpur, S. Roychowdhury, J. Li, "Raga Classification from Vocal Performances Using Multimodal Analysis", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

Figure 1. Video stills of singers, from left to right: Apoorva Gokhale (AG), Chiranjeeb Chakraborty (CC), and Sudokshina Chatterjee (SCh).

With raga serving as the melodic framework in Indian art music, raga characteristics have been extensively explored via melodic features computed from the predominant pitch contour extracted from performance audio [10, 11]. There has not been any work that has similarly employed the visual data of performances, let alone the combination. In their survey paper [12], Duan et al. reviewed past research categorised by type of instrument and analysis task; no work was found on audiovisual analysis for singing. The nearest instrument task they reviewed was automatic transcription aided by visual analysis such as hand and finger tracking, helping play/non-play detection and note onset localization [13].

In the case of motion capture, the difficulty of data collection, particularly in natural contexts, limits the scope of research. Capture of full-body position information from video, such as is now possible using pose estimation algorithms, significantly increases the potential scope of multimodal analysis, with the possibility of collecting movement data from natural performance contexts extending over long durations. This makes it possible to explore sound-movement relationships of many kinds. Many different aspects of hand movement have been linked to musical sound and structure: for example, the tapping or beating of hands against knees may indicate the tempo and serve to instruct and coordinate performers; continuous hand movements seem to match aspects of the flow and organisation of sung phrases; some gestures seem to indicate analogies with the manipulation of physical objects such as elastic bands, or to describe dimensions of the sound (an open hand shape is understood to be linked to an open-throated voice production; hands gradually moving apart match an increase in volume). Gesture is observed

Raga	Bag	Marwa	Bahar	Kedar	Shree	Nand	MM	Jaun	Bilas	**Sum**
Sum	9	10	8	10	9	8	11	10	11	**86**

Table 1. Number of pieces from each raga in our dataset.

to be idiosyncratic, and yet there may be common features of the sound-movement relationship: Leante argued that specific gestures such as the deliberate vertical raising of a hand linked melodic aspects to visual imagery in Shree Rag [2]. Each of these possibilities suggests different empirical multimodal studies. The study presented here investigates the possibility that manual gesture is sufficiently closely related to the melodic movement of Hindustani ragas (melodic modes) that movement data may be used to help predict the identity of the raga being sung. We achieve this with suitable deep learning models applied to pose data and, further, to the audio and combined audio and visual streams.

In the next section, we present the data set. This is followed by a description of the audio, visual and combined-modality methods explored in this work. Experiments that compare the distinct approaches in terms of raga prediction performance for two different training conditions are discussed next, followed by the presentation of the results and key insights.

2. DATASET AND PROCESSING

We exploit a dataset comprising solo recordings of a common set of 9 Hindustani ragas recorded by 3 Hindustani singers [14]. For each singer and raga, we have 2 distinct takes of alap singing (duration of a take ranges from 165-221 s, with a median duration of 187 s) for 55 recordings in all (for one combination we have only one take, and for two combinations we have three takes). An alap is the improvised opening section of a concert and introduces the raga. We also have a set of 31 pakad (catch phrases) recordings (duration ranging from 18-96 s with a median duration of 19 s). Table 1 shows the raga distribution of the 86 pieces across the 3 singers combined. The ragas, as listed in Table 2, offer a cross-section of raga features in aspects such as the mood or character with which they are associated (serious, joyful, etc.), typical speed and complexity of melodic movement, and predominant melodic range (i.e. favouring the upper or lower tetrachord). The singers, as captured in Figure 1 are all professional performers, two female (Apoorva Gokhale and Sudokshina Chatterjee, here abbreviated AG and SCh) and one male (Chiranjeeb Chakraborty, CC). The pose of the singer's upper body skeleton is estimated for each frame in the corpus using the OpenPose system [15] for skeleton extraction from the video at 25 frames per second. The selected 11 key-points are from the upper body as shown as video input in Figure 2.

Raga	**Scale**
Bageshree (Bag)	S R g m P D n
Bahar	S R g m P D n N
Bilaskhani Todi (Bilas)	S r g m P d n
Jaunpuri (Jaun)	S R g m P d n
Kedar	S R G m M P D N
Marwa	S r G M D N
Miyan ki Malhar (MM)	S R g m P D n N
Nand	S R G m M P D N
Shree	S r G M P d N

Table 2. The pitch sets employed by the nine ragas. Lower case letters refer to the lower (flatter) alternative and upper case to the higher (sharper) pitch in each case.

The recordings are split into clips of 12 s each with starting times separated by a randomly chosen value in the interval [0.8, 2.4] s. The 12 s duration was set in order to encompass the typical duration of vocal phrases in this music. We then investigate the task of raga identification for each clip – first with each modality separately and then with combining the audio and video modalities using different methods to test whether this can enhance overall classification accuracy. This task is attempted under two conditions. In the first (termed 'seen singer'), each singer contributes to both the training and test data, as explained later; in the second (termed 'unseen singer'), we attempt classification of one singer's clips based purely on training using the other singers' clips. Table 3 summarises the number of examples (clips) in the training and validation sets which are distributed almost equally across the 9 ragas for each singer.

3. METHODS

Figure 2 depicts our overall system for raga classification from the audiovisual data. As discussed next, each of the multiple pathways from the input data to the final prediction represent distinct approaches that differ in which modality is exploited or in how the two are brought together.

3.1 Audio Features

Melodic aspects that distinguish ragas include the tonal material, the hierarchy of notes and their sequences leading to characteristic phrases [10]. Phrases and motifs are recognised by their melodic shape which can be represented by the computed pitch contour. Melodic phrases cannot be regarded simply as sequences of the distinct pitch classes, since raga also helps determine features such as oscillations around certain pitches and distinctive pitch transitions (i.e. slides): thus, the pitch contours contain rich information. Our dataset of alap and pakad pieces comprises solo singing with drone. We apply source separation [16] followed by pitch and voicing detection at 10 ms intervals using short-time autocorrelation analysis

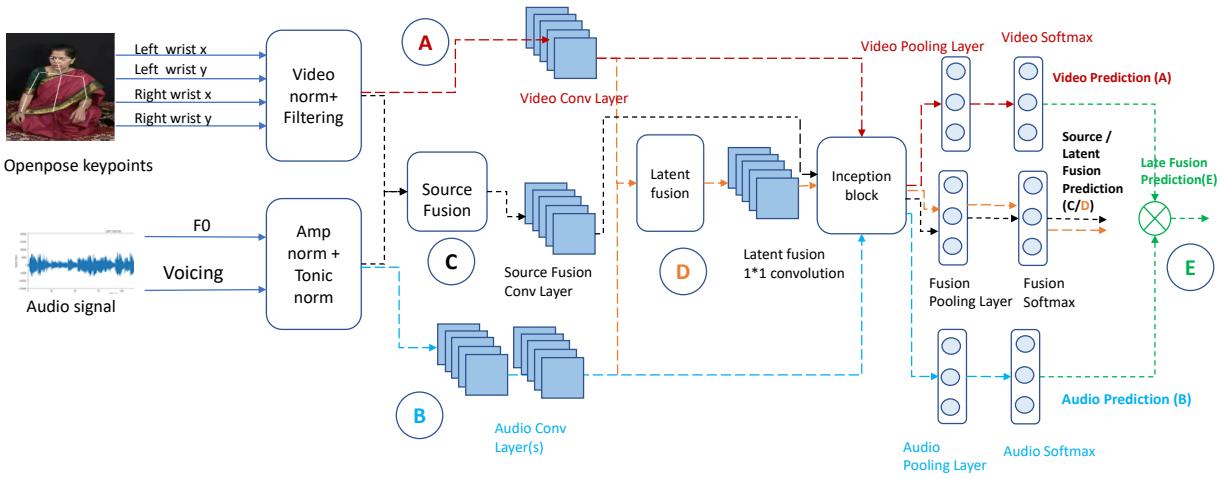


Figure 2. Proposed system for multimodal classification from pose and audio time series extracted from 12s video examples. This diagram shows 5 such configurations. A and B represent unimodal classification for video and audio streams respectively. C and D represent multimodal classification using data fusion at different layers. E represents model fusion using unimodal classification results.

[17, 18]. An important analysis parameter (that also dictates the window size) is the pitch search range, limiting which can help minimize octave errors. The tonic, automatically detected and manually verified [19, 20], was obtained for each performance and used to achieve the needed tonic-normalization of the extracted pitch contours. The detected tonic is also used to define the expected 2 octave pitch range for each piece. The voicing is a binary variable per frame that is set to zero in detected silence frames corresponding to singing pauses. Brief unvoiced regions (less than 500 ms) arising from short silences and consonant utterances are filled in via cubic spline interpolation to obtain the continuous pitch contours associated with melodic movements. The resulting time series from each clip thus comprises of 1200 samples (12s x 100/s).

3.2 Video Features

Real-time skeleton data, such as that obtained via OpenPose, can be handled by a graph model such as Graph CNN. We treat it instead as multivariate time-series data, each time series corresponding to one of the position coordinates of a tracked joint, similar to the processing of data from body-worn sensors in human activity recognition tasks. The 2D positions of the 11 keypoints of the upper body (eyes, nose, neck, shoulders, mid-hip, elbows and wrists) are recorded and used to obtain normalised (x,y) coordinates for each of the two wrists at the frame rate of 25/s. The position data for the singer's two wrists alone is retained for the analysis since (a) these points are more reliably estimated by OpenPose than others such as the elbows or shoulders, and (b) hand gestures most clearly relate to raga expression, and this data is most likely to contribute to raga classification. We thus have 4 time-series representing the x,y positions of each wrist. Any missing data are interpolated and each of the wrist position time series is low-pass filtered to remove any jitter. The length of each of the time series is 300 samples (12s x 25 fps).

3.3 Network Architectures and Hyperparameters

Given that our music audio and video time series data embed information at multiple time scales, we choose an inception network for its multiple kernel sized filters [21,22]. Inception networks have been previously used for multivariate time series classification [23, 24]. We empirically observed that preceding the inception block with two convolutional layers led to superior audio performances while a single convolutional layer worked best for the video input. The relatively high frame rate of audio also necessitated greater stride choices through the convolutional layers that helped reduce the time series dimension to 200 at the input to the inception block. The convolutional layers help in learning audio features which may be relevant for processing via the inception network which further has a range of receptive fields for the convolution. A similar approach to use prior convolution layers with inception blocks was adopted in the original work introducing the inception network for image classification [21]. We further exploit these prior convolutional layers learned for the individual unimodal (audio and video) channels in our latent representation fusion model described in the following section. The inception block is followed by pooling and softmax layers.

Figure 3 shows the architecture of the inception block. Overall, the hyperparameters tuned include the kernel size and number of filters in the convolution layers, the common kernel size, number of filters, pool size and pool type in the inception block and the pool type in the final pooling layer. Hyperparameter tuning was carried out with sweeping across the chosen ranges using Bayesian optimization [25, 26].

3.4 Multimodal Analysis

It is widely believed that integrating features from multiple modalities can lead to more robust classification due

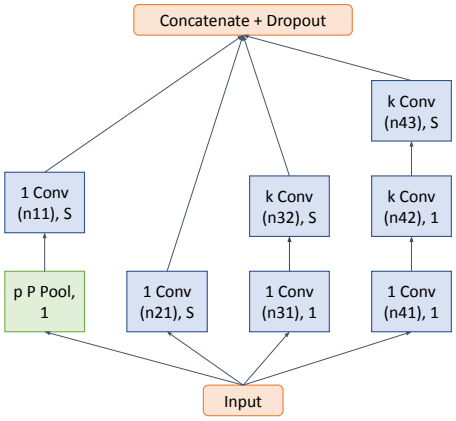


Figure 3. Structure of the inception block. 'k Conv (n), S' indicates a convolution layer with n k-sized kernels and a stride S. 'p' indicates the pooling size of the pool layer and 'P' the type of pooling. k, p, P and the number of filters were determined by hyperparameter tuning. S=1 for the video and combined models and S=2 for the audio model.

to potentially complementary information across the different modalities, all observing the same phenomenon. Multimodal classification has been an active research area in machine learning [27] where the precise approaches to combining information have been broadly categorised as early (feature-based), late (decision-based) and hybrid (their combination). While late fusion builds on combining the decisions of the individual unimodal predictors, early fusion can potentially exploit the low-level correlations between features across modalities. In our task, we have audio and video time series that actually co-vary as the singer makes continuous movements with her hands while singing. This relationship between melodic pitch variation and wrist movements makes it attractive to investigate the combination of the time series for classification.

In Figure 2, the flow-graphs labeled A and B depict the individual video and audio classification paths respectively. While the basic audio-only and video-only models differ only in the number of convolutional layers employed before the inception block, the hyperparameters assume values that are influenced by the time resolution of the corresponding time series with their different sampling rates. Early fusion is then obtained by first downsampling the pitch and voicing time series to the lower rate of 50/s and interpolating the wrist position data from its original 25/s to 50/s. The six time series form a 6-channel input to the convolutional layers thus realizing a form of source fusion. This is shown in position C in Figure 2 and a separate convolutional layer is used to learn the joint features before passing them onto the inception block.

Late fusion is achieved at the point labeled E (decision stage) in Figure 2 by combining the softmax outputs of the ensemble of the best model for each modality. We use soft voting [28] between the two classifiers by averaging the softmax outputs and then choosing the class with the maximum average as the predicted class. In addition, we learn

Data split	**Seen split**			**Unseen split**		
	AG	**CC**	**SCh**	**AG**	**CC**	**SCh**
Train	5590	5487	5588	4715	4105	4304
Validation	972	1075	974	1847	2457	2258

Table 3. Number of 12 s duration examples in each singer's train-validation data split

classifiers using the concatenated softmax outputs of the best models of each modality. This is a common approach in model stacking [29,30] and has been used in multimodal fusion earlier [27,31]. We try multiple models for stacking viz. Random Forests [32, 33] and logistic regression [34] and hyperparameter tune each of them using scikit-learn's GridSearchCV routine [35].

While decision fusion is the most straightforward approach to combining modalities, we also consider fusion at an intermediate stage given that the dynamic correspondence between the movement in the video and the aligned audio is expected to persist in the ealier convolutional layers. We achieve this by fusing the latent representations from each of the subnetworks where each has been optimised for a simpler task, viz. audio-only or video-only classification. Here we use the pre-trained weights of the convolutional layers of the best models of each modality and freeze the weights. This ensures that the features learnt for the individual modalities are maintained and the following inception block is tuned to amplify the interrelations between the two modalities. We call this novel method of fusion as latent fusion and depict it by D in Figure 2. Given that the input audio and video streams are at different effective temporal rates (lengths of 200 and 300 respectively) due to their original sampling and/or the different convolutional layer strides, we need to do a pooling on each individual modality convolutional output to bring the number of steps in the output sequence of the frozen models in sync. In addition, post the fusion we apply a convolution with filter size of 1 and learn the number of required filters via hyperparameter tuning thus letting the model learn the number of channels needed for the inception network.

4. EXPERIMENTS AND RESULTS

The training and validation sets are designed separately for the two tasks. We report experimental results for different model architectures (including one or both modalities) for each singer. In the seen singer task, we create training and validation data splits for each singer. Raga classification accuracies are reported for each method on the validation dataset of each singer and also averaged across the singers. For a given singer, the validation set comprises the set of examples from one of the singer's alap takes for each raga. The corresponding training dataset is then all the *remaining* pieces by the singer plus all the pieces of the other 2 singers. We have thus a validation and train set for each of the 3 singers. The similar exercise is carried out for the unseen singer task. Here, all the pieces by a given singer

are placed in the validation set and the pieces sung by the other two singers in the training set. Table 3 summarises the number of 12 s examples in each split as used in the experiments presented next.

For each task, we carry out model hyperparameter tuning on each individual singer's train-val dataset to obtain 3 sets of hyperparameters in all. These 3 hyperparameter-tuned models next have their weights recomputed on the training data of a given singer (say, AG) to obtain 3 trained models. Finally, the evaluation of Singer AG validation data is obtained via the ensemble of the three models. Similarly, we obtain the ensembled models for each of the singers CC and SCh and evaluate the methods on their respective validation datasets.

Our first set of experiments tests separately the performances of the audio modality and the video modality for the seen and unseen singer raga classification tasks. Table 4 presents the obtained validation accuracies. We observe that the unseen singer task is more challenging as expected. The audio based accuracies are significantly higher than those from video data on both tasks. The video based accuracies in the unseen singer task are close to chance (in the 9-way raga classification) indicating that the association between raga identity and gesture is highly singer dependent.

Our next set of experiments involve different approaches to multimodal classification. Given the non-informative nature of video cues in the unseen singer task, we restrict our attention here to the seen singer task. Table 5 presents the obtained validation accuracies across the different classification methods for each singer and the resulting mean across singers. We observe that early fusion is on the average at the performance level of the weaker modality. This is similar to the observations of Oramas [36] where the learning over very unequal modalities can be overwhelmed by either one. The results of late fusion appear in the final two rows and we find that they fall slightly short of the audio performances, all pointing to the challenge of actually realizing the benefits of the complementary information.

We look for another opportunity to combine audio and video information streams at the output of the convolutional layers. The latent representations at this stage are generated from convolutional layers that are frozen in pre-trained unimodal classification tasks. Rather than simple concatenation of the two representations, we use pooling to first align the representations along the time axis reducing both audio and video to a length of 100 from 200 and 300 respectively. We obtain a performance that is slightly, but consistently, superior to that from audio alone (D vs B in Table 5) for every one of the 3 singers, indicating that the combination of aligned latent representations successfully exploits the joint information. This is also borne out by the histogram summary provided in Figure 4.

5. DISCUSSION

As expected, prediction accuracy is significantly higher for the audio modality than the video, and higher in the 'seen singer' than the 'unseen singer' condition. Prediction accuracy is increased slightly when audio and video data are combined in comparison with audio data alone (from 82.8 to 85.1 %). It should be noted that some extracts score much more highly than others: SCh's Shree scores highly for prediction accuracy in both modalities; CC's MM is unusual in that video prediction seems more accurate than the audio; some extracts score very poorly on prediction from video only, such as AG's Kedar and SCh's Bahar. Some of the wrong predictions can probably be attributed to lack of data: for example, if the singer is silent for a significant part of the 12 second clip, or their hands do not move significantly. Other mismatches may be explained by features of either the melodic movement or the singers' gestures.

We present confusion matrices for the audio, video and the latent fusion bimodal classification in Figure 5. The video-only matrix has significant off-diagonal dispersion.

Data Split	Seen Singer		Unseen Singer	
	Audio	Video	Audio	Video
AG	92.1	36.3	76.9	14.3
CC	79.4	31.8	60.4	13.8
SCh	77.0	39.2	67.2	10.0

Table 4. Validation accuracy (%) of only audio and only video modalities on seen/unseen singer train-val splits.

Model Type	Model Name	AG	CC	SCh	Mean
A	Video	36.3	31.8	39.2	35.8
B	Audio	92.1	79.4	77.0	82.8
C	Source fusion	30.1	42.4	35.8	36.1
D	Latent fusion	**93.3**	**82.7**	**79.2**	**85.1**
E1	Equal voting	85.9	73.7	67.9	75.8
E2	Stacking classifier – RF	81.9	74.2	76.3	77.5

Table 5. Validation accuracy (%) from each singer's split for different model architectures in the seen singer task.

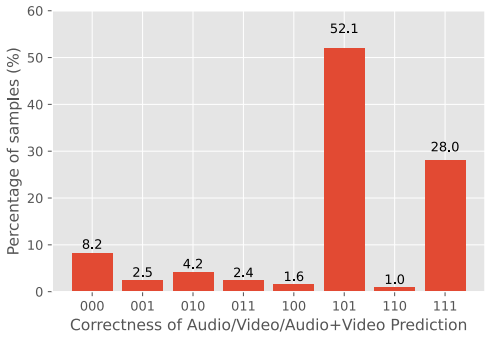


Figure 4. Histogram indicating percentage of the 3021 validation data examples predicted correctly (1) or incorrectly (0) by audio, video and the latent fusion based methods. For example, 011 indicates incorrect prediction by audio but correct predictions by video and bimodal classifiers.

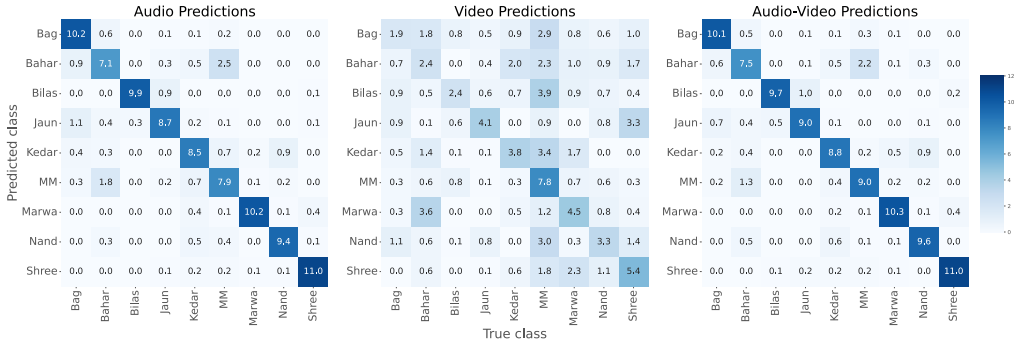


Figure 5. Confusion matrices of predictions made from audio, video and audio-video modalities. Numbers are represented in percentages of the total number of test examples across the three singers combined.

On the other hand, the multimodal matrix visibly improves upon the audio-only by further moving examples into the diagonal. The similar behaviour was noted in the individual singer confusion matrices. We also observe that the most common mis-predictions in the audio modality occur when the scale is the same or similar, and particularly where both pitch material and melodic movements are similar, as is the case between Miyan ki Malhar (MM) and Bahar. The main differences between these two closely-related ragas is that Bahar favours a higher pitch range and faster movement [37]: qualitative review of the prediction results shows that these factors are at play (e.g., in the faster portions of the extracts MM is more likely to be mis-classified as Bahar, at least for the two female singers). Other confusions are observed between Kedar and Nand (same scale), between Bageshree and Bahar (the latter uses one additional note), or between Jaunpuri and Bageshree (one note is different). Relatively few cases occur where the pitch material is very different (e.g. in SCh between MM and Nand or Kedar), which are harder to interpret. Some prediction errors seem to be related to the use of a low pitch (below Pa, the fifth degree, in the lower octave), as for example CC's Shree between 38-53s: this can be explained by the fact that the pitch extraction was constrained to a range of two octaves, with the low Pa as its limit. We find in this case clear instances of video and multimodal predictions doing well.

Confusion matrices for the video prediction are harder to read, given the lower accuracy overall, but some patterns can be observed. The Bahar/MM confusion is also present in the video domain (for the female singers), perhaps related to the fact that these two ragas share not only a scale but also many aspects of their melodic movement. Bahar can sometimes be confused with Nand and Marwa. The Bahar/Nand confusion is not surprising, since both are associated with lively, complex melodic movement and a joyful mood. The confusion between these two ragas and Marwa is less expected, since Marwa's mood contrasts strongly (being serious and often described as unsettled or restless). It may be that there is a similarity in the singers' hand movements between Bahar and Nand's liveliness and Marwa's restlessness. MM, Kedar and Bilaskhani seem to get confused when the singers make rounded, bimanual gestures, as when the hands seem to be moving round each other. These gestures are associated with specific melodic movements, an andolan (slow oscillation) on the Ga (3) in MM and distinctive crooked (vakra) pitch movements in the other two ragas. For SCh, several ragas are wrongly predicted as Shree when she makes a direct upward hand movement, which is often distinctive of this raga [2]. Such qualitative observations suggest that the video prediction system may be classifying the hand movements in meaningful ways even when the predictions are wrong, pointing to similarities between the ways each singer gestures in different ragas.

When we look at the proportion of clips correctly identified in at least one of the two modalities in Figure 4, the most common result is to be correctly classified from the audio but incorrectly classified from the video. The small percentage of clips for which the opposite is true (c. 6.6 %) suggests there is some scope for the video information to improve the prediction accuracy achieved through audio alone, although this seems like a difficult challenge as the audio prediction accuracy is already high. Even so, we note that the 6.6 % where the audio is incorrect but video correct, the multimodal condition actually recovers about a third of this. Further, we have 2.5 % of the total set of clips (75/3021) correctly predicted only in the multimodal condition (i.e. audio and video data are combined using latent fusion). This amounts to a quarter of all clips that were wrongly classified in both audio-only and video-only conditions. It is interesting to note that of the 75 clips correctly predicted only in the multimodal condition, the most common ragas represented were Bahar (19) and MM (14), which as noted above share the same scale and many melodic movements. This study suggests that the combination of coordinated audio and gesture features can improve the classification of ragas from short clips. This approach could be extended to other musicological investigations such as the musical expression of mood or character, melodic phrase segmentation, and interpersonal coordination.

Supplementary material: https://dap-lab.github.io/multimodal-raga-supplementary/.

6. REFERENCES

[1] M. Clayton, "Time, gesture and attention in a khyāl performance," *Asian Music*, vol. 38, no. 2, pp. 71–96, 2007.

[2] L. Leante, "The lotus and the king: Imagery, gesture and meaning in a hindustani rāg," *Ethnomusicology Forum*, vol. 18, no. 2, pp. 185–206, 2009.

[3] ——, "Gesture and imagery in music performance: Perspectives from north indian classical music," in *The Routledge Companion to Music and Visual Culture*, T. Shephard and A. Leonard, Eds. Routledge, 2013, pp. 145–152.

[4] ——, "The cuckoo's song : imagery and movement in monsoon ragas." in *Monsoon feelings : a history of emotions in the rain.*, I. Rajamani, M. Pernau, and K. R. B. Schofield, Eds. New Delhi: Niyogi Books, 2018.

[5] M. Rahaim, *Musicking Bodies: Gesture and Voice in Hindustani Music*. Wesleyan University Press, 2012.

[6] S. Paschalidou, T. Eerola, and M. Clayton, "Voice and movement as predictors of gesture types and physical effort in virtual object interactions of classical indian singing," in *Proc. of the 3rd Int. Symposium on Movement and Computing*, 2016.

[7] M. Clayton, K. Jakubowski, and T. Eerola, "Interpersonal entrainment in indian instrumental music performance: Synchronization and movement coordination relate to tempo, dynamics, metrical and cadential structure," *Musicae Scientiae*, vol. 23, no. 3, pp. 304–331, 2019.

[8] L. Pearson, "Gesture and the sonic event in karnatak music," *Empirical Musicology Review*, vol. 8, no. 1, pp. 2–14, 2013.

[9] ——, "Coarticulation and gesture: An analysis of melodic movement in south indian raga performance," *Music Analysis*, vol. 35, no. 3, pp. 280–313, 2016.

[10] G. Koduri, S. Gulati, P. Rao, and X. Serra, "Rāga recognition based on pitch distribution methods," *Journal of New Music Research*, vol. 41, no. 4, pp. 337–350, 2012.

[11] K. K. Ganguli and P. Rao, "On the distributional representation of ragas: Experiments with allied raga pairs," vol. 1, no. 1, pp. 79–95, 2018.

[12] Z. Duan, S. Essid, C. C. Liem, G. Richard, and G. Sharma, "Audiovisual analysis of music performances: Overview of an emerging field," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 63–73, 2019.

[13] A. Bazzica, J. C. van Gemert, C. C. S. Liem, and A. Hanjalic, "Vision-based detection of acoustic timed events: a case study on clarinet note onsets," 2017. [Online]. Available: https://arxiv.org/abs/1706.09556

[14] M. Clayton, J. Li, A. R. Clarke, M. Weinzierl, L. Leante, and S. Tarsitani, "Hindustani raga and singer classification using pose estimation," 2021. [Online]. Available: https://doi.org/10.17605/OSF.IO/T5BWA

[15] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 1, pp. 172–186, 2021.

[16] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, p. 2154, 2020.

[17] Y. Jadoul, B. Thompson, and B. de Boer, "Introducing parselmouth: A python interface to praat," *Journal of Phonetics*, vol. 71, pp. 1–15, 2018.

[18] P. Boersma and D. Weenink, "Praat: doing phonetics by computer [Computer program]," http://www.praat.org/, 2021, version 6.1.38, retrieved May 2022.

[19] J. Salamon, S. Gulati, and X. Serra, "A multipitch approach to tonic identification in indian classical music," in *Proc. of the 13th Int. Soc. for Music Information Retrieval Conference*, Porto, Portugal, 2012.

[20] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor *et al.*, "Essentia: an audio analysis library for music information retrieval," in *Proc. of the 14th Int. Soc. for Music Information Retrieval Conference*, Curitiba, Brazil, 2013.

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[22] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[23] C.-L. Liu, W.-H. Hsaio, and Y.-C. Tu, "Time series classification with multivariate convolutional neural network," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4788–4797, 2018.

[24] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. Schmidt, J. Weber, G. I. Webb, L. Idoumghar, P. Muller, and F. Petitjean, "Inceptiontime: Finding alexnet for time series classification," *Data Mining and Knowledge Discovery*, vol. 34, no. 6, pp. 1936–1962, 2020.

[25] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, 2011.

[26] L. Biewald, "Experiment tracking with weights and biases," 2020, software available from wandb.com. [Online]. Available: https://www.wandb.com/

[27] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 2, pp. 423–443, 2018.

[28] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[29] N. Hatami and R. Ebrahimpour, "Combining multiple classifiers: diversify with boosting and combining by stacking," *Int. Journal of Computer Science and Network Security*, vol. 7, no. 1, pp. 127–131, 2007.

[30] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Machine learning*, vol. 54, no. 3, pp. 255–273, 2004.

[31] J. H. Koo, S. W. Cho, N. R. Baek, M. C. Kim, and K. R. Park, "Cnn-based multimodal human recognition in surveillance environments," *Sensors*, vol. 18, no. 9, p. 3040, 2018.

[32] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[33] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152.

[34] R. E. Wright, "Logistic regression." 1995.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] S. Oramas, F. Barbieri, O. Nieto, and X. Serra, "Multimodal deep learning for music genre classification," *Transactions of the Int. Soc. for Music Information Retrieval*, vol. 1, no. 1, pp. 4–21, 2018.

[37] S. Rao and W. van der Meer, "Miyan ki malhar," https://autrimncpa.wordpress.com/miyan-ki-malhar/, accessed: May-2022.

TRACES OF GLOBALIZATION IN ONLINE MUSIC CONSUMPTION PATTERNS AND RESULTS OF RECOMMENDATION ALGORITHMS

Oleg Lesota[1,2] **Emilia Parada-Cabaleiro**[1,2] **Stefan Brandl**[1]
Elisabeth Lex[3] **Navid Rekabsaz**[1,2] **Markus Schedl**[1,2]
[1]Multimedia Mining and Search Group, Institute of Computational Perception, JKU Linz, Austria
[2]Human-centered AI Group, AI Lab, Linz Institute of Technology (LIT), Austria
[3] Graz University of Technology, Austria
markus.schedl@jku.at

ABSTRACT

Music streaming platforms allow users to enjoy music from all over the globe. Such opportunity speeds up cultural exchange between different countries, a process often associated with globalization. While such an exchange could lead to more diverse music consumption, empirical evidence on its influence on online music consumption is limited. Besides, the extent to which music recommender systems foster exchange or amplify globalization in music remains an understudied problem.

In this paper, we present findings from an empirical study to detect traces of globalization in domestic vs. foreign online music consumption. Besides, we investigate if popular recommendation algorithms, specifically ItemKNN and NeuMF, are prone to amplifying globalization processes. Our experiments on Last.fm listening data show nuanced patterns of globalization in music consumption. We observe a strong position of US music in all considered countries. In countries such as Sweden, Great Britain, or Brazil, US music shows various levels of coexistence with domestic music. We find that Finland is least influenced by US music, while greatly consuming and "exporting" domestic music. With respect to recommendation algorithms, ItemKNN tends to recommend domestic music to users of many countries, while NeuMF contributes to accelerating globalization and shifting balance towards dominance of US music on the market.

1. INTRODUCTION AND RELATED WORK

Globalization can be defined as an *"expanding cultural exchange between countries, which may imply an increasing consumption of foreign cultural goods beside the local ones"* [1]. Among others, previous research proposes two interpretations of globalization: (i) *Cultural Imperialism* [2], i. e., the growing cultural exchange triggered by globalization is mostly profitable for certain dominant western cultures (in particular American culture) and thereby threatens to overwhelm the others; and (ii) *Glocalization* [1, 3], i. e., fostering the development of local cultures through the globalization process, by means of adaptation of global cultural forms and strengthening local identity as a counterbalancing mechanism against global influences. These interpretations imply that globalization exposes local cultures to pressure from global trends, and while some of them adapt and confront the threat, others weaken and decline.

The realizations of these interpretations in various domains have been confirmed by several studies. For instance, Crane [2] shows the dominance of US film industry in most regions of the world, while Chen and Shen [4] display the ability of some cultures to adapt and develop under the pressure of more dominant ones. Approaching globalization, most of the previous works resort to the analysis of various aggregated popular charts representing mainstream consumption trends [1, 5–7]. Specifically, Achterberg [1] show that US music has become increasingly popular in the Netherlands, Germany, and France until the late 1980s', while starting in the 90s', more local music has been produced. The revival of domestic music consumption is also evidenced by Bekhuis et al. [6], who show that popularity of domestic artists is positively correlated with a high sentiment of national pride. Similarly, Verboord and Brandellero [7] conduct a multilevel analysis of popcharts' evolution, showing that the consumption of foreign music has increased in many countries except in the US.

The mentioned studies conduct the analysis on aggregated pop-music charts, neglecting the nuances of music consumption by the consumers with less mainstream tastes. Including less-mainstream listeners in country-specific analyses is particularly important, as listeners in different countries display diverse tendencies towards listening to mainstream music [8], which also gets translated into significant differences in the *performance* of recommendation algorithms [9, 10]. However, to the best of our knowledge, little research has been dedicated to investigate the *influence* of streaming platforms on globalization processes [5]. Furthermore, the extent to which existing globalization processes might be amplified by music rec-

© O. Lesota, E. Parada-Cabaleiro, S. Brandl, E. Lex, N. Rekabsaz, and M. Schedl. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** O. Lesota, E. Parada-Cabaleiro, S. Brandl, E. Lex, N. Rekabsaz, and M. Schedl, "Traces of Globalization in Online Music Consumption Patterns and Results of Recommendation Algorithms", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

ommender systems has, to the best of our knowledge, not yet been investigated.

To bridge these gaps, we leverage online music listening data, allowing us to analyze globalization patterns based on actual per-user consumption of music from a wider range of countries. In addition, we study potential impacts music recommender systems may have on globalization. To this end, we strive to answer the following three research questions:

- **RQ1**: How prominent is the aspect of US "cultural imperialism" in the sphere of online music consumption? Is its influence uniform across countries?
- **RQ2**: How significant is domestic music consumption in different countries? Are there any signs of "glocalization"?
- **RQ3**: Do music recommender systems influence users' inclination towards music coming from certain countries?

We highlight the importance of RQ3, considering recent research showing that recommender systems can amplify biases from underlying data in their output, as well as inflict additional algorithmic biases. An overview of biases recommender systems are prone to is given in [11,12]. Among them is popularity bias, i. e., the tendency to favor popular items at the expense of niche and less popular items, often leading to dissatisfaction of users [13,14]. We pose RQ3 to investigate possible occurrence of analogous "globalization bias" in recommender systems.

The remainder of the paper is structured as follows. Section 2 describes our methodological approach to the research questions, including data and methods used. We report and analyze our results in Section 3. Section 4 is dedicated to limitations of our study. Finally, we sum up our findings and list potential future directions in Section 5.

2. METHODS AND MATERIALS

In the following, we detail the methodology we adopt to answer the research questions and describe the dataset our analysis is based upon. We first clarify and formally define the terminology we use in the methodological description: A *listening event* (or *interaction*) is a tuple $< u, i >$, indicating that a user u consumed an item i, which is a music track or a song in our case. Each track i has been created or performed by an artist a.[1] Each artist a originates from a country c_a. Each user u is from a country c_u. $\mathcal{I}_{c_u}$ denotes the set of all user–item interactions made by users in country c_u. $\mathcal{I}_{c_a}$ refers to the set of all interactions with items created by artists from country c_a. $\mathcal{I}_{c_a,c_u}$ denotes the set of all interactions with items created by artists from country c_a, listened to by users from country c_u.

2.1 Investigating US Cultural Imperialism

To answer RQ1, we compute, for all users in a given fixed country c_u, their aggregate share of consumed music that has been created by artists from each country under consideration. Since we are interested in this share for local and US music (artists), we define the following function

$$IC_{c_u \to c_a} = \frac{|\mathcal{I}_{c_a,c_u}|}{|\mathcal{I}_{c_u}|}$$

for which we consider three cases:

- *Local:* $c_a = c_u$,
- *US:* $c_a = US$, and
- *Other:* $\sum_{c_a \in C \setminus \{c_u, US\}} IC_{c_u \to c_a}$, where C is the set of all countries considered (sum over all artist countries that are not the local country nor the US).

In simple words, this formal framework can answer how much of the music all users in country c_u consume was created by artists from the same country (local consumption), was created by artists from the US, and was created by artists from other countries (neither local nor the US); all expressed in relative numbers.

2.2 Investigating Traces of Glocalization

To answer RQ2, we consider all listening events of songs by artists from the country under investigation, i. e., we fix c_a. We then define $IC_{c_u \leftarrow c_a}$, analogously to $IC_{c_u \to c_a}$ above, i. e., for a given country c_a, the share of its artists' listening events that originate from users in each country c_u under investigation.

$$IC_{c_u \leftarrow c_a} = \frac{|\mathcal{I}_{c_a,c_u}|}{|\mathcal{I}_{c_a}|}$$

Here we are mostly interested in this share between local and foreign music consumers, and accordingly consider two cases:

- *Local:* $c_u = c_a$, and
- *Other:* $\sum_{c_u \in C \setminus \{c_a\}} IC_{c_u \leftarrow c_a}$, where C is the set of all countries considered (sum over all user countries that are not the local country).

In simple words, this formal framework can answer how much of the music created by artists from c_a is consumed by users in the same country (local consumption of local artists), and how much by users in other countries; all expressed in relative numbers.

2.3 Influence of Recommendation Algorithms on Globalization Patterns

To approach RQ3, we consider two popular recommendation algorithms, a classical ItemKNN [15] and a more recent, deep-learning-based NeuMF [16]. ItemKNN assigns a recommendation score to an item for a given user based on how similar this item is to the items already consumed by the user. Similarity is computed based on the interactions of other users. This algorithm does not learn any special representations for users and items, operating directly on the interaction matrix. On the other hand, NeuMF is a matrix factorization approach. Not only does it learn user and item embeddings, but also a dedicated scoring

[1] Being aware that "artist" in the context of music can refer to different subjects, we adopt a pragmatic definition here. Owed to the type of user-generated data we investigate in our study, we consider both composers and performers as artists.

Country	Tracks	Users	Artists	Interactions	
				Users	Artists
US	252,370	1,763	15,440	2,057,684	6,607,441
GB	99,911	890	5,271	1,095,637	2,767,202
DE	42,799	890	3,077	1,012,806	697,866
SE	29,108	348	1,970	393,348	672,944
CA	24,005	232	1,565	304,817	594,868
FR	17,718	281	1,815	337,739	357,730
AU	14,770	208	1,306	261,965	343,892
FI	14,673	448	1,093	508,934	286,145
BR	14,091	1,138	1,022	1,312,909	232,640
RU	11,779	1,288	848	1,202,064	155,409
JP	11,731	115	1,228	92,459	143,203
NO	11,282	224	765	256,921	238,427
PL	11,145	1,121	883	1,249,746	186,032
NL	10,958	406	1,018	573,307	186,117
IT	9,633	252	1,058	237,708	131,769
ES	6,115	257	765	297,364	71,862
BE	4,204	141	586	166,247	64,765
MX	2,881	213	323	295,140	33,887
UA	1,849	317	160	348,706	27,339
TR	1,478	115	286	113,165	14,868
Other	44,736	2,228	4,947	2,521,335	825,595
Total	637,236	12,875	45,426	14,640,001	14,640,001

Table 1: Basic statistics of the dataset. For each country, we report the number of users, tracks, artists, and interactions made by all users in the country as well as interactions made to artists from the country.

function, thanks to the multi-layer perceptron constituting a part of the model.

To estimate potential influence of the recommender algorithms on globalization patterns in different countries, we train them on subsamples of the dataset used to answer RQ1 and RQ2, and then analyze the recommendations they produce. To this end, we consider top 10 recommendations provided to each user and then calculate $IC_{c_u \to c_a}$ for $c_u = c_a$ and $c_a = US$ exactly like for RQ1.

Because the dataset under investigation contains a large number of items (see Table 1), we use its subsamples (containing around 100K items each) to run recommendation experiments, thereby avoiding computational limitations. For the subsamples, we enforce the following standard limitations: every track has to be interacted with at least 5 times and every user is required to have at least 5 interactions. To ensure robustness of the results, we conduct the experiment on three such random subsamples and report average recommendation levels.

2.4 Dataset

We conduct our experiments on the LFM-2b dataset [17] of listening events created by users of the online music platform Last.fm. [2] Unlike stand-alone streaming services such as Spotify or Deezer, Last.fm is based on the concept of "scrobbling", meaning that its users can share on the platform which music they are listening to, regardless of the actual service, device, or application they are using for music consumption. Therefore, Last.fm can be regarded as an aggregator that reflects the entire music consumption history of its users. This is desirable for our analysis since we aim at capturing in a more comprehensive way each listener's (and country's) music consumption behavior instead of focusing on one particular streaming platform.

The full LFM-2b dataset contains listening histories of ∼120K users, totaling to ∼2B interactions. Since the dataset's temporal coverages spans 15 years, from 2005 to 2020, and we intentionally leave out aspects of temporal dynamics from our analysis (see future work), we only consider a subset covering the years 2018-2019. Furthermore, we exclude potentially accidental interactions by removing listening events $< u, i >$ that only occurred once. [3]

Since our analysis necessitates country information of both users and artists, we first remove all users for which no such information is available in LFM-2b. We then collect information about artists' country from Musicbrainz [4] and only retain those artists for whom country data could be retrieved. After these steps we end up with a dataset of ∼14,640K interactions triggered by ∼13K users from 143 countries with ∼637K music tracks produced by ∼45K artists from 155 countries.

Finally, to reduce the complexity of the analysis and concentrate on reasonably represented countries, we select for the detailed analyses countries with at least 100 users and only countries whose artists created at least a total of 1,000 tracks. These filtering steps result in 20 countries [5] from all over the world (see Table 1 for basic statistics), which we further analyze. Note that the countries beyond these 20 still contribute to the results mentioned as a part of the aggregation over "other" countries.

3. RESULTS

We illustrate our findings with a series of figures. Figure 1 shows how popular is domestic music, music produced by US artists, and music produced by artists from other countries in every of the 20 investigated countries. In case of DE, a little under 40% of listening events generated by German users are allocated to music produced by US artists (orange bar on the left). Under 20% of listening events generated by them is allocated to domestic music, i. e., produced by German artists (blue bar on the right). Figure 2 indicates the degree to which domestic music of different countries is consumed in the country of origin and outside. For example, BR has most of the listening events allocated to its domestic music coming from Brazilians, showing that it is not as popular in other countries. Figure 4 shows the spread of listening interactions with music from every country across other countries. In other words, this matrix shows how uniform the "export" of domestic music from every country to other countries is. Every row

[2] `https://last.fm`

[3] Since the dataset provides no information about the duration of a listening event, those single user–item interactions are often the result of a recommender engine starting to play a new track to the user, which the user skips after a few seconds. Therefore, we exclude those single interactions to remove this kind of noise.

[4] `https://musicbrainz.org`

[5] Throughout the paper, we use ISO codes to abbreviate countries. US: United States, GB: United Kingdom, DE: Germany, SE: Sweden, CA: Canada, FR: France, AU: Australia, FI: Finland, BR: Brazil, RU: Russia, JP: Japan, NO: Norway, PL: Poland, NL: Netherlands, IT: Italy, ES: Spain, BE: Belgium, MX: Mexico, UA: Ukraine, TR: Turkey

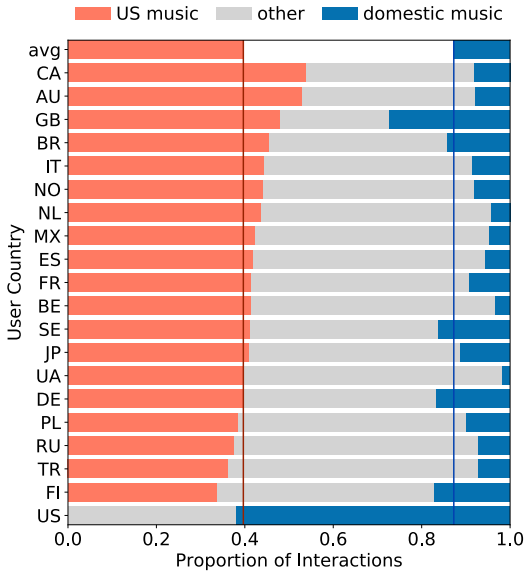


Figure 1: Distribution of listening activity over foreign (US and other) and domestic music. Every row corresponds to listeners of one country. Average proportions of interactions with music produced in the US (orange) and domestically (blue) are shown in the first row.

corresponds to the share, among all listening interactions, of the music produced in a single country. Every cell in the row shows the percentage of listening events coming from listeners of the corresponding country on the x-axis. For example, we can observe that 14.6% of listening interactions with JP music comes from US users (row JP, column US). Figure 3 demonstrates the potential impact two recommendation algorithms, ItemKNN and NeuMF, may have on the consumption balance in different countries. In each of the two subplots, empty bars reflect proportions of music actually consumed by the users (similar to Figure 1): orange bars on the left indicate listening events allocated to US artists, blue bars on the right refer to domestic artists. The filled bars illustrate the same concept applied to items recommended to users of different countries. For example, in Figure 3b, the row corresponding to DE shows that about 50% of items recommended to German users come from US artists, while at the same time only about 40% of their actual listening activity belongs to tracks from US.

We make the following observations answering **RQ1**. First, from the listening activity on Last.fm in 2018-2019, we see that 39.6% of all items interacted with were produced by US artists. Second, as Figure 1 shows, over 60% of listening events generated by US users are allocated to domestic music (bottom row, blue bar). This marks domestic superiority of US music, unmatched by any other country. The runner-up, being GB, shows only about 30% of listening events allocated to the domestic market. Third, on average, listeners of considered countries allocate 40% of their consumption to music produced by US artists (top row and vertical red line), with a maximum of about 50% shown by Australia and Canada. For many countries, above-average consumption of US music is combined with below-average consumption of domestic music, e. g., AU,

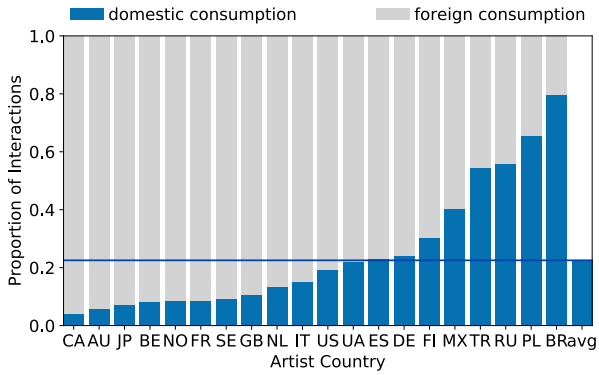


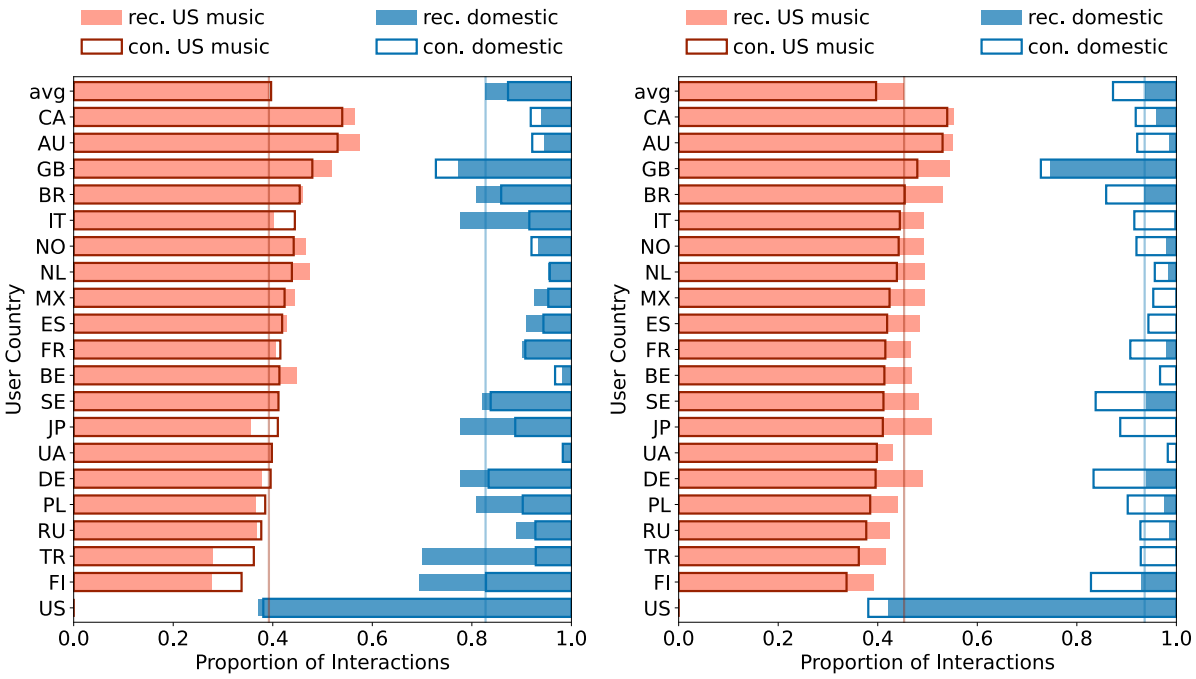
Figure 2: Consumption distribution of music produced in different countries, between local (blue) and international (gray) audiences.

CA, NL, IT, and UA. Forth, some countries such as GB, BR, and SE show above-average consumption of US music combined with also above-average consumption of domestic music. This is an indication of local musical culture comfortably coexisting and possibly interacting with the incoming US culture. Fifth, other countries such as TR, PL, FI, and RU consume below average of US music, remaining more open for music coming from other countries. In addition, FI also displays significant attention to domestic music.

From these observations, we conclude that music produced by US artists maintains strong positions in the considered countries. In particular, it dominates its own domestic market unlike domestic music of other countries. While there are countries combining low consumption of domestic with high consumption of US music, it is hard to call US music globally dominating. Many regions are also comparably influenced by other countries (if combined) and some, like FI or GB, show very strong positions of domestic artists. Thus, we hesitate to call US "cultural imperialism" absolute and homogeneous across countries.

We approach **RQ2** by defining three indicators related to domestic music in every country: *international consumption*, i. e., how big is the proportion of interactions with its domestic music coming from other countries (in other words, how widely exported domestic music is, see Figure 2 and 4); *domestic popularity*, i. e., the proportion of interactions from listeners of the country with domestic music (how popular domestic music is in its home region, see Figure 1); *popularity of US music* (used to detect hints of cultural dialog between US and considered country, see Figure 1). We analyze these indicators in terms of below/above average across considered countries for every particular country.

Using the indicators described above, we identify four patterns in behavior of domestic music scenes. First, glocalization through adaptation. Countries such as SE and GB score above average on all three indicators: their music is appreciated abroad while also being popular locally, and in addition these countries consume above average US music. We interpret this pattern as comfortable coexistence of global and domestic cultures with the latter likely adapt-



(a) Impact of ItemKNN

(b) Impact of NeuMF

Figure 3: Potential impact of recommender algorithms on music consumption balance in different countries. Empty boxes reflect actual consumption (denoted con.) extracted from the data. Filled boxes represent the output the two recommender algorithms produce (denoted rec.). Orange-tinted boxes on the left correspond to consumed / recommended music from US. Blue-tinted on the right - consumed / recommended domestic music.

ing and contributing to the former. Second, heavier adaptation of global trends. Domestic music is popular in BR as well as US music (both indicators above average). Domestic BR music receives most of its interactions from BR and little international attention. This pattern may imply even heavier adaptation of global influences to domestic circumstances. Third, non-US influenced. Countries such as PL, TR, and RU score below average on all three indicators: their music is predominantly consumed on domestic market while being less popular than US-produced music (which in turns enjoys below average popularity in these countries). We interpret this combination in a way that in these countries domestic music competes with a wide array of incoming music and US music is not necessarily the strongest influence there. Fourth, as the sole representative of this pattern, FI demonstrates high popularity of domestic music, combined with decent international consumption and lower popularity of US music. This pattern may express successful adaptation of less mainstream (beyond US) influences as well as strong distinct national culture.

We answer **RQ3** by analyzing results of the recommendation experiment detailed in Section 2.3. As shown in Figure 3b, NeuMF consistently recommends more US-produced items to listeners of every non-US country than these non-US listeners used to consume. This happens at the expense of the share of recommended domestic items. The character of the change implies that the share of recommended items from other countries is also larger than their actual consumption share (for all countries). We conclude that NeuMF amplifies globalization patterns and in particular dominance of an already dominant player, i. e., the US. On the contrary, ItemKNN (see Figure 3a) shows a different and less consistent behavior. Listeners in JP, TR, IT, and FI are recommended a considerably bigger share of domestic tracks than the share of listening events they actually allocate to domestic music. Interestingly, the shares of US music recommended to users in these four countries is lower than the share of US music consumed by them. The average share of recommended US tracks (see top row and red vertical line) is very close to the share of attention US music actually receives. On average, ItemKNN shows less changes than NeuMF. These observations are in line with the results of [14] where ItemKNN shows most calibrated recommendation results in terms of track popularity, especially when compared to methods with a higher number of trainable parameters, such as ALS and Variational Autoencoders. Our experiment shows that music recommendation algorithms can considerably contribute to the process of globalization, and the exact contribution largely depends on the algorithm. Therefore, recommender system designers need to be aware of such potential impacts.

4. LIMITATIONS

While our analysis captures listeners' behavior beyond pop charts, it bears a number of limitations. First, the data we base our analysis upon reflects a particular audience: Last.fm users (likely active internet and social media users), and therefore might not be representative of the population at large. Some countries and social groups are underrepresented on Last.fm (e. g., female users). In some

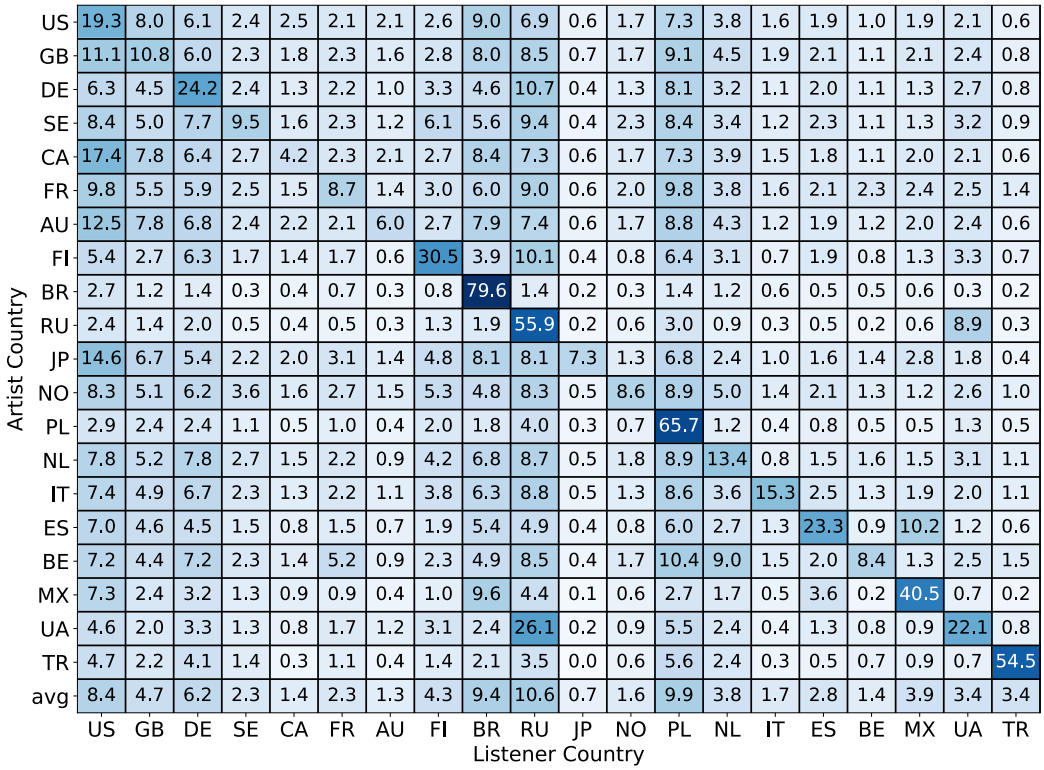


Figure 4: Music export matrix. Every cell shows the proportion of interactions with music from a given artist country allocated to users from a given listener country. For instance, 19.3% of all the music created by US artists is consumed by US listeners, while 14.6% of interactions with the entirety of Japanese music are made by US users.

countries Last.fm is not very popular and their listeners are likely to use different channels of music consumption. Therefore, our dataset may overrepresent listeners who are open to and interested in global culture, in particular in countries where Last.fm is not popular. Furthermore, the data gathered in the LFM-2b dataset are already affected by recommender systems of different platforms users connect to their Last.fm accounts, which means that our insights about "actual music preference" can be, to a certain degree, distorted.

5. CONCLUSION AND FUTURE WORK

We addressed three research questions related to patterns of music consumption on online music platforms. Regarding the dominance of US vs. domestic music (RQ1), we found that the former maintains strong positions in all considered countries. However, the US does not display signs of absolute and homogeneous "cultural imperialism". While it dominates its domestic market, in other countries, US music shows various levels of coexistence with domestic music. While some countries such as Australia and Canada find their domestic music competing with US music, others, such as Great Britain, Brazil, and Sweden, display high consumption levels of both US and domestic music. Countries like Finland and Germany, on the other hand, are open to music both from other countries and domestically created.

Looking for traces of glocalization (RQ2), we distinguish several ways domestic music can behave under the pressure of global cultural trends. Music from countries like Great Britain and Sweden shows signs of adaptation of global cultural trends, with their music coexisting with US music and being greatly listened to outside their domestic markets. Music from Brazil appears to be highly influenced by global trends but mostly consumed locally. Countries like Poland and Turkey also display signs of their music being localized and influenced by global trends, however, to a lesser extent influenced by US music than others. Finland combines competitive "export" of their music with high local consumption and relatively low consumption of US music, showing signs of a strong and distinctive musical culture.

Finally, we show that recommender systems may have a considerable impact on globalization patterns (RQ3). We investigate this for a traditional ItemKNN approach and the deep-learning-based NeuMF algorithm. While the former fosters consumption of local music in most of the considered countries, the latter supports internationalization and, in particular, cultural imperialism of the US.

Directions for future research include temporal analysis of consumption trends over the 15-year-span covered by the LFM-2b dataset, in particular considering possible alterations caused by the global pandemic [18, 19]. Other directions include exploring the link between globalization amplification and popularity bias, and investigating the effectiveness of different mitigation strategies.

6. ACKNOWLEDGMENTS

This research was funded in part, by the Austrian Science Fund (FWF) [P33526] and [DFH-23]; as well as by the State of Upper Austria and the Federal Ministry of Education, Science, and Research, through grant LIT-2020-9-SEE-113 and LIT-2021-YOU-215.

7. REFERENCES

[1] P. Achterberg, J. Heilbron, D. Houtman, and S. Aupers, "A cultural globalization of popular music? American, Dutch, French, and German popular music charts (1965 to 2006)," *American Behavioral Scientist*, vol. 55, no. 5, pp. 589–608, 2011.

[2] D. Crane, "Cultural globalization and the dominance of the American film industry: Cultural policies, national film industries, and transnational film," *International Journal of Cultural Policy*, vol. 20, no. 4, pp. 365–382, 2014.

[3] M. Castells, *The power of identity*. Malden, MA, USA: Blackwell, 1997.

[4] X. Chen and S. Shen, "Review of the impact of cultural imperialism in the context of globalization to the film industry," in *Proc. of the International Conference on Language, Art and Cultural Exchange*. Luoyang, China: Atlantis Press, 2021, pp. 205–212.

[5] P. Bello and D. Garcia, "Cultural divergence in popular music: the increasing diversity of music consumption on spotify across countries," *Humanities and Social Sciences Communications*, vol. 8, no. 1, pp. 1–8, 2021.

[6] H. Bekhuis, M. Lubbers, and W. Ultee, "A macrosociological study into the changes in the popularity of domestic, European, and American pop music in western countries," *European Sociological Review*, vol. 30, no. 2, pp. 180–193, 2014.

[7] M. Verboord and A. Brandellero, "The globalization of popular music, 1960-2010: A multilevel analysis of music flows," *Communication Research*, vol. 45, no. 4, pp. 603–627, 2016.

[8] C. Bauer and M. Schedl, "Global and country-specific mainstreaminess measures: Definitions, analysis, and usage for improving personalized music recommendation systems," *PLOS ONE*, vol. 14, no. 6, pp. 1–36, 2019.

[9] G. Vigliensoni and I. Fujinaga, "Automatic music recommendation systems: Do demographic, profiling, and contextual features improve their performance?" in *Proc. of the International Society for Music Information Retrieval Conference*. New York, NY, USA: ISMIR, 2016, pp. 94–100.

[10] D. Kowald, P. Müllner, E. Zangerle, C. Bauer, M. Schedl, and E. Lex, "Support the underground: Characteristics of beyond-mainstream music listeners," *EPJ Data Science*, vol. 10, no. 1, pp. 1–26, 2021.

[11] A. Ashokan and C. Haas, "Fairness metrics and bias mitigation strategies for rating predictions," *Information Processing & Management*, vol. 58, no. 5, pp. 1–18, 2021.

[12] J. Chen, H. Dong, X. Wang, F. Feng, M. Wang, and X. He, "Bias and debias in recommender system: a survey and future directions (2020)," *arXiv preprint arXiv:2010.03240*, 2020.

[13] H. Abdollahpouri, M. Mansoury, R. Burke, B. Mobasher, and E. C. Malthouse, "User-centered evaluation of popularity bias in recommender systems," in *Proc. of the Conference on User Modeling, Adaptation and Personalization*. Utrecht, the Netherlands: ACM, 2021, pp. 119–129.

[14] O. Lesota, A. Melchiorre, N. Rekabsaz, S. Brandl, D. Kowald, E. Lex, and M. Schedl, "Analyzing item popularity bias of music recommender systems: Are different genders equally affected?" in *Proc. of the Conference on Recommender Systems*. Amsterdam, the Netherlands: ACM, 2021, pp. 601–606.

[15] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," *ACM Transactions on Information Systems - TOIS*, vol. 22, pp. 143–177, 2004.

[16] X. He, L. Liao, and H. Zhang, "Neural collaborative filtering," *Proc. of the International Conference on World Wide Web*, pp. 173–182, 2017.

[17] M. Schedl, S. Brandl, O. Lesota, E. Parada-Cabaleiro, D. Penz, and N. Rekabsaz, "Lfm-2b: A dataset of enriched music listening events for recommender systems research and fairness analysis," in *Proc. of Conference on Human Information Interaction and Retrieval*. Regensburg, Germany: ACM SIGIR, 2022, pp. 337–341.

[18] N. C. Hansen, J. M. G. Treider, D. Swarbrick, J. S. Bamford, J. Wilson, and J. K. Vuoskoski, "A crowdsourced database of coronamusic: Documenting online making and sharing of music during the covid-19 pandemic," *Frontiers in Psychology*, vol. 12, pp. 1–9, 2021.

[19] M. Liu, E. Zangerle, X. Hu, A. Melchiorre, and M. Schedl, "Pandemics, music, and collective sentiment: Evidence from the outbreak of covid-19," in *Proc. of the International Society for Music Information Retrieval Conference*. Virtual: ISMIR, 2020, pp. 157–165.

NETWORK ANALYSES FOR CROSS-CULTURAL MUSIC POPULARITY

Kongmeng Liew[1] **Vipul Mishra**[1] **Yangyang Zhou**[1] **Elena V. Epure**[2]
Romain Hennequin[2] **Shoko Wakamiya**[1] **Eiji Aramaki**[1]
[1] Graduate School of Science and Technology, Nara Institute of Science and Technology, Japan
[2] Deezer Research, Paris, France
{liew.kongmeng, vipul-mi, zhou.yangyang.zr4, wakamiya, aramaki}@is.naist.jp
{eepure, rhennequin}@deezer.com

ABSTRACT

Anglo-American popular culture has been said to be intricately connected to global popular culture, both shaping and being shaped by popular trends worldwide, yet few research has examined this issue empirically. Our research quantitatively maps the extent of these cultural influences in popular music consumption, by using network analyses to explore cross-cultural popularity in music from 30 countries corresponding to 6 cultural regions (N = 4863 unique songs over six timepoints from 2019-2021). Using Top100 charts from these countries, we constructed a network based on the co-occurrence of songs in charts, and used eigencentrality as an indicator of cross-cultural song popularity. We then compared the country-of-origin of the artists, arousal music features, and socioeconomic indicators. Songs from artists with Anglo-American backgrounds tended to have higher eigencentrality overall, and mixed effects regressions showed that eigencentrality was negatively associated with danceability, and positively associated with spectral energy, and the migrant population of the country (of the charts). Next, using community detection, we observed 11 separate 'communities' in the network. Most communities appeared to be limited by region/culture, but Anglo-American music seemed disproportionally able to transcend cultural boundaries far beyond their geographical borders. We also discuss implications pertaining to cultural hegemony, and the effectiveness of our method in estimating cross-cultural popularity.

1. INTRODUCTION

How do songs become international hits? To some extent, research suggests that one answer is simply being based in America, and singing in English [1]. Numerous studies have documented the prevalence of Anglo-American artists on the charts of 'foreign' countries throughout the 1960s to the 2000s [2, 3]. Yet, at the same time, the 1980s saw a nationalistic increase in the consumption of domestic, and localized artists alongside widely dominant Anglo-American music [1, 4]. However, these papers rely on country-determined charts as indicators of music-popularity, which raises problems of irregularity. For example, the metrics/methods used to determine rankings on national top charts may differ between countries. Moreover, charts may also be defined through radio plays (e.g., Billboard), which biases the consumption pattern towards the distributor of the music, rather than bottom-up listening preferences. The advent of music streaming services in recent years provides a possible solution to both these issues: within a platform, charts are calculated through consistent metrics. Secondly, users choose the songs they want to listen to. While music recommendation systems do provide suggestions on songs for the user, these are still largely based on the users' preferences and listening history [5], and users still retain agency in choosing whether or not to listen to a recommended song. This is a shift away from the music-purchase model [6], allowing for greater flexibility and choice. Accordingly, streaming charts offer increased granularity, whereas music-purchase models only monitor sales. In this regard, country charts from music streaming services may be more representative of bottom-up music consumption.

In assessing the global top charts provided by streaming services, we are unable to decompose the cultural contribution of individual countries in determining overall global popularity, as such information is typically not made public. We thus needed a method to combine music from various country-charts to composite a marker of global popularity. To this end, we constructed a network from Top 100 charts from 30 countries (over 6 cultural regions), and relied on the co-occurence of songs within and across charts to determine its cross-cultural popularity through centrality. Constructing a network would also allow us to quantify the cultural diversity and clusters present across these 30 charts through community detection algorithms. More information on how networks are used in this paper is available in Section 2 (Methods).

Accordingly, our research aims to use network analyses to model the cultural 'Communities' present across these charts, and in doing so, assess the dynamics of cultural influences in music consumption around the world. We identified 11 Communities, most bounded by shared cultures,

© K. Liew, V. Mishra, Y. Zhou, E. V. Epure, R. Hennequin, S. Wakamiya, and E. Aramaki. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** K. Liew, V. Mishra, Y. Zhou, E. V. Epure, R. Hennequin, S. Wakamiya, and E. Aramaki, "Network Analyses for Cross-Cultural Music Popularity", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

with the exception of the largely Anglo-American Community, which was prevalent across charts from all cultural regions.

1.1 Network analyses

Network analyses model the complex relational structures present in a data, and have been used in a diverse range of applications, from web search engines [7], to modeling language evolution [8], and are used widely in the social sciences [9, 10]. They examine relationships between nodes in a network: nodes refer to any entity that is treated as fundamental in the analysis. For example, in a study on social networks, individuals are often treated as nodes. The relationships between nodes are represented by edges. edges may be weighted, to reflect the strength of a connection between two nodes. For example, two individuals (nodes) that frequently interact with each other, may have a higher weight for that relationship (edge), than between two strangers. Accordingly, some nodes may be more strongly connected with other nodes in a network. These nodes exert a greater influence on the network, and are more 'central': centrality refers to the relative importance of a node to all other nodes within the network [11].

In this paper, we rely strongly on the notion of eigencentrality, which additionally accounts for the corresponding centrality of linked nodes in determining the centrality of a given node [12]. For our analysis, we define nodes to be unique songs, and the edge weights as the frequency by which they co-occur on any country's Top 100 charts. Songs with high eigancentrality scores are more influential within the network, which means that they carry a greater presence across the various country-based Top 100 charts used in this analysis, which could be in turn indicative of greater cross-cultural popularity.

1.2 Contextualizing factors

For interpretation, we also compared the eigencentrality of songs on countries' charts with socioeconomic indicators, as well as song-level information on the country-of-origin or residence of its artist, and features of rhythmic and intensity arousal.

1.2.1 Artists' Country-of-residence and origin

While constructing a network of music charts would allow us to visualize and compare the influence of individual songs, we still needed to compare this information to the cultural background of these songs. As such, we obtained the country-of-residence (where possible) or country-of-origin of all artists that were represented on the charts. We then grouped these countries according to their cultural region to facilitate cultural comparison.

1.2.2 Gross Domestic Product (GDP), income inequality, and migration

Going beyond the network, we wanted to examine if economic development, social inequality, and migration could offer explanations for cross-cultural popularity. Past research has identified links between music preferences, and socialeconomic status (SES; [13]), such as social class [14]. At a country-level, Woolhouse and Bansal [15] found a direct relationship between economic development and music consumption patterns. Accordingly, we examined economic development through GDP per capita, and income inequality (Gini coefficient). Finally, we also examined migration statistics, as music has strong functions for identity formation in immigrants (see [16, 17]), who may bring with them differing music consumption patterns into the country they relocate to. Moreover, societies with high openness may be more accepting of immigrants and more open towards foreign cultures [18], and consequently, may be more receptive towards music from beyond their cultural region.

1.2.3 Arousal features in music

Rhythmic (danceability) and intensity (energy) arousal features of songs in national charts has been shown to reflect that country's affordances for high-arousal negative emotional experiences in daily life [19]. Energy, in particular, has been shown to reflect the use of cathartic emotional downregulation of anger experiences [20]. Given that past research has found links between music popularity and arousal features (e.g., [21, 22], we also examined eigencentrality in the context of musical arousal.

1.3 Related work: Network analyses in music research

Networks have long been used to analyze and visualize relationship structures in music. In predicting artist popularity, Matsumoto et al. [23] constructed a context-aware network combining Spotify-based audio features with biographic metadata and 'related artist' lists. South et al. [24] examined a dataset of musical collaborations on Spotify, and used eigencentrality to estimate the influence of artists on that dataset. Zinoviev [25] similarly examined the mobility of individual musicians amongst music groups in the Russian music industry. Finally, Ortega [21] examined music covers to estimate the influence of the original artist. While these projects largely used network analyses to estimate the popularity or influence of an artist or song, our research is unique in using chart co-ocurrence to derive cross-cultural popularity, through accounting for the relative popularity of songs from different regions around the world.

2. METHODS

2.1 Data Acquisition

Our analysis utilized consumption charts of Top 100 songs (by monthly play count on Deezer) from 30 countries (see Table 1: Chart Country) for March 2019, September 2019, March 2020, September 2020, March 2021, and September 2021, for a total $N_{songs} = 16998$ (unique $N_{songs} = 4863$ from $N_{artists} = 1001$).

For each song, we obtained the titles, artists, and ISRC codes from the Deezer Application Program Interface

Cultural Region	Chart Country	Country-of-Origin
African-Islamic	*UAE, Saudi Arabia, Turkey, South Africa	*Bahrain, *Morocco, Lebanon, *UAE, Saudi Arabia, Egypt, *Syria, South Africa, Iraq, Nigeria, Turkey, Algeria, Uzbekistan, Iran, Azerbaijan, *Kazakhstan, *Kuwait, Tunisia, Jordan, Palenstine, *Comoros, *Yemen, *Congo
Catholic-Europe	Croatia, Hungary, Poland, Austria, Belgium, Spain, France, Italy	France, Italy, Belgium, Spain, Austria, Croatia, Slovenia, Hungary, Poland, *Greece, Portugal
Confucian (Asia)	Not Applicable	South Korea, Japan
English-Speaking (Anglo-American)	Australia, Canada, *Ireland, United Kingdom, United States	United Kingdom, United States, Canada, Australia, *New Zealand, *Ireland
Latin America	Argentina, Brazil, Chile, Colombia, Guatemala, Mexico	Colombia, Philippines, *Puerto Rico, Argentina, Uruguay, Chile, *Panama, Venezuela, *Dominican Republic, Peru, *Jamaica, Brazil, Mexico, Guatemala
Orthodox-Europe	Russia, Ukraine	*Lithuania, *Kosovo, Bosnia and Herzegovina, Ukraine, Romania, Serbia, Bosnia, Belarus, Bulgaria, Russia, Armenia
Protestant-Europe	Switzerland, Germany, Denmark, Finland, Netherlands	Netherlands, Sweden, Norway, Germany, Switzerland, Denmark, Iceland, Finland, Estonia
West-South Asia	Not Applicable	Israel, Vietnam, Malaysia

Table 1. List of countries where charts were sampled from (Chart Country) and where sampled artists either resided in or originated from (Country-of-Origin). These were grouped according to Inglehart & Welzel's Cultural Map of the World (Cultural Region), following their macro-level orientation on Traditionalism and Secularity. Countries with '*' were not included in the original Cultural Map, but were categorized following the categories of their geographical neighbours.

(API) [1], and obtained artists' country of residence/origin from the MusicBrainz API [2]. Through this method, we managed to obtain the country or residence for artists from 2942 songs, and manually searched for the country of residence or origin (if country of residence was unavailable) for the artists of the remaining 1921 songs, through a combination of databases from Google, Wikipedia, LastFM, and Popnable. For analyses, we labelled this variable as Country-of-Origin. We also obtained danceability, based on detrended fluctuation analysis [26], and spectral energy scores for each song through the Essentia [27] library.

Country-level indicators of economic development were obtained through Gross Domestic Product (GDP) per capita, percentage of migrants, and income inequality (Gini) from the World Bank Databank [3]. We also categorized countries according to their respective Inglehart-Welzel cultural regions [4]: African-Islamic, Catholic-Europe, Confucian, English-Speaking, Latin-America, Orthodox-Europe, Protestant-Europe, and West-South Asia. If the country was not officially categorized in this system, we approximated the category based on the categorization of its geographical neighbours. Country categorizations are in Table 1, and data is available on our online repository [5].

2.2 Data Handling and Analysis

To construct the networks, we created undirected edge lists for each timepoint, consisting of nodes (song), weight: the number of times any two songs (or nodes) appeared on the same country's charts (denoted by W). In other words, for any given timepoint, we define a network G, where $G = (V, E)$. V refers to the set of unique songs in all charts, and E refers to the set of edges. W is the weighted adjacency matrix of the graph and is defined as follows for any pair of songs (i, j):

$$W_{ij} = \sum_{C_k \in C} Cooccur(C_k, i, j) \quad (1)$$

where $C = \{C_1, C_2, \ldots, C_K\}$ is the set of all considered charts and C_k is a particular chart (for a particular country and month), *i.e.* the set of 100 songs that were most listened in this country during this month. $Cooccur$ is an indicator function which takes the value 1 if and only if song i and song j co-occur in C_k, and 0 otherwise. To assess the centrality of a node, we examined the eigencentrality (x) of a node i, which is written in Equation 2 as:

$$x_i = \frac{1}{\Lambda} \sum_{j \in V} W_{ij} x_j \quad (2)$$

Here, Λ refers to the largest eigenvalue of the matrix W. Centrality of a node (song) x_i takes into account the sum of centralities of its neighbors, which was a reason why we used this measure (as opposed to degree centrality, which relies only on the number of connections). Eigencentrality parameters (e.g., number of iterations) relied on igraph defaults obtained from arpack [28].

For community detection, we used the modularity metric, which identifies nodes with statistically higher numbers of connections (edges) than random chance levels, and partitioning the interconnected nodes according to the boundaries of these 'above random' interconnections [29]. We used Clauset et al's fastgreedy function ('cluster_fast_greedy' [30]) that repeatedly combines lower-level communities to maximize modularity, for a bottom-up determination of the number and structure of communities. These analyses were conducted through the igraph package [31] in R [32].

For statistical analyses, we fitted separate χ^2 tests and ordinary least squares (OLS) or linear mixed effects (LME)

[1] https://developers.deezer.com/api
[2] https://musicbrainz.org
[3] https://databank.worldbank.org/source/world-development-indicators
[4] https://www.worldvaluessurvey.org/
[5] https://osf.io/uyh6d/

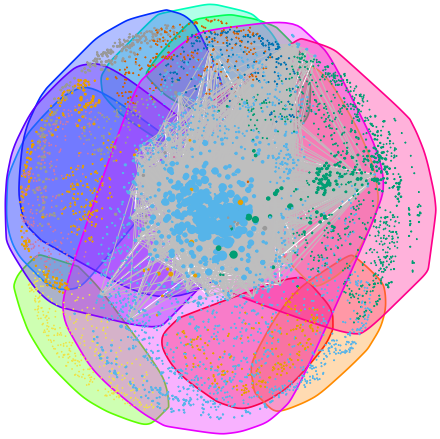


Figure 1. Network visualization using the Kamada-Kawai force-directed algorithm. Songs are represented by nodes, and node size corresponds to eigencentrality. Communities are displayed through colors. A version with node labels (titles and artists) is available on our online repository.

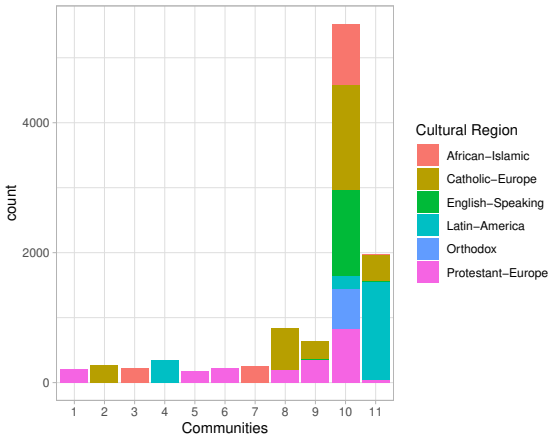


Figure 2. Visualizing culture-of-consumption: the distribution of charts that make up each Community, categorized by cultural region. Community 10 (C10) for example, comprises 5807 songs that are in charts from all 6 cultural regions sampled in this study.

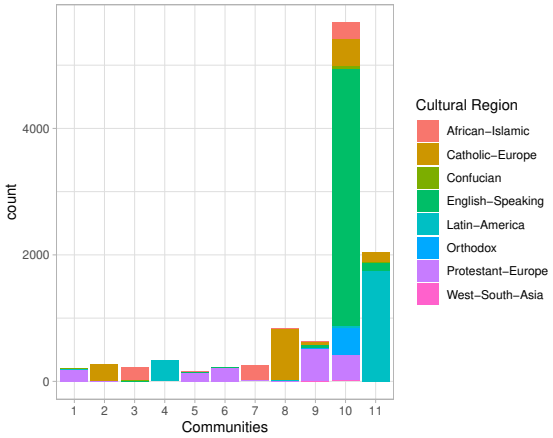


Figure 3. Visualizing culture-of-origin: the distribution of artists' countries for each Community, categorized by cultural region. Community 10 (C10) for example, comprises 5807 songs, of which 4707 are from the English-Speaking (Anglo-American) cultural region.

regression models [33] predicting node (song) eigencentrality from Community (OLS), cultural region (LME), economic development (LME), and danceability/spectral energy features (LME). For LME models, random intercepts were included for country-of-consumption (countries' charts), and random slopes were included for danceability and spectral energy by country where applicable. Degrees of freedom were estimated using Satterthwaite approximation. All categorical variables were deviation coded to examine the relative effect of a specific cultural region against the mean of all regions.

3. RESULTS

3.1 Descriptive statistics of detected Communities

A total of 11 Communities (C1-11) were detected from the network. Of which, Communities 10 (C10) and 11 (C11) had substantial amounts (5807 and 2056 songs respectively), but the other Communities ranged between sizes of 179 (C5) to 832 (C8). Due to space concerns we report only the results on cultural regions in this paper, but contingency tables and lists describing the country-of-consumption of the charts and country-of-origin of the artists' are available on our online repository [6]. Figure 1 displays a visualization of the network using a Kamada-Kawai layout [34].

We then tested the Communities' association with country-of-consumption (that the Top 100 chart was from) and country-of-origin of the artist. These were grouped according to their cultural region, described in Table 1. We observed a significant association between Communities and songs' culture-of-consumption, $\chi^2(50, 10681) = 15433, p < .001$, and between Communities and artists' culture-of-origin, $\chi^2(35, 10459) = 16253, p < .001$. Figures 2 and 3 visualize the cultural make-up by culture-of-consumption and culture-of-origin. For the most part, with the exception of C10, we find the relative consistency between culture-of-consumption and culture-of-origin, suggesting that the artists residing or originating from a specified cultural Community were largely listened to (popular) within the same cultural Community. For example, C1 comprised charts from the Netherlands, and a majority (87.6%) artists were of Dutch origin. Other Communities represented a linguistic cultural sphere over several countries: in C8, French-origin artists comprised 88.7% of the Community, which was largely split amongst Belgian (36.4%), French (39.3%), and Swiss (27.8%) Top100 charts. However, for C10, we observed that Anglo-American artists' formed the majority (70.1%), but consumption was spread out over all cultural regions and all 30 countries.

[6] https://osf.io/uyh6d/

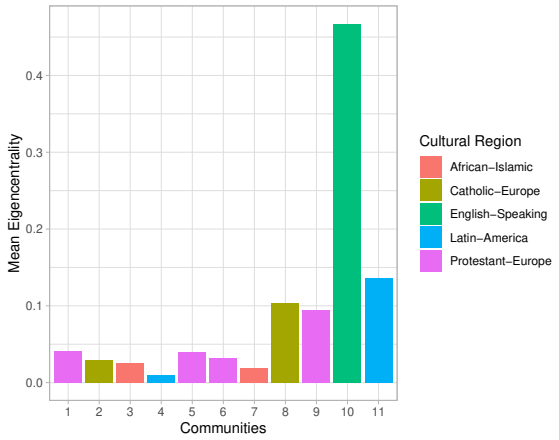


Figure 4. Mean eigencentrality of songs in each Community. Colors denote the most common culture-of-origin for artists in that Community.

3.2 Cross-cultural popularity (eigencentrality)

An OLS model ($R^2 = 0.36, F(10, 11033) = 614, p < .001$) with eigencentrality as the outcome variable, Communities as the predictor, and C1 as the reference level, found that, C2 ($\beta = -0.19, t = -4.15, p < .001$), C3 ($\beta = -0.20, t = -4.01, p < .001$), C4 ($\beta = -0.25, t = -6.10, p < .001$), C5 ($\beta = -0.16, t = -2.87, p = .004$), C6 ($\beta = -0.18, t = -3.69, p < .001$), and C7 ($\beta = -0.19, t = -4.15, p < .001$) showed significantly lower eigencentrality (than the average across Communities), and C10 ($\beta = 1.17, t = 72.08, p < .001$), and C11 ($\beta = 0.14, t = 6.92, p < .001$) showed significantly higher eigencentrality. These suggest that C2-7 had songs with cross-cultural popularity significantly lower than the Grand mean, and C10 and C11 had songs with cross-cultural popularity significantly above the Grand mean. These results are visualized in Fig 4.

Next, we examined the eigencentrality of songs by the culture-of-origin of the artist. As one artist could have several songs across various countries' charts, we used a LME model, with random intercepts for country-of-consumption (Model $R^2_{marginal} = 0.42, R^2_{conditional} = 0.44$). As countries' economic situation may have also influenced their consumption preferences, GDP per capita was added as a control variable. We observed a significant effect of culture-of-origin ($F(7, 6166.9) = 778.73, p < .001$), but not GDP per capita ($F(1, 27.9) = 0.90, p = .350$). With African-Islamic culture as a reference level, we found that songs originating from artists in Catholic-Europe ($b = -0.11, t = -9.31, p < .001$), Latin America ($b = -0.11, t = -8.43, p < .001$), and Orthodox Europe ($b = -0.06, t = -3.73, p < .001$) cultures had eigencentrality significantly below the Grand mean, songs originating from artists in Confucian Asia ($b = 0.12, t = 4.49, p < .001$) and English-Speaking (Anglo-American) cultures ($b = 0.34, t = 34.93, p < .001$) had eigencentrality significantly above the Grand mean. This suggests that songs from Confucian Asian and Anglo-American artists appeared to be more popular cross-culturally.

3.3 Additional analyses

We fitted two LME models exploring possible factors behind cross-culturally song popularity. These include socioeconomic reasons, such as GDP, income inequality, and immigration, and musical reasons, where we focus on danceability and energy as measures of rhythmic and intensity arousal inherent in the song. For the socioeconomic model (Model $R^2_{marginal} = 0.05, R^2_{conditional} = 0.10$), only immigration, measured using 2015 migrant percentages within a country's population, significantly predicted the eigencentrality of the songs on its charts ($b = 0.01, t = 3.83, F(1, 20.1) = 14.67, p = .001$). However, we noted the possibility that Anglo-American cultures, that typically have higher migrant percentages, also have artists with songs that have higher cross-cultural popularity. As such, we repeated the analysis with the exclusion of these countries (USA, UK, Australia, Canada). While still significant, migrant percentages predicted eigencentrality to a smaller extent (Model $R^2_{marginal} = 0.03, R^2_{conditional} = 0.07; b = 0.007, t = 2.21, F(1, 16.9) = 4.90, p = .041$, suggesting that countries with larger migrant populations also consume songs that are more cross-culturally popular.

While both are used as measures of arousal, danceability (rhythmic arousal) and spectral energy (intensity arousal) are only weakly correlated on a song-level ($r = 0.043, p, .001$). The LME model (Model $R^2_{marginal} = 0.02, R^2_{conditional} = 0.12$) showed a significant effect of danceability ($F(1, 30.4) = 12.4, p = .001$) and spectral energy ($F(1, 26.9) = 16.3, p < .001$). Eigencentrality was negatively predicted by danceability ($b = -0.121, t = -3.52, p = .001$), but positively predicted by spectral energy ($b = 2.68, t = 4.04, p < .001$). In sum, after controlling for country-specific effects, the cross-cultural popularity of a song was negatively associated with its danceability but positively associated with energy.

4. DISCUSSION

Our network appears to model the cultural spread of music consumption amongst the 30 countries studied. The 11 Communities detected largely reflect cultural consumption, which to some extent, seems to quantify linguistic boundaries as a common denominator across cultures. C8 for example, comprises French music, and is consumed in countries where French is widely spoken: Belgium, France, and Switzerland. Similarly, C9 comprises German artists, and is most consumed in Austria, Germany, and Switzerland. However, C10 appears unique in that that is largely comprised of songs by Anglo-American artists, yet is common in charts from countries where English is not primarily spoken. This appears to go beyond the nation or language-based consumption patterns of the other Communities. Given the high eigencentrality scores associated with songs from these cultures, our results may thus be indicative of a disproportional popularity of English-medium songs in global music consumption.

This interpretation would be consistent with Bello and Garcia's [4] research on nationalism and cultural diver-

gence in local charts in recent years. Our results show that many charts retain a large amount of songs by local or regional artists, and despite past research suggesting that American cultural hegemony in popular music might have decreased in recent years (see [1]), still contain strong Anglo-American music presences.

In contextualizing cross-cultural popularity in the socioeconomic conditions of a country, we found that migrant population, and not economic development (GDP) or income inequality (Gini coefficient), significantly predicted a country's consumption of cross-cultural popular songs. As many such artists are of Anglo-American origins, and Anglo-American countries typically have large migrant populations, we repeated this analysis after systematically excluding these countries. Despite a smaller observed effect, our findings remained consistent, in suggesting that migrant population is associated with wider consumption of cross-culturally popular songs. We speculate that a larger migrant population could indicate a more diverse population [35, 36]. If so, consumption patterns may be less subject to nationalistic tendencies, and the drive for local music in recent years may not have had as strong an effect in these Communities. One possibility could be that this creates a larger space and demand for music that is popular elsewhere, but caution that these interpretations are speculations, and more research is needed to uncover the antecedents of cross-cultural popularity.

We also examined cross-cultural popularity by analyzing song arousal. Following research by [22, 37] that found popular music comprised high intensity and strong rhythms, we show that intensity arousal appears to be more cross-culturally popular, but rhythmic arousal appeared inversely related to popularity. This suggests a conceptual differentiation between both arousal domains, and more research is needed to identify the distinct functions and qualities of both.

Regarding the high eigencentrality of artists from Confucian Asian cultures, we note however that this could be a result of sampling bias. Given that we did not examine charts from Confucian Asian cultures, the presence of artists with this cultural background would mean that they had to have a level of global visibility to appear in the charts we sampled. In our data, these mainly consisted of South Korean artists, like BTS and Blackpink. Nevertheless, that these artists had songs with larger-than-average eigencentrality, despite our sample not having any East Asian charts (where they would have had a natural advantage), is a testament to the success of the South Korean Hallyu movement. Yet, research has also suggested that this may be due to the hybridization of Korean popular music with American influences and trends [38].

One limitation of our research is in the narrow sample of countries included in the analysis. For validity, we chose only to focus on countries where Deezer occupies a sizeable market share, inevitably under-representing users in Asia. Secondly, some songs may be region-locked, in that users from Country A may not necessarily be able to listen to a song from Country B. However, we think that this affects only a minority of songs, and most songs appear to be accessible from different geographical regions; region specificity may not be a strong-enough limiter in preventing German artists (for example) from being played in the United States.

Next, we did not conceptually distinguish popularity from influence. Cross-cultural popularity (i.e., the consumption of foreign music) may not necessarily suggest cross-cultural influence (i.e., the pervasiveness of foreign aesthetics and values in local culture), and some of our arguments are built on the assumption that popularity reflects a certain amount of influence. A thorough distinction on mechanisms are beyond the scope of the current paper, but is an area that we feel is in need of more research. Finally, we had difficulties in differentiating country-of-origin from country-of-residence for the artists sampled. These issues were often technical, in that the databases we relied on did not necessarily specify these differences. Consequently, some issues on validity remain: an artists originating from Puerto Rico but residing in New York may be coded inconsistently. Nevertheless, we think our data is sufficient in supporting the broad claims on cross-cultural popularity made in this paper, but we defer to ethnomusicologists working in localized geographical areas for greater depth on the role of specific cultural influences within local societies.

5. CONCLUSION

Given that our aim was to empirically quantify the dynamics of cultural influence in music charts, we think that the findings demonstrate a limited usefulness of our method of using song co-occurence on countries' charts for network construction, to empirically examining the extent of a culture's influence. While we focused solely on music consumption and popularity, which at best can be considered just one aspect of influence, we nevertheless found convergence in our findings on Anglo-American musical influence, with evidence from sociological and anthropological literature: Anglo-American dominance remains unmatched in its prevalence around the world. Moving forward, we propose that our method of network construction can also be used to examine the dynamics of regional influences from within the other Communities. For example, C11 comprises songs mostly on Latin American charts, but shows that artists tend to be from Puerto Rico or Colombia, and these artists from these countries may thus have a stronger regional influence within this Community. Thus, even within these regional cultural spheres, artists may be concentrated within industries centered on a small number of countries, and this may be of interest to researchers. All supplementary materials are available on our online OSF repository [7], and we hope that this can be a resource for the music science community in researching the dynamics of cultural products and cross-cultural influence.

[7] https://osf.io/uyh6d/

6. REFERENCES

[1] P. Achterberg, J. Heilbron, D. Houtman, and S. Aupers, "A cultural globalization of popular music? american, dutch, french, and german popular music charts (1965 to 2006)," *American Behavioral Scientist*, vol. 55, no. 5, pp. 589–608, 2011. [Online]. Available: https://doi.org/10.1177/0002764211398081

[2] H. Bekhuis, M. Lubbers, and W. Ultee, "A Macro-Sociological Study into the Changes in the Popularity of Domestic, European, and American Pop Music in Western Countries," *European Sociological Review*, vol. 30, no. 2, pp. 180–193, 10 2013. [Online]. Available: https://doi.org/10.1093/esr/jct028

[3] M. Verboord and A. Brandellero, "The globalization of popular music, 1960-2010: A multilevel analysis of music flows," *Communication Research*, vol. 45, 01 2016.

[4] P. Bello and D. Garcia, "Cultural divergence in popular music: the increasing diversity of music consumption on spotify across countries," *Humanities and Social Sciences Communications*, vol. 8, pp. 1–8, 2021.

[5] M. Schedl, "Deep learning in music recommendation systems," *Frontiers in Applied Mathematics and Statistics*, vol. 5, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fams.2019.00044

[6] T. Wagner, M. Rose, C. Baccarella, and K.-I. Voigt, "Streaming killed the download star! how the business model of streaming services revolutionizes music distribution," *Journal of Organizational Advancement, Strategic and Institutional Studies*, vol. 8, pp. 29–39, 03 2015.

[7] P. Chen, H. Xie, S. Maslov, and S. Redner, "Finding scientific gems with google's PageRank algorithm," *Journal of Informetrics*, vol. 1, no. 1, pp. 8–15, jan 2007. [Online]. Available: https://doi.org/10.1016%2Fj.joi.2006.06.001

[8] J. C. Jackson, J. Watts, T. R. Henry, J.-M. List, R. Forkel, P. J. Mucha, S. J. Greenhill, R. D. Gray, and K. A. Lindquist, "Emotion semantics show both cultural variation and universal structure," *Science (New York, N.Y.)*, vol. 366, no. 6472, p. 1517—1522, December 2019. [Online]. Available: https://doi.org/10.1126/science.aaw8160

[9] N. Hummon and P. Doreian, "Computational methods for social network analysis," *Social Networks*, vol. 12, pp. 273–288, 12 1990.

[10] S. P. Borgatti, A. Mehra, D. J. Brass, and G. Labianca, "Network analysis in the social sciences," *Science*, vol. 323, no. 5916, pp. 892–895, 2009. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1165821

[11] D. Hevey, "Network analysis: a brief overview and tutorial," *Health Psychology and Behavioral Medicine*, vol. 6, no. 1, pp. 301–328, 2018, pMID: 34040834. [Online]. Available: https://doi.org/10.1080/21642850.2018.1521283

[12] M. E. J. Newman, *Mathematics of Networks*. London: Palgrave Macmillan UK, 2016, pp. 1–8. [Online]. Available: https://doi.org/10.1057/978-1-349-95121-5_2565-1

[13] M. Liu, X. Hu, and M. Schedl, "The relation of culture, socio-economics, and friendship to music preferences: A large-scale, cross-country study," *PLOS ONE*, vol. 13, no. 12, pp. 1–29, 12 2018. [Online]. Available: https://doi.org/10.1371/journal.pone.0208186

[14] K. van Eijck, "Social Differentiation in Musical Taste Patterns*," *Social Forces*, vol. 79, no. 3, pp. 1163–1185, 03 2001. [Online]. Available: https://doi.org/10.1353/sof.2001.0017

[15] M. H. Woolhouse and J. Bansal, "Work, rest and (press) play: Music consumption as an indicator of human economic development," *Journal of Interdisciplinary Music Studies*, vol. 7, pp. 45–71, 2013.

[16] E. Fock, "With the background in the foreground - music among young danes with immigrant backgrounds," *YOUNG*, vol. 7, no. 2, pp. 62–77, 1999. [Online]. Available: https://doi.org/10.1177/110330889900700205

[17] J. Louie, "Media in the lives of immigrant youth," *New directions for youth development*, vol. 2003, pp. 111–30, 02 2003.

[18] P. Dinesen, R. Klemmensen, and A. Nørgaard, "Attitudes toward immigration: The role of personal predispositions," *Political Psychology*, vol. 37, 06 2014.

[19] K. Liew, A. H. Q. Koh, N. R. Fram, C. M. Brown, C. dela Cruz, L. L. Neng, R. Hennequin, A. E. Krause, and Y. Uchida, "Groovin' to the cultural beat: Preferences for danceable music represent cultural affordances for high-arousal negative emotions," Dec 2020. [Online]. Available: psyarxiv.com/eqd8t

[20] K. Liew, H. Domae, A. H. Q. Koh, and Y. Uchida, "Energetic music is used for anger downregulation: A cross-cultural comparison," In review.

[21] J. L. Ortega, "Cover versions as an impact indicator in popular music: A quantitative network analysis," *PLOS ONE*, vol. 16, no. 4, pp. 1–18, 04 2021. [Online]. Available: https://doi.org/10.1371/journal.pone.0250212

[22] H.-M. Wang and S.-C. Huang, "Musical rhythms affect heart rate variability: Algorithm and models," *Advances in Electrical Engineering*, vol. 2014, 09 2014.

[23] Y. Matsumoto, R. Harakawa, T. Ogawa, and M. Haseyama, "Context-aware network analysis of music streaming services for popularity estimation of artists," *IEEE Access*, vol. 8, pp. 48 673–48 685, 2020.

[24] T. South, M. Roughan, and L. Mitchell, "Popularity and centrality in Spotify networks: critical transitions in eigenvector centrality," *Journal of Complex Networks*, vol. 8, no. 6, 03 2021, cnaa050. [Online]. Available: https://doi.org/10.1093/comnet/cnaa050

[25] D. Zinoviev, "Networks of music groups as success predictors," *CoRR*, vol. abs/1709.08995, 2017. [Online]. Available: http://arxiv.org/abs/1709.08995

[26] P. Herrera and S. Streich, "Detrended fluctuation analysis of music signals: Danceability estimation and further semantic characterization," in *Audio Engineering Society Convention 118*, May 2005. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=13125

[27] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. Zapata, and X. Serra, "Essentia: An open-source library for sound and music analysis," in *Proceedings of the 21st ACM International Conference on Multimedia*, ser. MM '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 855–858. [Online]. Available: https://doi.org/10.1145/2502081.2502229

[28] *1. Introduction to ARPACK*, pp. 1–7. [Online]. Available: https://epubs.siam.org/doi/abs/10.1137/1.9780898719628.ch1

[29] M. E. J. Newman, "Modularity and community structure in networks," *Proceedings of the National Academy of Sciences*, vol. 103, no. 23, pp. 8577–8582, 2006. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.0601602103

[30] A. Clauset, M. E. J. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, p. 066111, Dec 2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.70.066111

[31] G. Csardi and T. Nepusz, "The igraph software package for complex network research," *InterJournal*, vol. Complex Systems, p. 1695, 2006. [Online]. Available: https://igraph.org

[32] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2020. [Online]. Available: https://www.R-project.org/

[33] D. Bates, M. Mächler, B. Bolker, and S. Walker, "Fitting linear mixed-effects models using lme4," *Journal of Statistical Software*, vol. 67, no. 1, pp. 1–48, 2015.

[34] T. Kamada, S. Kawai *et al.*, "An algorithm for drawing general undirected graphs," *Information processing letters*, vol. 31, no. 1, pp. 7–15, 1989.

[35] P. Collier, *Exodus: How migration is changing our world*. Oxford University Press, 2013.

[36] V. Bove and L. Elia, "Migration, diversity, and economic growth," *World Development*, vol. 89, pp. 227–239, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305750X16304466

[37] R. T. Dean, F. Bailes, and E. Schubert, "Acoustic intensity causes perceived changes in arousal levels in music: An experimental investigation," *PLOS ONE*, vol. 6, no. 4, pp. 1–8, 04 2011. [Online]. Available: https://doi.org/10.1371/journal.pone.0018591

[38] G. Kim, "Korean wavel between hybridity and hegemony in k-pop's global popularity: A case of ğirls' generation'sämerican debut," *International Journal of Communication*, vol. 11, no. 0, 2017. [Online]. Available: https://ijoc.org/index.php/ijoc/article/view/6306

THREE RELATED CORPORA IN MIDDLE BYZANTINE MUSIC NOTATION AND A PRELIMINARY COMPARATIVE ANALYSIS

Polykarpos Polykarpidis[1] **Dionysios Kalofonos**[2] **Dimitrios Balageorgos**[1]
Christina Anagnostopoulou[1]
[1] Department of Music Studies, National and Kapodistrian University of Athens, Greece
[2] Independent researcher, Manchester, UK
polpol@music.uoa.gr, peitemou@gmail.com, dbalageorgos@music.uoa.gr, chrisa@music.uoa.gr

ABSTRACT

The Middle Byzantine notation (MBn) is used to capture the plainchant melodies of eastern Orthodox Christian music from the middle of the 12th century until 1814. In the context of this research, we study the evolution of a subgenre of Byzantine music known as Heirmologic. We present three Heirmologic corpora spanning the periods before, during and after the 16th century. We discuss the challenges we faced during the digitisation process, and the steps we took to overcome them. For the analysis of the three corpora, we apply the three methods, namely notational texture, melodic arch similarity, and Jensen-Shannon distances of Markovian models, the second of which is novel and inspired by the idea of melodic arches [1, 2]. Through these methods, we aim at highlighting the differences of the corpora in order to obtain an outline of the evolution of the subgenre. We observe that the post-16th century Heirmologic pieces are more similar to the 16th century ones, while there is a greater difference with the pre-16th century pieces. This indicates that the 16th century constitutes a turning point in the melodic features of the Heirmologic subgenre.

1. INTRODUCTION

Byzantine music has a known history of 2000 years and it constitutes a part of the human world heritage [3]. Roughly, the first millennium is characterised as pre-notational period, while the second as notational period [4]. It was mainly developed in the Eastern Byzantine Empire and up to this day has been influencing the cultures of the Southeast and Eastern Europe, and parts of the Caucasus regions. It has also influenced, among others, the Western and the Slavic music [5, 6].

The Middle Byzantine notation (MBn) is used to capture the plainchant melodies of Byzantine music from the middle of the 12th century until 1814 [7]. At the moment of writing, the cataloged manuscripts of Byzantine music are numbered approximately 10,000, most of them written in MBn [8]. The absence of MBn corpora prevents the computational process of this music genre and makes impossible any cross-cultural research, what is described in [9, 10]. Previous attempts have been made to transcribe Byzantine music into staff notation, but they were rejected due to loss of the information that is rooted in the symbols' orthography [7].

We present three related corpora of MBn scores which are part of the Knowledge Representation presented in [11]. The corpora are available to MBn researchers as well as to the wider musicological community [12]. These corpora are used in a study to outline the evolution of the Heirmologic subgenre of Byzantine music that took place in the 16th century. For this reason, we present a preliminary comparative analysis of the corpora.

This paper is organised as follows. Section 2 introduces the Heirmologic subgenre that the pieces of the corpora belong to, the manuscripts that the corpora consist of, and the reason why we study them. It closes with the discussion on the challenges in the digitisation and how we dealt with them. Section 3 presents the methods we applied for the musicological analysis of the corpora. Section 4 presents the results of the analysis. This paper closes with section 5 which discusses our results.

2. THE THREE CORPORA

Byzantine music consists of three subgenres: Heirmologic, Sticheraric, and Papadic. The subgenre of a Byzantine music piece depends on the poetic form of its lyrics [1]. Generally, Byzantine music compositions of a hymn are influenced by (a) the subgenre that the hymn belongs and (b) the preceding compositions of the same hymn (different melodies using the same lyrics). Regarding point (b), Manuel Chrysaphes (15th century) in his treatise [14, pp. 44–47], states that new compositions of Hymns follow the music of the preceding compositions of these Hymns. Since music can be expressed as a series of viewpoints,

© P. Polykarpidis, D. Kalofonos, D. Balageorgos, and C. Anagnostopoulou. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** P. Polykarpidis, D. Kalofonos, D. Balageorgos, and C. Anagnostopoulou, "Three related corpora in Middle Byzantine music notation and a preliminary comparative analysis", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] Specifically, the Heirmologic music pieces (called *Heirmoi*) use lyrics that belong to the poetic form of *Kanon* [7, 13]. At a high level, a *Kanon* consists of a number of stanzas, some of which work as model melodies. These stanzas are called *Heirmoi* (singular: *Heirmos*). The remaining stanzas of *Kanon* follow the melodies of the *Heirmoi* [7]. The book that contains the Heirmoi is called Heirmologion.

we explore this observation and we model the similarity of the corpora through the distances of their respective viewpoints.

In the context of this research, we study the evolution of the Heirmologic subgenre. Using the Knowledge Representation (see section 2.5), we focus our attention on the Heirmologions of 16th century, and specifically the Karykis Heirmologion. This selection is not random. According to the existing historical research, in the 16th century a great change happened to the melodies of the Heirmologion [13, 15, 16]. In order to study the Karykis' Heirmologion and to evaluate its contribution to the tradition, we select the Heirmologions of pre-Karykis, Karykis and post-Karykis.

2.1 A representative sample of Heirmologion of pre-Karykis era

In our research we evaluate the contribution of Karykes' Heirmologion in the context of tradition. As a representative example of the previous tradition, the Heirmologion of manuscript 1101 from the Iveron (mount Athos) library was chosen. This Heirmologion was written somewhere between 1535-1540 by the monk Pachomius Rousanos [17].

There are several reasons for choosing Rousanos' Heirmologion as a representative of the pre-Karykis era: (a) the Rousanos' Heirmologion follows the prevailing tradition of the subgenre, (b) it is dated close to Karykis' era, (c) Rousanos was a well-known lettered monk, who had a good knowledge of both notation and theory, so his manuscripts are considered reliable sources.

2.2 A representative sample of Karykis' Heirmologion

So far, no autograph of Karykis' Heirmologion has been identified. From various copies of the Karykis' Heirmologion, the Iveron 1167 was chosen for these analyses. This manuscript dates to the early 17th century.

2.3 A representative sample of Heirmologion of post-Karykis era

As a representative example of the post-Karykis tradition, we take the Balasis' Heirmologion of Sinai 1433 (Jerusalem). This Heirmologion was written in 1690 by Kosma the Macedonian [18]. In the period between the Balasis' and Karykis' Heirmologions, appeared also some other Heirmologion anthologies. However, based on the number of copies, Balasis' Heirmologion was especially popular among the post-Karykis' Heirmologions. Therefore, the next main tradition of the Heirmologion subgenre is considered to be that of Balasis. Regarding the choice of the copy, Kosmas the Macedonian was a peer of Balasis in music studies and his manuscripts are considered to be a reliable source [19].

2.4 Corpora

Having defined the previous and the next tradition of the Karykis' Heirmologion, it becomes possible to create three corpora that represent these traditions. Specifically, 128 Heirmoi were received from each Heirmologion. The sample is evenly distributed in terms of Echoi (modes): 16 Heirmoi per Echos. The same Heirmoi were taken from each corpus, i.e., the same text (lyrics) composed by different composers. This sample is approximately 25% of the number of Heirmoi contained in each Heirmologion.

2.5 Knowledge Representation

The Knowledge Representation we use is a tree structure capturing the viewpoints [20, 21] of the music piece. In our case, these viewpoints are the *syllable*, *interval*, *pitch*, and *voiced sign*. The *syllable* is the viewpoint that captures in a sequential form the syllables of the lyrics. The *interval* and *pitch* viewpoints capture the information that provides us with the Metrophonia [2] of the piece (i.e., basic melodic line without duration information). The *voiced sign* viewpoint captures the aspect of the notation that provides us with the Metrophonia. Naturally the tree captures the relations of the viewpoints and their sequence in the music piece. Exploiting the structure of the tree, the researcher can then perform a computational analysis of the properties of the music piece. Moreover, the researcher can post-process the tree to highlight specific properties of the music piece by reorganising the connections of the viewpoints. For more information see [11].

2.6 Digitisation challenges

During the process of digitisation of the music pieces, we were faced with a number of challenges. Firstly, notational errors in the manuscripts due to the scribe is a common issue. Such errors are identified through the Metrophonia of the notation and are corrected by comparing these excerpts containing the error with similar excerpts from other manuscripts. For the pre-Karykis Heirmologion (manuscript Iveron 1101), the following manuscripts were used for the corrections: Grottaferrata EgII [22], Sinai 1256 [23] and Iveron 1185 [24]; for Karykis Heirmologion (manuscript Iveron 1167) manuscripts Iveron 1154, Iveron 1155 and Iveron 1231 were used [24]; and for post-Karykis Heirmologion (manuscript Sinai 1433) manuscripts NLG 946, NLG 936 and NLG 967 were used [25]. All the imposed corrections are documented within the corpus (GitHub link).

Secondly, the signs of MBn are grouped into two main categories: Voiced signs and Voiceless signs. One of the uses of the latter is to group the former. Many times it does not seem clear which Voiced signs the Voiceless signs are grouping (e.g., Figure 1). This ambiguity particularly concerns the Heirmologions of Karykis' era and onwards. This problem does not have a unique solution and individual rules may apply. A general principle that we followed in our corpora was that Voiceless signs group all the Voiced signs in a syllable. Also, sometimes we juxtaposed the am-

[2] Although the interpretation of this notation remains an open question, the basic melodic line as evidenced by the signs (called Metrophonia) is unquestionable.

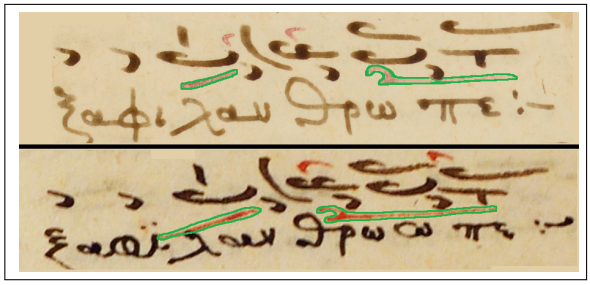


Figure 1. The top excerpt is from the Heirmologion of the manuscript Iveron 1167 f 4r, while the bottom is the same excerpt from the Heimrologion of the manuscript Iveron 1155 f 4r. As can be seen, the same Voiceless sign (red with a green border) spans several Voiced signs (black) in the two manuscripts.

biguous excerpts from corresponding manuscripts to locate a kind of pattern.

Thirdly, at a higher level, the MBn can be understood as a sequence of signs that indicate intervals. The pitches of these intervals are determined by a series of specific signs which are always placed at the beginning of the pieces. This group of signs is called Martyriai. Choosing a pitch as representative of each Martyria is not always straightforward [26]. The choices made in the specific music pieces are such as to facilitate the comparison of the three corpora.

Fourthly, our Knowledge Representation gives us the ability to capture the melodic phrases of each piece. The choice of end points of music phrases is not objective. In order to be consistent with our choices across the corpora, we have created a set of general rules and separation preferences.

Rule 1.1: Phrases are separated in the last syllable of a word. **Rule 1.2:** Phrases are separated in the last syllable of an *Enclisis* group (linguistic term). **Rule 2.1:** The last syllable of the phrase usually contains at least one of the following signs [3] : *Diple Apostrophos*, *Diple*, *Apoderma*, *Kratema* (rarer), *Klasma* (rarer). **Rule 2.2:** The penultimate syllable of the phrase can have the *Koufisma* sign. **Rule 2.3 (intuitive):** The last syllable of the phrase may have one of the following compound signs: *Omalon*, *Piasma*, and *Vareia*. **Rule 3:** There should be a colon and/or a comma in the lyrics of the music text. **Rule 4 (tendency):** Small phrases rather than large ones are preferred. **Rule 5:** The *Martyriai* appearing mid-text usually separate phrases.

3. METHODS OF COMPARISON

In order to draw an outline of the evolution of the Heirmologic subgenre of Byzantine music, we use three methods to capture the differences of some aspects of the corpora: Notational texture (method 1), similarity of the melodic arches (method 2), and distances of Markovian models of the attributes (method 3).

[3] These are the names of the signs of MBn that etymologically come from Greek words [7].

3.1 Method 1: notational texture.

In plainchant music (e.g., Byzantine, Gregorian, Mozarabic etc.) a characteristic of the style of a piece is its average number of notes, time beats, or signs per syllable [27–33]. Based on this characteristic, we can cluster the music pieces into three categories: syllabic, neumatic, and melismatic. In order to eliminate any ambiguities in these categories, we define two types of texture: Notational and Durational texture. *Notational texture* of a corpus is defined as the average number of pitches or of signs on a syllable, while *Durational texture* of a corpus is defined as the average number of durations per syllable. When the average number of the measured quantity (pitch, duration, signs etc.) of a corpus is between 1 and 2, then the texture of the corpus is characterized syllabic, when it is between 2 and 4 then the texture of the corpus is characterized neumatic, and when it is 4 or more then the texture of the corpus is characterized melismatic. Hence, a music corpus in terms of pitch can belong to syllabic style, while in terms of duration can belong to neumatic style. The interpretation of the MBn in terms of the duration of the syllables remains an open question. Consecutively, as the music pieces considered in this research are written in MBn, our measurements cover the notational texture.

All the pieces of our corpora belong to Heirmologic subgenre. This means that the general notational texture of these three corpora is syllabic. Nevertheless, through this measurement we can obtain a more subtle distinction of the notational texture of the syllables as examined by Troelsgard [30]. Specifically, we measure the number of pitches per syllable, $notational\_texture = \frac{|voiced\_units|}{|syllables|}$.

3.2 Method 2: similarity of the melodic arches

As we discussed in section 2, tradition plays an influential role in the composition of a hymn. In this research, we are interested in (a) identifying the areas where the new compositions follow the tradition, (b) the areas where the new compositions diverge from the tradition, and (c) quantify the divergence of the new and prior compositions.

Plotting the melodic lines of the different compositions of the same poetic verses lined up by syllable, we observe that the compositions usually present similar melodic shapes in corresponding areas (Figure 2). Based on this observation and inspired by the idea of melodic arches [1, 2], we create a novel method that, given two melodies of the same number of syllables, returns a percentage of similarity of their melodic arches. The design of this algorithm is driven by (a) the fact that the compositions in Byzantine Music are rooted on the syllables of the lyrics, (b) the duration of the pitches is unknown, and (c) the need for the derivation of a comparison metric instead of a metric for the identification of a dominant melodic arches [1].

The algorithm consists of three phases, (a) the plotting of the pitches (Algorithm 1), (b) the extraction of the arches (Algorithm 2), and (c) the comparison of the arches of the melodic lines (Algorithm 3).

Given two music pieces, the algorithm extracts the pitches, and plots them on a two dimensional grid (Algo-

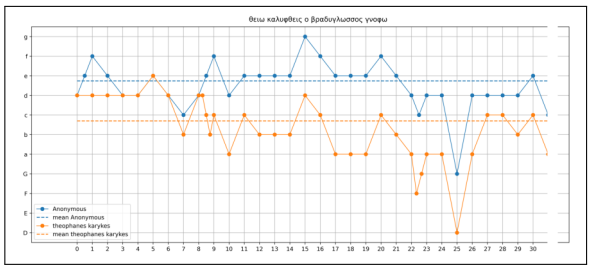


Figure 2. The melodic line of the first 30 syllables of Heirmos 'Theio kalyptheis' by the two compositions (Iveron 1101 and Iveron 1167). The x-axis represents the syllables, where at x=0, the value of y is the Martyria, i.e., the initial pitch (see [11]). The y-axis represents the steps (pitches).

rithm 1). The two pieces are aligned by syllables, and are of equal length. As a result, the pitches give us a view of the shape of the melodic lines, and highlights the similarities and differences of the compositions (Figure 2).

In the next phase the algorithm extracts the arches that are formed by the pitches (Algorithm 2), i.e. shape of the melody. Using these arches, we translate the two melodies into sequences of two types of arches: convex and concave.

In the final phase, the algorithm compares the compositions through their arches (Algorithm 3). When two areas are translated into the same arch type, they are considered to be similar. When two areas are translated into different arch types, they are considered to be dissimilar. When the areas cannot be translated into arches, they are classified as dissimilar due to different shapes.

When the arch of one melodic line corresponds to more than one arches of the other, the former is compared with the second degree polynomial regression of the latter. The second degree polynomial will give us a curve that is either convex or concave. Through the comparison of the two melodic lines, we obtain (a) the degree of similarity, (b) the degree of dissimilarity due to difference arches, and (c) the degree of dissimilarity due to difference shapes (Figure 3).

3.3 Method 3: distances of Markovian models of the attributes

As a third method, we capture the viewpoints as sequences (n-grams). We apply Markovian models on the n-grams to examine the divergence of the corpora. The melodic features of the notation are those that refer directly to the basic melodic line of the chant: pitch, interval, voiced unit. The information carried by the first two attributes is identical to that of the western staff. Voiced units are those that encapsulate the music information of the main melodic line as it is imprinted on the other two features (pitch, interval). In essence, the voiced unit attribute concerns the sign itself as it is imprinted in the score, while the pitch and the interval concern the aspects of the melody that the voiced units indirectly indicate (signified and signifier).

From the pitch and interval we also extract three others features which are redundancies of the melodic line: general syllabic pitch, general syllabic interval, and melodic

Algorithm 1 Pitch plotting algorithm.

1: Let $syllables$ be the sequence of syllables of a music piece.
2: Let $pitches$ be the sequence of pitches of a music piece.
3: Let $f^{assoc} : syllable_k \rightarrow pitches_k$ where $pitches_k$ is the sequence of pitches associated with $syllable_k$ and $pitches_{n_k}$ is the nth pitch of $pitches_k$.
4: Let $f^{pos} : syllable_{i=1}^{n} \rightarrow \mathbb{Z}^+$ be the mapping of the position of a syllable i in the sequence of the music piece to positive integers, this mapping defines the values of the x-axis.
5: $\forall n \in [\ 1,\ |pitches_k|\ ]$ where $pitches_k = f^{assoc}(syllable_k)$, we plot the points, $P = \{\ (x, y)\ :\ x = f^{pos}(syllable_{k-1}) + n/|pitches_k|,\ y = pitches_{n_k}\ \}$. Point (x_0, y_0) is reserved for the initial pitch given by a special sign known as Martyria.
6: Create an abstraction by removing the embellishments, in the context of syllable, from the melodic line, defined as $\mathbb{P}^- = \{(x, y) : (x, y) \in P, x \in \mathbb{Z}^+\}$, i.e. for every syllable keep the last pitch.
7: For each pair of points in $\mathbb{P}^-$, we identify the melodic contour (i.e., step up or down). Zero intervals inherit the melodic contour of the previous pair, hence they do not change the melodic contour.

contour of syllable. For the general syllabic pitch, instead of extracting every pitch from every sign of the syllable, we extract only the last pitch of the syllable. For the general syllabic interval, instead of extracting every interval from every sign of the syllable, we extract the sum of all intervals of the syllable. For the melodic contour of the syllable, we extract the values $(-, +, 0)$, which describe the contour of the general syllabic interval.

Using our Knowledge Representation [11], the corpora is translated into sequences of values based on the logic of viewpoints [20, 21]. For the aforementioned viewpoints, we train Markov models of different order, and we measure the distance of the models using the Jensen-Shannon distance (i.e., the square root of the Jensen-Shannon divergence) [34, 35]. To tackle the zero-probability problem we use Laplace smoothing [36].

Markov models express the probability of an n-gram appearing in the corpus. However, Markov models do not capture the position of the n-gram in the corpus and since we are restricted to low order Markov models due to overfitting, two different corpora can theoretically result in the same probability distribution. As we are using this method to find the relation of two corpora, we apply it both on the detailed dataset that reflects the music surface and on a simplified dataset that has a single value associated with each syllable. Since the corpora contain the same lyrics and in the simplified dataset every syllable has one value (one pitch, one interval, or one contour), the n-grams span the same regions for both melody and lyrics.

Furthermore, since the divergence is derived from the

Algorithm 2 Arch extraction algorithm.

1: Let $\alpha = (start, peak, end, shape)$ be an arch, where $start,\ peak,\ end \in \mathbb{P}^-$ and $shape \in \{CONVEX, CONCAVE\}$.
2: Let point x be the starting point, then $start^\alpha = x$.
3: Find x_i such that all pairs in the range $[x, x_i]$ have the same contour or zero intervals, then $peak^\alpha = x_i$.
4: Find x_j such that all pairs in the range $[x_i, x_j]$ have the same contour or zero intervals, then $end^\alpha = x_j$.
5: Set $shape^\alpha = CONCAVE$ when $start^\alpha < peak^\alpha$, otherwise $shape^\alpha = CONVEX$.
{Since zero intervals do not change the contour, the arch will include zero intervals.}

Markov tables, and since we are using it for comparison, we make sure that the Markov tables of the corpora under comparison consist of a common set of n-grams, spanning the union of the set of symbols found in the corpora under comparison. As such, the divergence reflects the distance of the probability distributions over a single set of n-grams.

We study n-grams between the orders of 2 and 4, and favour higher orders until the point where all lines show decrease due to over-fitting. As we are interested in the comparison of the corpora, higher orders with the same probability indicate areas of similarity between the corpora. However, with higher order Markov, we tend towards the scenario where an n-gram appears once in one corpus and not the other, the Markov table is dominated by noise due to smoothing, and Jensen-Shannon distance reflects the noise rather than the relation of the data.

Finally, the absolute values of the Jensen-Shannon distance are not meaningful without a reference. As we are studying the distances of corpora, we need a reference in order to quantify the magnitude of their distance. For this reason, we make use of a case study corpus consisting of 16 Heirmoi in first echos which have the property that the 8 pre-Karykis [4] and 8 Karykis [5] Heirmoi belong to the pre-Karykis' music style. Generally, Karykis changes the compositional style in first echos, but for some reason, he keeps the previous style (pre-Karykis) for only these 8 Heirmoi. This is confirmed by the similar cadential pitches and pitch profiles, the highest melodic arch similarity (similar arches 84.4%, dissimilar arches 12.9% and dissimilar shapes 2.7%), and low Jensen-shannon distance (gray line in Figures 4 & 5). For these reasons, we consider the distance of these case study corpora as a threshold for similarity.

4. RESULTS

Table 1 presents the notational textures (i.e. the number of voiced units per syllable) of the three corpora. The three corpora show a syllabic notational texture. We observe that Karykis tends even closer to compose Heirmos with absolute syllabic notational texture (i.e., near to 1).

[4] Manuscript Iveron 1101, folios 7v-9r
[5] Manuscript Iveron 1167, folios 138r-141v

Algorithm 3 Arch comparison algorithm.

1: Let A and A' be the sequences of arches of the two melodic lines under comparison.
2: **while** $A \neq \emptyset$ and $A' \neq \emptyset$ **do**
3: Let α_1 be the first arch in A, and α'_1 be the first arch in A'.
4: Let $L = \emptyset,\ R = \emptyset$.
5: **if** $start^{\alpha_1} < start^{\alpha'_1}$ **then**
6: $L = \{\alpha_1\}$.
7: $R = \{\alpha'_n \in A' : end^{\alpha'_n} \leq end^{\alpha_1}\}$.
8: **else if** $start^{\alpha_1} > start^{\alpha'_1}$ **then**
9: $L = \{\alpha'_1\}$.
10: $R = \{\alpha_n \in A : end^{\alpha_n} \leq end^{\alpha'_1}\}$.
11: **else**
12: $L = \{\alpha \in A : end^\alpha \leq \max_{\alpha'' \in \{\alpha_1, \alpha'_1\}} end^{\alpha''}\}$.
13:
14: $R = \{\alpha' \in A' : end^{a'} \leq \max_{\alpha'' \in \{\alpha_1, \alpha'_1\}} end^{\alpha''}\}$.
15: **end if**
16: Let $area^L = \sum_{i=0}^{n} end^{\alpha_i} - start^{\alpha_i}$ where $\alpha \in L$. Equally we define $area^R$.
17: **if** $area^L \neq area^R$ **then**
18: Expand smallest area by adding padding, i.e. zero intervals.
19: **end if**
20: Let $quad(area^L)$ and $quad(area^R)$ be the two quadratic regressions using the points of the arches and the padding included in these areas. The two areas are considered similar if the curves have the same shape, i.e., both curves being convex or concave.
21: Remove from A and A' the arches in L and R.
22: **end while**

corpora	pre-karykis	karykis	balasis
notational texture	1.22	1.13	1.23

Table 1. The notational texture of the corpora.

Figure 3 shows the results of the similarity of the melodic arches. The melodic arches of the Karykis-Balasis corpora show the greatest similarity (78.99%). In the second place we have the pre-Karykis – Karykis pair with similarity (68.18%), and in the last place the pre-Karykis – Balasis pair with similarity (62.34%). The exact opposite is true for dissimilarity and the percentage of dissimilar shapes. Our results follow the observation of Stathis [13, pp. 26], quoting "the seventeenth-century Heirmologia contain kallopismoi [literally, to make beautiful or embellishments, but not to be confused with improvisations] of the Heirmologia of Theophanes Karykes and Iohasaph the New Koukouzeles", i.e. the 17th onwards Heirmologions are not considered original compositions as they use as a basis the 16th century Heirmologions.

Figures 4 & 5 present the Jensen-Shannon distances of the corpora on the studied viewpoints. We observe that Karykis-Balasis show the smallest difference, while pre-Karykis – Karykis and pre-Karykis – Balasis show similar

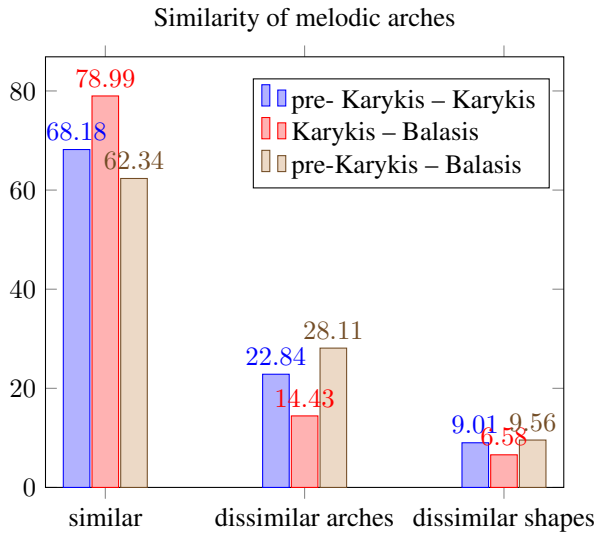


Figure 3. The bar charts of the melodic arches' similarity.

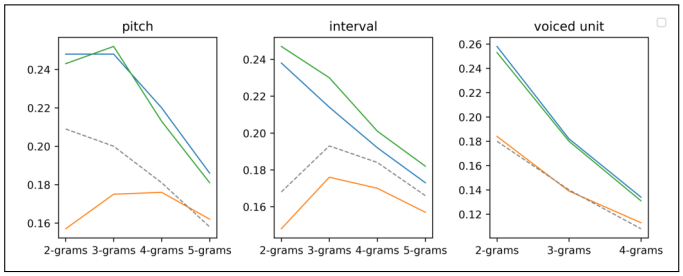


Figure 4. The Jensen-Shannon distances (abbr. JSd) of the Markov models. The reference line shows the JSd of the case study corpora (pre-Karykis and Karykis) where Karykis chooses exceptionally to keep the pre-Karykis tradition. The Blue line shows the JSd of the pre-Karykis – Karykis, the orange shows the JSd of the Karykis - Balasis, and the green line shows the JSd of the pre-Karykis – Balasis. The orders of n-grams are kept low in order to reduce the negative effect of higher order n-grams [38].

distances which are greater than of Karykis – Balasis. The close relation of Karykis – Balasis is confirmed by the near placement of the orange and gray lines of the case study corpus (section 3.3). The low rate of dissimilarity of the Karykis – Balasis Markov models is consistent with the observation of Makris [37] that the modality of Karykis – Balasis remains the same in contrast to the modality of the pre-Karykis era.

5. DISCUSSION

The notational texture (i.e. the number of voiced units per syllable) of the corpora outlines a change that occurred in the Heirmologic style. As shown in Table 1, the notational texture of Karykis presents a drop ($1.22 \rightarrow 1.13$) which leads us to a more syllabic texture. At the same time, however, Balasis shows an increase of notational texture, which reaches the same value as pre-Karykis ($1.13 \rightarrow 1.23$). If we consider that there is no important difference in the

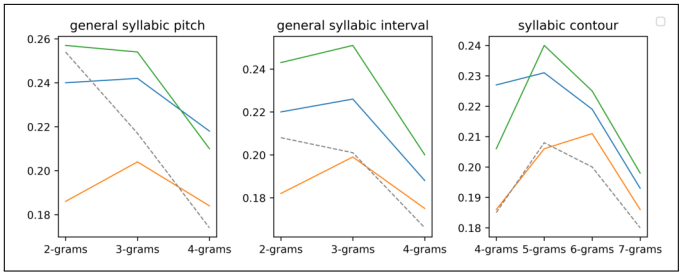


Figure 5. As of Figure 4, the Jensen-Shannon distances of redundant viewpoints.

interpretation-performance of the notation of the three corpora, then we can conclude that the values of the notation texture also correspond to values of durational texture. From these changes to the values of the notational texture, we can observe that Karykis simplifies the pre-Karykis melos and Balasis makes it sophisticated again by increasing the textures (notational and durational).

According to the bar chart of similarity of the melodic arches, the Karykis and Balasis pair present the greatest similarity than the other two. This presupposes that Karykis changes the tradition of the Heirmologion and Balasis largely follows Karykis. Also, we observe that all pairs present a melodic similarity that is more than 60%. This can be justified by the fact that the compared pieces are different compositions of the same lyrics (different melodies of the same Heirmoi). It seems that there are some constraints between the text and the melody, especially in the accented syllables. For example, we can guess that the composers want the accented syllables to be in local maxima. So, the peak areas influence the melodic contour of the melodic lines. This presupposes that Karykis changes the tradition and Balasis largely follows Karykis. Specifically, Karykis – Balasis have less distances than pre-Karykis – Karykis and pre-Karykis – Balasis ones.

These results can be viewed within the context of the *Theses* concept in Byzantine music. *Theses* are *formulae* that work as building blocks in Byzantine music pieces. Since the Markov models capture the melodies as frequencies of small patterns, we can infer that similar models of two corpora indicate that the two corpora use similar *Theses*.

6. CONCLUSIONS

This paper presented three related corpora belonging to the Heirmologic subgenre of the Byzantine music. The corpora are digitised using the Knowledge Representation of [11] which captures the MBn. Through this effort we envisage the creation of a large database of MBn scores. Through the computational analysis we observed that Karykis changes the style of the subgenre while the succeeding tradition of the subgenre (Balasis) follows the Karykis style. Even though the analysis presented addresses a specific research question, it offers us insights which will become the inspiration for further research in the domain of Computational Byzantine Musicology.

7. ACKNOWLEDGMENTS

We would like to thank Fabian Moss for his insights in the digitisation of the corpora.

8. REFERENCES

[1] D. Huron, "The Melodic Arch in Western Folksongs," *Computing in Musicology*, vol. 10, Mar. 1996.

[2] N. Cook, *A guide to musical analysis*, reprint ed. Oxford: Oxford Univ. Press, 2009.

[3] "Convention for the Safeguarding of the Intangible Cultural Heritage Intergovernmental Committee for the Safeguarding of the Intangible Cultural Heritage Fourteenth Session Bogotá," Dec. 2019.

[4] G. Stathis, *Ta protographa tes exegeseos eis ten nean methodou semiographias: ta prolegomena*. Athens, Greece: Institute of Byzantine Musicology, 2016, vol. 1.

[5] M. Velimirović, *Byzantine elements in early Slavic chant: The Hirmologion*. Munksgaard, 1960.

[6] E. Wellesz, *Eastern Elements in Western Chant: Studies in the Early History of Ecclesiastical Music*. Byzantine Institute, 1947, google-Books-ID: uXzQvwEACAAJ.

[7] C. Troelsgård, *Byzantine neumes: a new introduction to the Middle Byzantine musical notation*, ser. Monumenta musicae Byzantinae. Subsidia. Copenhagen : Lancaster: Museum Tusculanum Press ; Gazelle [distributor], 2011, no. 9, oCLC: ocn750956349.

[8] M. Alexandru, *Palaiographia Byzantines Mousikes [Paleography of Byzantine Music]*, Apr. 2017. [Online]. Available: https://repository.kallipos.gr/handle/11419/6487

[9] P. Savage, "An overview of cross-cultural music corpus studies," in *Oxford Handbook of Music Corpus Studies*. New York: Oxford University Press. [Online]. Available: https://www.researchgate.net/publication/326143358_An_overview_of_cross-cultural_music_corpus_studies

[10] X. Serra, "Creating research corpora for the computational study of music: the case of the compmusic project," in *Audio engineering society conference: 53rd international conference: Semantic audio*. Audio Engineering Society, 2014.

[11] P. Polykarpidis, C. Anagnostopoulou, and D. Kalofonos, "Towards the digitisation of Middle Byzantine notation: a formal model and Knowledge Representation," in *Music Encoding Conference 2022*, Halifax, Canada, To be published.

[12] ——, "MBnHeirmologicCorpora," Jul. 2022, publisher: OSF. [Online]. Available: https://osf.io/f97a5/?view_only=131fae4a07a04792986a12c43327709a

[13] G. Stathis, *Introduction to Kalophony, the Byzantine Ars Nova: The Anagrammatismoi and Mathēmata of Byzantine Chant*. Peter Lang, 2014, google-Books-ID: gAvwoQEACAAJ.

[14] D. Conomos, *The treatise of Manuel Chrysaphes, the Lampadarios: On the theory of the art of Chanting and on certain erroneous views that some hold about it*, ser. Corpus Scriptorum de re Musica. Wien: Österreichische Akademie der Wissenschaften, 1985, no. II.

[15] E. Makris, "Adjustments of modality in the Postbyzantine Heirmologion," in *Tradition and Innovation in Late- and Postbyzantine Liturgical Chant. Acta of the Congress Held at Hernen Castle, the Netherlands, in April 2005*. Hernen Castle, Netherlands: Peeters, 2008, pp. 37–63.

[16] N. Bouka, "E paradose ton Eirmologikon Byzantinon melodion tou Bareos echou apo ton 10o aiona eos ton 16o aiona," PhD, Ionian University, Corfu, Greece, 2004.

[17] A. Chaldaeakes, *Ermeneia syntomos eis tin kath' emas mousikin Pachomiou monachou*, ser. Mnimeia Ellenorthodoxis Mousikes. Athens, Greece: Athos, 2015, vol. 5, no. 1.

[18] D. Balageorgos and F. Kritikou, *Ta cheirographa tes Byzantines Mousikes: Sina*. Athens, Greece: Institute of Byzantine Musicology, 2021, vol. 2.

[19] G. Zesimos, "Kosmas Iberites kai Makedon: Domestikos tes mones ton Iberon," PhD, National and Kapodistrian University of Athens, Athens, Greece, 2007.

[20] D. Conklin and C. Anagnostopoulou, "Representation and discovery of multiple viewpoint patterns," in *Proceedings of the International Computer Music Conference (ICMC 2001)*, Havana, Cuba, 2001, pp. 479–485.

[21] D. Conklin and I. Witten, "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, Mar. 1995, publisher: Routledge_eprint: https://doi.org/10.1080/09298219508570672. [Online]. Available: https://doi.org/10.1080/09298219508570672

[22] L. Tardo, *Hirmologium Cryptense*, ser. principale (Facsimilés). Rome, Italy: Monumenta Musicae Byzantinae, 1951, vol. 3, no. 1.

[23] D. Balageorgos and F. Kritikou, *Ta cheirographa tes Byzantines Mousikes: Sina*. Athens, Greece: Institute of Byzantine Musicology, 2008, vol. 1.

[24] G. Stathis, *Ta cheirographa tes Byzantines Mousikes: Agion Oros*. Athens, Greece: Institute of Byzantine Musicology, 2015, vol. 4, no. 1.

[25] D. Touliatos-Miles, *A Descriptive Catalogue of the Musical Manuscript Collection of the National Library of Greece: Byzantine Chant and Other Music Repertory Recovered*. London: Routledge, Oct. 2017.

[26] J. Raasted, *Intonation Formulas and Modal Signatures in Byzantine Musical Manuscripts*. Copenhagen: E. Munksgaard, 1966.

[27] "Syllabic style," 2001. [Online]. Available: https://doi.org/10.1093/gmo/9781561592630.article.27236

[28] "Neumatic style," 2001. [Online]. Available: https://doi.org/10.1093/gmo/9781561592630.article.19788

[29] "Melismatic style," 2001. [Online]. Available: https://doi.org/10.1093/gmo/9781561592630.article.18333

[30] C. Troelsgård, "Stylistic variation in the old Sticherarion," in *Theoria and Praxis of the Psaltic Art*. Athens, Greece: Institute of Byzantine Musicology, 2006, pp. 225–232.

[31] G. S. Papadopoulos and P. Polykarpidis, "Elements in the development of how the Great Doxologies were sung from the 18th to the 19th century," *Journal of the International Society for Orthodox Music*, vol. 4, no. 2, pp. 293–322, Nov. 2020, number: 2. [Online]. Available: https://journal.fi/jisocm/article/view/95235

[32] P. Polykarpidis and G. S. Papadopoulos, "Melismatic elements in heirmologic chants from the 16th to the 19th century." Thessaloniki, Greece: IMS Study Group, 2022. [Online]. Available: https://www.academia.edu/52588244/Melismatic_elements_in_heirmologic_chants_from_the_16th_to_the_19th_century

[33] D. Spanoudakis, "Octaechia and Melismaticity in Byzantine Music. Collations and Musicological Reductive – Statistical Analysis," *IMS Regional Association for the Study of Music of the Balkans*, pp. 181–199, 2021. [Online]. Available: https://ejournals.lib.auth.gr/smb/article/view/7939

[34] F. C. Moss, "Transitions of Tonality: A Model-Based Corpus Study," PhD, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, 2019. [Online]. Available: https://infoscience.epfl.ch/record/273178

[35] "scipy.spatial.distance.jensenshannon — SciPy v1.7.1 Manual." [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.jensenshannon.html

[36] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, UK: Cambridge University Press, 2008.

[37] E. Makris, "Adjustments of modality in the Postbyzantine Heirmologion," in *Tradition and Innovation in Late- and Postbyzantine Liturgical Chant. Acta of the Congress Held at Hernen Castle, the Netherlands, in April 2005*. Hernen Castle, Netherlands: Peeters, 2008, pp. 37–63.

[38] J. Sakellariou, F. Tria, V. Loreto, and F. Pachet, "Maximum entropy models capture melodic styles," *Scientific Reports*, vol. 7, no. 1, Aug. 2017. [Online]. Available: https://www.nature.com/articles/s41598-017-08028-4

PLAYING TECHNIQUE DETECTION BY FUSING NOTE ONSET INFORMATION IN GUZHENG PERFORMANCE

Dichucheng Li[1] **Yulun Wu**[1] **Qinyu Li**[2] **Jiahao Zhao**[1]
Yi Yu[3] **Fan Xia**[2] **Wei Li**[1,4]

[1] School of Computer Science and Technology, Fudan University, Shanghai, China
[2] College of Experimental Art, Sichuan Conservatory of Music, Sichuan, China
[3] National Institute of Informatics (NII), Tokyo, Japan
[4] Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, China

21210240219@m.fudan.edu.cn, {20110240018, 20210240306, weili-fudan}@fudan.edu.cn,
yiyu@nii.ac.jp, {935362804, 409769992}@qq.com

ABSTRACT

The Guzheng is a kind of traditional Chinese instruments with diverse playing techniques. Instrument playing techniques (IPT) play an important role in musical performance. However, most of the existing works for IPT detection show low efficiency for variable-length audio and provide no assurance in the generalization as they rely on a single sound bank for training and testing. In this study, we propose an end-to-end Guzheng playing technique detection system using Fully Convolutional Networks that can be applied to variable-length audio. Because each Guzheng playing technique is applied to a note, a dedicated onset detector is trained to divide an audio into several notes and its predictions are fused with frame-wise IPT predictions. During fusion, we add the IPT predictions frame by frame inside each note and get the IPT with the highest probability within each note as the final output of that note. We create a new dataset named GZ_IsoTech from multiple sound banks and real-world recordings for Guzheng performance analysis. Our approach achieves 87.97% in frame-level accuracy and 80.76% in note-level F1-score, outperforming existing works by a large margin, which indicates the effectiveness of our proposed method in IPT detection.

1. INTRODUCTION

The Guzheng (古箏), which is also known as the Chinese zither, is a plucked 21-string Chinese musical instrument existing for over 2,500 years [1]. Chinese traditional music attaches great importance to the melody, so a large number of playing techniques are used to enhance the vividness of Guzheng performance. The pitch variation produced by pressing the strings with the left hand is even regarded as "the soul of Guzheng music" [2]. Instrument playing technique (IPT) detection aims to classify the types of IPTs and locate the associated IPT boundaries in an audio clip. However, there were only few researches about IPTs in the field of Music Information Retrieval (MIR). Particularly, most researches on automatic music transcription (AMT) only consider pitch estimation, while a complete transcription of a Guzheng performance should contain the notations of the playing techniques as shown by the Guzheng numbered musical notation [1] in Figure 1. In this work, we propose an IPT detection method that can further be incorporated into a full transcription system of Guzheng music.

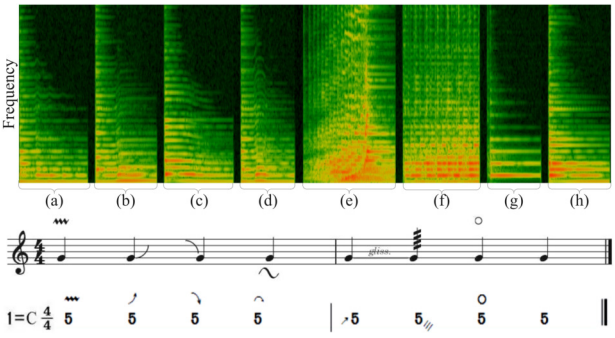


Figure 1. The spectrogram, staff notation and numbered musical notation of a Guzheng phrase that contains 8 notes with different playing techniques: vibrato (a), upward portamento (b), downward portamento (c), returning portamento (d), glissando (e), tremolo (f), harmonic (g), plucks (h)

One major difficulty for the IPT research is the lack of IPT sound databases. Most of the researches [3–6] on IPT recognition limited their experiments to samples from isolated notes in a single sound bank. However, according to [7], the performance of classifiers trained and tested with a single sound bank provides no assurance in the generalization capabilities of the classifiers. Ducher et al. [8] extended the above theory to IPT recognition research. They

© D. Li, Y. Wu, Q. Li, J. Zhao, Y. Yu, F. Xia, and W. Li. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** D. Li, Y. Wu, Q. Li, J. Zhao, Y. Yu, F. Xia, and W. Li, "Playing Technique Detection by Fusing Note Onset Information in Guzheng Performance", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] a musical notation system widely used in China and called *Jianpu* in Chinese

used five available IPT sound banks in their experiments but did not annotate any recorded corpora of audio.

Existing methods for IPT detection are mainly implemented through two steps [5,6,9]: locating and classifying. These methods are sensitive to the errors caused in locating. Recently, some end-to-end methods [10] have been proposed, while they are weak for variable-length audio and show low accuracy at the boundary of adjacent notes.

Note onset detection aims to localize the very beginning of each note. The beginning of a Guzheng note is easier to identify because the amplitude of that note is at its peak. In Guzheng performance, strings are usually plucked using special nails attached to fingers bottoms. When string is plucked, it has a pre-attack before the string reaches its full level vibration so the onset has a unique broadband spectrum. As shown in Figure 1, each Guzheng playing technique is applied to a note and each note in Guzheng music starts with a clear onset. So onset information is crucial to the Guzheng playing technique detection. However, to the best of our knowledge, no research has taken use of the onset information in the end-to-end IPT detection field.

The main contributions of this paper are as follows: 1) We create a new dataset, GZ_IsoTech, which consists of Guzheng playing technique clips from two Guzheng sound banks and real-world recordings recorded by a professional Guzheng performer; 2) We propose the first end-to-end method that can be applied to variable-length audio for Guzheng playing technique detection; 3) We propose a decision fusion method where an onset detector is trained to divide an audio into several notes and the predicted IPTs produced by the IPT detector are added frame by frame inside each note to get the IPT class with the highest probability within each note as the final output of that note.

2. RELATED WORK

Although the research on IPT detection is still in its early stage, we can summarize relevant researches and divide the development of this field into three periods.

The researches in the first period mainly focus on the playing technique classification of isolated notes [4, 11]. However, in reality, there are often continuous notes with varying playing techniques in an audio. We not only need to classify the IPT types but also locate the IPT boundaries.

The methods for IPT detection proposed in the second period can be divided into two steps: locating and classifying. In [5,6], signal processing methods were used to select candidates of the playing techniques in the melody contours extracted from 42 electric guitar solo tracks. Then the timbre and pitch features of the candidates were input into the classifiers such as Support Vector Machine (SVM). The candidates were manually selected based on the melody contour, so the methods are sensitive to errors in melody extraction and can hardly be generalized to other IPTs.

In [9], the candidates of guqin playing techniques were firstly located according to the IPT onset and duration annotations in 39 guqin solo recordings and then classified into six left-hand IPT types. Yet, we generally do not have onset and note duration as prior information in reality.

In [12], two binary classifiers bases on Convolutional Neural Networks (CNN) are firstly used to decide whether a fixed-length portion in a piano recording is played with the sustain pedal. Then sliding windows are used to apply the method to pieces with variable lengths. In [8], five different methods were proposed to classify 18 different IPTs in the cello solo audio concatenated by isolated cello notes with different IPTs from 5 different sound banks. Although the models proposed in [8, 12] can detect the IPTs in audio with continuous IPTs, the essence of them is to classify IPTs in a single block. These methods have redundancy in the computation because the input sound is split into overlappping frames that are fed individually to the network through a sliding window. Each convolution is thus computed several times on the overlapping parts.

We regard end-to-end IPT detection as the approach in the third period. In [10], the authors presented an end-to-end method based on Fully Convolutional Networks (FCN) to detect the IPTs in 10-second segments concatenated by isolated erhu notes. However, the method shows low accuracy at the boundary of adjacent notes and has computational redundancy when applied to variable-length audio.

Our model is built upon the FCN model presented in [10], with some improvements. We add a dedicated onset detector to take advantage of the significance of note onsets in Guzheng music. A decision fusion is implemented for the onset and IPT prediction to make a note-level prediction. In this way, we achieve a better performance for IPT predictions at the note boundaries. To apply our method to audio with variable lengths, 1D max-pooling layers which only halve the length of frequency axis are used.

3. GZ_ISOTECH DATASET

In this section, we introduce the Guzheng playing techniques considered in our work and the process of making the dataset.

3.1 Guzheng playing techniques

The traditional Guzheng playing techniques can be divided into two classes: plucking string with the right hand and bending string with the left hand [13]. In our work, we consider the following eight playing techniques which are the most frequently used in Guzheng solo compositions.

Left-hand playing techniques:

- **Vibrato (chanyin颤音):** the periodic oscillation of tones caused by the periodic pressing of a string with left hand (region (a) of Figure 1).
- **Upward Portamento (shanghuayin上滑音) (*UP* for short thereinafter):** press a string with left hand to increase the pitch of a ringing note to a desired pitch within major third (region (b) of Figure 1).
- **Downward Portamento (xiahuayin下滑音) (*DP* for short thereinafter):** decrease the pitch of a ringing note by releasing a bended string (region (c) of Figure 1). It is the opposite of Upward Portamento.

IPT	VSB		RW	
	num	seconds	num	seconds
vibrato	192	261.1	42	51.5
UP	488	752.0	48	94.4
DP	333	508.5	51	87.5
RP	272	327.0	94	94.7
glissando	316	595.3	42	95.9
tremolo	205	259.8	23	59.7
harmonic	318	305.8	61	42.0
plucks	204	204.0	135	99.7

Table 1. Quantity of data in the virtual sound banks (VSB) and real-world (RW) recordings. The Guzheng playing technique clips from VSB is used in the training set and that from RW is used in the test set.

- **Returning Portamento (huihuayin回滑音) (*RP* for short thereinafter):** pluck a string, press it with left hand, and then release it, creating an up-down slide (region (d) of Figure 1).

Right-hand playing technique:

- **Glissando (guazou刮奏, huazhi花指):** play several strings consecutively up or down with thumb or index finger rapidly (region (e) of Figure 1).
- **Tremolo (yaozhi摇指):** pluck a single string in rapid succession by "shaking" the index finger or thumb of the right hand back and forth across it as shown in the region (f) of Figure 1.
- **Harmonic (fanyin泛音):** tap a special location of a string with the left hand, and lift the left hand while playing the string with the right hand to produce a corresponding harmonic effect as shown in the region (g) of Figure 1.
- **Plucks (gou勾, da打, mo抹, tuo托...):** as shown in the region (h) of Figure 1, plucks are defined as simply plucking a string with a finger, including gou, da, mo, tuo [2], and so on.

3.2 Data collection and labelling

In order to ensure the diversity of the data, we collected audio clips of single playing techniques (we called these audio clips *short clips* thereinafter) from real-world recordings (RW) and 2 virtual Guzheng sound banks (VSB) which had different players and recording setups: Yellow River Sound and Kong Audio 3 KONTAKT. We sampled 2328 *short clips* covering almost all the tones in the range of Guzheng from the sound banks and annotated them into 8 classes one by one according to the sound bank manuals. Their durations range from about 0.6 to 11.5 seconds, in total 3213.4 seconds (see Table 1). The 496 real-world recordings of *short clips*, in total 625.3 seconds, were recorded and annotated by a professional Guzheng performer.

We use the *short clips* sampled in VSB as the training split and the set of *short clips* recorded in real world as the test split. This division well simulates the test situation in the real world as we usually do not have access to recordings of a testing Guzheng at training time. The dataset with detailed information and demos is available on the website [3] [14].

4. METHODS

As shown in Figure 2, our proposed model consists of two modules: the IPT detector and the onset detector. In the training phase, we train the onset detector and the IPT detector separately. Then, in the fusion phase, we apply decision fusion to the thresholded output of the onset detector and the raw output of the IPT detector to get the final result.

4.1 Input Representation

Since our goal is to generalize to actual solo recordings, we have generated series of audio sequences which simulate such recordings. Firstly, on both training and test, we concatenated our *short clips* in the dataset mentioned in Section3 one after another randomly until the length of the concatenated audio sequence is greater than 12.8 seconds. A check operation was executed to confirm that every generated audio sequence is unique. A 50 milliseconds crossfade was made in each boundary of adjacent *short clips* to ensure the audio sequence sound more realistic. We cut the audio sequence generated from the training set directly to 12.8 seconds long for training and use the unsplit audio sequence whose length is greater than 12.8 seconds generated from the test set for testing.

We also label the start time and duration of the corresponding playing techniques. Then we generate two kinds of labels (the onset lables and the IPT labels) by quantizing the continuous time labels into timestamps in the unit of 0.05-second-long frame.

The input representation is a log mel-spectrogram. Following [10], we use *librosa* [15] to compute log mel-spectrogram with 128 logarithmically-spaced frequency bins, an FFT window of 2048, a hop size of 2205, and a sample rate of 44.1kHz.

4.2 IPT Detector

Guzheng playing technique detection can be regarded as a part of the task of Guzheng music transcription. As most of the transcription task [16, 17], we treat the frame as a basic unit so we need to identify the playing techniques frame by frame. We can treat the task as a semantic segmentation task since there is no overlap of playing techniques in our data and frames in music are similar to pixels in images. Inspired by the success of FCN in semantic segmentation [18], which is also extended to audio processing research [10, 19], we applied FCN in our task.

[2] gou, da, mo, tuo are the names of the Guzheng playing techniques for simply plucking a string using the middle finger, ring finger, index finger, and thumb respectively.

[3] https://ccmusic-database.github.io/en/database/ccm.html#GZTech

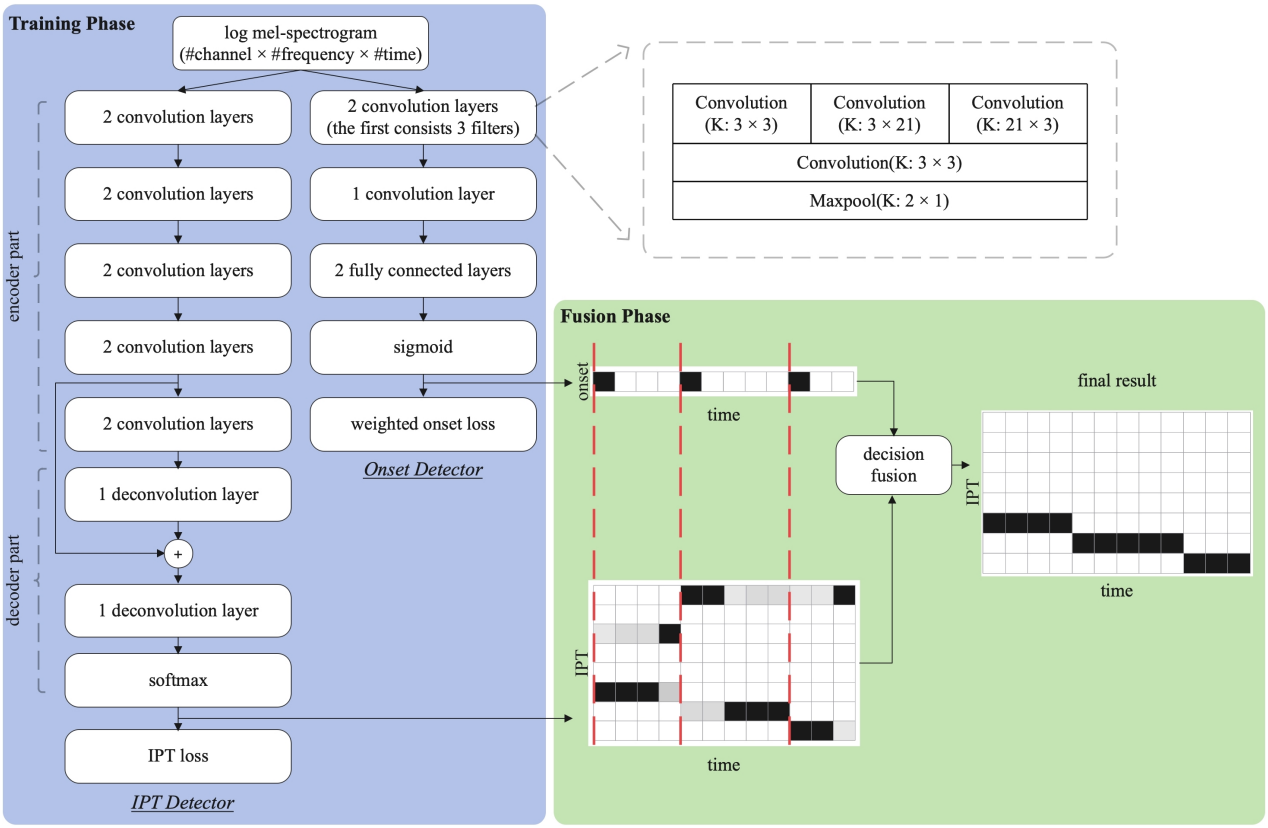


Figure 2. System architecture of our proposed model. In the training phase, the IPT detector and the onset detector were trained separately and then decision fusion was applied to their outputs in the fusion phase to get the final result. The vertical red dotted lines in the "fusion phase" stand for the starts of all onset predictions.

As shown in Figure 2, the IPT detector can be divided into the encoder and the decoder. The encoder part consists of five convolution modules. Two identical convolutional layers were stacked in each module and each of them has square (3×3)[4] filters. At each convolutional layer, zero padding was applied to make the output have the same length as the input and the output of every convolutional layer was then passed through a Rectified Linear Unit (ReLU). To apply our model to variable-length audio, a 1D max-pooling layer which only halve the length of frequency axis was implementd at the output of each convolution module followed by a dropout layer with the probability 0.25 to aid generalization. The output of the encoder part is the input of the decoder part which is made of two deconvolution layers. The decoder part is used for upsampling the feature map to the shape that we need and then make a frame-level prediction to the IPT. A skip connection of element-wise adding was used from the output of the fourth module in the encoder part to the first module output in the decoder part to help the decoder fuse the information.

4.3 Onset Detector

The onset detector is composed of two convolution modules, followed by two fully connected layers. The last layer is a fully connected sigmoid layer with 1 output for representing the probability of an onset for this frame. It has been suggested in recent research [20] that using different musically motivated filter shapes in the first convolutional layer of CNN could improve the model performance. As we discussed in Section 3, different playing techniques in Guzheng vary a lot in the temporal and spectral domain. Thus, we apply three filters of different shape (3×3, 3×21, 21×3) to the first convolutional layer and the 3×21 filter is designed to better capture the feature of long playing technique such as *portamento* or *glissando*. The following two convoliutional layers each has 3×3 filters.

We use weighted binary cross entropy (WBCE) loss as the loss function for our onset detector.

$$L_{onset} = mean(\sum_{t=0}^{T} WBCE(\beta, x_t, y_t)) \quad (1)$$

$$WBCE(\beta, x_t, y_t) = \beta y_t \log x_t + (2-\beta)(1-y_t)\log(1-x_t) \quad (2)$$

where T is the number of frames in the example, y_t is the label that is 1 when there is a ground truth onset at the frame t, x_t is the probability output by the model at frame t and β is the weight factor of the positive samples. Recent models in the onset detection research showed a lower recall than precision [16, 17] and it is largely caused by class imbalance that positive samples for onset are far less than the negative samples. So we decided to increase the weight of the positive samples to balance the performance of the model. We set β to 1.94 by coarse hyper-parameter search.

[4] 2D filters are noted: N frequency bins x M time frames.

model	Frame-level	Note-level		
	accuracy	precision	recall	F1-score
FCN+Onsets	**87.97**	**78.20**	**83.78**	**80.76**
Z.Wang's model [10]	69.44	35.18	47.45	39.53
J.-F. Ducher's model_Reproduced [8]	66.93	19.37	21.31	20.05
B.Liang's model_Reproduced [12]	53.98	41.38	44.50	42.81

Table 2. Frame-level accuracy and note-level precision, recall and F1-score on GZ_IsoTech dataset. Note-based scores calculated by the *mir_eval* library. Final metric is the mean of scores calculated per piece in the test set.

4.4 Fusion of IPT Detector and Onset Detector

In evaluation, the IPT detector and the onset detector are firstly used separately. Then we use a threshold of 0.5 to make the onset output binary. We separate the IPT output into several clips by the binary onset output. Then we selected an IPT as the final result of each clip by "voting" within it as described by Algorithm 1. We set n as the number of IPTs and t as the number of time frames. D_{onset} whose shape is [t] is the binary output of the onset detector. D_{IPT} whose shape is [n, t] is the raw output of the IPT detector, representing the probability of each IPT for each frame. D is the final one-hot result that implies the IPT at every frame. There are two intermediate variables in our algorithm. V whose shape is [n] represents the total "voting" score for each IPT in the current clip and b denotes the frame number where the current clip starts.

Algorithm 1 Decision fusion

Input: the number of IPTs n, the number of time frames t, the thresholded output of the onset detector D_{onset}, the output of the IPT detector D_{IPT}

Output: final result D

$b \leftarrow 0$
$D \leftarrow zeros(D_{IPT})$
$V[0, \ldots, n-1] \leftarrow 0$
for $i \leftarrow 0$ **to** $t-1$ **do**
  for $j \leftarrow 0$ **to** $n-1$ **do**
    $V[j] \leftarrow V[j] + D_{IPT}[j][i]$
  end for
  if $i+1 = t$ or $D_{onset}[i+1] = 1$ **then**
    $D[\text{argmax}(V)][b, \ldots, i] \leftarrow 1$
    $V[0, \ldots, n-1] \leftarrow 0$
    $b \leftarrow i+1$
  end if
end for
return D

5. EXPERIMENTS

In this section, we introduce the evaluation metrics we used, the details of the experiment and the result analyses.

5.1 Evaluation Metrics

The metrics used to evaluate a model consist of frame-level and note-level metrics. Note-level metrics include precision, recall, and F1 scores. They are calculated by the *mir_eval* library [21]. We require that both the IPT type and the onset are correct. In detail, an onset estimation is considered correct if it is within ± 0.05 seconds of the ground-truth. Frame-level accuracy is calculated by comparing the output of our model to the ground truth label frame by frame. More specifically, we calculated the ratio of the number of correct frames to the total number of frames as frame accuracy. Both note and frame scores are calculated every audio sequence and we calculated the mean of these scores as the final metric.

5.2 Results

To examine whether our proposed architecture (namely `FCN+Onsets`) can better detect IPTs in audio with continuous IPTs, we reproduced J.-F. Ducher's model [8], B.Liang's model [12] and Z.Wang's model [10] for comparison, while adapting their capacity to our data.

Because Z.Wang's model [10] cannot be tested on variable-length audio, we cut all the audio from the test set to 12.8 seconds long for the test of Z.Wang's model [10] while other models are still tested on variable-length data.

Table 2 shows the frame-level and note-level results of Guzheng playing technique detection on the GZ_IsoTech dataset. Our `FCN+Onsets` model reaches 87.97% in frame-level accuracy and 80.76% in note-level F1-score, producing the best scores in both frame-level and note-level metrics. Besides, our proposed model outperforms other models by a large margin in all metrics and results in over a 100% relative improvement in note-level F1-score compared to other models. This is mainly because our model takes better advantage of the note onset detection results which is highly interrelated to the note-level metrics. Because the test set consists of audio of real-world recordings that is different from training set in performer, instrument and recording environment, the results show that our model offers good generalization capabilities.

The visualization of the input spectrogram, the thresholded onset output, the raw IPT output, the final fusion result and the label for a recording from the test set are shown in Figure 3. Through the images in Figure 3, we can clearly see the importance of fusing IPT activations with onset predictions. The vertical red lines in the third and fourth images are the onset predictions. We can separate the audio into several notes using the onset predictions. Each note in our dataset only corresponds to one playing technique, but there may be several IPT predictions in one note as

model	Frame-level	Note-level		
	accuracy	precision	recall	F1-score
FCN+Onsets	**87.97**	78.20	**83.78**	**80.76**
CNN+Onsets	60.80	45.01	48.30	46.51
No Onset Fusion	76.48	27.74	50.55	35.47
Onset3×3	87.70	72.67	82.76	77.19
No Skip Connection	82.45	71.05	76.15	73.39
No Weighted Loss	86.26	**80.88**	79.46	80.03

Table 3. Ablation studies with frame-level accuracy and note-level precision, recall and F1-score on GZ_IsoTech dataset.

shown in the "Raw IPT Output" image. So we implement the decision fusion method to choose one IPT as the final output of a note as illustrated in Algorithm 1. The IPT results after being restricted by the onset are presented in the second-to-bottom image ("Final Fusion Result"). The "Target" image is the visualization of the truth label. As shown by Figure 3, the accuracy of the predictions in the "Final Fusion Result" image is far exceed that in the "Raw IPT Output" image.

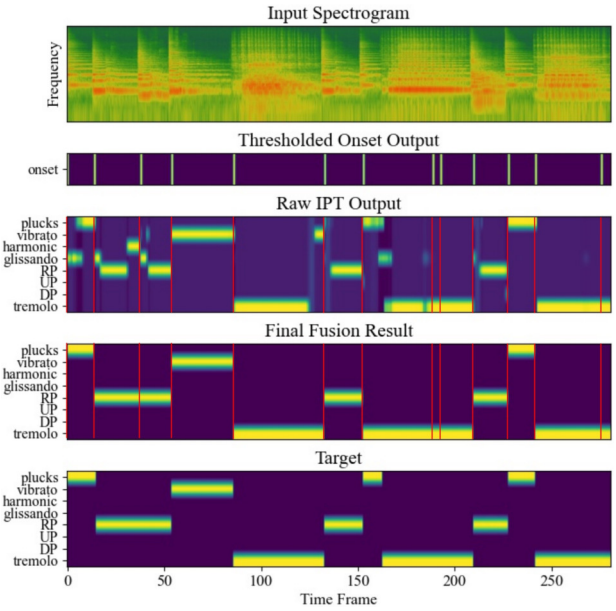


Figure 3. Inference on an audio in the test set. The log mel-spectrogram input, the thresholded onset output, the raw IPT output, the final fusion result and the target are shown from top to bottom. The vertical red lines in the third and fourth images are the onset predictions.

5.3 Ablation Studies

We conducted an ablation study where the performance of `FCN+Onsets` is compared with the following five models (namely `CNN+Onsets`, `No Onset Fusion`, `Onset3×3`, `No Skip Connection`, `No Weighted Loss`). In `CNN+Onsets`, we used the same CNN architecture for the IPT detector as presented in the onset detector instead of FCN. The result of `No Onset Fusion` is the output of the IPT detector without the decision fusion with the onset detector. In `Onset3×3`, we replaced the whole first convolutional layer of the onset detector by 3×3 filters. In `No Skip Connection`, we remove the skip connection part of the `FCN+Onsets` model. `No Weighted Loss` is the model that uses ordinary binary cross entropy (BCE) Loss in the onset detector.

The results show the importance of the onset information - `No Onset Fusion` results in a significant 11.49% decrease in the frame-level accuracy and a 45.29% decrease in the note-level F1-score. Our model can locate the IPT boundaries precisely and rectify some mispredicted frames for IPTs through fusing the IPT results with the note onset information. `CNN+Onsets` results in decreases in all metrics. It shows the FCN can better distinguish different IPTs than CNN. `No Skip Connection` shows a 5.52% decrease in frame-level accuracy and a 7.37% decrease in note-level F1-score. This proves the effectiveness of our skip-connection operation. `No Weighted Loss` has a little rise in note precision but decrease in all other metrics that shows weighted loss can better balance the precision and recall.

6. CONCLUSION

In this paper, we create a new dataset consisting of abundant Guzheng playing technique clips from multiple sound banks and real-world recordings. We propose an end-to-end Guzheng playing technique detection system using FCN that can be tested on variable-length audio. We propose a new decision fusion method where an onset detector is trained to divide an audio into several notes and the predicted IPTs are added frame by frame inside each note during fusion. The final output of each note is the IPT class with the highest probability within the note. Our method outperforms existing works by a large margin, which proves that our model is effective and offers good generalization capabilities that transfer well between different training and test sets. Moreover, by visualising the results, we find that the onset information is crucial for the Guzheng playing technique detection task.

In different genres of Guzheng music, there are slight differences in the same playing technique. In the future, we will continue to expand our dataset and add real-world Guzheng music pieces from different genres. Moreover, Guzheng is polyphonic and even one note may have more than one playing techniques. We will investigate how to detect more complicated playing techniques.

7. ACKNOWLEDGEMENT

This work was supported by National Key R&D Program of China (2019YFC1711800), NSFC (62171138). Wei Li, Fan Xia and Yi Yu are corresponding authors of this paper.

8. REFERENCES

[1] S. Zhuo, *The Chinese zheng zither: contemporary transformations*. Routledge, 2016.

[2] X. Zhao, "From 'complementing tone with yun' to 'expressing with sound': On the changes of left-hand playing techniques in zheng playing and the thinking aroused from them (In Chinese)," *Yellow River of the Song*, no. 19, pp. 33–37, 2009.

[3] V. Lostanlen, J. Andén, and M. Lagrange, "Extended playing techniques: the next milestone in musical instrument recognition," in *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, 2018, pp. 1–10.

[4] L. Su, L.-F. Yu, and Y.-H. Yang, "Sparse cepstral, phase codes for guitar playing technique classification." in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, 2014, pp. 9–14.

[5] Y.-P. Chen, L. Su, and Y.-H. Yang, "Electric guitar playing technique detection in real-world recording based on f0 sequence pattern recognition." in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR*, 2015, pp. 708–714.

[6] T.-W. Su, Y.-P. Chen, L. Su, and Y.-H. Yang, "Tent: Technique-embedded note tracking for real-world guitar solo recordings," *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, pp. 15–28, 2019.

[7] A. Livshin and X. Rodex, "The importance of cross database evaluation in sound classification," in *Proceedings of the 4th International Society for Music Information Retrieval Conference, ISMIR*, 2003, pp. 1–1.

[8] J.-F. Ducher and P. Esling, "Folded cqt rcnn for real-time recognition of instrument playing techniques," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 708–714.

[9] Y.-F. Huang, J.-I. Liang, I.-C. Wei, and L. Su, "Joint analysis of mode and playing technique in guqin performance with machine learning," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020, pp. 85–92.

[10] Z. Wang, J. Li, X. Chen, Z. Li, S. Zhang, B. Han, and D. Yang, "Musical instrument playing technique detection based on fcn: Using chinese bowed-stringed instrument as an example," *arXiv preprint arXiv:1910.09021*, 2019.

[11] J. Abeßer, H. Lukashevich, and G. Schuller, "Feature-based extraction of plucking and expression styles of the electric bass guitar," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2010, pp. 2290–2293.

[12] B. Liang, G. Fazekas, and M. Sandler, "Piano sustain-pedal detection using convolutional neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2019, pp. 241–245.

[13] J. Qiu, "On the classification and evolution of zheng playing techniques (In Chinese)," *Chinese Music*, no. 4, pp. 215–224, 2004.

[14] Z. Li, S. Yu, C. Xiao, Y. Geng, W. Qian, Y. Gao, and W. Li, "Ccmusic database: Construction of chinese music database for mir reaserch (In Chinese)," *Journal of Fudan University (Natural Science)*, vol. 58, no. 3, pp. 351–357, 2019.

[15] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, pp. 18–25.

[16] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, 2018, pp. 50–57.

[17] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution piano transcription with pedals by regressing onset and offset times," *IEEE/ACM Transactions on Audio, Speech, and Language Processing, TASLP*, vol. 29, pp. 3707–3717, 2021.

[18] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition, CVPR*, 2015, pp. 3431–3440.

[19] L. Ardaillon and A. Roebel, "Fully-convolutional network for pitch estimation of speech signals," in *20th Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2019, pp. 2005–2009.

[20] J. Pons and X. Serra, "Designing efficient architectures for modeling temporal features with convolutional neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2017, pp. 2472–2476.

[21] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. Ellis, "mir_eval: A transparent implementation of common mir metrics," in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, 2014, pp. 367–372.

KDC: AN OPEN CORPUS FOR COMPUTATIONAL RESEARCH OF *DASTGĀHI* MUSIC

Babak Nikzat
Kunstuniversität Graz
b.nikzat@kug.ac.at

Rafael Caro Repetto
Kunstuniversität Graz
rafael.caro-repetto@kug.ac.at

ABSTRACT

Iranian *dastgāhi* music is considered as the classical repertory of contemporary Iran. In the 19th century, the melodic modes that developed during its long history were grouped in categories, each of them known as *dastgāh*. The *dastgāhi* system presents unique features, that have been object of musicological study since its inception. However, computational methods for its research are still scarce, due in good part to the lack of open, well curated corpora. The aim of the KUG *Dastgāhi* Corpus (KDC) is to contribute to the development of computational corpus driven research for this tradition. KDC is created following the FAIR principles, and in close collaboration with performers and scholars, who contribute to it with annotations and qualitative evaluations. Besides presenting the first version of KDC, in this paper we explore the possibilities that Iranian *dastgāhi* music offers to computational research. In order to test the performance of state-of-the-art technologies applied to this music tradition, we present preliminary results for several analytical tasks, and discuss their opportunities and limitations learnt in the process.

1. INTRODUCTION

The potential of computational methods for supporting musicological research tasks is increasingly being recognized and adopted by musicologists. Due to the high degree of specialization required both from the technical and musicological sides of this kind of research, collaboration between specialists from both disciplines is one of the most fruitful approaches for computational musicology. Developed at the Institute for Ethnomusicology of the Kunstuniversität Graz (KUG), the KUG *Dastgāhi* Corpus (KDC) aims to offer a growing collection of well curated and annotated data of Iranian *dastgāhi* music around which engineers, musicologists and researchers from other disciplines interested in this music tradition can collaborate. In this first section we describe the main features of the *dastgāhi* musical system and offer an overview of the state-of-the-art of its computational research. Then, we introduce and describe in detail the first version of KDC. In section 3 we discuss the potential of Iranian *dastgāhi* music for computational research. In the following sections, preliminary analyses carried out to test such potential are presented and discussed. The paper closes with our plan for expanding KDC and future work.

1.1 Iranian *dastgāhi* music

Iranian *dastgāhi* music is modal. The basic modal entities in this tradition are known as *guše*s and they are defined by a model melody built on a specific intervallic structure, a series of characteristic melodic patterns, and a modal centre known as *šāhed* [1]. Related *guše*s are grouped in specific collections, and in a specific order. These collections are known either as *dastgāh* or *āvāz*, each of them with its own specific name. A *dastgāh* is larger than an *āvāz* in terms of number of *guše*s, and an *āvāz* is considered to be a subcategory of a specific *dastgāh*. Nowadays, the canonical repertoire consists of 7 *dastgāh*s and 5 *āvāz*es (see Table 1). For simplicity, in this paper we are using *dastgāh* to refer to both of them collectively. The *guše*s in a particular *dastgāh* are organized in a fixed order, and even though the performance of a *dastgāh* does not required all its *guše*s to be performed, the order has to be maintained. The first *guše* in all *dastgāh*s is called *darāmad*, and it has a very significant meaning since it introduces the *dastgāh*. Most performances start and end with *darāmad*.

Along the history of Iranian *dastgāhi* music, great masters (*ostād*s) of this tradition defined their own canonical repertory of the 7 *dastgāh*s and 5 *āvāz*es, fixing the number of *guše*s in each of them, their particular order, their specific melodic contour and their rhythmic rendition. These canonical repertoires are known as *radif*s, always associated with their corresponding *ostād*. These fixed *radif*s are nowadays only used for learning the tradition. Contemporary musicians perform their own interpretation of the repertory, but through the rules they experientially extracted from the *radif* they learnt. As a more encompassing concept, different performance schools are recognized, each of them with idiosyncratic performance styles. The most extended ones today are Tabriz, Tehran and Isfahan.

Iranian *dastgāhi* music uses a microtonal tuning system, resulting in specific intervallic structures defined for each *guše*. Three basic interval categories are considered, namely tone, semitone and an intermediate one between the previous two, whose specific width is not fixed but flexible [1], depending on *guše* or even performer, and that is the result of altering one natural tone less than a semitone

© B. Nikzat, and R. Caro Repetto. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** B. Nikzat, and R. Caro Repetto, "KDC: An Open Corpus for Computational Research of *Dastgāhi* Music", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

up (*sori*) or down (*koron*). These intervallic structures form specific scales, in which particular tones receive predominant roles, such as *šāhed* and *ist*, which are fundamental for defining the modal features of a *guše*.

Specific techniques for vocal and instrumental performance, including an extensive repertory of ornamentation, is another essential aspect of this music, showing a great diversity depending on school, instrument, and personal style. The most common instrumental ornaments include: *ešāreh*, *tekiyeh*, *qalt*, *dorrāb*, *riz*, and *šalāl*. *Tahrir* is the most significant singing ornament, which requires a specific technique in which the singer switches between chest and head voice (see Fig. 5).

*Guše*s can be performed in free rhythm or using rhythmic cycles. Even though this is fixed per each *guše* in the *radif*s, it is decided by each musician in contemporary performance. When a *guše* is sung in free rhythm, the prosodic rules, known as *aruz*, of Persian classical poetry, to which most of the sung lyrics in Iranian *dastgāhi* music belong, informs the rhythmic rendition of the singing. This results in a characteristic rhythmic style that is also adopted when a *guše* is perfomed only instrumentally [2, 3]. *Aruz* not only informs musical rhythm through the sequence of long and short syllables, but also through the qualitative features of the vowels (*hejā*) [4, 5]. For metred performance, musicians can draw on rhythmic cycles known as *adwār-e iqāi*, the most common ones being *kerešme*, *čāhār-pāreh*, *zanguleh* and *gereyli* [4]. Each *dowr-e iqāi* is defined by a number of pulses, known as *naqareh*, and a structure of accents.

A typical ensemble of Iranian *dastgāhi* music is formed by *santur* (hammered zither), *tār* (long-neck lute), *setār* (long-neck lute), *ney* (bamboo flute), *kamanče* (spike fiddle), *tonbak* (goblet drum) and a singer. The music, however, is also commonly performed by a solo singer or instrumentalist.

Dastgāhi music is an oral tradition and conventionally the students learn it by memorizing a specific *radif* taught in private lessons by an *ostād*. The learning process is based on observation and imitation. In the contemporary context some teachers use transcriptions in staff notation as teaching material. However the notation functions just as memory aid, since many details cannot be represented in the staff notation system.

1.2 Computational analysis of Iranian *dastgāhi* music

Considering the magnitude of Iranian *dastgāhi* music, in terms of history, wide performance realm, complex theorization and extensive musicological research, this music tradition has received comparatively little attention from computational approaches. Pioneering work has been carried out by the researcher and *santur* performer Peyman Heydarian. Since his master thesis [6], Heydarian is working on the computational analysis of this instrument, especially its acoustic features [7–9], as well as on *dastgāhi* music performed in this instrument, with a special focus on *dastgāh* classification [10–12]. In fact, this second task, automatic *dastgāh* identification and classification, has been the most common one by other researchers who approached this tradition from computational methods [13–16].

Computationally aided musicological research on *dastgāhi* music is still very rare. Shafiei used score to audio alignment for the analysis of *tahrir* [17], while Sanati analysed intervals from *ney* recordings [18]. There has also been some work on *dastgāhi* music generation using deep neural networks [19].

We argue that one of the reasons that explains that the computational analysis of *dastgāhi* music is not yet been fully established in an unified trend, instead of occasional isolated works, is the lack of a comprehensive, publicly accessible datasets upon which continuous research can be built. All the publications referenced in the previous paragraphs gathered their own research datasets, but none of them offer any indication about possibilities for accessing them. Only Heydarian has been able to sustain a decades long research on this tradition based on his own dataset [20], but it is not public, and it only includes recordings of *santur*. Hence, the goal of KDC precisely is to contribute to the establishment of a coherent line of computational research on Iranian *dastgāhi* music by providing an open, well curated corpus of high quality recordings, upon which this research can be built.

2. DESCRIPTION OF KDC

KDC is conceived as an ever growing corpus, so in this paper we present its first version. Although far from its ideal size in terms of coverage, it already offers a coherent and comprehensive representation of the *dastgāhi system*. Currently, KCD contains 213 solo recordings by four professional musicians, 65 of which consist of complete *guše* performances and hence form the core of the corpus.

In order to analyse the corpus, we draw on the five criteria defined by Serra [21]:

Purpose: As described in section 1, KDC is created for the analysis of Iranian *dastgāhi* music using corpus driven computational methods. Due to the lack of a notational system originally devised for this music tradition, and the difficulties of staff notation for capturing key elements of this music, especially regarding tuning, intonation and ornamentation, KDC primarily aims at gathering audio recordings. However, the inclusion of other data types, such as scores, lyrics, videos, images, etc., that might contribute to the general purpose of the corpus, is not excluded. A key point in the creation of KDC is the close collaboration with musicians. They do not only contribute with recordings and annotations, but also with their knowledge, evaluation, and research ideas. Musicians are thus considered as active collaborators of KDC.

Coverage: According with KDC's purpose, we focus on voice and melodic instruments. In this first version of the corpus, being one of our goals to test state-of-the-art technologies for the analysis of this music tradition, we have focused on solo performances, and consciously excluded ensemble performances, that would present a greater challenge for computational analysis. In order to

		Performance		*Šāhed*	*Ist*	Pitch Set	*Radif*	
Dastgāh	*Čāhārgāh*	7	9:02	5	3	2	5	5:21
	Homāyun	5	7:35	5	4	2	3	3:50
	Māhur	6	8:37	5	2	2	3	2:03
	Navā	6	9:19	5	4	2	3	4:01
	Rāst-Panjgāh	4	5:56	4	3	2	2	2:15
	Segāh	5	8:57	5	2	2	3	2:12
	Šur	7	9:55	5	2	2	3	2:12
Āvāz	*Abuatā*	5	6:47	5	3	1	2	2:01
	Afšāri	5	9:36	5	4	1	2	1:56
	Bayāt-e Tork	5	8:03	5	3	1	2	1:47
	Dašti	5	6:53	5	4	1	2	2:10
	Esfahān	5	9:21	5	3	2	2	2:40
	Total	65	100:01	59	38	20	31	31:40

Table 1. Number of recordings in KDC according to *dastgāh* and *āvāz*. For performance and *radif* recordings, the duration is also given.

Inst.	Musician	Rec.	Dur.
voice	F. Sahebghalam, R. Zalpour	24	37:50
ney	R. Zalpour, M. Khodadadi	37	45:54
setār	M. Shaari	35	47:57
Total		96	131:41

Table 2. Number of performance and *radif* recordings in KDC according to instrument.

build a first dataset that provides a coherent and comprehensive representation of the *dastgāh* system, this first version of KDC contains the first *guše* of all 7 *dastgāh*s and 5 *āvāz*es, known as *darāmad*, which holds great significance (see section 1.1). Four professional musicians (see Table 2) have contributed to KDC with original recordings of complete performances [1] of these 12 *darāmad*s. Hence, the corpus currently contains five full renditions of the 12 *darāmad*s, namely, two vocal ones, one female and one male (with the exception of the *darāmad* of *dastgāh rāst-panjgāh*, whose female vocal rendition could not be recorded), two by *ney*, and one by *setār*. According to their own decision, the musicians occasionally recorded more than one version of a particular *darāmad*, and all of them are included in the corpus. This results in 65 performance recordings, which is the core of KDC (see Table 1).

Completeness: KDC includes a csv file with metadata and annotations for all recordings, including the corresponding *dastgāh*, *guše*, artist, instrument, recording type, and duration. Considering the methodological purpose of the corpus, the musicians who collaborate with KDC were asked to record the *šāhed* of the performed *darāmad* as a single, stable pitch. The 59 resulting recordings are included in KDC, and they were used to compute the frequency of the *šāhed*, [2] which is added as an annotation in the metadata file. The musicians who collaborate with KDC show a great interest in the project and actively propose contributions to it, according to their understanding of which data could benefit the purpose of KDC. As a consequence, some musicians recorded the *ist* of the performed *darāmad* in a separate file as a single, stable pitch, resulting in 38 recordings, the corresponding pitch set, resulting in 20 recordings, and one or more related performances according to *radif* (see Table 1). Since pattern analysis is a key task for the study of Iranian *dastgāh*s, two of the collaborator musicians manually annotated the characteristic patterns that identify the performed *guše* using the software Praat [23]. Each of them annotated two full renditions of the 12 *darāmad*s, so that only the *setār* recordings are still not annotated in terms for patterns.

Quality: all the recordings in KDC are provided in the lossless compression format flac. The recordings by M. Shaari and F. Sahebghalam were recorded in a professional recording studio at KUG, those by M. Khodadadi were recorded by the first author using a Zoom H4 recorder, and those by R. Zalpour by himself using his cell phone. The musicians, all having previous recording experience, evaluated the quality of their own recordings both in terms of sound and performance quality and agreed to include them in KDC as good representatives of their art. To validate the pitch frequencies of *šāhed*, mp3 files were generated, and R. Zalpour assessed their correctness.

Reusability: All the collaborator musicians contributed their recordings to KDC for them to be published under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0). [3] In order to ensure the reusability of KDC, we follow the FAIR principles. [4] To ensure that all recordings are findable and reusable, they are made available through the KUG's Phaidra repository. [5] Regarding accessibility, the

[1] Given its improvisational nature, there is no standard rendition of a particular *guše*. By complete performance we mean a rendition that completely conveys the performed *guše*.

[2] The pitch of the *šāhed* was computed with the pYin: Notes plugin [22] of Sonic Visualiser

[3] https://creativecommons.org/licenses/by-nc-nd/4.0/legalcode

[4] https://www.go-fair.org/fair-principles/

[5] The following Phaidra collection is created for KDC: https://phaidra.kug.ac.at/view/o:127195

metadata of all recordings are also stored in MusicBrainz [6] in its original Farsi and with English aliases, and linked to their corresponding Phaidra objects. A specific protocol for interoperable structuring of Iranian *dastgāhi* music is still being developed, and is expected to be evolving along with the expansion of the corpus.

3. RESEARCH POTENTIAL OF KDC

The hypothesis that motivated the KDC project is the conviction that musicological study of Iranian *dastgāhi* music can deeply benefit from computational methods. Conversely, we argue that the characteristics of this music raise interesting research tasks for music information retrieval. In this section we point out some of the most relevant of these tasks. Since KDC was created with the purpose of researching the modal aspect of this music, tasks related to this domain are described in more detail.

3.1 Melody and mode

The study of melody in Iranian *dastgāhi* music is one of the most essential tasks, especially for the purpose of KDC, therefore pitch track extraction is essential. According to our preliminary analyses, state-of-the-art algorithms perform satisfactorily for monophonic recordings as those by voice or *ney*. However, further research is still needed to obtain equally satisfactory results for instruments with high resonance as the *setār* (see section 4). A relevant research task based on these pitch tracks is the corpus driven analysis of vibrato and *tahrir* (see Fig. 5). Their automatic identification and measurement would contribute to their deeper understanding, but also to better define school styles, or idiomatic performance for specific instruments.

Tasks conducing to the characterization of *guše*s are also of key importance. A particularly relevant one is automatic pattern analysis. A main challenge for this task from the computational point of view is evaluation. In order to contribute to that, KDC includes manual annotations by expert musicians. Analyses of each *guše*'s tonal hierarchy are also essential, especially those contributing to the understanding of concepts like *šahed* or *ist* (see section 1.1).

Given the classification system of *guše*s, *dastgāh*s and *āvāz*es, this music tradition is well suited for automatic recognition and classification tasks, which in fact are the ones most commonly explored to date (see section 1.2).

Finally, even though Iranian *dastgāhi* music is not easily represented on staff notation, a tradition of transcriptions using this notation system exists. This music then can be also taken as a challenging case for automatic transcription, for which existing scores (currently only in print) can be used for evaluation.

3.2 Other musical aspects

Iranian *dastgāhi* music offers interesting tasks for rhythm research. In metred performance, automatic *dowr-e iqāi* detection becomes a significant challenge, given the heterophonic nature of ensemble performance and the timbral and stylistic characteristic of solo performance. Expressive microtiming deviations from isochrony is a meaningful analysis for characterising performance style. Unmetred performance of *dastgāhi* music offers a great opportunity for multimodal analysis combining audio and lyrics analysis (this would require incorporating text data to KDC). Natural Language Processing methods can be applied to model the prosodic features of *aruz*, that can later be mapped to music performance.

Timbre is a very diverse and expressive feature of Iranian *dastgāhi* music performance. Combined with nuances in dynamics, singers use vowel shades to introduce variability in sustained notes. Equally, *ney* performers use a wide range of timbres from bright, clean ones to raspy, breathy ones. Automatic analysis of the timbral categories can inform about stylistic preferences and if a relationship exists with specific *guše*s. The use of dynamics is another important element in Iranian *dastgāhi* music expressivity, and its computational analysis can shed light about its systematic or stylistic use.

4. PRELIMINARY ANALYSES

In order to test the potential of KDC as described in the previous section, we carried out a series of preliminary analyses. The current size of KDC does not allow to obtain general conclusions about Iranian *dastgāhi* music. Our aim is to explore the possibilities for computational methods, and therefore to suggest further lines of research for future stages of KDC.

Since our main interest is the melodic dimension of *dastgāhi* music, our analysis started with pitch track extraction. To that aim, we applied the CREPE algorithm [24] to all monophonic recordings of KDC, that is, those by voice and *ney*. After visual and aural examination, we concluded that state-of-the-art algorithms, both for monophonic and polyphonic signals, [7] do not produce results usable for further analysis on the recordings of *setār*, an instrument with high, and sought after, resonance and common use of chords. Consequently, the recordings by *setār* were excluded for these analyses. In order to ensure reproducibility, the data, metadata and annotations used for these analyses are shared as a dataset. [8] The code and resulting plots are also shared in a public repository. [9]

4.1 *Šāhed*

In order to study *šāhed*, we compute pitch histograms from the previously extracted pitch tracks using the algorithm developed in [27]. The histograms are computed in cents aligned to the *šāhed* that KDC has annotated for each individual recording. We also compute pitch histograms for the aggregated pitch tracks of all recordings of the same

[6] The following MusicBrainz collection is created for KDC: https://musicbrainz.org/collection/6ee214c7-f116-4691-b46d-36033c8b17b0

[7] Pitch extraction was attempted on the *setār* recordings using the algorithms CREPE [24], pYin [22] and Melodia [25], the last two in their Sonic Visualiser [26] plug-in version.

[8] https://phaidra.kug.ac.at/o:127202

[9] https://github.com/Rafael-Caro/KDC-v1.0

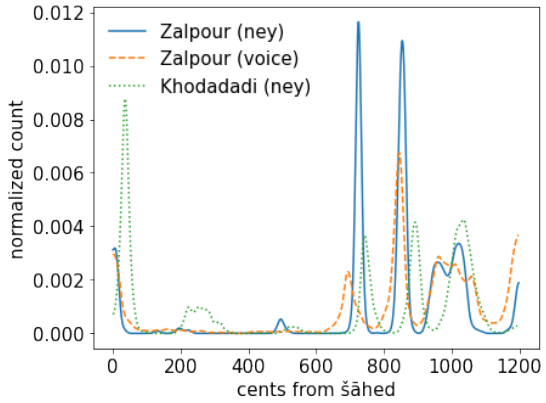


Figure 1. Folded pitch histograms of three recordings of *darāmad abuatā*

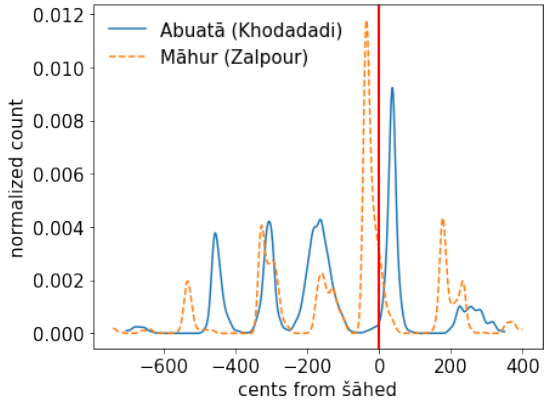


Figure 2. Pitch histograms of two *ney* recordings. The vertical line indicates the *šāhed*

dastgāh. Besides, histograms folded to one octave are also computed for individual recordings and for *dastgāh*s.

One of the most discussed issues in Iranian classical music is the concept of *šāhed*. Different definitions for this concept can be found in the existing literature. According to Farhat, "[i]n most of the Persian modes one tone assumes a conspicuously prominent role. It may or may not be the finalis" [1]. Nooshin argues that *šāhed* is "[t]he most prominent pitch, functioning as the tonal center of the mode" [28], while Talai defines it as "[t]he most frequently repeated pitch" [29]. We argue that corpus driven analyses can help to examine which of the definitions for *šāhed*, namely, "prominent" tone, "tonal center" or "most frequently repeated" tone better represents our results.

Fig. 1 shows the folded pitch histograms of three different performances of the same *guše*, *darāmad abuatā*. It can be seen how the *šāhed* is the most frequent tone in M. Khodadadi's performance, but that is not the case in the two ones by R. Zalpour, both vocal and *ney*. This preliminary result, together with other examples not mentioned here, suggests a more nuanced understanding of *šāhed*, that would escape a unique, universally applicable definition.

Mus. (inst.)	Mean	SD	Lowest	Highest
Saheb. (voice)	4.19	10.28	-10.11	28.10
Zalp. (voice)	6.92	7.26	-5.02	16.46
Zalp. (*ney*)	-3.19	16.87	-35.43	23.99
Khod. (*ney*)	13.95	13.70	-7.69	37.07

Table 3. Mean and SD of the differences in cents between the *šāhed*'s pitch in their isolated recordings and in performance per musician and instrument. The difference between the lowest and highest performance of the *šāhed* compared with the isolated recording is also given in cents.

4.2 Intonation

One interesting issue is the musicians' abstract notion of the *šāhed*'s pitch and its actual intonation in performance. Since all the collaborator musicians recorded the *šāhed* of their performances as an extra, isolated file according to their aforementioned abstract notion, we can compare its pitch in both contexts. Fig. 2 shows how M. Khodadadi performs the *šāhed* 37.07 cents higher than the isolated version he recorded himself, whilst R. Zalpour performs it 35.43 cents lower. We computed the difference between isolated *šāhed* and its performed version (obtained from the value of the closest peak in the histogram) for all recordings (excluding *setār*), and calculated the mean and SD for each musician and instrument, as it can be seen in Table 3. The results show a great variability in the divergence between conceptualized and performed *šāhed*, frequently reaching easily audible deviations.

These observations points to an interesting line of future research regarding the perception of intervallic structures by musicians. The specific width of the intervals of a particular *guše*, an essential element for the conceptualization of Iranian *dastgāhi* music, is part of the implicit knowledge of each individual musician, enacted mainly during performance. Their intellectual formulation, as the examples here analysed suggest, might no correspond with this implicit knowledge.

4.3 Pitch alteration

In some *gušes* specific degrees can be rendered by two neighbouring pitches. For example, in the case of *darāmad afšāri*, the degree above the *šāhed* can be performed at 119 cents above it, which is considered the main tone for this degree, or at 204 cents over the *šāhed*, which is considered an altered version of the same degree, as it can be seen in Fig. 3 (cent values computed according to [1]). These altered pitches are known as *not-e moteġayer* [1,30]. The use of this pitch alteration can be observed in pitch histograms as flat peaks, as it can be seen in Fig. 4 around 200 cents. These results point to another interesting line of research, for the computational detection of *not-e moteġayer* that can contribute to the better characterization of *gušes*.

4.4 Vibrato and *tahrir*

In order to analyse vibrato, we ran the Vibrato algorithm available in the Essentia library [31] on all recordings, and

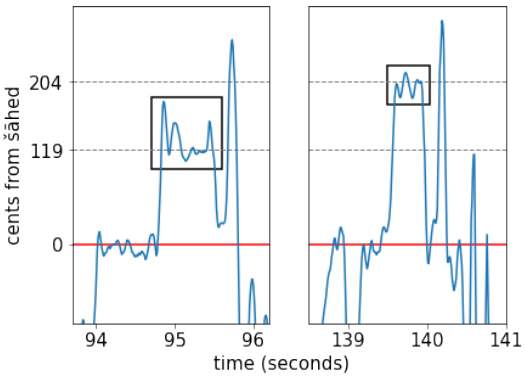


Figure 3. Two excerpts of the the pitch track of the of the vocal recording of *darāmad afšāri* by F. Sahebghalam. The horizontal red line indicates the *šāhed*, the gray line at 119 cents indicates the main pitch, whose corresponding *not-e moteġayer* is indicated by the gray line at 204 cents. The boxes mark the performance of these two notes.

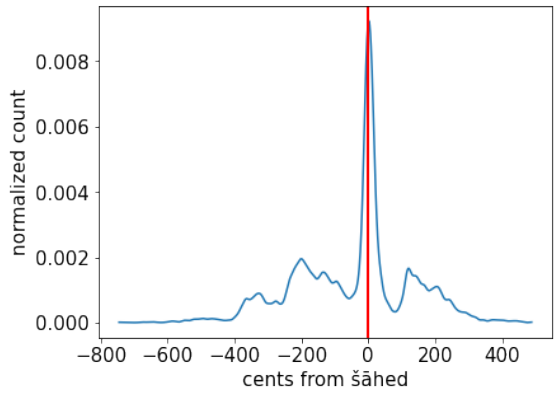


Figure 4. Pitch histogram of the vocal recording of *darāmad afšāri* by F. Sahebghalam. The vertical line indicates the *šāhed*

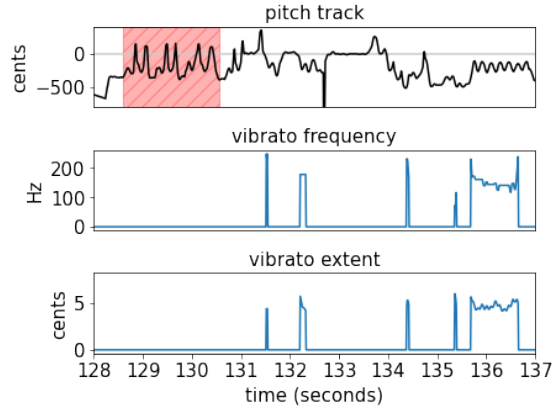


Figure 5. Excerpt of the recording of *darāmad abuatā* by F. Sahebghalam. Upper plot shows pitch track (horizontal line indicates *šāhed*), middle plot detected vibrato frequency, and lower plot detected vibrato extent. Highlighted section shows *tahrir*.

the results were plotted against the pitch track (see Fig. 5 for an example). These results, stored as csv files, together with the plots for all recordings are available in the code repository. Through visual evaluation of the results we argue that further research is required in order to improve automatic detection of vibrato for this tradition.

Dastgāhi music is an idiomatic tradition. Vocal and instrumental performances are characterized by specific techniques and sound qualities, being vibrato a relevant one. Its extent, frequency and occurrence are idiomatically specified by instrument, school or personal style. Automatic vibrato detection and measurement can contribute to characterizing idiomatic styles. Another important vocal technique is *tahrir*, which consists in wide cyclic pitch alteration of a particular tone in terms of pitch and timber. The pitch fluctuation is not realized as glissandi but jumps, and this might explain why it is not detected by the used algorithm, as shown in Fig. 5. Analysing the difference between these two techniques would help to understand the idiomatic practices in Iranian *dastgāhi* music.

5. CONCLUSIONS AND FUTURE WORK

With KDC we set a very ambitious goal: to create an open, well curated, ever growing corpus of Iranian *dastgāhi* music data, metadata and expert annotations which can contribute to the development of a coherent line of computational research for this music tradition. This is a task that requires collaborative efforts from musicologists, musicians and computer engineers. KDC is still in its infancy, but its first version presented in this paper is complete enough to test state-of-the-art methodologies, obtain preliminary results, and consequently propose new directions for future research.

The expansion of KDC is and is going to be a constant future task. As any other aspect of our lives in the past two years, the COVID-19 pandemic has hindered the development of KDC with the suspension of international travels (economic sanctions against Iran prevent working with musicians living in the country from outside). Besides the addition of recordings that cover new *gušes*, instruments and schools, increasing the number of expert annotations is one of our short term goals. In a first stage, we aim at having the 5 renditions of the whole set of 12 *daramād*s annotated at least by three different musicians, so that we can start studying this phenomenon from the perception of the performers.

Regarding computational analysis, a short term goal is to seek collaborations to perform automatic pattern analysis. This is a fundamental element for the characterization of *gušes*, and most of our collaborator musicians have highlighted the importance of this task and their interest in it. The annotations currently existing in KDC are aimed to this goal. Other tasks that are of our interest, and for which collaboration with computer engineers is also necessary, is the improvement of pitch track extraction for instruments with great resonance as *setār*. Finally, collaboration will also be sought for the development of methods for automatic *tahrir* detection and analysis.

6. ACKNOWLEDGEMENTS

The authors would like to express their sincerest gratitude and acknowledgement to their collaborator musicians Masoud Shaari, Farahnaz Sahebghalam, Reza Zalpour and Mohammad Khodadadi, not only for contributing their recordings and ideas to KDC, but also for their enthusiastic support to the project. We would also like to thank Dr. Negar Bouban and Reza Zalpour for their very valuable manual annotations of melodic patterns. Finally, we thank Kunstuniversität Graz for the financial support that makes KDC possible.

7. REFERENCES

[1] H. Farhat, *The Dastgāh Concept in Persian Music*. Cambridge and New York: Cambridge University Press, 1990.

[2] T. Tsuge, "Rhythmic aspects of the Âvâz in persian music," *Ethnomusicology*, vol. 14, no. 2, pp. 205–227, 1970.

[3] F. Amoozegar-Fassie, "The poetics of persian music: The intimate correlation between prosody and persian classical music," Ph.D. dissertation, Faculty of Graduate Studies, University of British Columbia, UK, 2008.

[4] M. Azadehfar, *Rhythmic structure in Iranian music*. Tehran: Tehran University Press, 2017.

[5] N. Bouban, "Barresi-ye Āvāi-ye ritm dar pāyehā-ye vājegan-e š'er va nasr-e fārsi-ye rasmi [the phonetic study of rhythm in units of poems and writings in standard farsi]," *Majaleh-ye pajuheš-hā-ye zabānšenasi*, vol. 1, no. 1, pp. 101–122, 2010.

[6] P. Heydarian, "Music note recognition for santoor," Master's thesis, Tarbiat Modarres University, Iran, 2000.

[7] P. Heydarian and J. D. Reiss, "The persian music and the santur instrument," in *Proc. of the 6th Int. Society for Music Information Retrieval Conf.*, London, UK, 2005, pp. 524–527.

[8] P. Heydarian and L. Jones, "Measurement and calculation of the parameters of santur," in *Proc. of the Annual Conf. of the Canadian Acoustical Association (CAA)*, Vancouver, Columbia, 2008.

[9] ——, "Tonic and scale recognition in persian audio musical signals," in *12th Int. Conf. on Signal Processing (ICSP)*, Hangzhou, China, 2014.

[10] P. Heydarian, L. Jones, and A. Seago, "Automatic mode estimation of Persian musical signals," in *133rd Audio Engineering Society Convention*, San Francisco, USA, 2012.

[11] P. Heydarian, "Automatic recognition of persian musical modes in audio musical signals," Ph.D. dissertation, Faculty of Art, Architecture and Design, London Metropolitan University, UK, 2016.

[12] P. Heydarian and D. Bainbridge, "Dastgàh recognition in Iranian music: Different features and optimized parameters," in *6th International Conference on Digital Libraries for Musicology (DLfM '19), November 9, 2019, The Hague, Netherlands.*, 2019, pp. 53–57.

[13] N. Darabi, N. H. Azimi, and H. Nojumi, "Recognition of dastgah and maqam for persian music with detecting skeletal melodic models," in *Proc. SPS-DARTS 2006-The second annual IEEE Benelux/DSP Valley Signal Processing Symposium*, Antwerp, Belgium, 2006.

[14] S. Abdoli, "Iranian traditional music dastgah classification," in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011*, 2011, pp. 275–280.

[15] M. A. Layegh, S. Haghipour, and Y. N. Sarem, "Classification of the radif of mirza abdollah a canonic repertoire of persian music using svm method," *Gazi University Journal of Science*, vol. 1, no. 4, pp. 57–256, 2013.

[16] L. Ciamarone, B. Bozkurt, and X. Serra, "Automatic Dastgah Recognition Using Markov Models," in *Proceedings of the 14th International Symposium on Computer Music Multidisciplinary Research*, 2019, pp. 51–58.

[17] S. Shafiei, "Analysis of vocal ornamentation in iranian classical music," in *Proc. of the 16th Sound and Music Computing Conf.*, Málaga, Spain, 2019, pp. 437–441.

[18] F. Sanati, "An investigation on the value of intervals in persian music," Master's thesis, Department of Music, University of Jyväskylä, Finland, 2020.

[19] S. Malekzadeh, M. Samami, S. R. Azar, and M. Rayegan, "Classical music generation in distinct dastgahs with alimnet ACGAN," *CoRR*, vol. abs/1901.04696, 2019.

[20] P. Heydarian and J. D. Reiss, "A database for persian music," in *Proc. of the Digital Music Research Network Summer Conference (DMRN 2005)*, Glasgow, UK, 2005.

[21] X. Serra, "Creating research corpora for the computational study of music: the case of the compmusic project," in *53rd AES International Conference on Semantic Audio*, New York, USA, 2014, pp. 1–9.

[22] M. Mauch and S. Dixon, "pYIN: A fundamental frequency estimator using probabilistic threshold distributions," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2014)*, Florence, Italy, 2014, pp. 659–663.

[23] P. Boersma and D. Weenink, "Praat: doing phonetics by computer [computer program]," 1992–2022. [Online]. Available: https://www.praat.org

[24] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, Canada, 2018, pp. 161–165.

[25] J. Salamon and E. Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, pp. 1759–1770, 2012.

[26] C. Cannam, C. Landone, and M. Sandler, "Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files," in *Proceedings of the ACM Multimedia 2010 International Conference*, Firenze, Italy, October 2010, pp. 1467–1468.

[27] G. K. Koduri, V. Ishwar, J. Serrà, and X. Serra, "Intonation analysis of rāgas in carnatic music," *Journal of New Music Research*, vol. 43, pp. 73–94, 2014.

[28] L. Nooshin, "The song of the nightingale: Processes of improvisation in dastgāh Segāh (Iranian classical music)," *British Journal of Ethnomusicology*, vol. 7, no. 1, pp. 69–116, 1998.

[29] D. Talai, "A New Approach to the Theory of Persian Art Music: The Radīf and the Modal System," in *Garland Encyclopedia of World Music Volume 6: The Middle East*, V. Danielson, S. Marcus, and D. Reynolds, Eds. New York: Routledge, 2001, pp. 894–903.

[30] H. Asadi, "Theoretical foundation of persian classical music: Dastgāh as a muti-modal cycle," *Mahoor*, vol. 6, pp. 43–56, 2004.

[31] D. Bogdanov, N. Wack, E. Gómez Gutiérrez, S. Gulati, H. Boyer, O. Mayor, G. Roma Trepat, J. Salamon, J. R. Zapata González, and X. Serra, "Essentia: An audio analysis library for music information retrieval," in *Proc. of the 14th Int. Society for Music Information Retrieval Conf.*, Curitiba, Brazil, 2013, pp. 493–498.

INACCURATE PREDICTION OR GENRE EVOLUTION? RETHINKING GENRE CLASSIFICATION

Ke Nie
University of California, San Diego
knie@ucsd.edu

ABSTRACT

The existing MIR research on genre classification primarily focuses on how to classify a song into the "correct" genre while downplaying the fact that genres mutate over time and in response to social change in terms of their musical properties. Songs claiming the same genre can sound very different if they are released years apart, and genres may revive musical traditions from the past. In this paper, I show that the performance of genre classifiers fluctuates as genres evolve. Unsatisfactory performance of the classifiers may not indicate algorithmic flaws but rather the change of genre characteristics. I demonstrate this by studying the case of Chinese Hip-Hop music. Specifically, I collected and analyzed 69,427 songs from four genres (Hip-Hop, Pop, Rock, and Folk) released on a Chinese music platform between 2009 and 2019. Using classifiers trained from the songs in different year cohorts to predict the genre of all the songs, I show how genre classifiers can be used to detect the stylistic shift in Hip-Hop that happened during this period. The paper thus offers a novel, sociological perspective on contending with the much-challenged idea of improving genre classification accuracy for its own sake. However, instead of questioning the effort, I argue that MIR research on genre classification can be helpful for studying genre as a social construct and cultural phenomenon if the pursuit of prediction performance and the cultural meaning of inaccurate prediction are carefully balanced.

1. INTRODUCTION

Past MIR research on genre classification has primarily strived to find ways for algorithms to correctly detect music genres. Numerous studies have experimented with various data sources, algorithmic approaches, and evaluation metrics to improve algorithmic attempts to grasp the characteristics of music genres [1-7]. Most of the existing studies rely on fixed datasets and metrics to evaluate the performance of classifiers for the convenience of comparison.

One major concern in this area, however, is that genre is an ambiguous concept [8]. After all, what distinguishes one genre from another is *subjective*, since it is "based upon subjective responses with little inter-participant consensus" [3]. This is because the boundaries according to which genres are distinguished are not entirely rooted in the musical or sonic elements; rather, they are based on people's understanding of the difference between genres, which depends on their cognitive perception and cultural knowledge [3,9]. In fact, research has found that human evaluators may also be ambivalent about the classification of genres, which problematizes the very idea of genre and challenges the purpose of classification tasks [10, 11].

Adding to this line of questioning, this paper underscores the social dimension of genres that further complicates the concept by exploring critical implications for the design and implementation of genre classifiers. I argue that genres are social constructs that constantly change over time and in response to social, economic, and political pressure [12-15]; therefore, if genres evolve, the performance of genre classifiers trained on songs from precedented cohorts will fluctuate in predicting future cohorts. Given this feature, I argue that the "inaccurate" prediction of genre classifiers can be used to detect and map the evolution of genres. I demonstrate this point by presenting a case study of Chinese Hip-Hop music. Using songs from different year cohorts as the training set, I show that the performance of the classifiers mutates over time as the genre evolves. The findings demonstrate that prediction accuracy may not be the most valuable feature of algorithmic classifiers and that inaccurate predictions may suggest genre evolution.

This paper is organized as follows. In Section 2, I review the current MIR research on genre classification tasks and identify the common concern shared in this research area regarding the ambiguity of the genre concept. I propose taking a deeper look at the social dimension of genres, a generally understudied subject in the MIR field, as a way to advance our understanding of the evaluation of genre classifiers. In Section 3, I present the case of Hip-Hop music in China, a genre that has enjoyed a dramatic turn of events in the past decade. Drawing from different pieces of evidence, in Section 4 I show how the evolution of genre could complicate the performance of genre classifiers and how classifiers can be used to help understand the development of a genre. I then conclude the paper by discussing the impacts of genre evolution on the design and performance of genre classifiers.


© K.Nie. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** K. Nie, "Inaccurate Prediction or Genre Evolution? Rethinking Genre Classification", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.*, Bengaluru, India, 2022.

2. RELATED WORK

The existing MIR literature on genre classification is primarily aimed at exploring, devising, and experimenting with algorithmic designs that could be used to understand and predict music genres automatically. These studies can be roughly categorized into three main themes.

First and foremost, researchers have attempted to identify different kinds of *data sources* that are useful for earmarking the characteristics of music genres, such as music and sonic features extracted from the digital signals of the songs [1], as well as information on the performers, musicians, communities, and other social-cultural features associated with the genre [2]. Second, contingent upon the data sources mentioned above, researchers have been trying to improve on the *algorithmic approaches* to achieve better performance in genre classification tasks by experimenting with various types of acoustic representations (MFCCs, spectrograms, etc.) and machine learning architectures (SVM, CNN, etc.) [4,5,6,7,8,16]. The primary goal here is to improve the accuracy of the classification algorithms, which are usually tested on an existing dataset.

Studies aligned with the above two themes have explored pathbreaking methods through which genre classification tasks are performed at higher accuracy. Nevertheless, some scholars have also reflected on the *evaluation metrics* of genre classifiers, arguing that "accuracy is not enough," as genre classification is essentially a task of musical recognition rather than classification accuracy [17,18]. Furthermore, researchers highlight the ambiguity of the concept of "genre" itself, pointing out that the notion is essentially "subjective" [3]; therefore, human evaluation needs to be included in the process [19]. While there is suspicion about the validity of genre classification tasks due to the ambiguity of the genre concept, most scholars still acknowledge the relevance of the research while looking for instruments to mitigate the problem of subjectivity [11]. Previous studies from this third, reflective approach have shared incisive thoughts on how to refine algorithmic frameworks for better classification performance.

Following past reflections on this issue, I want to further point out that genre is a social construct that depends on mutual agreement among evaluators or the community as a whole. This view is close to that of the social scientists and business scholars of the music industry, who have long argued that genres are classificatory apparatuses used by producers, consumers, and intermediaries to make sense of a distinct type of music [12-14]. This sociological view of genre claims that while music content may matter in the identification of genre, genres are, above all, divided by communities. This helps explain cases where different genres sound similar to each other or where the same genre may sound very different in different regions. For example, sociologist William Roy pointed out that "hillbilly music" and "race records" in the 1920s America were two genres that were associated with the same type of music, although the former was used as a market label for African American audience while the latter was for the "rural whites" due to segregations at the time [20]. They are, nevertheless, considered two disparate "genres" as they are tied to different communities.

A more important component of the sociological view of genre that is usually underplayed in the MIR research is that genres can evolve. In other words, the way in which genre classification systems upon which producers, consumers, and intermediaries agree to distinguish different types of music can change across time and localities [12,13]. Additionally, genre categories may also evolve in response to external shocks to the industry or the social environment in general. One example is the case of censorship, in which censored genres have to change their content in response to political pressure [15]. Similarly, advancements in technology or sound devices also help to define or shift music genres, such as the use of the Roland TR-808 as the defining sound of early-1980s Hip-Hop music. The idea here is that genres are constantly shaped and reshaped by social, economic, and political forces; therefore, they cannot be seen as fixed, immutable entities.

The mutability of genre adds a further question to the design of genre classification algorithms in terms of their data sources, algorithmic architecture, and evaluation metrics. This paper attempts to address the issue of evaluation specifically. If genres evolve over time, one should then expect that the classification accuracy of a genre classifier will drop when it is used to predict the songs that are released a long time apart from the songs that were used to train the classifier. Specifically, we may expect that a classifier that was trained on a set of songs released in the 1980s will have a harder time predicting the correct genre of songs from the 2010s than songs from the 1990s. In this case, the accuracy metric is not useful for understanding the performance of the classifier. Rather, the evaluation metrics may have a richer meaning than simply as a criterion for making the genre distinction: they may be used to understand how genre evolves or how genres influence each other. For example, inaccurate predictions may not necessarily indicate algorithmic flaws; they may imply that the actual genre is moving toward the predicted genre in terms of its sound. The trickiest part of making such a claim, however, is how to distinguish genre evolution from the drawbacks of the algorithmic design when prediction accuracy is low. This will be discussed further in the analysis and discussion sections below.

3. DATA & METHOD

To demonstrate how genre evolution may confound the performance of genre classifiers, I study the case of Chinese Hip-Hop music from the past decade. Hip-Hop has long been a relatively low-profile genre in China since it was introduced to the country in the 1990s [21]. In recent years, however, Chinese Hip-Hop experienced a series of dramatic external shocks, which elevated the genre's prominence extraordinarily. In summer 2017, the first season of *The Rap of China* started to stream on iQiyi, one of the most popular streaming platforms in China. The reality-show-style music competition attracted an unexpectedly expansive audience, securing 2.7 billion views in the

course of three months and becoming the most successful online program ever in the history of the sector. The contestants featured in this program, as well as their music, soon became the iconic representatives of an unprecedented fad. Most significantly, many of the high-profile contestants, including the co-winners of the competition, GAI and PG One, were known for their involvement in Trap music, a style that originated in the Southern United States and is characterized by its distinct use of synthesized drums (above all, Roland 808 drums). Moreover, it is remarkably different from the jazzy, pop-oriented Hip-Hop style prevalent in the genre before. In January 2018, Hip-Hop's rise was met with state censorship aimed at blocking Hip-Hop musicians from mainstream media, as the state was concerned about past Hip-Hop figures' problematic practices of including subversive and delinquent content in their songs. Although many musicians had to remove some of their songs from the internet under political pressure, Hip-Hop songs, in general, were still allowed to circulate online via music streaming platforms, and the censorship loosened up six months later without any official announcement [15].

The story of Chinese Hip-Hop music thus provides an interesting case for investigating the issue of genre evolution, as the genre became remarkably popular and potentially turned in a new stylistic direction due to the success of *The Rap of China*. To understand the impact of this stylistic change on machine learning classifiers, I collected songs from one of the most popular music streaming platforms in China. I collected the songs' audio files, as well as their metadata, including their genre and tags. The platform's setup requires each song to have only one identified genre, while the song can have multiple tags that are considered its "subgenres." Because of copyright issues, only a portion of the available songs could be downloaded at the time of the data collection in September 2019. Eventually, I collected 69,427 songs released between 2009 and September 2019 from four genres (Hip-Hop, Pop, Rock, and Folk), constituting the dataset on which I conducted my analysis. The other three genres are also well established in China in that they have distinct fanbases, although the Chinese Hip-Hop community was more connected to the Rock community particularly in the early 2000s for sharing resources and venues [22].

The structure of the dataset is given in Figure 1 below. As shown in the graphic, the number of total song releases consistently increases over the years, with the exception of the year 2019, when the data collection was suspended. The data and the code for the following analysis are publicly available,[1] although the metadata of the songs is left out due to copyright issues.

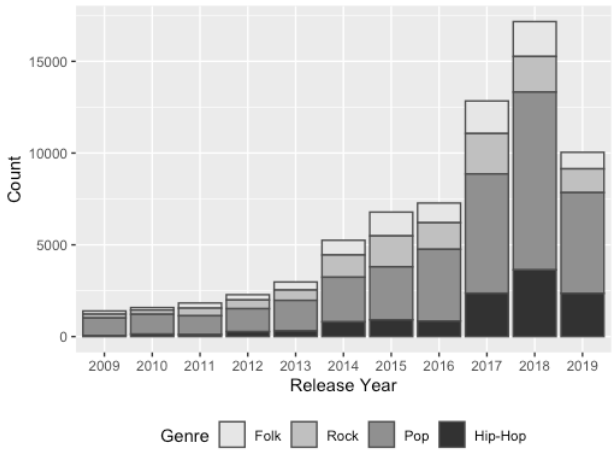


Figure 1. The structure of the dataset. The height of the bar represents the number of releases.

I used librosa [23] to extract sonic features from the audio files to prepare for training the classifiers. As my goal was not to improve classification accuracy, I followed Tzanetakis & Cook's [1] classic approach for computational efficiency. Specifically, I sliced a 1-minute long excerpt of audio signals from each song starting at 30 seconds into the song, which supposedly captures a substantive part of the song, and extracted 11 timbral texture features from this slice, including the chroma features (chroma_stft), root-mean-square of the spectrogram (rms), spectral centroid (spectral_centroid), spectral bandwidth (spectral_bandwidth), spectral roll-off (spectral_rolloff), zero-crossing rate (zero_crossing_rate), and the first five Mel-frequency cepstral coefficients (MFCC). I normalized the value of the features using a 0-1 scale and took the means of the features for each song, resulting in an 11-dimensional feature vector.

I used the songs released in 2009 and 2018, respectively, to train the genre classifiers. They represent the first and the last full year in the dataset, a fact that can be conveniently used to track changes over the year. I also used the widely studied GTZAN dataset [1] — its filtered version [24] due to the errors in the original collection [25] — as a supplementary check of the classifiers' performance trained on a dataset from a different time and region. For each of these three cohorts, I trained four classifiers: a Gaussian Naïve Bayes classifier (hereafter, GNB), a K-Nearest Neighbor classifier (hereafter, KNN), a Random Forest classifier (hereafter, RF), and a classifier trained by Extreme Gradient Boosting, a more recently prevailing tree-based algorithm. They were used in parallel to triangulate the results. For the 2009 cohort, I picked the top 50 most streamed songs of each genre, for a total of 200 songs, to compile the training set, whereas for the 2018 cohort, I picked the top 1900 songs of each genre, totaling 7600 songs. My reason for choosing these two numbers is that they are the closest number that can be divided by ten to the total number of releases of a genre with the least releases in that year (54 Hip-Hop songs in 2009; 1902 Folk

[1] The data can be found at https://github.com/norzvic.

songs in 2018), making it convenient for a 4:1 split for training and testing the classifiers. For the GTZAN dataset, I only used the data for three genres — Pop, Rock, and Hip-Hop — to train the classifiers, as there is no "Folk" in the GTZAN dataset. I extracted the data on the 11 features, normalized them, and used all 300 songs from the three genres in the dataset to train four classifiers similarly. The performance of the classifiers is shown in Table 1. I then used all four sets of classifiers to predict the genre of all the songs in the dataset, the results of which are reported in the next section.

		2009	2018	GTZAN
GNB	Acc	0.575	0.472	0.750
	AUC	0.832	0.755	0.867
KNN	Acc	0.625	0.584	0.800
	AUC	0.646	0.578	0.608
RF	Acc	0.800	0.580	0.833
	AUC	0.908	0.786	0.927
XGB	Acc	0.700	0.629	0.732
	AUC	0.905	0.864	0.916

Table 1. The performance of the classifiers across cohorts. Acc refers to the accuracy of predicting the test set, whereas AUC refers to the corresponding area under the ROC line for multi-classes.

4. FINDINGS

4.1 Overall Accuracy and Recall

As the focus of this study is Hip-Hop, I will primarily heed two metrics: Hip-Hop accuracy and Hip-Hop recall. Here, Hip-Hop accuracy refers to the rate at which the classifier correctly identifies Hip-Hop versus non-Hip-Hop, whereas Hip-Hop recall refers to the rate at which the classifier correctly identifies Hip-Hop among true Hip-Hop songs. The reason to incorporate recall is that the metric can be useful to detect potential genre evolution: the lower the recall is, the more probable that a Hip-Hop song is classified to other genres, which could mean that the genre is straying away from the sound captured by the training set.

The overall performance of the classifiers is shown in Figure 2. The "Average" column on the right of the graphic takes the average value of the corresponding metrics of the left three classifiers, showing an average trend of the metrics. The graphic demonstrates that the performance of most classifiers follows a fuzzy U-shape trend, with higher values at the start and the end than in the middle years of the dataset. The U-shape trend is statistically tested by a series of polynomial regressions of the metric on year and its quadratic term. In particular, the U-shape is demonstrable in the performance of the averaged classifier. All three recalls from the averaged classifier (bottom right of Figure 2) fit a polynomial regression on year and its quadratic term, where their coefficients are all statistically significant ($p < 0.05$). On the other hand, only the averaged classifier trained from the 2018 cohort songs yields significant results in the regression of accuracy on year and its quadratic term, while the other two averaged classifiers did not.

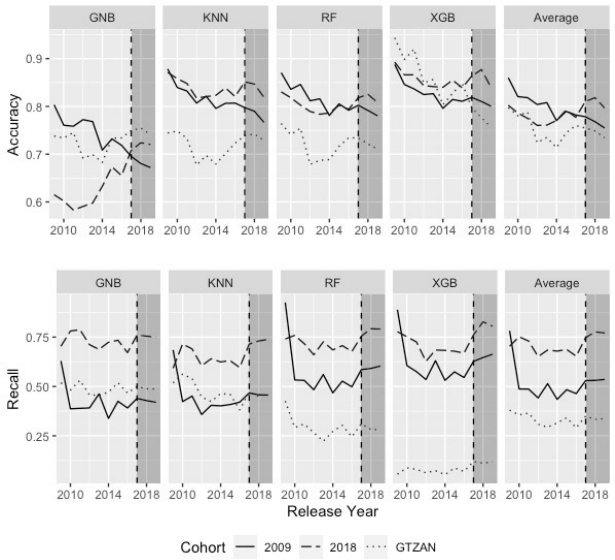


Figure 2. Hip-Hop accuracy and recall across cohorts and classifiers. The darker section on the right of each grid indicates the years since the first season of *The Rap of China* in 2017.

The results from the recalls, which imply how the true Hip-Hop songs of each year are similar to those of the benchmark cohort, indicate that Hip-Hop deviated from its late 2000s conventions in the middle of the 2010s but bounced back later, in particular after 2017, when the first season of *The Rap of China* aired. The downward trend of the prediction accuracy of the 2009 cohort classifiers may suggest more nuanced changes in the genre: given that the recall of the 2009 cohort is also U-shaped, it implicates that the 2009 cohort classifiers are able to detect the bouncing back of the genre conventions but confused other genres with Hip-Hop when the bounce took place in the late 2010s. This may be primarily because of the relatively small size of the 2009 cohort classifiers' training set, but it also suggests the possibility of other genres' cross-over to Hip-Hip, which I will discuss in 4.3.

4.2 Metrics and Subgenre Salience

Genres are comprised of "subgenres," which are subsets of their parent genre but can be distinctly different from each other in terms of their sound [1,26]. The idea of distinct subgenres under the same genre category thus suggests that genre evolution may be understood as changes in the most salient subgenres, i.e., subgenres that take up the highest proportion of the genre. It is intuitive, then, to speculate that the classifier's performance is better when it predicts a set of songs whose most salient subgenres are more similar to those of the training set.

Testing this speculation requires, above all, identifying the most salient subgenres of the training set, or the subgenre to which most songs of that genre are affiliated. I thereby designed an original approach to measuring subgenre salience, and I used the approach for measuring the

		GNB			KNN			RF			XGB			Average		
		Subg	Acc	Rec	Subg	Acc	Rec	Subg	Acc	Rec	Subg	Acc	Rec	Subg	Acc	Rec
2009	Top1	OS	0.236* (0.103)	0.625*** (0.095)	OS	0.188* (0.066)	0.636** (0.149)	OS	0.165* (0.067)	1.010*** (0.182)	OS	0.180** (0.050)	0.773** (0.165)	OS	0.154* (0.062)	0.761*** (0.139)
	Top5	OS; I; Con; A; CR	0.143* (0.045)	0.265** (0.075)	OS; I; Con; A; CR	0.123*** (0.021)	0.329** (0.072)	OS; A; U; Pop; I	0.090 (0.065)	0.001 (0.314)	OS; A; U; Pop; I	0.082 (0.058)	-0.049 (0.251)	OS; I; Con; A; CR	0.138*** (0.029)	0.301 (0.168)
2018	Top1	HH	0.211** (0.045)	0.065 (0.264)	HH	-0.001 (0.029)	0.208** (0.055)	HH	0.033 (0.023)	0.149* (0.048)	HH	0.003 (0.028)	0.206* (0.071)	HH	0.071* (0.022)	0.157** (0.047)
	Top5	HH; Pop; T; OS; CU	0.245** (0.058)	0.016 (0.070)	HH; Pop; T; OS; CU	-0.022 (0.034)	0.150 (0.093)	HH; Pop; T; OS; CU	0.245** (0.058)	0.106 (0.074)	HH; Pop; T; OS; CU	-0.013 (0.033)	0.136 (0.109)	HH; Pop; T; OS; CU	0.082** (0.028)	0.102 (0.077)

Note: Standard errors are in parentheses. Among the subtitles, Subg refers to subgenre, Acc refers to accuracy, and Rec refers to recall. Abbreviations in the Subg column: OS for Old-School Hip-Hop; I for Instrumental Hip-Hop; Con for Conscious Hip-Hop; A for Alternative Hip-Hop; CR for Cloud Rap; U for Underground Hip-Hop; Pop for Pop Rap; HH for Hip-Hop; T for Trap; CU for Chinese Underground Hip-Hop.
*p < .05; **p <.01; ***p<.001

Table 2. Performance of genre classifiers on songs released 2010-2019.

salience of Hip-Hop subgenres as follows. First, I extracted the subgenres from all the Hip-Hop songs in the dataset and counted their appearances each year. In many cases where a song has multiple subgenres, one more appearance was added for each of them. To account for the fact that prediction performance on each subgenre can significantly impact the prediction performance on the genre as a whole, I used the 2009 cohort classifiers and the 2018 cohort classifiers, respectively, to predict the genre of the Hip-Hop songs released in the year of the same cohort. After that, I calculated the prediction accuracy of each subgenre (which is also identical to recall, as only true Hip-Hop songs are examined here) by computing the proportion of correct predictions among all the songs with the subgenre in question. In this case, the prediction accuracy of each subgenre also indicates how the classifiers were trained to favor certain subgenres over others. I multiplied the prediction accuracy of each subgenre by their number of appearances. The product is the measurement of subgenre salience, which nicely captures the size of the impact that a subgenre can make on the pooled metrics of the genre. I identified the top 1 subgenre and top 5 subgenres with the highest salience score, which are presented in the note in Table 2. Then, I calculated their proportion in the total number of releases each year, respectively. Finally, I employed linear regression of Hip-Hop accuracy and recall on the proportion of the most salient subgenre(s) each year to understand the association between them. The objective of the linear regression is to see if the classifier performs better when the salient subgenres, which are favored by the classifiers by design, are more dominant among all subgenres of a year. The GTZAN cohort was not included in the analysis in this part, as it does not have a corresponding year in the dataset.

The coefficient estimates of the proportions of salient subgenres according to the regression models are presented in Table 2. The overall pattern generally substantiates a positive relationship between performance metrics and the size of the salient subgenres' proportions. In other words, the classifiers perform worse in years where there are fewer salient subgenres among the total releases of the year in question. Specifically, the average classifier trained on 2009 songs performs better when there are more songs in that year that are Old School Hip-Hop, Instrumental Hip-Hop, Conscious Hip-Hop, Alternative Hip-Hop, or Cloud Rap, whereas the average classifier trained on 2018 songs performs better when there are more Pop Rap, Trap, Old School Hip-Hop, or Chinese Underground Hip-Hop present in that year. The results imply that it is possible to detect genre evolution from the performance of the classifiers, which is demonstrable in the metrics when the genre deviates from the salient subgenres of the classifier.

4.3 Metrics and Genre-Crossing

As implied in Figure 2, the evolution of a genre may involve other genres. This could either mean "assimilation," where the genre in question is absorbing musical elements from other genres, or "dispersion," where the musical elements of the genre in question penetrate other genres. In either case, it is reasonable to speculate that the interaction between genres will likely do harm to the performance of the classifiers that are trained on songs where genre-crossing is not prominent. To investigate this issue, I first identified the number of releases in non-Hip-Hop genres (i.e., Folk, Rock, and Pop) with at least one subgenre that is explicitly associated with Hip-Hop. I did so by searching all the songs whose subgenres incorporate strings such as "Hip-Hop," "Hip Hop," "Rap," "Trap," and their Chinese equivalent. Table 3 presents the number of Hip-Hop-crossing non-Hip-Hop songs and their percentage among the total releases over the years studied. The table has some important implications for understanding classifiers' performance, as illustrated in Figure 2. First, there are no Hip-Hop-crossing non-Hip- Hop songs from 2009-2011. This

	Hip-Hop-crossing Non-Hip-Hop	Total Non-Hip-Hop	(%)
2009	0	1344	0
2010	0	1446	0
2011	0	1717	0
2012	22	1991	1.1
2013	59	2603	2.2
2014	98	4347	2.2
2015	184	5682	3.1
2016	163	6264	2.5
2017	94	10381	0.9
2018	135	13394	1.0
2019	118	7565	1.5

Table 3. The number of Hip-Hop-crossing non-Hip-Hop songs and their percentage in the total non-Hip-Hop releases over the years studied.

	GNB		KNN		RF		XGB		Average	
	Diff. Acc	Diff. FN	Diff. Acc	Diff. FN	Diff. Acc	Diff. FN	Diff. Acc	Diff. FN	Diff. Acc	Diff. FN
2009	-0.063*** (0.015)	0.087*** (0.016)	-0.020 (0.017)	0.163*** (0.015)	-0.089*** (0.015)	0.162*** (0.016)	-0.076*** (0.016)	0.181*** (0.016)	-0.062*** (0.008)	0.148*** (0.008)
2018	-0.090*** (0.012)	0.217*** (0.017)	-0.125*** (0.016)	0.270*** (0.017)	-0.131*** (0.016)	0.245*** (0.017)	-0.161*** (0.016)	0.261*** (0.017)	-0.127*** (0.008)	0.249*** (0.008)
GTZAN	-0.079*** (0.011)	0.182*** (0.017)	-0.120*** (0.016)	0.134*** (0.017)	-0.086*** (0.015)	0.263*** (0.015)	-0.081*** (0.014)	0.073*** (0.010)	-0.092*** (0.007)	0.120*** (0.008)

Note: Standard errors are in parentheses. *p < .05; **p <.01; ***p<.001

Table 4. Two-sample T-tests of key metrics between Hip-Hop-crossing non-Hip-Hop songs and those that do not cross the Hip-Hop boundary. Among the subtitles,

means that the 2009 cohort classifiers were not well-trained to correctly detect Hip-Hop-crossing non-Hip-Hop songs, which occur more frequently in the subsequent years. This possibly explains why 2009 cohort classifiers have a downward trend in their accuracy, as they are underperformed when identifying crossover songs. Second, the fact that there are more Hip-Hop-crossing non-Hip-Hop songs in terms of their percentage in the total non-Hip-Hop releases in the middle years than in the early and the late 2010s coincides, inversely, with the U-shaped trend of the metrics in Figure 2. This makes sense, as the prominence of genre-crossing is supposed to curb classification performance.

To further understand the relationship between genre-crossing and the performance of the classifiers, I also analyzed the difference in performance between genre-crossing non-Hip-Hop songs and those that did not cross the boundary of Hip-Hop. To do so, I conducted two-sample t-tests on two metrics between the two groups. The first metric was prediction accuracy, indicating the rate at which the non-Hip-Hop genre of the song was correctly identified. The second metric was the False Negative rate, indicating the rate at which the non-Hip-Hop song was incorrectly identified as Hip-Hop. Table 4 shows the results of the test, which unanimously demonstrate how genre-crossing discounts classification performance. Specifically, the consistent and statistically significant negative values under the columns of the difference in prediction accuracy suggest that the classifiers' ability to predict non-Hip-Hop songs that do not cross the boundary of Hip-Hop is always better than their ability to predict those that do so. Similarly, all the positive values under the False Negative columns provide strong evidence that classifiers are better at identifying non-Hip-Hop songs when they do not claim Hip-Hop as one of their subgenres. In short, the performance of genre classifiers may be an indicator of genre evolution, as bad performance points to genre-crossing.

5. DISCUSSION

Genres will evolve, and this will affect the performance of genre classifiers regardless of the data sources or algorithmic approaches they use. By analyzing the case of Chinese Hip-Hop, I demonstrate that the performance of genre classifiers is significantly impacted by genre evolution, particularly when the training set has less songs of the salient subgenres and when genres start to cross boundaries.

The goal of the present study is to highlight genre as a mutable cultural construct and to examine what this means for MIR research on genre classification. This study has at least three implications. First, similar to what has been argued in some of the previous MIR studies of genre, I contend that musical genre is a cultural construct and that the musical elements associated with a genre can change over time. My findings also suggest that the prevalence of a particular music style in a genre may be revived after a period of relative silence, which echoes what music industry scholars call a "revival" of music styles [13].

Second, because genres change and evolve, one possible application of genre classifiers is to understand the patterns of such change. I show it is possible to use genre classifiers, if trained on a particularly designed dataset, to detect the way genres evolve. It is difficult, though, to separate genre evolution from flaws in algorithmic design; therefore, it is important to supplement the analysis with a more detailed investigation of the soundscape of the songs across subgenres and songs that cross genres.

Third, the findings suggest that using accuracy or accuracy-related metrics to train genre classifiers might not be the only best application in practice. Regardless of their algorithmic framework, genre classifiers are almost always better at predicting the genre of songs released within a similar timeframe as those in the training set. As a result, if genre classifiers are meant to help classify newly released songs, it might be futile to aim at improving prediction accuracy, as the genre might evolve. This implies that we need to find an optimal balance between raising the prediction accuracy of the algorithms and understanding the developing trend of genres, which might require data exploration or even qualitative data beyond the design and optimization of machine learning algorithms.

There are obvious limitations in the data analysis; however, I argue that they will not severely affect the findings. Since a significant number of songs were inaccessible, especially in very distant years, it may twist the conclusion that Hip-Hop has been "revived" from its convention 10 years ago. However, it remains true that, if a genre classifier is trained on the songs of a particular composition of subgenres, the classifier might better predict the genre of a song when it shares commonalities with that subgenre's composition. In that case, classifiers can still be used to understand the evolution of genres, although they would be better "detectors" if more data were available.

6. ACKNOWLEDGEMENT

The author would like to thank Jin Ha Lee, Oriol Nieto, and the anonymous reviewers for their insightful comments and suggestions. The author would also like to acknowledge Leo Y. Yang for his assistance with data collection.

7. REFERENCES

[1] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on speech and audio processing*, vol. 10, no. 5, pp. 293–302, 2002.

[2] P. Knees, E. Pampalk, and G. Widmer, "Artist Classification with Web-Based Data.," 2004.

[3] A. J. Craft, G. A. Wiggins, and T. Crawford, "How Many Beans Make Five? The Consensus Problem in Music-Genre Classification and a New Evaluation Method for Single-Genre Categorisation Systems.," in *ISMIR*, 2007, pp. 73–76.

[4] J. Lee, J. Park, K. L. Kim, and J. Nam, "SampleCNN: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, no. 1, p. 150, 2018.

[5] J. Nam, K. Choi, J. Lee, S.-Y. Chou, and Y.-H. Yang, "Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach," *IEEE signal processing magazine*, vol. 36, no. 1, pp. 41–51, 2018.

[6] S. Oramas, F. Barbieri, O. Nieto Caballero, and X. Serra, "Multimodal deep learning for music genre classification," *Transactions of the International Society for Music Information Retrieval. 2018; 1 (1): 4-21.*, 2018.

[7] J. Pons and X. Serra, "Randomly weighted cnns for (music) audio classification," in *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, 2019, pp. 336–340.

[8] N. Scaringella, G. Zoia, and D. Mlynek, "Automatic genre classification of music content: a survey," *IEEE Signal Processing Magazine*, vol. 23, no. 2, pp. 133–141, 2006.

[9] M. Schedl, E. Gómez Gutiérrez, and J. Urbano, "Music information retrieval: Recent developments and applications," *Foundations and Trends in Information Retrieval. 2014 Sept 12; 8 (2-3): 127-261.*, 2014.

[10] M. Sordo, O. Celma, M. Blech, and E. Guaus, "The quest for musical genres: Do the experts and the wisdom of crowds agree?," in *ISMIR*, 2008, pp. 255–260.

[11] C. McKay and I. Fujinaga, "Musical genre classification: Is it worth pursuing and how can it be improved?," in *ISMIR*, 2006, pp. 101–106.

[12] J. C. Lena and R. A. Peterson, "Classification as culture: Types and trajectories of music genres," *American Sociological Review*, vol. 73, no. 5, pp. 697–718, 2008.

[13] J. C. Lena, *Banding together: how communities create genres in popular music*. Princeton, N.J: Princeton University Press, 2012.

[14] P. DiMaggio, "Classification in Art," *American Sociological Review*, vol. 52, no. 4, pp. 440–455, 1987.

[15] K. Nie, "Disperse and preserve the perverse: computing how hip-hop censorship changed popular music genres in China," *Poetics*, p. 101590, 2021.

[16] D. Diakopoulos, O. Vallis, J. Hochenbaum, J. W. Murphy, and A. Kapur, "21st Century Electronica: MIR Techniques for Classification and Performance.," in *ISMIR*, 2009, pp. 465–470.

[17] B. L. Sturm, "Two systems for automatic music genre recognition: What are they really recognizing?," in *Proceedings of the second international ACM workshop on Music information retrieval with user-centered and multimodal strategies*, 2012, pp. 69–74.

[18] B. L. Sturm, "Classification accuracy is not enough," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 371–406, 2013.

[19] S. Lippens, J.-P. Martens, and T. De Mulder, "A comparison of human and automatic musical genre classification," in *2004 IEEE international conference on acoustics, speech, and signal processing*, 2004, vol. 4, pp. iv–iv.

[20] W. G. Roy, "'Race records' and 'hillbilly music': institutional origins of racial categories in the American commercial recording industry," *Poetics*, vol. 32, no. 3–4, pp. 265–279, Jun. 2004.

[21] J. Sullivan and Y. Zhao, "Rappers as knights-errant: classic allusions in the mainstreaming of Chinese rap," *Popular Music and Society*, pp. 1–18, 2019.

[22] J. De Kloet, *China with a cut: globalisation, urban youth and popular music*, vol. 3. Amsterdam University Press, 2010

[23] B. McFee *et al.*, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015, vol. 8.

[24] C. Kereliuk, B. L. Sturm, and J. Larsen, "Deep learning and music adversaries," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 2059–2071, 2015

[25] B. L. Sturm, "The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use," *Journal of New Music Research*, vol. 43, no. 2, pp. 147–172, Apr. 2014

[26] A. Caparrini, J. Arroyo, L. Pérez-Molina, and J. Sánchez-Hernández, "Automatic subgenre classification in an electronic dance music taxonomy," *Journal of New Music Research*, vol. 49, no. 3, pp. 269–284, 2020.

IN SEARCH OF SAÑCĀRAS: TRADITION-INFORMED REPEATED MELODIC PATTERN RECOGNITION IN CARNATIC MUSIC

Thomas Nuttall[1] **Genís Plaja-Roglans**[1] **Lara Pearson**[2] **Xavier Serra**[1]
[1] Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
[2] Max Planck Institute for Empirical Aesthetics, Frankfurt am Main, Germany
[1]{thomas.nuttall, genis.plaja, xavier.serra}@upf.edu, [2]lara.pearson@ae.mpg.de

ABSTRACT

Carnatic Music is a South Indian art and devotional musical practice in which melodic patterns (motifs and phrases), known as *sañcāras*, play a crucial structural and expressive role. We demonstrate how the combination of transposition invariant features learnt by a Complex Autoencoder (CAE) and predominant pitch tracks extracted using a Frequency-Temporal Attention Network (FTA-Net) can be used to annotate and group regions of variable-length, repeated, melodic patterns in audio recordings of multiple Carnatic Music performances. These models are trained on novel, expert-curated datasets of hundreds of Carnatic audio recordings and the extraction process tailored to account for the unique characteristics of *sañcāras* in Carnatic Music. Experimental results show that the proposed method is able to identify 54% of all *sañcāras* annotated by a professional Carnatic vocalist. Code to reproduce and interact with these results is available online. [1]

1. INTRODUCTION

Musical styles around the world are often structured in relatively idiosyncratic ways, which can require bespoke tools for MIR tasks and computational music analysis [1]. This is the case in Carnatic Music, a South Indian art and devotional music tradition, in which several characteristic features of the style necessitate tailored MIR approaches.

Relative to other musical styles, Carnatic Music is heavily ornamented. This ornamentation is not superficial decoration but rather is integral to musical meaning [2]. The ornaments, known as *gamakas*, can greatly alter the sound of the notated *svaras* (notes); for example, some *gamakas* do not rest at all on the theoretical pitch of the notated *svara*, and instead involve oscillations between pitches either side of it [3, 4]. This oscillatory movement is particularly characteristic of the style, and can often subsume individual *svaras* [4]. The surface effect on the melodic line is that it typically has fewer stable pitch regions than many other styles. Such qualities makes it impossible for researchers who are not themselves Carnatic musicians to reliably identify *svaras* from audio recordings, although recent work has attempted to automate this task by creating descriptive transcriptions that assign *svara* names to key points in the melodic flow [5–7].

Another important feature of the style is the structural and expressive significance of motifs and phrases known as *sañcāras*, which can be defined as coherent segments of melodic movement that follow the grammar of the *rāga* (melodic framework) [2, 8]. Some musicians use the term in a narrower sense to refer to phrases that are particularly characteristic of the *rāga*, but here we use the term in its broader sense, referring to any melodic segment that is coherent from a Carnatic performer's perspective. These melodic patterns are the means through which the character of the *rāga* is expressed and form the basis of various improvisatory and compositional formats in the style [2,9]. There exists no definitive lists of all possible *sañcāras* in each *rāga*, rather the body of existing compositions and the living oral tradition of *rāga* performance act as repositories for that knowledge.

In this work we take advantage of two deep learning models, CAE [10] and FTA-Net [11], and the Dunya Carnatic corpus [12] for an improved, variable-length, and tradition-informed melodic pattern discovery in Carnatic Music. We first introduce the collections of data we build on top of, and we present our process relating each decision with the tradition-specific characteristics introduced in Section 1.1. We also include an empirical evaluation against expert annotations, brief musicological discussion of results and make available all code, features, models, results and an interactive application to explore them.

1.1 Characteristics of Carnatic Music

Many of the decisions taken in developing the methodology presented here are informed by certain characteristics of Carnatic Music and *sañcāras*, which to varying extents may be shared with other musical traditions around the world. We list some of those most relevant to this task, assigning each a unique code which will be referenced later.

CHAR1. *Sañcāras* conform in some ways to concepts of phrase more widely held; for example in Western Art Music, phrases are understood as influenced by gestalt

[1] https://github.com/MTG/searching_for_sancaras

© T. Nuttall, G. Plaja-Roglans, L. Pearson, X. Serra. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** T. Nuttall, G. Plaja-Roglans, L. Pearson, X. Serra, "In Search of Sañcāras: Tradition-informed Repeated Melodic Pattern Recognition in Carnatic Music", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

principles wherein segments are separated by features such as silence and long periods of stasis [13]. Also, as in other musical styles, longer phrases/*sañcāras* can be understood as comprising shorter but still meaningful segments [14].

CHAR2. The ideal boundary between these shorter segments within a longer phrase is not always clear-cut, as there might be no point of rest or silence between them. Based on discussions with the expert annotator during this project, it is clear that often a longer phrase may be plausibly segmented in two or three different ways. However, there are rules of *rāga* grammar, and understandings of typical *sañcāras* that restrict the number of plausible segmentation points. This is similar to the segmentation of Western Art Music, where in any given section, several plausible segmentations may be made, but where the options are not unlimited [15].

CHAR3. A given *sañcāra* when repeated in a performance may be immediately preceded or followed by different melodic material. This feature is relatively common in many musical styles.

CHAR4. When a *sañcāra* is repeated, it is often elaborated on, which involves the insertion of additional *svaras* and *gamakas*. So the same basic or underlying *sañcāra*/phrase may appear many times in a composition with different elaborations. The commonly occurs in the style because the main compositional format, the *kriti*, has a theme and variation structure.

CHAR5. There can be tempo variations between instances of the same *sañcāra*. This can range from minor fluctuations due to extemporisation, to playing a *sañcāra* at double or half the speed it was originally performed, which is a particular feature of some musical formats such as the *varṇam*.

CHAR6. Single performances within a concert can typically range from between approximately 6 to 60 minutes in length. In the longer examples, the presentation is made up of different musical formats, often involving a composition (e.g., a *kriti*) as well as a number of more extemporised formats (*ālāpana*, *niraval*, *kalpana svaram*), based on *sañcāras* and the *rāga* grammar. This freedom for performers to combine different musical formats to create a larger whole is a typical feature of the style.

CHAR7. A final characteristic feature of the style is its instrumentation. The most widely found contemporary ensemble consists of a vocalist accompanied by violin, *mridangam* (double ended drum) and *tambura* (plucked lute, which creates the drone). Other ensembles led by instrumentalists can also be found.

1.2 Related Work

The automatic retrieval of melodic patterns is a prominent problem in MIR given its applications for music analysis [16], segmentation [17] and music theory [18]. Despite efforts to settle consensus in melodic pattern discovery [19, 20], said task may be built on top of diverse representations of audio signals, such as mel-frequency cepstral coefficients (MFCC) [21], chroma [22], or predominant pitch time-series [23]. In a Carnatic Music context, most existing work relies on processes that consider pairwise distance metrics, such as dynamic time-warping (DTW), between sub-sequences of pitch transcriptions of the predominant sang melody [23–29]. However, this can often be computationally expensive and does not necessarily account for some of the idiosyncrasies of the tradition. To reduce the complexity, several works notate the patterns for an optimized similarity computation [30].

In a Carnatic context, gathering musically-relevant pattern annotations for entire recordings is expensive and time-consuming, and requires the involvement of experts. The most common approach is to ask experts to judge the relevance of the retrieved patterns and evaluate via standard classification metrics [9, 23].

2. DATA

For this research we use the Dunya Carnatic corpus [12], which includes $\approx$ 500 hours or music, divided into 2,380 audio recordings and organized in 235 concerts. Up to 259 artists and 227 unique *rāga*s are present in the corpus. The Saraga dataset is extracted from such corpus and is one of the largest and more complete, open-access datasets for research on the Carnatic and Hindustani music traditions [31]. For a total of 168 Carnatic recordings in Saraga, close-microphone tracks are available. Since the vocalist is the soloist and lead performer in these recordings, for this research we create a sub-dataset comprised of such tracks (168 in total) and denote it Saraga Carnatic Vocal (SCV). All tracks have a sampling rate of 44.1kHz.

We also rely on the Saraga-Carnatic-Melody-Synth (SCMS) dataset [29], a large dataset of vocal melody ground-truth for Carnatic Music compiled using a Carnatic-specific Analysis/Synthesis method, which is currently the largest open source collection of such data and has been shown to positively impact the melodic analysis for Carnatic Music [29].

3. METHOD

Our goal is to extract and group regions of repeated melodic patterns from audio recordings of Carnatic Music performances, Figure 1 provides an overview of this process. Many design decisions are informed by the characteristics introduced in Section 1.1, where relevant, we reference these characteristics using their corresponding **CHARx** code.

3.1 Features

Two feature sets are extracted from each recording in SCV: (1) An automated transcription of the predominant sung pitch in Hz (Section 3.1.1), from which we derive a mask corresponding to silent/stable regions (Section 3.1.1), and (2) Transformation-invariant melodic features extracted using a Complex Autoencoder (CAE) from audio in CQT representation (Section 3.1.2), which we use for self-similarity computation.

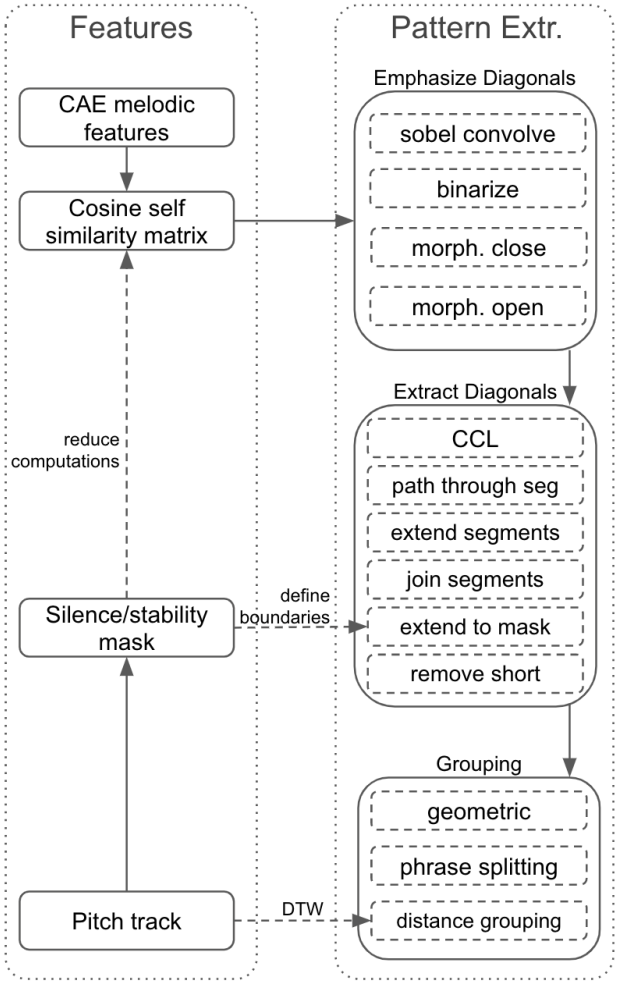


Figure 1. Pipeline overview - two feature sets are extracted from audio (pitch track and CAE features) and used to inform the extraction process.

3.1.1 Pitch Track and Mask

Recent work on vocal pitch extraction for Carnatic Music shows that state-of-the-art heuristic and data-driven methods for such tasks struggle to discriminate between singing voice and violin, and are not able to fully capture the vocal ornamentations in this tradition [29] **[CHAR7]**. To overcome this problem, a frequency-temporal attention network denoted FTA-Net [11] is re-trained using the SCMS dataset, further illustrating the positive impact this has for downstream tasks such as melodic pattern recognition. In this work we take advantage of this procedure to extract cleaner and more versatile pitch tracks from the audio in the Saraga dataset. We prioritize the close-microphone singing voice track for the extraction if available, however, there is no significant performance drop when running the Carnatic-tuned FTA-Net on top of mixed recordings. To account for incorrectly annotated silence that occur within ornaments due, for example, to glottal closure or other rapid vocal movements, we interpolate the pitch tracks to fill gaps shorter than or equal to 250ms [32].

As noted in [29] and [27], sañcāras are unlikely to contain long regions of sustained silence or pitch stability. We identify these regions in our pitch tracks and use this information to inform our search [**CHAR1**].

Silence is annotated as the regions of the pitch track that correspond to 0 Hz, in theory these are all regions where there is no sung voice (remembering that all silences of 250ms or less have been interpolated).

Stability is computed using the method outlined in [27, 29] - for a hop size of 0.2s, a window is passed across the entirety of the pitch track, those windows in which the maximum/minimum frequency is within $\pm$8Hz of it's mean are considered stable. Regions of the pitch track that contain consecutive stable windows whose total length sum to more than 1s are annotated as corresponding to a held note. The remaining windows, stable or not, are left unannotated. The final silence/stability mask, M_{ss}, annotates regions that correspond to silence or a held note with 1, and 0 otherwise.

3.1.2 Melodic Features and Self-similarity

We extract melodic features using a Complex Autoencoder (CAE). Mapping a signal, x, onto complex basis functions learnt by the CAE results in a transformation-invariant "magnitude space", $\mathbf{r}_x$, and a transformation-variant "phase space", ϕ_x. Exploiting the invariance-property of $\mathbf{r}_x$ has proven to achieve state-of-the-art results in repeated section discovery for audio [10].

We train a CAE on the vocal tracks in SCV, represented by their constant-Q transformed spectrogram with a hop size of 1984. The range comprises 120 frequency bins (24 per octave), starting from a minimal frequency of 80 Hz (selected as the minimum frequency expected in Carnatic vocal performance [33]). The spectrogram is split into n-grams of 32 frames. Before training, samples from the training data are randomly transposed by [-24 .. 24] frequency bins or time shifts of [-12 .. 12].

Each audio recording in SCV is mapped onto the complex basis functions learnt by the CAE and the transformation invariant "magnitude space", $\mathbf{r}_x$, used as melodic features relevant to the task of repeated pattern discovery.

As in [10], we compute a self-similarity matrix, $\mathbf{X}$, of $\mathbf{r}_x$, using the reciprocal of the cosine distance (plus some negligible epsilon to avoid division by zero). With some Carnatic performances lasting upwards of one hour, this computation rapidly becomes expensive. We use the masks learnt in Section 3.1.1, M_{ss}, to reduce the computation to include only those areas corresponding to regions we are interested in, reducing the number of pairwise comparisons by around 67%. Typically, $\mathbf{X}$ is between around 10,000 and 50,000 elements squared **[CHAR1, CHAR6]**.

Diagonals in $\mathbf{X}$ of high value correspond to two regions of continued similarity (repeated pattern). It is these diagonals we want identify and extract.

3.2 Repeated Pattern Extraction

Strong diagonal segments in $\mathbf{X}$ correspond to two occurrences of a repeated pattern. We define and extract them using a series of image processing techniques.

3.2.1 *Emphasizing diagonals*

To accentuate the diagonal segments, $\mathbf{X}$ is first convolved with a Sobel Filter to emphasize edges and then binarized about some experimentally set threshold, T_B. Since the emphasized edges correspond to the borders of a diagonal segment and not the segment itself, the binary matrix is morphologically closed to fill gaps and morphologically opened to smooth and remove noisy artifacts. Figure 2 illustrates this process.

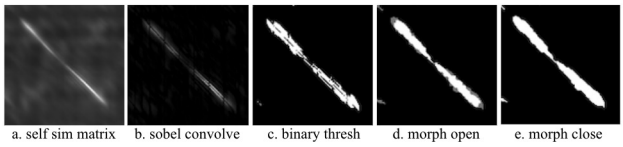


Figure 2. Emphasizing diagonals with convolution and morphological transformations: (a) Self-similarity matrix, (b) convolve with sobel filter, (c) binarize, (d) morphological opening, (e) morphological closing.

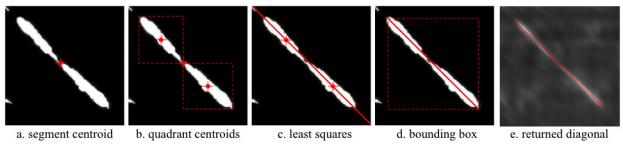


Figure 3. Defining diagonals from non-zero regions. Each segment is a thick region of non-zero values in $\mathbf{X}$, we extract a single line through this region corresponding to the underlying diagonal segment: (a) Identify segment centroid, (b) identify quadrant centroids, (c) least squares line, (d) identify bounding box, (e) return diagonal within bounding box.

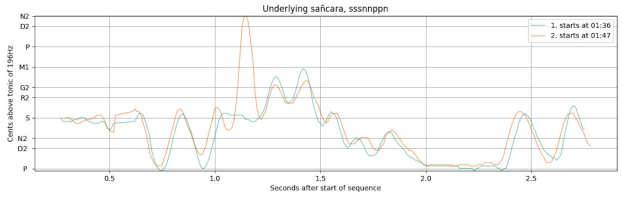


Figure 4. Two instances of the underlying sañcāra, sssnnppn, corresponding to the diagonal segment in Figures 2 and 3

3.2.2 *Extracting Diagonals*

Since instances of the same sañcāra can be performed at differing tempos, segments in $\mathbf{X}$ are not necessarily parallel to the $x = y$ diagonal, as such the method used to extract them in [10, 34] is not appropriate. We instead use a two-pass binary connected-components labelling algorithm (CCL) with a structuring element that considers elements that touch diagonally [35] to identify and group connecting non-zero elements, this can be conceptualized as a *flood fill* algorithm applied to all non-zero regions of $\mathbf{X}$. Each non-zero group corresponds to multiple x, y coordinates, we want to define a single path through this group to nominate as a candidate for the underlying true diagonal segment. First we compute the centroid of these coordinates and split the bounding box into an upper left quadrant and lower right quadrant centered on it. A line is learnt using least squares regression on the points connecting the two quadrant centroids and the underlying diagonal segment defined along that line between the bounding box of the group as a whole (Figure 3) **[CHAR5]**.

Segments are extended along their trajectory for a maximum of up to 50% of their length either side if the similarity in $\mathbf{X}$ for these regions is below some reduced threshold, T_{ext} where $T_{ext} < T_B$ (Figure 5) **[CHAR3]**.

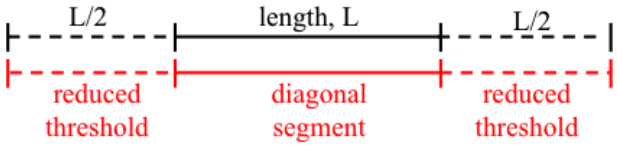


Figure 5. Reducing threshold for significant similarity in regions bordering returned segments. The two regions bordering the trajectory of a returned segment are subject to a lower threshold for similarity to account for variations in preceding and succeeding svaras in sañcāras.

Neighboring segments are joined if they are within 0.5s of each other and their gradients differ by $\leq$0.07rad **[CHAR4]**.

Segments that traverse regions identified in M_{ss} are broken along the center points of these silent/stable regions. Segments that are within 50% of their length to a silent or stable region (annotated in M_{ss}) are extended to the centroid of that silent/stable region **[CHAR1]**.

3.3 Grouping

Relationships between identified patterns are implicit in the geometry of $\mathbf{X}$ - each segment corresponds to two separate instances of the same pattern, those segments that intersect in x correspond to the same region in the recording (that represented on the x-axis) and those in y, vice versa. This information is used to iteratively group all patterns.

3.3.1 *Phrase Splitting*

We can further exploit segment positions in $\mathbf{X}$ to learn boundaries between shorter segments and longer phrases. If two segments intersect on one axis but are of different lengths, new segments are created corresponding to the intersecting portion of the segments **[CHAR2]**. Figure 6 illustrates an example of such. Segment A (corresponding to two patterns, $[x_1 : x_2]$ and $[y_1 : y_2]$) intersects with Segment B (corresponding to two patterns, $[x_1 : x_3]$ and $[y_3 : y_5]$). Consequently, two new segments are created from Segment B, $[y_3 : y_4]$ and $[y_4 : y_5]$ where $[y_3 : y_4]$ is another occurrence of $[x_1 : x_2]$. All patterns corresponding to Segment A, Segment B and the two new sub-segments are included in the final results.

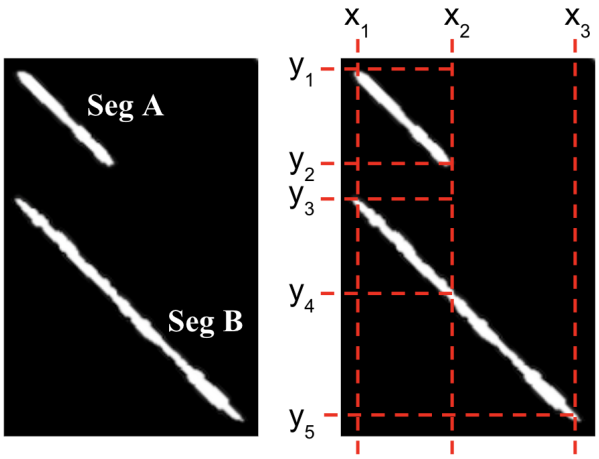


Figure 6. Learning sub-phrase relationships using geometry in $\mathbf{X}$ - Segment B is split into two since a subsequence of Segment B exists in isolation elsewhere (Segment A), both the entire Segment B and it's two subsequences are returned.

Performance	N_r	N_a	Recall	Precision	F1
KJ	174	168	0.73	0.76	0.74
SJ	214	106	0.53	0.54	0.52
VNK	90	144	0.33	0.42	0.37
Overall	478	418	0.54	0.60	0.57

Table 1. Melodic pattern discovery results for the three performances in Table 2. N_r and N_a are the number of returned and number of annotated patterns respectively

3.3.2 Distance Grouping

Further grouping is attempted by comparing each pairwise combination of groups. In each comparison, the dynamic time-warping distance is computed between 10 sets of two randomly selected patterns (one from each group) using the pitch tracks. If the average DTW distance between two groups is below some threshold, T_{dtw}, the groups are considered to represent the same pattern and merged to form one **[CHAR4]**.

4. EXPERIMENTAL SETUP

We apply our approach to three Carnatic performances (Table 2) and compare the results against expert annotations.

4.1 Annotations

Ground-truth annotations of all *sañcāras* in the audio recordings were created by a professional Carnatic vocalist who has 21 years of performance experience in South India. Annotations were created in Carnatic notation, known as *sargam* [36], using the software ELAN [37]. As there is no definitive list of *sañcāras* in a given *rāga*, the segmentations are based on the annotator's experience as a professional performer and student of a highly esteemed musical lineage. These annotations are therefore subjective to some degree, but have the benefit of being based on expert performer knowledge rather than on an externally imposed metric that may be irrelevant to musical concepts held by culture bearers.

In *kritis*, one of the most popular compositional formats, initial phrases are repeated with variations, known as *saṅgatis*. This means that there will be initial *sañcāras* that are subsequently varied. These musical connections are captured by grouping the related musical material together with an 'underlying *sañcāra*' annotation, which refers to the first occurrence and typically simplest version of the *sañcāra*. To reflect the hierarchical nature of plausible musical segmentations, longer full phrase-level annotations that can comprise several shorter *sañcāras* are also created **[CHAR1]**. The evaluation reported here is made on the 'underlying *sañcāra*' and 'underlying full phrase' levels.

4.2 Method and metrics

The various parameters at every step are optimized on one performance, Koti Janmani, representing 35% of the total number of minutes performed across the three performance. Those parameters related to the diagonal segment extraction are selected so as to maximize recall in identifying 'underlying *sañcāra*s' and 'underlying full phrases' that occur more than once in the ground-truth annotations. To consider whether a returned pattern, *R*, is a match with an annotation, *A*, we consider the overlap between them. Since the ideal boundary between shorter segments within a longer phrase is not always clear cut **[CHAR2]**, the expectation of 100% overlap is unrealistic **[CHAR1]**. As in [29], we consider *A* and *R* to be a match if the intersection between them is more than two-thirds the length of *A and* more than two-thirds the length of *R*, inspection of results find this limit to be sufficient in identifying and exploring regions of melodic interest.

5. RESULTS

Table 1 presents the results for the three performances. Our process is able to find 226 of the 418 annotations across the three recordings, corresponding to a total recall of 0.54, precision of 0.60 and F_1 of 0.57. The reader is encouraged to browse and listen to the results using the online visualisation tool provided in our Github repository.

6. DISCUSSION

Although far from 100 percent of annotated patterns were found, we significantly exceed what would have been achieved if we had used out-of-the-box methods without any consideration for the specifities of the Carnatic traditon. For example, when applying the pattern extraction process presented in [10] using our CAE model trained on the SCV, we are able to identify only 8% of annotated patterns at a precision of 7%. It is unsurprising that our performance exceeds this figure since our process is built, in part, on the same underlying model, replacing the pattern extraction steps with more tradition-informed ones.

The results achieved could contribute to a useful tool for musicologists needing to find many instances of the

Alias	Artist	Title	*Rāga*	Composer	Duration
KJ	Akkarai Sisters	Koti Janmani	*Ritigowla*	Oottukkadu Venkata Kavi	08:46
SJ	Salem Gayatri Venkatesan	Sharanu Janakana	*Bilahari*	Purandara Dasa	07:02
VNK	Sumitra Vasudev	Vanajaksha Ninne Kori	*Ritigowla*	Veenai Kuppaiyer	08:37
				Total:	24:25

Table 2. Three Carnatic performances used for evaluation.

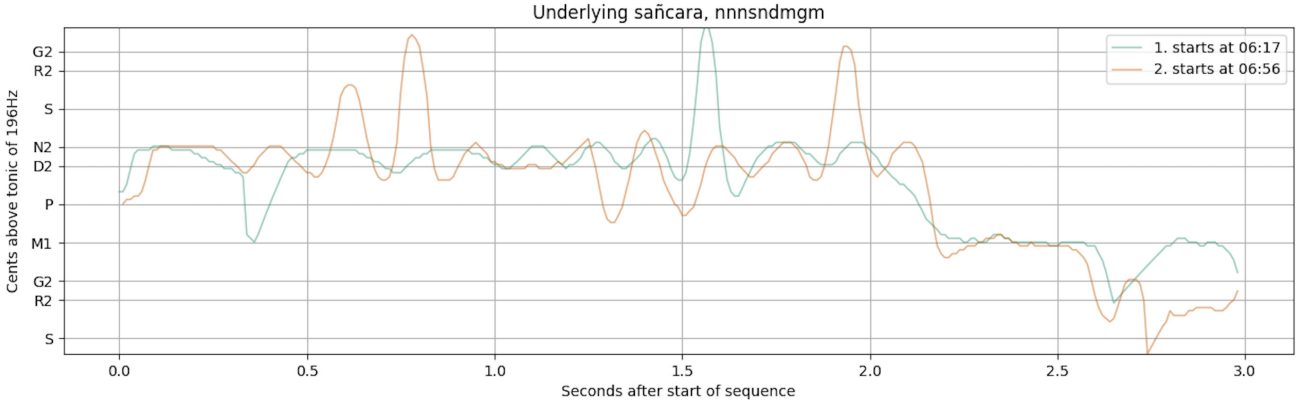


Figure 7. The pattern "nnnsndmgm" in Koti Janmani. The underlying *sañcāra* (blue line) is performed "nnnsndmgm", then later as a distinct variation "nnssndpdnndmmgrgm" (orange line).

same *sañcāra* across a long or many performances, a process that would otherwise involve extremely time consuming labour. While yet to be evaluated formally, the nature of the groupings returned appear promising from a musicological perspective. Figure 4 displays the pitch tracks for the two patterns corresponding to the diagonal segment in Figures 2 and 3. Each pattern would be considered instances of the same underlying sañcāra, "sssnnppn", but in fact are realized slightly differently ("snsgrsnnpn" at 01:36 and "snsmgrsnnpn" at 01:47). The steps outlined in Figures 2 and 3 illustrate how this subtle variation in performance is captured by considering the structural features that are particular to this musical style. However, there comes a point where the variation performed on the underlying or initial *sañcāra* is too great to be found by the process, and indeed it would also not be considered the same *sañcāra* by musicians, but rather a variation or new *saṅgati*. For example, in Koti Janmani, the initial *sañcāra* "nnnsndmgm" is later performed as a distinct variation "nnssndpdnndmmgrgm", and these two are successfully separated by the matching process into two motif groups (11 and 12) (Figure. 7).

While in the *kritis*, Koti Janmani and Sharanu Janakana, the majority of patterns annotated are found, fewer were found in the *varṇam*, Vanajaksha Ninne Kori. This might be explained by the differing structural features of the two compositional styles and of these particular compositions. This *varṇam* has a long 28 beat metrical cycle that lasts approximately 22 seconds. Furthermore, the melodic line has a more discursive and wandering character than the two *kritis* in this analysis. *Kritis* are based on a theme and variation (*saṅgati*) structure, while *varṇams* typically have fewer repetitions of any given phrase. As a result of these qualities, it might be harder for our process to find the borders of segments that match with those of the annotator. As discussed, there is often more than one plausible option for segmentation within longer units. It is possible that even expert annotators would find a larger number of plausible segmentation options in this *varṇam*, something that we could test for in future work by asking two or more annotators to annotate all plausible options, rather than only those that seem optimal **[CHAR2]**.

To demonstrate how our results could be useful to musicologists and fans of the style, we provide a high-level tool to explore and listen to them. We also include the results for 6 more performances across 6 ragas not mentioned here for a total of 9 performances across 8 distinct ragas. We do not have expert annotations for these extra performances and hence cannot empirically evaluate the results. One important factor in how valuable such a tool is, is the extent to which retrieved *sañcāras* are effectively sorted into groups of high melodic similarity that matches musicians' concepts of musical similarity in the style. It is obvious when exploring the results, that even for a listener with little musical background, there is a strong degree of similarity between patterns returned in the same group, both for those patterns that correspond to annotations and otherwise.

7. CONCLUSION

By considering and incorporating the characteristic features of a musical tradition into the development process, we obtain musicologically relevant results for the task of repeated melodic pattern recognition in Carnatic Music. In an effort to provide value for researchers, musicologists and fans of the Carnatic Music tradition, we provide all code, data and a high-level visualisation tool to explore the results.

8. ACKNOWLEDGEMENTS

Thanks to the Carnatic vocalist, Brindha Manickavasakan for her considered annotations of the audio recordings.

This research work was carried out under the project Musical AI - PID2019- 111403GB-I00/AEI/10.13039/501100011033, funded by the Spanish Ministerio de Ciencia e Innovación and the Agencia Estatal de Investigación.

9. REFERENCES

[1] X. Serra, "A multicultural approach in music information research," *In Proc. of the 14th Int. Society for Music Information Retrieval Conf., (ISMIR),* Miami, Florida, pp. 151–156, 2011.

[2] T. Viswanathan, "The analysis of rāga ālāpana in South Indian music," *Asian Music*, vol. 9, no. 1, pp. 13–71, 1977.

[3] T. M. Krishna and V. Ishwar, "*Carnatic music: Svara, gamaka, motif and raga identity*," *2nd CompMusic Workshop*, pp. 12–18, 2012.

[4] L. Pearson, "Coarticulation and gesture: an analysis of melodic movement in South Indian raga performance," *Music Analysis*, vol. 35, no. 3, pp. 280–313, 2016.

[5] V. S. Viraraghavan, A. Pal, H. Murthy, and R. Aravind, "State-based transcription of components of carnatic music," *In Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 811–815, 2020.

[6] H. G. Ranjani, D. Paramashivan, and T. V. Sreenivas, "Quantized melodic contours in indian art music perception: Application to transcription," *In Proc. of the 18th Int. Society for Music Information Retrieval (ISMIR),* Paris, France, 2017.

[7] H. G. Ranjani, D. Paramashivan, and T. Sreenivas, "Discovering structural similarities among rāgas in indian art music: a computational approach," *Sādhanā*, vol. 44, 05 2019.

[8] L. Pesch, *The Oxford illustrated companion to South Indian classical music*. Oxford University Press, USA, 2009.

[9] V. Ishwar, S. Dutta, A. Bellur, and H. A. Murthy, "*Motif Spotting in an Alapana in Carnatic Music.*" *In Proc. of the 14th Int. Society for Music Information Retrieval Conf. (ISMIR),* Curitiba, Brazil, pp. 499–504, 2013.

[10] S. Lattner, A. Arzt, and M. Dörfler, "Learning complex basis functions for invariant representations of audio," *In Proc. of the 20th Int. Society for Music Information Retrieval Conf. (ISMIR),* Delft, The Netherlands, 2019.

[11] S. Yu, X. Sun, Y. Yu, and W. Li, "Frequency-Temporal Attention Network for Singing Melody Extraction," *In Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Toronto, Canada, pp. 251–255, 2021.

[12] A. Srinivasamurthy, G. K. Koduri, S. Gulati, V. Ishwar, and X. Serra, "Corpora for Music Information Research in Indian Art Music," *Int. Computer Music Conf./Sound and Music Computing Conf.,* Athens, Greece, pp. 1029–1036, 2014.

[13] D. Deutsch, "Grouping mechanisms in music," in *The psychology of music*. Academic Press, 1982, pp. 99–134.

[14] J. Tenney and L. Polansky, "Temporal gestalt perception in music," *Journal of Music Theory*, vol. 24, no. 2, pp. 205–241, 1980.

[15] C. Hasty, "Segmentation and process in post-tonal music," *Music Theory Spectrum*, vol. 3, pp. 54–73, 1981.

[16] A. Volk and P. van Kranenburg, "Melodic similarity among folk songs: An annotation study on similarity based categorization in music," *Musicae Scientiae*, vol. 16, no. 3, pp. 317–339, 2012.

[17] P. Boot, A. Volk, and W. Bas de Haas, "Evaluating the Role of Repeated Patterns in Folk Song Classification and Compression," *Journal of New Music Research*, vol. 45, no. 3, pp. 223–238, 2016.

[18] P. Rao, J. C. Ross, and K. K. Ganguli, "Distinguishing raga-specific intonation of phrases with audio analysis," *Ninaad*, vol. 26, no. 1, pp. 59–68, 2013.

[19] I. Y. Ren, A. Volk, W. Swierstra, and R. C. Veltkamp, "In search of the consensus among musical pattern discovery algorithms," *In Proc. of the 18th Int. Society for Music Information Retrieval (ISMIR),* Paris, France, pp. 671–680, 2017.

[20] B. Janssen, W. Bas de Haas, A. Volk, and P. van Kranenburg, "Finding repeated patterns in music: State of knowledge, challenges, perspectives," *Lecture Notes in Computer Science*, no. 8905, pp. 277–297, 2014.

[21] M. Thomas, Y. Murthy, and S. G. Koolagudi, "Detection of largest possible repeated patterns in Indian audio songs using spectral features," *In Proc. of the IEEE Canadian Conf. on Electrical and Computer Engineering (CCECE)*, 2016.

[22] C. Wang, J. Hsu, and S. Dubnov, "Music Pattern Discovery with Variable Markov Oracle: A Unified Approach to Symbolic and Audio Representations," *In Proc. of the 16th Int. Society for Music Information Retrieval Conf. (ISMIR)*, pp. 176–182, 2015.

[23] G. Sankalp, J. Serrà, V. Ishwar, and X. Serra, "Mining melodic patterns in large audio collections of indian art music," *In Proc. of the Int. Conf. on Signal Image Technology and Internet Based Systems (SITIS),* Marrakech, Morocco, pp. 264–271, 2014.

[24] P. Rao, J. C. Ross, K. K. Ganguli, V. Pandit, V. Ishwar, A. Bellur, and H. A. Murthy, "Classification of Melodic Motifs in Raga Music with Time-series

Matching," *Journal of New Music Research*, vol. 43, no. 1, pp. 115–131, 2014.

[25] P. Rao, J. C. Ross, K. K. Ganguli, V. Pandit, V. Ishwar, A. Bellur, and H. Murthy, "Melodic motivic analysis of indian music," *Journal of New Music Research*, vol. 43, no. 1, pp. 115–131, 2014.

[26] S. Dutta and H. A. Murthy, "Discovering typical motifs of a raga from one-liners of songs in Carnatic Music," *In Proc. of the 15th Int. Society for Music Information Retrieval*, pp. 397–402, 2014.

[27] T. Nuttall, G. Plaja-Roglans, L. Pearson, and X. Serra, "The matrix profile for motif discovery in audio - an example application in carnatic music," *In Proc. of the 15th Int. Symposium on Computer Music Multidisciplinary Research (CMMR),* Tokyo, Japan, pp. 109–118, 2022.

[28] G. Padmasundari and H. A. Murthy, "Raga identification using locality sensitive hashing," *In Proc. of the 23th National Conf. on Communications (NCC)*, 2017.

[29] G. Plaja-Roglans, T. Nuttall, L. Pearson, and X. Serra, "Repertoire-specific vocal pitch data generation for improved melodic analysis of carnatic music," 2022. [Online]. Available: 10.5281/zenodo.7036117

[30] A. Lele, S. Pinjani, K. K. Ganguli, and P. Rao, "Improved melodic sequence matching for query based searching in Indian Classical Music," *In Proc. of the Frontiers in Signal and Music Processing (FRSM)*, 2016.

[31] A. Srinivasamurthy, S. Gulati, R. Repetto, and X. Serra, "Saraga: Open datasets for research on indian art music," *Empirical Musicology Review*, 2020.

[32] S. Gulati, J. Serrà, K. Ganguli, S. Şenturk, and X. Serra, "Time-delayed melody surfaces for Raga recognition," *In Proc. of the 17th Int. Society for Music Information Retrieval Conf. (ISMIR),* New York City, USA, pp. 751–757, 2016.

[33] M. Venkataraman, P. Boominathan, and A. Nallamuthu, "Frequency Range Measures in Carnatic Singers," *Journal of Voice*, 2020.

[34] S. Lattner, M. Grachten, and G. Widmer, "Learning transposition-invariant interval features from symbolic music and audio," *In Proc. of the 19th Int. Society for Music Information Retrieval Conf. (ISMIR),* Paris, France, 2018.

[35] J. R. Weaver, "Centrosymmetric (cross-symmetric) matrices, their basic properties, eigenvalues, and eigenvectors," *The American Mathematical Monthly*, vol. 92, no. 10, pp. 711–717, 1985.

[36] N. Ramanathan, "Sargam and musical conception in karnataka system'," *Sargam as a musical material*, 2004.

[37] H. Sloetjes and P. Wittenburg, "*Annotation by category-ELAN and ISO DCR*," Paper presented at the6th Int. Conf. on Language Resources and Evaluation (LREC), May, 2008.

AUTOMATIC CHINESE NATIONAL PENTATONIC MODES RECOGNITION USING CONVOLUTIONAL NEURAL NETWORK

Zhaowen Wang[1] **Mingjin Che**[2] **Yue Yang**[1] **Wenwu Meng**[2]
Qinyu Li[2] **Fan Xia**[2] **Wei Li**[3,4]

[1] Department of Music AI and Information Technology, Central Conservatory of Music, China
[2] College of Experimental Art, Sichuan Conservatory of Music, China
[3] School of Computer Science and Technology, Fudan University, China
[4] Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, China

{wzw, yewyang}@mail.ccom.edu.cn, weili-fudan@fudan.edu.cn,
{993393523, 1056844091, 935362804, 409769992}@qq.com

ABSTRACT

Chinese national pentatonic modes, with five tones of Gong, Shang, Jue, Zhi and Yu as the core, play an essential role in traditional Chinese music culture. After the early twentieth century, with the development of new Chinese music, the ancient Chinese theory of scales gradually developed into a new pentatonic modes theory under the influence of western music. In this paper, we briefly introduce our self-built CNPM (Chinese National Pentatonic Modes) Dataset, then design residual convolutional neural network models to identify which TongGong system the mode belongs, the pitch of tonic, the mode pattern and the mode type from audio signals, in combination with musical domain knowledge. We use both single-task and multi-task models with three strategies for identification, and compare them with a simple template-based baseline method. In experiments, we use seven accuracy metrics to evaluate the models. The results on identifying both the tonic pitch and the pattern of mode correctly achieve an average accuracy of 69.65%. As an initial research on automatic Chinese national pentatonic modes recognition, this work will contribute to the development of multicultural music information retrieval, computational ethnomusicology and five-tone music therapy.

1. INTRODUCTION

The modern Chinese national pentatonic modes theory based on the five tones of Gong, Shang, Jue, Zhi and Yu was created by Chinese musicians who combined music theories such as absolute pitch, twelve-tone equal temperament and major/minor modes used in western music with Chinese music theory after the early twentieth century.

© Z. Wang, M. Che, Y. Yang, W. Meng, Q. Li, F. Xia and W. Li. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Z. Wang, M. Che, Y. Yang, W. Meng, Q. Li, F. Xia and W. Li, "Automatic Chinese National Pentatonic Modes Recognition Using Convolutional Neural Network", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

Figure 1. Examples of mode scales. Chinese national pentatonic modes include pentatonic mode, hexatonic mode and heptatonic mode which are based on the five tones. Only pentatonic mode scales are shown here. The C Gong mode and the D Gong mode share the same mode pattern but with different pitch of tonic. The C Gong mode and the four modes in the second and third rows all belong to the C TongGong system.

From the 1920s to the 1960s, Chinese musicians such as Guangqi Wang [1], Zhengya Wang [2] and Yinghai Li [3] gradually proposed the theory of Chinese national pentatonic modes based on the ancient Chinese scales and related concepts, with reference to western scales and modes. The theory was later written by Chongguang Li into the basic music theory textbook [4], and became the version commonly used in Chinese music teaching. The musical data selected and the analyses conducted in this paper are based on the modern Chinese national pentatonic modes theory. It can be applied to most of the traditional music pieces, especially those works of Chinese national orchestral instruments adapted from traditional tunes. Further explanation and clarification of the theory and ancient Chinese music are listed here [1] .

Gong, Shang, Jue, Zhi and Yu can be called step names in the modern theory. They have a fixed interval relationship which can be described as Gong-Shang major second, Shang-Jue major second, Jue-Zhi minor third, Zhi-Yu major second and Yu-Gong minor third. In addition, Qingjue indicates the tone a minor second above Jue. Biangong and

[1] https://github.com/Hellwz/CNPM-ISMIR2022/blob/main/notes.md

Type	Step names
Pentatonic	Gong / C - Shang / D - Jue / E - Zhi / G - Yu / A
Hexatonic (Qingjue)	Gong / C - Shang / D - Jue / E - Qingjue / F - Zhi / G - Yu / A
Hexatonic (Biangong)	Gong / C - Shang / D - Jue / E - Zhi / G - Yu / A - Biangong / B
Heptatonic Yayue	Gong / C - Shang / D - Jue / E - Bianzhi / $^{\sharp}$F - Zhi / G - Yu / A - Biangong / B
Heptatonic Qingyue	Gong / C - Shang / D - Jue / E - Qingjue / F - Zhi / G - Yu / A - Biangong / B
Heptatonic Yanyue	Gong / C - Shang / D - Jue / E - Qingjue / F - Zhi / G - Yu / A - Run / ^{b}B

Table 1. Six mode types in Chinese national pentatonic modes. The pitch after each step name is just one of the possible matches, listed for better understanding.

Bianzhi indicate the tones a minor second below Gong and Zhi, respectively. And Run [2] is the tone a major second below Gong. The original Gong, Shang, Jue, Zhi and Yu tones are called Zheng-tones (i.e. main tones), and others are considered as Bian-tones (i.e. changed tones). Figure 1 shows several pentatonic scales for better understanding.

A complete mode name in Chinese national pentatonic modes can be divided into three parts: the pitch of tonic, the mode pattern, and the mode type. The pitch of tonic is indicated by pitch names such as C, D and E. The mode pattern has five categories according to the step name of tonic. Gong mode use Gong tone as tonic, Shang mode use Shang tone as tonic, and by analogy, there are five mode patterns of the Gong, Shang, Jue, Zhi and Yu. The mode type has six categories which are called Pentatonic, Hexatonic (Qingjue), Hexatonic (Biangong), Heptatonic Yayue, Heptatonic Qingyue and Heptatonic Yanyue mode. The step names these six mode types contain are shown in Table 1. Any Zheng-tone (i.e. Gong, Shang, Jue, Zhi and Yu) in it can be used as the tonic to form a mode. Therefore, combining the pitch of tonic, we get a total of 12x5x6=360 kinds of modes theoretically.

There is also a unique concept in Chinese national pentatonic modes theory called the TongGong system. If the Gong's pitch of two modes are the same, they are said to belong to the same TongGong system. For instance, the C Gong mode, D Shang mode, E Jue mode, G Zhi mode and A Yu mode all belong to the C TongGong system. The TongGong system to which the mode belongs, the tonic pitch and the pattern of the mode are related. Identifying any two of them can infer the third one. This is also a main idea of the approach in this paper.

Automatic key/mode recognition is an important research direction in MIR, but little research has been done on automatic Chinese national pentatonic modes recognition for audio. The main reasons include the lack of annotated data, the variety of national modes, and the difficulty of accurately identifying the pitch of Chinese instruments, etc. These reasons make recognition more difficult.

In this paper we develop deep learning models based on residual convolutional neural network for automatic recognition of Chinese national pentatonic modes. By entering the spectrogram of the audio, the models will output the full name of the national mode it belongs to. The correct recognition of the mode pattern and the pitch of tonic is the main focus of this paper. The benefits of using neural networks are that they can automatically extract features from spectrograms and have better generalizability to different musical genres and instruments. We use both single-task and multi-task models with three strategies, as described in section 3, for identification, and compared them with a simple template matching approach, as shown in section 4. Data augmentation is also adopted for better results.

The significance of Chinese national pentatonic modes automatic recognition includes: 1) Strengthening computer's understanding of music that can assist humans in musicological and ethnomusicological analysis; 2) Used to carry out the research of five-tone music therapy. Music in Chinese national pentatonic modes can regulate the body, assist in treatment and has a positive effect on some mental diseases [5, 6]; 3) Enhancing and further understanding the algorithmic mechanism of automatic mode recognition itself; 4) Assisting in the preservation of traditional music culture; 5) Auxiliary to other MIR studies.

2. RELATED WORK

The related work on automatic mode recognition and the application of convolutional neural networks in Chinese music information retrieval are introduced in this section.

2.1 Automatic Key/Mode Recognition

Automatic Key/Mode Recognition is one of the core tasks of MIR. The early studies on western major and minor key recognition were traditionally handled by template matching and Hidden Markov Models (HMMs). [7] proposed 24 major/minor key templates and algorithms for automatic key detection. [8] improved these templates and focused more on harmonic minor in minor key templates. [9] calculated Harmonic Pitch Class Profile (HPCP) for key matching of audio. [10] obtained templates from real instruments. [11] utilized probabilistic models (HMMs), combined with templates, to assist in key recognition. [12] extracted Pitch Class Profile (PCP) sequences and then used a single HMM directly.

[2] Run is also referred to as Qingyu (a minor second above Yu) by some scholars.

Classification basis	Names	Amount	Labels
TongGong System	C; ♯C (♭D); D; ♯D (♭E); E; F; ♯F (♭G); G; ♯G (♭A); A; ♯A (♭B); B	12	0-11
Pitch of Tonic	C; ♯C (♭D); D; ♯D (♭E); E; F; ♯F (♭G); G; ♯G (♭A); A; ♯A (♭B); B	12	0-11
Mode Pattern	Gong; Shang; Jue; Zhi; Yu	5	0-4
Mode Type	Pentatonic; Hexatonic (Qingjue); Hexatonic (Biangong); Heptatonic Yayue; Heptatonic Qingyue; Heptatonic Yanyue	6	0-5

Table 2. Definition of the mode category

With the development of deep neural networks, Convolutional Neural Network (CNN) has also been applied to the study of automatic key recognition. [13] proposed an end-to-end framework based on CNN for global key detection and can be adapted to different music styles. [14] improved the above model by removing the dense layer and keeping only the convolutional and pooling layers to obtain a genre-independent model and enhance generalization ability. [15] recognized local key for classical music, comparing the HMM and CNN methods and the effect of segmenting the dataset by song or version. [16] compared the influence of different directional convolutional architectures on the results along VGG-style networks. [17] designed a complex CNN architecture based on InceptionV3 to recognize 24 major/minor keys and evaluated a broad range of data augmentation methods.

In the literature, we found three works related to the recognition of Chinese national pentatonic modes, with two from symbolic format and only one from audio signals. None of them used neural networks. [18] used a manually designed decision tree for Chinese pentatonic mode recognition, and [19] used a template-matching based algorithm for automatic recognition of Chinese pentatonic mode and heptatonic mode. These two studies aim at MIDI files that are more conducive to recognizing musical meta-information such as pitch rather than audio signals. [20] designed uniform and ordinal templates for mode recognition of guqin music, using template matching to identify Gong, Shang and Zhi pentatonic modes from audio. We compare this method with our baseline in subsection 4.2.

2.2 Convolutional Neural Networks in CMIR

There have been several studies related to Chinese national music in the field of MIR in recent years, and some relevant datasets have emerged [21–24]. CNN models such as VGG [25], GoogLeNet [26] and ResNet [27] have made great progress on image recognition tasks. Inspired by this, CNN has also been applied in the CMIR (Chinese Music Information Retrieval) field. [28] utilized Convolutional Recurrent Neural Network (CRNN) for activity detection of Chinese national polyphony instruments and achieved a temporal resolution of seconds. [29] extracted Mel Frequency Cepstral Coefficients (MFCC) features of the audio and used VGGish network to classify 78 Chinese musical instrument timbres. [30] adopted Fully Convolutional Networks (FCN) to achieve the best results in the playing technique detection task of Chinese Erhu and Bamboo Flute instruments.

3. METHODS

The methods we propose and use are described below, including template matching and neural network models.

3.1 Definition

According to the definition in this paper, there are 360 kinds of Chinese national modes theoretically. We show their category definitions based on different classification basis and the numbering method of data annotation in Table 2. For simplicity, in the following we refer TongGong system to which the mode belongs as "system", pitch of the tonic as "tonic", mode pattern as "pattern", and mode type as "type". The primary task when classifying is to identify pattern and tonic, with system as auxiliary item, followed by type classification as a secondary task.

Based on the tonic t and the system s, we can infer the mode pattern. When t equals s, it is Gong mode. When t is 2 semitones higher than s, it is Shang mode. 4 semitones higher is Jue mode, 7 semitones higher is Zhi mode and 9 semitones higher is Yu mode.

3.2 Baseline

In order to compare the knowledge-based method with the data-driven method (i.e. neural network method), we use a template matching method designed for Chinese national pentatonic modes as the baseline.

The chroma features of the entire audio are obtained using the librosa package [31] and summed to get a twelve-dimensional chroma vector which reflects the energy of each pitch among the whole audio without octave information. Firstly, classify the TongGong system to which the audio belongs. The template of C TongGong system is [1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0] and others can be obtained by shifting the template in a loop. The Pearson correlation coefficient between chroma vector and each template is calculated respectively. The larger the correlation coefficient is, the higher the matching degree is. The template with the maximum value is the recognition result. Then for the tonic pitch, since most of the music we analyze returns to the tonic at the end, we use a simple way to identify the tonic: directly sum the chroma features of the last 500 frames according to the pitches, and regard the pitch name

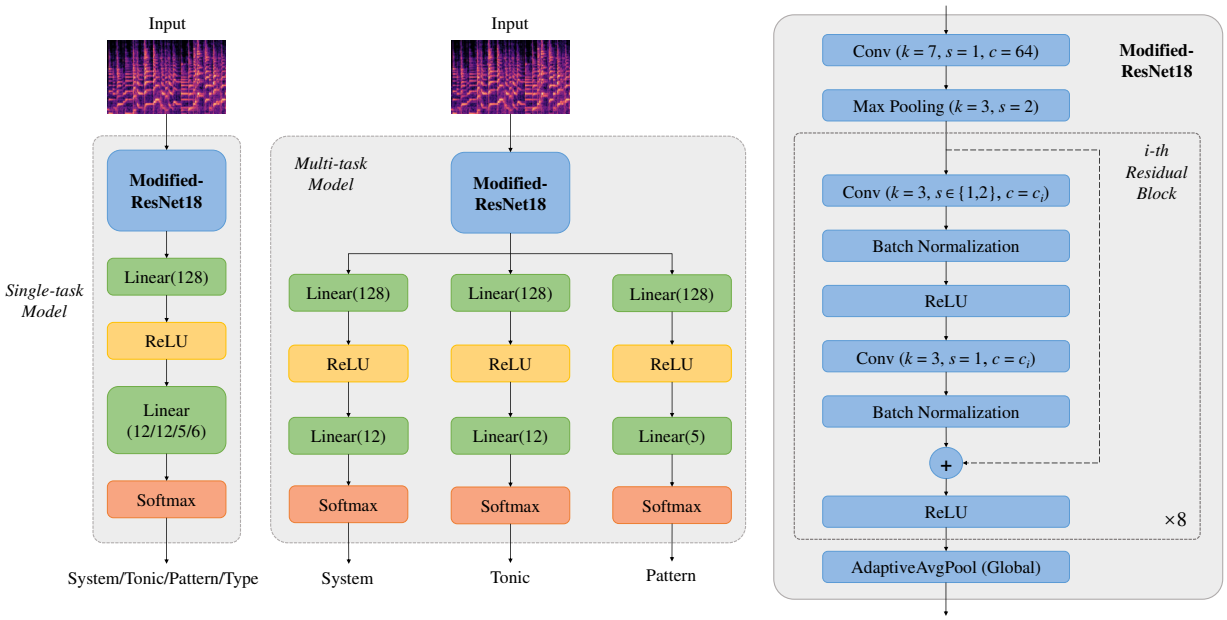


Figure 2. Model architectures. The shape of input is [Batch_size, Channel=1, F=168, T]. There are 4 single-task models for predicting TongGong system, tonic pitch, mode pattern and type respectively. Specifically, the single-task model for tonic prediction has no linear layer of 128 neurons and the following ReLU layer. In modified-ResNet18, k means the kernel size is $k \times k$, s indicates the stride and c stands for the number of channels. For 3rd, 5th and 7th residual block, the first s of conv layer equals 2, and others are 1. $c_i \in$ [64, 64, 128, 128, 256, 256, 512, 512]. A 1×1 convolutional layer is also used on the dashed line in the residual block if the number of channels does not match.

corresponding to the maximum value as the tonic pitch. As for the mode type recognition, the templates consisting of 0s and 1s are obtained according to the scale corresponding to each mode, and the results are acquired using a calculation method similar to that of the TongGong system identification. At last, the pattern of the mode can be obtained through the TongGong system and the tonic pitch.

3.3 CNN Models

Our CNN models use a modified ResNet18 [27] structure as the backbone. ResNet models have reached SOTA on multiple classification tasks in history and have simple structures which make them easy to tune and train. Their residual blocks are also widely used in various neural networks. For the input of our models, we use Constant-Q Transform (CQT) spectrogram, which is more appropriate for music audio than other spectrograms and contains more information than chroma features. Referring to the configuration of [14] and [17], we set the spectrogram frequency range from C1 to C8 with 168 frequency bins totally, and then downsample to 5 frames per second in the time axis. Because of the specific structure of our models, we can use any length of spectrogram as input.

The architectures of our models are shown in Figure 2. Compared to the original ResNet, our main changes include: 1) Modify the three channels of the input to a single channel for audio task; 2) Remove the downsampling from the first convolutional layer to accommodate a smaller size input; 3) Add the fully-connected layer in the output section to facilitate establishing different mapping relationships for different subtasks, and fit our tasks of less categories, compared to image classification task.

Due to the diverse types of modes but small amount of data, it is not reasonable to directly classify 60 or 360 categories. Therefore we use the following three strategies for recognition: 1) Directly use two single-task models to predict the tonic and pattern respectively; 2) Use two single-task models to predict the system and tonic respectively, and then indirectly calculate the pattern from these two results; 3) Use one multi-task model to recognize system, tonic and pattern, where pattern can be derived both directly and indirectly. When training the multi-task model, the losses of three outputs are summed for back propagation. As for type recognition, due to its unbalanced data distribution, difficulty of identification and little relationship with the other three, we directly use a single model to predict without adding it to multi-task model.

For the training input, we divide the spectrogram without overlap into 20s snippets, which is the same length as [17], to perform mini-batch training. Especially, the last snippet must be taken out regardless of whether it overlaps with the previous one, because it is critical for the tonic recognition. All snippets in the training set will be involved in the training, except that only the last snippet of each recording will be used when training the tonic model due to the characteristic of music. The tonic labels of non-final snippets will be masked when training the multi-task model for the same purpose, with other labels unchanged.

In the inference process, it is possible to either input the whole spectrogram directly or divide it into snippets,

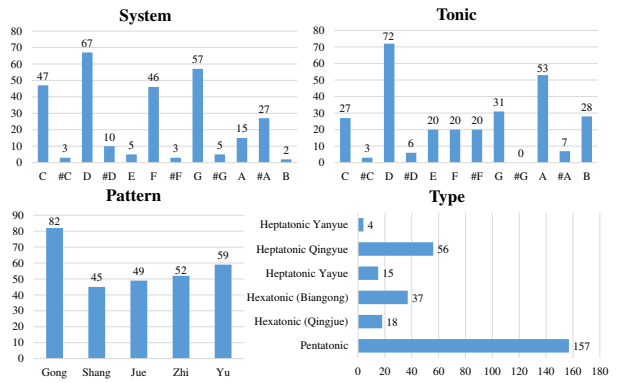


Figure 3. Data distribution of the CNPM Dataset

and then combine the results of each snippet to obtain the final result. In preliminary experiments we discover that for system, pattern and type tasks, using the whole spectrogram as input gives better results. In contrast, the tonic recognition only requires the last snippet of each song.

4. EXPERIMENTS

In the following we present the dataset we use and the experimental results we obtain using different methods.

4.1 Dataset

We use and expand our self-built CNPM (Chinese National Pentatonic Modes) Dataset. After expanding, the dataset contains 287 music recordings, including instrumental and vocal music, with instruments mainly being traditional Chinese instruments such as Guzheng, Guqin, Pipa, etc. The complete mode information is also included as labels, annotated by faculty and students in related disciplines. The average audio duration is 179.5s and the distribution of data is shown in Figure 3. Most of the audio comes from the Internet, with some Shang and Jue mode pieces being created and recorded by our musicians. In collecting and labeling the data, we select music pieces with clearer modes, more stable pitches, and mostly performed by contemporary musicians to avoid controversy in judging the modes. For music with modulation, we divide and label them accordingly. The dataset can be found here [3] .

The distribution of mode pattern in the database is relatively balanced, but system and tonic are not. To address the problem of fewer samples in some categories, we perform data augmentation by pitch shifting. To be specific, our processing of audio includes ascending a semitone, ascending a whole tone, descending a semitone and descending a whole tone with labels changing accordingly. Note that this augmentation only changes system and tonic, not pattern or type. And it is only applied to the training data.

4.2 Results

To ensure the effectiveness of the baseline approach, we first compare it with the 12-D ordinal template matching method proposed in [20] which also uses the chroma feature of audio. Different from our indirect prediction of five mode patterns from system and tonic, the ordinal templates directly predict three mode patterns of Gong, Shang and Zhi. The recognition accuracy is shown in Table 3, and our baseline method achieves better results.

Accuracy (%)	Ordinal	Baseline
Gong	62.20	**71.95**
Shang	55.56	**64.44**
Jue	-	**63.27**
Zhi	46.15	**67.31**
Yu	-	**62.71**

Table 3. Accuracy results of mode pattern recognition using template matching methods.

Metric	Object
ACC1	System
ACC2	Tonic
$ACC3_D$	Pattern (Directly)
$ACC3_I$	Pattern (Indirectly)
ACC4	Tonic and Pattern
ACC5	Type
ACC6	Tonic, Pattern and Type

Table 4. Accuracy metrics. Indirectly here means we get the mode pattern via system and tonic recognition results. Higher ACC4 is our primary task. And ACC6 means all correct.

For the neural network training, we use the Adam optimizer, cross-entropy loss function and a learning rate of 0.001. The batch size of each model is set to 32. For recognition results, we develop seven accuracy metrics to evaluate, as shown in Table 4.

In preliminary experiments, we adopt and modify the main structure of the InceptionKeyNet [17] network which is designed for major/minor keys recognition to perform our modes recognition. The usage method is similar to the modified-ResNet single-task models. We conduct experiments on the same random test set and find the two models have similar performance. But because the original InceptionKeyNet is implemented by TensorFlow and our models are implemented by PyTorch, it is hard to achieve a completely fair comparison and claim which structure is better. Since our models based on ResNet have a simpler structure and can converge in fewer epochs, making them easier to implement and train, the following experiments are performed using modified-ResNet18-based models.

The main experiments are conducted on the template-based baseline approach, the ResNet-based single-task models, as well as the ResNet-based multi-task model, and the results are shown in Table 5. To fully evaluate the models, we use a 5-fold cross-validation to obtain more con-

[3] https://ccmusic-database.github.io/en/database/ccm.html

ACC (%)	**Baseline**	**Res-ST**	**Res-MT**
ACC1	85.71	**88.50±1.80**	85.36±1.44
ACC2	71.78	74.85±9.28	**79.07±3.25**
$ACC3_D$	-	58.58±6.24	**63.79±7.74**
$ACC3_I$	66.55	71.01±10.30	**71.74±5.64**
ACC4	64.46	69.27±9.70	**69.65±5.21**
ACC5	48.08	**57.87±4.81**	-
ACC6	31.01	**42.15±3.65**	-

Table 5. Accuracy results under 5-fold cross-validation. Baseline is the template matching method. Res-ST is short for the modified-ResNet18 single-task model and Res-MT is the modified-ResNet18 multi-task model.

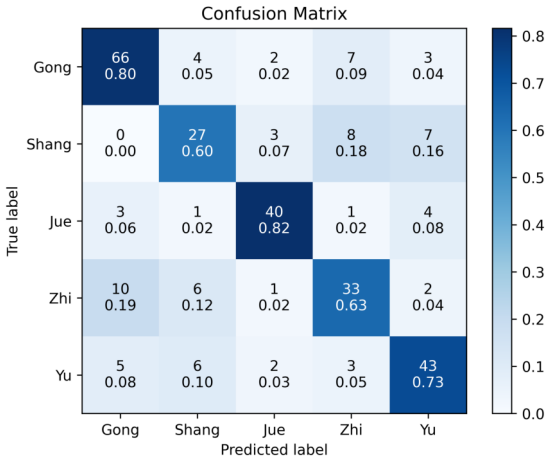


Figure 4. Normalized confusion matrix for mode pattern recognition results using the multi-task CNN model.

vincing accuracy results and compare them with the baseline method acting on the whole dataset. When dividing the data, we divide it by music tracks, keep the mode pattern ratio constant and ensure that only the corresponding training data are involved in back propagation in each fold of training.

As seen in the results, CNN-based methods outperform template matching method, and indirect mode pattern prediction is much better than direct prediction. The correct system and tonic predictions can introduce the correct pattern, but the correct pattern prediction can also be brought out by the wrong but relatively correct system and tonic. Therefore we need ACC4 to make sure all three (i.e. system, tonic and pattern) are correct. Since $ACC3_I$ is higher, we use the pattern results of indirect prediction to calculate ACC4 and ACC6. The accuracy of the system, tonic and pattern being predicted correctly at the same time achieved by our ResNet-based models is a satisfactory result. Due to the very uneven data distribution of mode type, the model responsible for type prediction is hard to train. So ACC5 and ACC6 are listed for reference only, and are not the primary tasks of this paper. In the comparison of single-task model and multi-task model, we find that the multi-task model has better ability to identify the tonic and directly recognize the mode pattern, which indicates the effectiveness of the multi-task learning strategy. Then for $ACC3_I$ and ACC4, the single-task and multi-task models perform similarly. The multi-task model performs more consistently over different folds with less parameters compared to using multiple single-task models.

Due to the relatively good performance of the multi-task model, we combined the indirect mode pattern recognition results of each fold to obtain a normalized confusion matrix and visualized it, as shown in Figure 4. Note that if the result of a certain indirect prediction is invalid, we use the direct prediction instead. From the confusion matrix, we can see that the CNN model has a high recognition ability for Gong, Jue and Yu modes, but the recognition of Shang and Zhi modes needs to be improved. This difference should be more related to the quality, distribution and representativeness of the data than the model itself. Besides, a larger value of k in k-fold cross-validation may lead to better results.

5. CONCLUSION & FUTURE WORK

In this paper, we explore the automatic mode recognition for Chinese national music composed after the early twentieth century with the characteristics of Chinese national pentatonic modes. For music from diverse ethnic and cultural backgrounds, the musical concepts and logic will be different, as well as the use of different tuning systems, rhythmic patterns, instruments and expressions. Traditional Chinese music played by traditional Chinese instruments is rich in the use of tuning systems, including compound tuning systems. This requires researchers to design more appropriate algorithms and models based on the characteristics of the music in order to get better results. Meanwhile, this can also facilitate the development of mainstream MIR algorithms.

We combine the musical domain knowledge when designing the methodology used in this paper. When identifying the mode pattern, we do so indirectly by considering its relationship to the TongGong system and the tonic pitch. As for the pitch of tonic, only the ending fragment of the music is considered according to the musical characteristic. We also use multi-task learning for recognition based on the correlation between system, tonic and pattern.

The accuracy of mode type is relatively low due to the difficulty of recognition and uneven data. In the future, more detailed annotation with segmentation can be carried out for audio to improve the accuracy. It is also possible to use other methods to identify each instrument and its pitches first before recognizing, but this remains a challenge for traditional Chinese instruments.

For the issue of small data size, more data augmentation methods can be used, such as time scaling, random masking, adding noise, etc. Transfer learning can also be utilized to train the backbone neural network on a large dataset first to have the ability of capturing tonality and pitch features, and then on a small dataset to accomplish a specific task.

6. ACKNOWLEDGEMENT

This work was supported by National Key R&D Program of China (2019YFC1711800), NSFC (62171138). Wei Li and Fan Xia are co-corresponding authors of this paper.

7. REFERENCES

[1] G. Wang, *Music of the oriental nation (in Chinese)*. Shanghai, China: Zhong Hua Book Company, 1929.

[2] Z. Wang, *The pentatonic scale and its harmony (in Chinese)*. Shanghai, China: Wenguang Bookstore, 1949.

[3] Y. Li, *The mode and harmony of Han nationality (in Chinese)*. Shanghai, China: Shanghai Literature and Art Publishing House, 1959.

[4] C. Li, *Fundamentals of music theory (in Chinese)*. Beijing, China: Music Publishing House (now People's Music Publishing House), 1962.

[5] J. Zhang, F. Xu, J. Du, X. Tan, C. Wu, and J. Kong, "Exploring and analyzing the five tone therapy in Chinese medicine (in Chinese)," *Journal of Changchun University of Traditional Chinese Medicine*, vol. 27, no. 5, pp. 702–704, 2011.

[6] S. Mi and M. Shi, "Discussion on the therapy of Wuyin (in Chinese)," *Clinical Journal of Chinese Medicine*, vol. 11, no. 29, pp. 12–14, 2019.

[7] C. L. Krumhansl, *Cognitive foundations of musical pitch*. Oxford, UK: Oxford University Press, 1990.

[8] D. Temperley, "What's key for key? the krumhansl-schmuckler key-finding algorithm reconsidered," *Music Perception*, vol. 17, no. 1, pp. 65–100, 1999.

[9] E. Gómez and P. Herrera, "Estimating the tonality of polyphonic audio files: Cognitive versus machine learning modelling strategies," in *Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, October 2004.

[10] Ö. Izmirli, "Template based key finding from audio," in *Proceedings of the 2005 International Computer Music Conference (ICMC)*, Barcelona, Spain, September 2005, pp. 211–214.

[11] G. Peeters, "Musical key estimation of audio signal based on hidden markov modeling of chroma vectors," in *Proceedings of the 9th International Conference on Digital Audio Effects (DAFx)*, Montreal, Canada, September 2006, pp. 127–131.

[12] W. Chai and B. Vercoe, "Detection of key change in classical piano music," in *Proceedings of the 6th International Conference on Music Information Retrieval (ISMIR)*, London, UK, September 2005, pp. 468–473.

[13] F. Korzeniowski and G. Widmer, "End-to-end musical key estimation using a convolutional neural network," in *25th European Signal Processing Conference (EUSIPCO)*, Kos, Greece, August 2017, pp. 966–970.

[14] F. Korzeniowski and G. Widmer, "Genre-agnostic key classification with convolutional neural networks," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, September 2018, pp. 264–270.

[15] C. Weiß, H. Schreiber, and M. Müller, "Local key estimation in music recordings: A case study across songs, versions, and annotators," *IEEE/ACM Transactions on Audio, Speech, and Language Processing (TASLP)*, vol. 28, pp. 2919–2932, 2020.

[16] H. Schreiber and M. Müller, "Musical tempo and key estimation using convolutional neural networks with directional filters," in *Proceedings of the 16th Sound & Music Computing Conference (SMC)*, Málaga, Spain, May 2019, pp. 47–54.

[17] S. A. Baumann, "Deeper convolutional neural networks and broad augmentation policies improve performance in musical key estimation," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, Online, November 2021, pp. 42–49.

[18] Y. Deng, L. Zhou, S. Ni, S. Zhang, and M. You, "Research on the recognition algorithm of Chinese traditional scales based on decision tree," in *37th Chinese Control Conference (CCC)*, Wuhan, China, July 2018, pp. 9527–9534.

[19] M. You, L. Chen, L. Zhou, and J. He, "Research on Chinese national music mode recognition based on template matching (in Chinese)," *Journal of Fudan University (Natural Science)*, vol. 59, no. 3, pp. 262–269, 2020.

[20] Y.-F. Huang, J.-I. Liang, I.-C. Wei, and L. Su, "Joint analysis of mode and playing technique in guqin performance with machine learning," in *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, October 2020, pp. 85–92.

[21] Z. Li, S. Yu, C. Xiao, Y. Geng, W. Qian, Y. Gao, and W. Li, "CCMusic Database: Construction of Chinese music database for MIR reaserch (in Chinese)," *Journal of Fudan University (Natural Science)*, vol. 58, no. 3, pp. 351–357, 2019.

[22] Z. Li and B. Han, "On database construction of acoustic system of Chinese traditional instrumental (in Chinese)," *Chinese Musicology*, no. 2, pp. 92–102, 2020.

[23] X. Liang, Z. Li, J. Liu, W. Li, J. Zhu, and B. Han, "Constructing a multimedia Chinese musical instrument database," in *Proceedings of the 6th Conference on Sound and Music Technology (CSMT)*, ser. Lecture Notes in Electrical Engineering (LNEE), vol. 568. Springer, 2019, pp. 53–60.

[24] X. Gong, Y. Zhu, H. Zhu, and H. Wei, "ChMusic: A traditional Chinese music dataset for evaluation of instrument recognition," in *4th International Conference on Big Data Technologies (ICBDT)*, Qingdao, China, September 2021, pp. 184–189.

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations (ICLR)*, San Diego, USA, May 2015.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the 28th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, USA, June 2015, pp. 1–9.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, USA, June 2016, pp. 770–778.

[28] Z. Li, C. Jiang, X. Chen, Y. Ma, and B. Han, "Instrument activity detection of China national polyphonic music based on convolutional recurrent neural network (in Chinese)," *Journal of Fudan University (Natural Science)*, vol. 59, no. 5, pp. 511–516, 2020.

[29] R. Li, Y. Xie, Z. Li, and X. Li, "Chinese musical instruments classification using convolutional neural network (in Chinese)," *Journal of Fudan University (Natural Science)*, vol. 59, no. 5, pp. 517–522, 2020.

[30] Z. Wang, J. Li, X. Chen, Z. Li, S. Zhang, B. Han, and D. Yang, "Deep learning vs. traditional MIR: a case study on musical instrument playing technique detection," in *Proceedings of 13th International Workshop on Machine Learning and Music (MML) at ECML/PKDD*, September 2020, pp. 5–9.

[31] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th Python in Science Conference (SciPy)*, Austin, USA, July 2015, pp. 18–25.

TEACH YOURSELF GEORGIAN FOLK SONGS DATASET: AN ANNOTATED CORPUS OF TRADITIONAL VOCAL POLYPHONY

David Gillman
New College of Florida
dgillman@ncf.edu

Uday Goyat
Georgia Institute of Technology
ugoyat3@gatech.edu

Atalay Kutlay
New College of Florida
atalay.kutlay18@ncf.edu

ABSTRACT

New datasets of non-Western traditional music contribute to the development of knowledge in MIR and allow computational techniques to inform ethnomusicology. We present an annotated dataset of traditional vocal polyphony from two regions of the Republic of Georgia with disparate musical characteristics. The audio for each song consists of four polyphonic recordings of one performance from different microphones. We present a process and workflow that we use to annotate the dataset, which takes advantage of the salience of individual voices in each recording. The process results in an f_0 estimate for each vocal part.

1. INTRODUCTION

To evaluate algorithms in Music Information Retrieval (MIR) it is essential to have a variety of extensive datasets of annotated music [1]. Annotated datasets of vocal *a cappella* music have been scarce until recently, particularly of non-Western music. The work of [2] provides a list of datasets of vocal polyphony that includes two datasets of songs from the Republic of Georgia: the Erkomaishvili dataset of [3] and the collection recorded in the work of [4].

Multi-f_0 estimation is a sub-problem of Automated Music Transcription (AMT) consisting of identifying the fundamental frequency f_0 of each part in a polyphonic recording. Datasets of polyphony that are annotated with the fundamental frequency of each part are useful as ground truth for multi-f_0 algorithms. Multi-f_0 estimation is a challenging problem for *a cappella* vocal music because of the variety of sounds produced by the human voice and the similarity in timbre of different voices [5].

In this work we present an annotated dataset of 38 three-part songs from the Republic of Georgia, including 29 from the region of Guria and nine from the region of Samegrelo. The total duration of the collection is 89 minutes. Gurian songs are a particular challenge for multi-f_0 estimation. The three parts are independent melodic lines that often cross and contain rapid movement. The top part often consists of *krimanchuli*, Georgian yodeling, with as many as six changes of octave per second. These features make our dataset a useful contribution as part of a training set for multi-f_0 algorithms. We have created a web-based visualization of the dataset that is potentially useful as an aid to singers learning their parts on Georgian songs, which was the purpose of the original recordings.

We also present a new process and workflow for multi-f_0 estimation where several recordings exist of a single performance. Our dataset is an unusual challenge for annotation in that it does not include isolated tracks for each vocal part. Instead there are four recordings of one performance made from different microphones. One recording presents a balanced mix of voices. In each of the other three recordings one of the voices is more salient than the other two, but all three voices are easily audible and create a polyphonic mixture. Our process and workflow consist of isolating the salient voice, applying several algorithms for monophonic f_0 estimation, and using a graphical interface to select the correct estimate. In addition to f_0 estimates we present the median absolute deviation of the estimates, a measure of confidence in the estimates.

Other interactive methods have appeared for extracting melody from audio. The work of [6] introduced the Tony software for monophonic audio. It presents to the user several pitch estimates generated by an early stage of the pYin algorithm, from which the user can select ranges of time and frequency. It is designed for ease of use and has many features such as octave correction [7]. The system of [8] designed for the Erkomaishvili dataset presents to the user a spectrogram with one melody highlighted as a result of dynamic programming performed on a set of salient frequencies at each time step, followed by automatic corrections that take into account musical knowledge such as voice ranges. The user is able to delete and replace lines in the spectrogram. There is a web interface for the Erkomaishvili dataset which plays the audio of each song accompanied by a scrolling score with lyrics. The work of [2] created the Dagstuhl dataset by recording each singer with a larynx microphone, a headset microphone, and a dynamic microphone. The researchers applied both the pYin and CREPE algorithms to each recording and derived confidences for each algorithm on each microphone using a subset of recordings manually annotated by a sound engineer who used Tony.

Our method differs from these methods in that it is designed for polyphonic recordings that contain one salient voice, it makes use of several f_0 estimates, and it presents

© D. Gillman, U. Goyat, and A. Kutlay. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** D. Gillman, U. Goyat, and A. Kutlay, "*Teach Yourself Georgian Folk Songs* Dataset: An Annotated Corpus Of Traditional Vocal Polyphony", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

a measure of confidence in the estimates. We have created a web interface for the dataset which plays the audio of each song accompanied by a scrolling visualization of the pitches. Unlike that of [2] our visualization shows the f_0 estimates of the three vocal parts, and the median absolute deviation of the each estimate is displayed in the manner of a confidence interval.

1.1 Georgian music datasets

The Republic of Georgia, bounded on the north by the Caucasus Mountains and on the west by the Black Sea, contains within its borders several starkly varying traditions of three-part *a cappella* vocal music [9]. Georgian singing is an aural tradition that uses a scale of its own [10]. The intervals that make up the Georgian scale are an area of research in ethnomusicology, as well as the extent to which singers in the tradition adjust to one another and deviate from a fixed scale to produce desired harmonies [11, 12]. Annotated datasets of Georgian music have been useful in advancing this research [9, 13].

The Erkomaishvili dataset contains Georgian sacred songs performed by a single performer on all three parts, with overdubbing. The most recent version of the dataset due to [3] includes f_0 annotations due to [8], onset annotations, and musical notation due to [14]. The dataset due to [4] includes Georgian folk songs from various regions recorded using individual larynx and headset microphones and a microphone for the ensemble. The authors observe that larynx microphones isolate the three parts well, and they illustrate this point by showing estimates of f_0 derived for one song in the collection.

The present dataset, like these other datasets, consists of recordings of performances by expert Georgian singers of interest to ethnomusicologists. However, unlike the dataset due to [3], our dataset consists of studio recordings of live ensembles, and unlike the dataset due to [4], our dataset was recorded without the benefit larynx microphones and therefore presents a greater challenge for pitch estimation.

Our dataset contains the recordings in the collections *Let Us Study Georgian Folk Songs (Gurian Songs)* and *Teach Yourself Georgian Folk Songs - Megrelian Songs*, both published in 2004 by the International Centre for Georgian Folk Song. The performers are highly trained musicians and experts in the music of their regions. [1] The performers were recorded together as an ensemble in close proximity in a single room in a studio [15]. There was one microphone for each part and one microphone for the room. This system of recordings was intended as an aid for singers learning their parts on Georgian songs, a setting in which it is beneficial to hear all three parts, but one above the other two. All songs are in three parts except during short intervals of overlap between antiphonal ensembles or solo and trio. The ensemble for each song is one high tenor (*pirveli* in Georgian), one middle tenor (*meore*) and one bass (*bani*), with a few exceptions. The song "Khasanbegura" has two antiphonal ensembles, one a trio and the other consisting of one high tenor and a choir of middle tenors and basses. The three songs "Maq'ruli", "Orira", and "Shvidk'atsa" also have two ensembles, one a trio and the other consisting of two tenors and a bass choir. Two songs, "Indi-Mindi" and "P'at'ara Saq'varelo", have a fourth part, a brief solo bass, that we have ignored for purposes of f_0 estimation. All but the two songs "Sabodisho" and "Mi Re Sotsodali" are *a cappella*. The accompaniment in these songs is a *chonguri*, a picked Georgian lute.

1.2 f_0 Estimation

Research on monophonic f_0 estimation has developed over several decades. The work of [16] introduced the use of the "cepstrum" in f_0 estimation, which exploits the fact that harmonics are regularly spaced in the frequency domain. Building on the work of [17], which exploited the same fact using the technique of spectral compression, the work of [18] introduced the technique of subharmonic summation to model the ability of the human ear to detect a weak fundamental pitch and used numerical methods to reduce the susceptibility of the estimate to noise [19]. The work of [20] introduced an algorithm for f_0 estimation and voiced-unvoiced classification which used the autocorrelation function of the signal to generate candidate pitches and dynamic programming to generate a sequence of pitches with few large jumps in frequency and few isolated voiced or unvoiced points. The well-known Praat software uses this algorithm [21]. The work of [22] introduced the well-known Yin algorithm, which used the sum of squared differences between the signal and lagged signals instead of the autocorrelation function. The work of [23] introduced the use of the fast-lifting wavelet transform for f_0 estimation. The work of [24] introduced the use of a neural network for f_0 estimation with the CREPE algorithm.

We use these six algorithms – Noll [16], Hermes [18], Boersma [20], Yin, Maddox [23], and CREPE – as a "panel of experts" to estimate f_0 in these recordings. [2] One may anticipate that algorithms based on autocorrelation or deep learning are strictly better than those based on Fourier or wavelet analysis. However, we find that for the high-quality non-monophonic recordings of this dataset, each algorithm produces correct estimates that tend to persist through the duration of a musical note or longer, and the algorithms err under different conditions. The errors we observe are mainly pitches in the wrong octave and pitches on the wrong part, including pitch estimates at times when the current part is silent or is a fricative consonant. Despite the presence of such errors, on harmonic content (judged subjectively as vowels sung by a healthy voice) at least one of the algorithms typically estimates f_0 correctly.

The paper is arranged as follows. In Section 2 we give details about the data and describe the process and workflow. [3] In Section 3 we present visualizations that com-

[1] The performers on the Gurian songs include Guri Sikharulidze, Tristan Sikharulidze, Otar Berdzenishvili, Anzor Erkomaishvili, Gedevan Mzhavanadze, Kote Papava, and Levan Goliadze. The performers on the Megrelian songs include members of the Odoia Choir directed by Polikarpe Khubulava.

[2] pYin did not improve over Yin in initial testing.

[3] The code resides in a public Github respository [25] The raw data and annotations are available to researchers upon request.

pare the performance of the six estimation algorithms on the dataset and that support the use of median absolute deviation as a measure of confidence in the estimates. We also introduce two types visualization that shed some light on musical content of the collection. Finally we present an illustration of the web interface of the dataset. [4]

2. PROCESS AND WORKFLOW

2.1 Estimates and alignment

The first step in our process is to apply six algorithms for monophonic f_0 estimation to each recording. The outputs of these algorithms are shifted in time, typically by a few hundredths of a second. We align the six estimates in time by fixing the estimates of one arbitrarily-chosen algorithm (Boersma) and shifting each of the others by multiples of $0.01s$. We choose the shift that minimizes $d(\delta t)$, the time-lagged ℓ_2-distance between the two time series at lag δt. We limit our search to the range $[-0.1, 0.1]$ because the vibrato of the human voice is $5-7Hz$, which is the Nyquist frequency of a sample rate of $10 - 14Hz$ [27].

Figure 1 shows a plot of $d(\delta t)$ for one song in the collection. In most cases δt_0 is very close to 0.

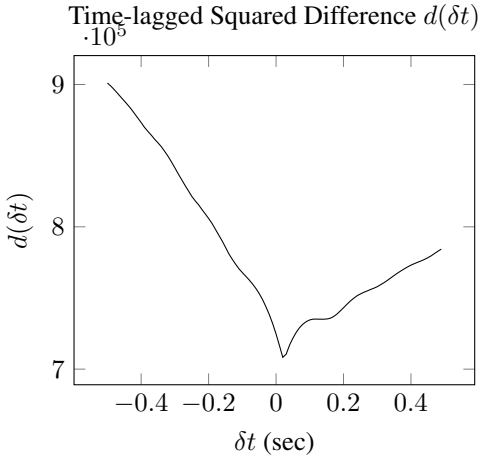


Figure 1. The squared difference between Boersma at time t and another f_0 estimator at time $t + \delta t$ for the middle part of *Ak'a Si Rekisho* from the Megrelian collection.

2.2 Note estimation

The next step in our process is a heuristic method for identifying notes in the target estimate. We chose a method that is simple to code and produces note values that serve well as a visual aid for the analyst, as opposed to a state-of-the-art method such as [28]. For purposes of post-processing non-note sections longer than $0.2s$ are considered unvoiced and the rest are considered voiced consonants.

The idea behind the heuristic is that a note transition is typically a monotone change in pitches between a local peak and a local trough in the singer's vibrato. The heuristic traverses the time series forward and identifies a note transitions as a change of 7% from the most recent local peak or trough. The pitch value of the note estimate is the average of the target estimate pitches over the duration of the note. The heuristic sets a minimum note length of $0.07s$. The heuristic also identifies as *transitions* monotone sequences of pitches between notes and increasing sequences of pitches before notes. Anything else is a non-note. We tuned the parameters during development. The 7% note transition threshold typically errs on the side of false negative note transitions.

[4] The web interface for the dataset makes available the f_0 estimates of the three vocal parts of each song [26].

2.3 Voiced-unvoiced detection

In parallel to the two steps just described, a heuristic "voiced/unvoiced" method removes non-salient voice parts that are present when the salient voice part is silent. This heuristic is a simple threshold based on the histogram of amplitude and the technique of weighted zero crossings. The recording is divided into overlapping frames of length $0.015s$ spaced $0.01s$ apart. For each frame the root mean squared of amplitude is divided by the zero crossing rate. A histogram of the ratio is created and smoothed using a Hanning window. The smoothed histogram typically has a local maximum near 0 that is due to sections of silence in the recording. There are typically one or two other local maxima near 0 that correspond to silence in the salient part. The "voiced" threshold is characterized by a high local maximum preceded by a low local minimum.

We tuned the parameters to result typically in some false positive sections of the recording that require manual correction but only isolated false negatives that can be corrected automatically. Figure 2 shows the smoothed histogram for one part of one song.

In Georgian music there is often a *decrescendo* in each voice at the end of a phrase. The threshold produced by the algorithm tends to correctly classify the ends of phrases but also tends to generate false *voiced* classifications during the subsequent silence. To combat this problem we added an automatic post-processing step which reclassifies short *voiced* sections within *unvoiced* sections.

We hand-tuned each of the heuristics mentioned above on a sample of varied musical sections, with the goal of minimizing the use of the interactive tool.

2.4 Interactive tool

The analyst uses a graphical interface to construct a target f_0 estimate for each part by stitching together the best estimates for different sections of the part. In sections of a part where no estimate is correct the analyst can specify a pitch range. The interface presents a window containing plots of the time series of f_0 estimates for one part of one song. The analyst can switch between parts and can turn the visibility of each time series on and off. The time series include the f_0 estimates of the six algorithms before and after application of the voiced-unvoiced algorithm. The window also includes the *target estimate*, which is initially equal to the CREPE time series, and note estimates for the target estimate. Selection tools and buttons enable the main functionalities of the tool, which are as follows:

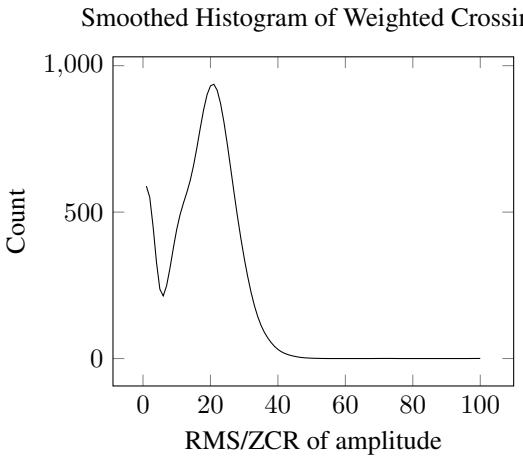


Figure 2. Smoothed histogram of weighted crossing for the top part of *Ak'a Si Rekisho* from the Megrelian collection

- to delete a section of the target estimate, i.e., set $f_0 = 0$ at selected times
- to set a section of the target estimate equal to the estimate of a selected algorithm
- to override a section of the target estimate with a pitch range

After using the tool on one part of a song, the analyst post-processes and saves the target estimate.

Figure 3 shows a screenshot of the tool in operation.

Figure 3. The interactive tool in operation at $84s$ in the middle part of *Adila-Alipasha* from the Gurian collection.

The last step of the process is to rerun the f_0 algorithms Noll, Hermes, Boersma, and Yin, constrained to be within a fixed percentage of the target estimate or within the pitch range. For CREPE and Maddox we use implementations that do not allow pitch constraints as input. If an algorithm is unable to find an estimate within the pitch range, its estimate is dropped.

2.5 Automatic Post-processing

The post-processing algorithm makes the following changes to the analyst's final choice of target estimate:

1. Revert isolated $(0.01 - 0.02s)$ unvoiced pitches to the estimates that were made before the application of the voiced-unvoiced algorithm.
2. Set unvoiced non-note sections (those shorter than $0.2s$) of the target estimate to 0.
3. Set voiced non-note sections of the target estimate to -10 to indicate *undecided*.

2.6 Manual Post-processing

The process described above generally results in a median absolute deviation of the estimates that is less than 1% of the median of estimates. (See Figure 6.) There are exceptions, sections of harmonic content where the estimates disagree. In some cases, typically sustained, well-tuned chords, it is clearly audible that two or more estimates are incorrect and skew the result. In these cases we have manually set a narrow pitch range in the interest of accuracy in the annotations.

3. RESULTS

In this section we discuss qualitatively the effectiveness of our method of combining multiple f_0 estimates. We provide descriptive statistics about the f_0 estimates that we include in the dataset of Gurian and Megrelian songs. First, we assess the estimates produced by each of the six algorithms on this dataset by comparing the initial estimate of each algorithm with the final estimate. Second, we assess the variability of the estimates as a measure of confidence in the final estimate. Third, we provide visualizations of the pitches and intervals of one song, *Gepshvat Ghvini*. Finally, we describe the web interface for the dataset and illustrate it with an example.

3.1 Discussion of the method

In most situations our process of combining multiple f_0 estimates "works" in the sense of producing near agreement among estimates where the fundamental frequency is unambiguous to the human ear. There is one noticeable failure mode. On non-monophonic input Hermes and Yin may fail to produce an estimate within given upper and lower frequency limits, and Boersma may label the sample *unvoiced*. In particular, Hermes, Yin, and Boersma tended to produce no estimate or incorrect estimates on prolonged, well-tuned chords. This is not a noticeable problem for Noll. We do not have a theoretical explanation, but we have found cases where the normalized difference function of Yin either does not have a minimum on the given interval or the location of minimum is unstable. On a small fraction of samples our process produced only two estimates or widely varying estimates.

3.2 Assessing each f_0 estimate

In this subsection we measure the accuracy of each monophonic f_0-estimation algorithm relative to the final f_0 estimate, separately on each collection. We consider the f_0 estimate of each algorithm, after applying alignment and the voiced-unvoiced algorithm, limited to samples that the note estimation algorithm labels as notes. Figures 4 and 5 show one smoothed histogram for each algorithm; it is a histogram of the ratio of the algorithm's f_0 estimate to the final f_0 estimate. By this measure Boersma is the most accurate when it is in the right octave (near 1.0), but Crepe and Noll are in the right octave more often.

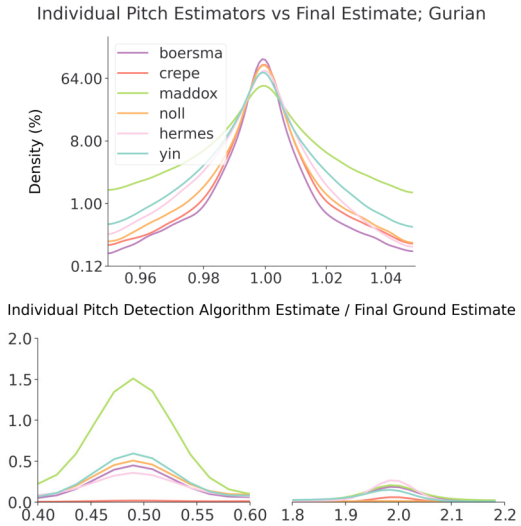


Figure 4. Smoothed histogram of ratio of f_0 estimates, algorithm/final, Gurian collection. Percentage of samples shown: boersma: 85.7%, crepe: 94.7%, maddox: 56.9%, noll: 85.9%, hermes: 86.0%, yin: 81.7%.

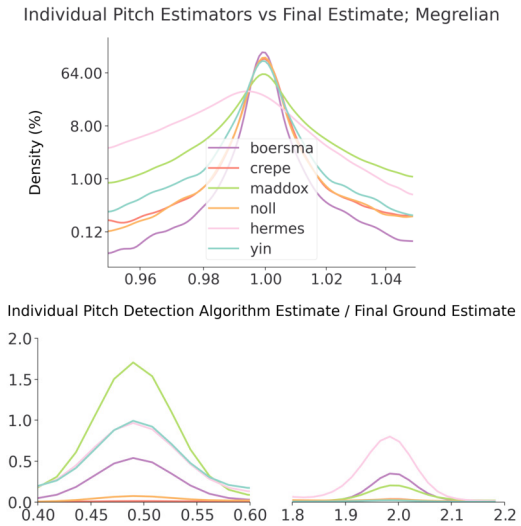


Figure 5. Smoothed histogram of ratio of f_0 estimates, algorithm/final, Megrelian collection. Percentage of samples shown: boersma: 84.3%, crepe: 96.3%, maddox: 63.1%, noll: 91.8%, hermes: 63.0%, yin: 79.4%.

3.3 Variability of the f_0 estimates

At each time step the final f_0 estimate is the median of the estimates of several algorithms. In this subsection we measure the variability of the estimates around the median. This measurement lends credibility to our method by showing that when there is a note (according to the note estimation algorithm) the variability is small (the estimates agree) and when there is not a note the variability is large (the estimates disagree). For our measure of variability we choose the median absolute deviation (MAD) around the median. This measure is robust to outliers, which means that when there is agreement among at least three estimates the MAD will be small. This choice corresponds to our intuition that during a vowel there will be general agreement among estimates and during a consonant there will be general disagreement. Figure 6 shows two graphs of smoothed histograms of MAD (in Hz), one for *note* samples and one for *non-note* samples. The graphs show 97.7% of Gurian note samples and 96.1% percent of Megrelian note samples. They show 70.9% percent of non-note Gurian samples and 74.7% of non-note Megrelian samples.

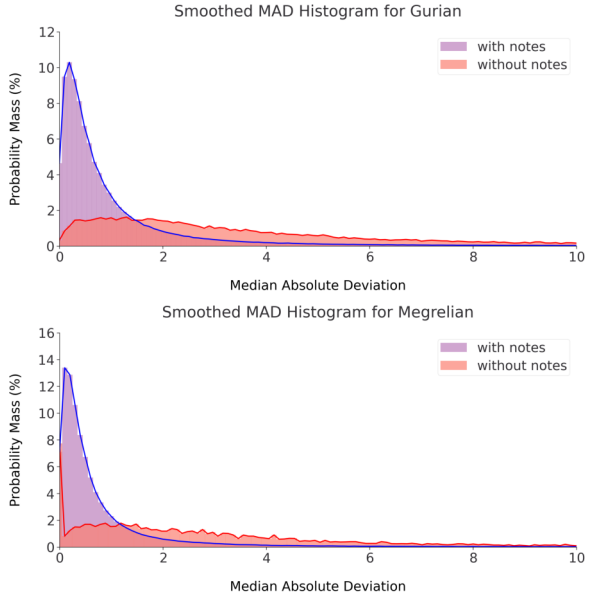


Figure 6. Smoothed histograms of MAD of each estimate, in Hz, *note* vs. *non-note* samples. Top: Gurian collection. Bottom: Megrelian collection.

3.4 Musical Observations

In this subsection we provide visualizations of the pitches and intervals in one Megrelian song, *Gepshvat Ghvini*. The purpose of these visualizations is to illustrate the potential for visualizations of digital data to enable and inform research in ethmomusicology.

We return to the question mentioned in the introduction of what intervals make up the Georgian scale, and to what extent singers deviate from the scale to produce desired harmonies. Figure 7 shows a histogram of the pitches of the three voices for *Gepshvat Ghvini*. The variation in

the peaks of the histograms for the three voices between 210Hz and 220Hz possibly indicates deviation from a fixed scale. Further investigation is needed.

Figure 8 shows a two-dimensional histogram of the chords of *Gepshvat Ghvini*. The x-axis is the ratio of pitches between the middle and bass parts. The y-axis is the ratio of pitches between the top and middle parts. The histogram is shown as a heatmap. The dark patch roughly at coordinates $(1.22, 1.22)$ shows that triads with a perfect fifth are common, and that these triads cluster around those with a "neutral" third between the minor and major third.

To exclude pitches in transition between notes we have limited the data in both figures to pitches at least three time steps away from the first and last pitch of each note, as designated by the note estimation algorithm. In the two-dimensional histogram we have limited the data to samples where we have estimates for all three parts.

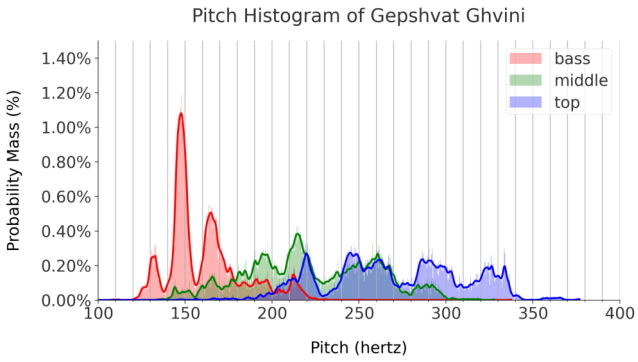


Figure 7. Histogram of pitches in *Gepshvat Ghvini*.

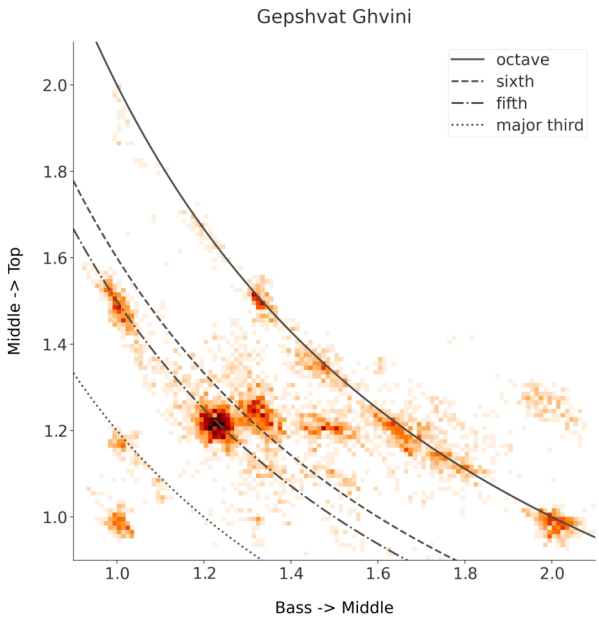


Figure 8. Histogram of chords in *Gepshvat Ghvini*. The x-axis is the ratio of middle to bass pitch. The y-axis is the ratio of top to middle pitch.

3.5 Web interface

In this subsection we describe the web interface for our dataset and illustrate the interface with an example. The web interface serves as a visual tool for the analyst to assess the accuracy of the results, as well as for ethnomusicologists to analyze of songs and for singers to learn the songs. The interface shows a scrolling graph of pitches that is synchronized to an audio player. Figure 9 shows a snapshot of the interface for the Megrelian song *Gepshvat Ghvini*. The graph optionally displays any of the median estimates of the three parts and the median absolute deviations (MAD) of the estimates.

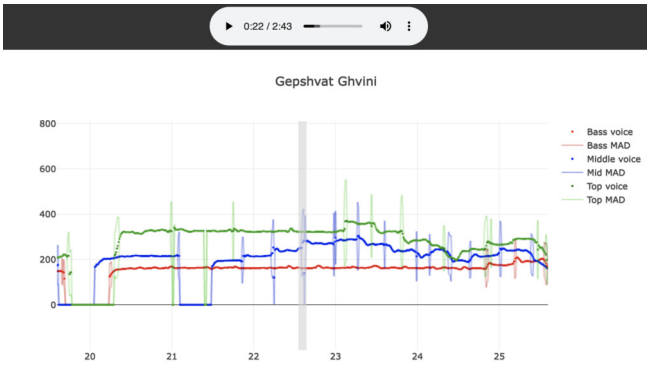


Figure 9. Web interface: *Gepshvat Ghvini*. The shaded line shows the current time. Each solid line shows the median frequency estimate for one part; the faint line of the same color shows the median absolute deviation.

4. CONCLUSION

We have presented a process and workflow for multi-f_0 annotation of a dataset of songs from the Republic of Georgia in which each song is represented by four recordings from different microphones. The annotations in our dataset represent the average of f_0 from different algorithms and are accompanied by the median absolute deviation of the estimates, which we have shown is a reasonable measure of confidence in the estimate. We hope our dataset and web interface are of interest to ethnomusicologists as audiovisual aids for analysis.

We plan to apply our process to collections of recordings from two other regions of Georgia which have been produced recently [29, 30]. We plan to use the resulting labeled datasets to develop algorithms for multi-f_0 estimation using the recordings of the mixed voices from the room microphones in our collections. We plan to develop the web interface further as a learning aid for singers by adding new features, including lyrics as subtitles.

5. ACKNOWLEDGEMENTS

The authors would like to thank the International Centre for Georgian Folk Song and the International Research Center for Traditional Polyphony, for permission to share the audio data with researchers.

6. REFERENCES

[1] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, "MedleyDb: A Multitrack Dataset for Annotation-Intensive MIR Research," in *Proceedings of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, Oct. 2014.

[2] S. Rosenzweig, H. Cuesta, C. Weiß, F. Scherbaum, E. Gómez, and M. Müller, "Dagstuhl ChoirSet: A Multitrack Dataset for MIR Research on Choral Singing," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 98–110, Jul. 2020, number: 1 Publisher: Ubiquity Press. [Online]. Available: http://transactions.ismir.net/articles/10.5334/tismir.48/

[3] S. Rosenzweig, F. Scherbaum, D. Shugliashvili, V. Arifi-Müller, and M. Müller, "Erkomaishvili Dataset: A Curated Corpus of Traditional Georgian Vocal Music for Computational Musicology," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 31–41, Apr. 2020, number: 1 Publisher: Ubiquity Press. [Online]. Available: http://transactions.ismir.net/articles/10.5334/tismir.44/

[4] F. Scherbaum, N. Mzhavanadze, S. Rosenzweig, and M. Müller, "Multi-Media Recordings Of Traditional Georgian Vocal Music For Computational Analysis," in *Proceedings of the 9th International Workshop on Folk Music Analysis*, Birmingham, Jul. 2019.

[5] H. Cuesta, B. McFee, and E. Gómez, "Multiple F0 Estimation in Vocal Ensembles using Convolutional Neural Networks," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, Montréal, Canada, 2020, arXiv: 2009.04172. [Online]. Available: http://arxiv.org/abs/2009.04172

[6] M. Mauch, C. Cannam, R. M. Bittner, G. Fazekas, J. Salamon, J. Dai, J. Bello, and S. Dixon, "Computer-aided Melody Note Transcription Using the Tony Software: Accuracy and Efficiency," in *Conference Proceedings of the First International Conference on Technologies for Music Notation and Representation*, 2015.

[7] ——, "Tony: a tool for melody transcription, https://code.soundsoftware.ac.uk/projects/tony," 2015. [Online]. Available: https://code.soundsoftware.ac.uk/projects/tony

[8] M. Müller, S. Rosenzweig, J. Driedger, and F. Scherbaum, "Interactive Fundamental Frequency Estimation with Applications to Ethnomusicological Research," in *Proceedings of the Audio Engineering Society Conference: 2017 AES International Conference on Semantic Audio*. Audio Engineering Society, Jun. 2017. [Online]. Available: https://www.aes.org/e-lib/browse.cfm?elib=18777

[9] F. Scherbaum, N. Mzhavanadze, S. Arom, S. Rosenzweig, and M. Müller, "Tonal Organization of the Erkomaishvili Dataset: Pitches, Scales, Melodies and Harmonies," *Computational Analysis Of Traditional Georgian Vocal Music*, 2020, publisher: Universitätsverlag Potsdam. [Online]. Available: https://publishup.uni-potsdam.de/frontdoor/index/index/docId/47614

[10] A. Erkomaishvili, *Georgian Folk Music. Guria. From Artem Erkomaishvili's Collection.* Tbilisi, Georgia: International Centre for Georgian Folk Song, 2005.

[11] S. Gelzer, "Testing a Scale Theory for Georgian Folk Music," in *The First International Symposium on Traditional Polyphony, Proceedings*. International Research Center for Traditional Polyphony, Sep. 2002, pp. 194–200. [Online]. Available: https://drive.google.com/file/d/1t1wNS_d2mBnPzeZnqSOe8QUymowiriwU/view?usp=drive_open&usp=embed_facebook

[12] M. Erkvanidze, "On Georgian Scale System," in *The First International Symposium on Traditional Polyphony, Proceedings*. International Research Center for Traditional Polyphony, Sep. 2002, pp. 178–185. [Online]. Available: https://drive.google.com/file/d/1MT2fdZwhrkMitpDjDYZBwTtxS_jl1DT0/view?usp=drive_open&usp=embed_facebook

[13] Z. Tsereteli and L. Veshapidze, "On the Georgian Traditional Scale," in *The Seventh International Symposium on Traditional Polyphony, Proceedings*. Tbilisi, Georgia: International Research Center for Traditional Polyphony, Sep. 2014, pp. 288–295. [Online]. Available: https://drive.google.com/file/d/1iVzCX1jAXnMnv_rJWNYEDdtOPTARqUzj/view?usp=drive_open&usp=embed_facebook

[14] D. Shugliashvili, *Georgian Church Hymns, Shemokmedi School.* Georgian Chanting Foundation, 2014.

[15] N. Razmadze, "Private communication," Mar. 2022.

[16] A. M. Noll, "Short-Time Spectrum and "Cepstrum" Techniques for Vocal-Pitch Detection," *The Journal of the Acoustical Society of America*, vol. 36, no. 296, 1964.

[17] ——, "Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum, and a maximum likelihood estimate," in *Proceedings of the Symposium on Computer Processing in Communication*, vol. 19. New York, NY, USA: Microwave Institute, University of Brooklyn, 1970, pp. 779–797.

[18] D. Hermes, "Measurement of pitch by subharmonic summation," *The Journal of the Acoustical Society of America*, vol. 83, no. 257, 1988.

[19] E. Terhardt, G. Stoll, and M. Seewann, "Algorithm for extraction of pitch and pitch salience from complex tonal signals," *The Journal of the Acoustical Society of America*, vol. 71, pp. 679–688, 1982.

[20] P. Boersma, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," in *IFA Proceedings 17*, 1993, pp. 97–110.

[21] P. Boersma and D. Weenink, "Praat, a system for doing phonetics by computer," *Glot International*, vol. 5, no. 9/10, pp. 341–345, 2001.

[22] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, Apr. 2002, publisher: Acoustical Society of America. [Online]. Available: https://asa.scitation.org/doi/10.1121/1.1458024

[23] E. Larson and R. K. Maddox, "Real-Time Time-Domain Pitch Tracking Using Wavelets," in *Proceedings of the University of Illinois at Urbana Champaign Research Experience for Undergraduates Program*, 2005.

[24] J. W. Kim, J. Salamon, P. Li, and J. Bello, "Crepe: A Convolutional Representation for Pitch Estimation," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[25] D. Gillman, U. Goyat, and A. Kutlay, "Teach Yourself Georgian Folk Songs Dataset: Code," 2021. [Online]. Available: https://github.com/New-College-of-Florida/Voice

[26] ——, "Teach Yourself Georgian Folk Songs Dataset: Website," 2021. [Online]. Available: http://131.247.152.67/georgian

[27] M. Vetterli, J. Kovačević, and V. K. Goyal, *Foundations of Signal Processing*. Cambridge University Press, Sep. 2014. [Online]. Available: https://www.cambridge.org/highereducation/books/foundations-of-signal-processing/DCC08E20D354F34E084FC11862E18F18

[28] R. Nishikimi, E. Nakamura, K. Itoyama, and K. Yoshii, "MUSICAL NOTE ESTIMATION FOR F0 TRAJECTORIES OF SINGING VOICES BASED ON A BAYESIAN SEMI-BEAT-SYNCHRONOUS HMM," *New York City*, p. 7, 2016.

[29] M. Khardziani and N. Razmadze, "Acharan Folk Songs – Collection Of Sheet Music With CD For Self-Study – International Research Center for Traditional Polyphony," 2020. [Online]. Available: http://polyphony.ge/en/acharan-folk-songs-collection-of-sheet-music-with-cd-for-self-study/

[30] R. Tsurtsumia and N. Razmadze, "Svan Folk Songs," May 2020, section: Books. [Online]. Available: https://chanting.ge/en/svan-folk-songs/

ADAPTING METER TRACKING MODELS TO LATIN AMERICAN MUSIC

Lucas S. Maia[1] **Martín Rocamora**[2] **Luiz W. P. Biscainho**[1] **Magdalena Fuentes**[3]
[1] PEE/COPPE, Federal University of Rio de Janeiro, Brazil
[2] FING, Universidad de la República, Uruguay
[3] MARL-IDM, New York University, United States
lucas.maia@smt.ufrj.br

ABSTRACT

Beat and downbeat tracking models have improved significantly in recent years with the introduction of deep learning methods. However, despite these improvements, several challenges remain. Particularly, the adaptation of available models to underrepresented music traditions in MIR is usually synonymous with collecting and annotating large amounts of data, which is impractical and time-consuming. Transfer learning, data augmentation, and fine-tuning techniques have been used quite successfully in related tasks and are known to alleviate this bottleneck. Furthermore, when studying these music traditions, models are not required to generalize to multiple mainstream music genres but to perform well in more constrained, homogeneous conditions. In this work, we investigate simple yet effective strategies to adapt beat and downbeat tracking models to two different Latin American music traditions and analyze the feasibility of these adaptations in real-world applications concerning the data and computational requirements. Contrary to common belief, our findings show it is possible to achieve good performance by spending just a few minutes annotating a portion of the data and training a model in a standard CPU machine, with the precise amount of resources needed depending on the task and the complexity of the dataset.

1. INTRODUCTION

Meter tracking means following the pulsating temporal structure of music from audio signals, which implies identifying at least beats and downbeats [1]. It is a long-standing area of research in music information retrieval (MIR) with applications ranging from automatic DJ mixing [2] to musicological studies [3]. Meter tracking has gone through a big transformation in the last decade due to the introduction of deep learning (DL) techniques [4–7], which brought an improvement in performance as well as a change in the design paradigm of related methods [8]. Nowadays, beat and downbeat tracking models rely mostly on supervised DL [8], and thus become data-driven and requiring large amounts of annotated data to generalize to different songs, genres or datasets.

This dependence on annotated data poses many challenges to the widespread use and adoption of such models, especially for culturally specific music traditions [9–11], which often lack annotated data as producing annotations requires culturally-aware expertise. For this reason, off-the-shelf general-purpose models typically underperform in these music genres since they are underrepresented in the datasets used for training. Nevertheless, previous work on some Latin American music traditions shows that if annotations are available, training statistical models can produce good performance results [12].

Recent works have started to look at beat tracking from a different perspective. Instead of developing "universal" models capable of performing equally well across various music genres (requiring large quantities of labeled data), recent efforts have shifted towards adapting preexisting models to succeed on a subset of interest [13], which can be as restricted as a single musical piece [14, 15]. This paradigm aligns well with real-world applications, where it is reasonable for a user to spend a short time producing a few seconds of annotations to get a good performance.

We apply this idea to the refinement of a meter tracking model so that it works well in a particular music genre. We argue that if the genre presents enough homogeneity in terms of its instrumentation and metric structure, as is the case with many Latin American music traditions, it is possible to adapt meter tracking models to perform notably well with just a few annotated data points. We explore this adaptability in terms of data, performance and computational cost. While focusing on two Latin American music genres, *samba* and *candombe*, we study the adaptation of a deep learning state-of-the-art model [16] and compare it with a simpler statistical model [17]. Our contributions are: 1) We perform a detailed analysis on how much annotated data and computation time in CPU are needed to achieve close to "full-dataset" performance in *samba* and in *candombe*, including models trained from scratch and fine-tuned, and compare them to off-the-shelf models trained with Western music; 2) We propose initial experiments to understand the homogeneity conditions under which this adaptation will be successful; 3) We open-source our experiments and provide pre-trained models.

© L.S. Maia, M. Rocamora, L.W.P. Biscainho, and M. Fuentes. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** L.S. Maia, M. Rocamora, L.W.P. Biscainho, and M. Fuentes, "Adapting Meter Tracking Models to Latin American Music", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

1.1 Other adaptive methods

Existing adaptive methods typically feature some form of transfer learning or fine-tuning, use deep learning models, and are concerned with either beat tracking [13–15] or onset detection [18]. Fiocchi et al. [13] adapt a beat tracking model via transfer learning from Western music to a dataset of Greek music. The authors explore both recurrent neural networks (RNNs) and long-short term memory networks (LSTMs), plus a dynamic Bayesian network (DBN) for inference. The model is fine-tuned using a big training set, though with limited success which the authors attribute to the challenges of the dataset. Even though this is an interesting approach for exploring the idea of adapting to a particular music genre, RNNs and LSTMs are known to be computationally expensive, so in the context of real-world model adaptation they are a concerning choice.

On the other hand, Pinto et al. [14] and Yamamoto [15] explore temporal convolutional networks (TCNs), and focus on the adaptation of models to a particular piece of interest of the user. These authors showed that is possible to adapt TCN models using very small quantities of data (in the order of seconds) to work well, in particular with musically challenging pieces. Furthermore, the TCN is a light-weighted model, computationally more efficient.

Finally, Fonseca et al. [18] apply similar ideas to the adaptation of an onset detection model (also featuring TCNs) to a Latin American music tradition: *maracatu de baque solto*. The authors fine-tune the last layers of the TCN with just a few seconds of manual annotations, and show the advantage of instrument-specific models for the automatic annotation of onsets for musicological studies.

1.2 Latin American Music Traditions

Candombe drumming is a musical tradition from Uruguay that constitutes an essential part of its popular culture and African heritage. Its rhythm is structured in 4/4 meter, and it is played while marching in the streets using three types of drums of different sizes and pitches: *chico*, *repique*, and *piano*. Each of these drums has a distinctive rhythmic pattern and musical role. An additional time–line pattern, called *clave* or *madera*, is shared by the three drums. The *chico* drum is the timekeeper; it repeats a one-beat pattern that establishes the pulse throughout the performance. The *repique* drum is the improviser; it alternates *clave* patterns and characteristically syncopated phrases. The *piano* drum delineates the timeline with distinctive one-cycle patterns and occasionally interposes ornamented *repique*-like figurations. The rhythm shares many traits with other musical traditions of the Afro-Atlantic world. Notably, some of its rhythmic patterns have strong phenomenological accents displaced with respect to the metric structure and divide the rhythmic cycle irregularly with few strokes on the beat.

In parallel, *samba* is a Brazilian musical genre deeply rooted in Brazilian culture, and also has African origins. The word "*samba*" actually describes a family of different subgenres, the most famous arguably being *samba de enredo*, *partido alto*, *bossa nova*, and *pagode*. Similarly to *candombe*, *samba* can be played while parading, which is most common during the festivities of *Carnaval*. It can also be performed in more informal settings, in *rodas* and bars, or even as a chamber-music-like style. Its rhythm is commonly perceived in 2/4 meter, and is conveyed by several types of percussion instruments —– *tamborim*, *pandeiro*, *surdo*, *cuíca*, *agogô*, among others. Each instrument has a handful of distinct patterns [19], and more than one instrument may act as the timekeeper. Because of this combination of timbres and pitches, the texture of a performance can become very complex. *Samba* has unmistakable characteristics as the strong accent on the second beat and the development of contrametric structures.

2. METHOD

Following our intuition about the high homogeneity of *candombe* and *samba* as discussed in Section 1.2, our objective is to understand if it is possible to train meter tracking models with small quantities of data from these music traditions, and if so, how much is needed. To that end, we train the models with increasing amounts of annotated data, ranging from less than a minute up to nearly 40 min, and compare the performance and computational cost of each configuration against the others. We contrast three different training strategies: 1) training the model from scratch with either *candombe* or *samba* snippets; 2) fine-tuning a model trained with 38 h of data from diverse datasets of Western music to work on either *candombe* or *samba*; and 3) same as the previous two, but training the models with data augmentation to artificially increase the "small data" input. We use a state-of-the-art temporal convolutional network model [16] for our experiments, as it presents a good compromise between performance and computational cost. We contrast this model against off-the-shelf models trained in Western music. To understand the adaptability and computational cost of deep learning based methods, we compare the TCN against another simple yet effective baseline, a Bayesian model (BayesBeat) [17]. In the following, we explain our methodology in detail.

2.1 Datasets

We have selected datasets of two different Afro-rooted Latin American music traditions for our experiments. First, the *Candombe* dataset [12, 20], which consists of 35 recordings of *candombe* drumming, for a total of nearly 2.5 h. Each track contains an ensemble recording of three to five drummers using different configurations of drums. Tempo varies greatly and often increases along the performance. To represent the *samba* genre, we use the "acoustic mixtures" data from the BRID dataset [21]. These correspond to 93 short tracks (about 30 s each) of musicians playing together rhythm patterns found in *samba* and two of its subgenres (*samba de enredo* and *partido alto*). Ten different instrument classes are represented, and two to four musicians take part in each track. The dataset contains a variety of tempi; however, tempo remains fairly constant within each track. In order to consistently train our models with about the same amount of data from both *candombe*

and *samba*, and also to allow the comparison between the results obtained for both sets, *candombe* tracks were segmented into non-overlapping 30 s excerpts. In each experiment repetition, we use a sample of 93 *candombe* excerpts.

We also used six datasets to train the baseline TCN model: Ballroom [17, 22], Beatles [23], GTZAN [24, 25], and RWC (Classical, Popular, Jazz) [26, 27]. These are commonly used in meter tracking tasks and together correspond to over 38 h of audio data. The Ballroom and GTZAN datasets comprise many diverse music genres (e.g., waltz, tango, rumba, rock, pop, country, etc.). We used the loaders from *mirdata v0.3.6* [28], except for a custom loader used with Ballroom.

2.2 Working with small size datasets

For our experiments, in all cases, we first separate train and test data (80% and 20% of 93 excerpts respectively) to ensure a fair assessment of the models. Then, we divide the training data into six subsets, spanning {4, 9, 18, 37, 55, 74} 30-second tracks. We want to determine how differently the models adapt to small quantities of data, so we followed a similar approach to that of [14] to define the amount of data to be used for training. We select short 10 s temporal regions at the beginning of the audio excerpts, along with the corresponding beat and downbeat annotations, and discard the remaining audio portion. Then we split each of these regions into two adjacent 5 s parts, the first to be used for training and the second reserved for validation in the TCN model; alternatively, we use the entire 10 s for training the Bayesian model with off-the-shelf parameters. Considering that each snippet only lasts 10 s, these data subsets add up to approximately 40 s, 1.5, 3, 6, 9, and 12 min of annotations, respectively. The rationale behind this strategy is that given a set of recordings of such Latin American music traditions in real-world applications, it would be reasonable to ask a user to annotate just a few seconds to a few minutes of data; of course, the less data needed, the better.

Given that we are using very few data points to train the models, performance is strongly affected by data sampling. To mitigate this, we repeat all of our experiments 10 times with different seeds for the random data split generation, which means that models are trained 10 times with each of the different subset sizes. Note that selecting the best strategies for data sampling is out of the scope of this work, and left to be addressed in the future. Test data are left uncut, i.e., we use the full 30 s, to keep compatibility with common model evaluation practices in meter tracking.

2.3 TCN Model

We use in our experiments the TCN multi-task model presented in [16], in particular the open-source implementation of [8]. In this work, we focus on meter tracking, and ignore the tempo estimation head of the network. First, the TCN estimates the beat and downbeat likelihood. Then, we use two different implementations of a DBN (*DBNBeatTracker* and *DBNDownBeatTracker* from *madmom v0.17.dev0* [29]) to infer the final positions of beats and downbeats respectively. Inferring them separately rather than jointly led to better results.

2.4 Training strategies

2.4.1 Training from scratch (TCN-FS)

For datasets with high similarity in terms of instrumentation, rhythmic patterns, and tempo, we expect that we can train a model from scratch with a few training points that would work well for most of the data.

Following the explanation in Section 2.2, we train one model per data subset, and repeat this 10 times with randomly initialized weights and seeds. We also consider the case in which all annotations are available and include the analysis of model performance when training with the entire 30-second excerpts. In this situation, we split the 74 train excerpts into train and validation (75%/25%). For every strategy, we use a learning rate of 0.005, and reduce it by a factor of 0.2 if validation loss did not improve after 10 epochs. We train for a maximum of 100 epochs, early stopping at 20 epochs.

2.4.2 Fine-tuning (TCN-FT)

We also approach the problem of meter tracking in a culture-specific setting from a "transfer learning" perspective. Following [13, 14, 18], we adapt a meter tracking model that was previously trained for a different musical context. The intuition here is that if the model is first trained on a large dataset, even if it was built around Western music, it can serve as a good starting point for a model that is to be tuned for a specific out-of-training music tradition. This is a realistic approach since most of the available annotated data and trained models are Western-based. For this purpose, we trained a baseline TCN model on the Ballroom, Beatles, GTZAN, and RWC datasets. Due to the nature of its training data, this baseline model has to cope with many different meters, genres and acoustic conditions, which makes it a good starting point. We fine-tuned it by using the same training procedure described previously with the initial learning rate reduced to 0.001, a fifth of the value used in the FS approach, as in [14].

2.4.3 Data augmentation (TCN-FTA, TCN-FSA)

Data augmentation techniques are useful for artificially increasing the number of training data points, which can be of great benefit in cases of low or insufficient data such as ours. In order to evaluate the impact of data augmentation in our models, we adopted a simple strategy inspired by [14, 16] in the experiments conducted with the TCN model: computing the input STFTs with different frame rates, i.e., varying hop sizes, so as to even out the distribution of tempi in the train set. Instead of randomly sampling from a normal distribution around the annotated tempo, we selected a set of frames rates $\pm 2.5\%$ and $\pm 5\%$ around its value. This allowed us to increase our sample size five-fold while maintaining the same amount of annotation effort. Models obtained with the data augmentation procedure are labeled TCN-FSA and TCN-FTA, for the training strategies described in Sections 2.4.1 and 2.4.2.

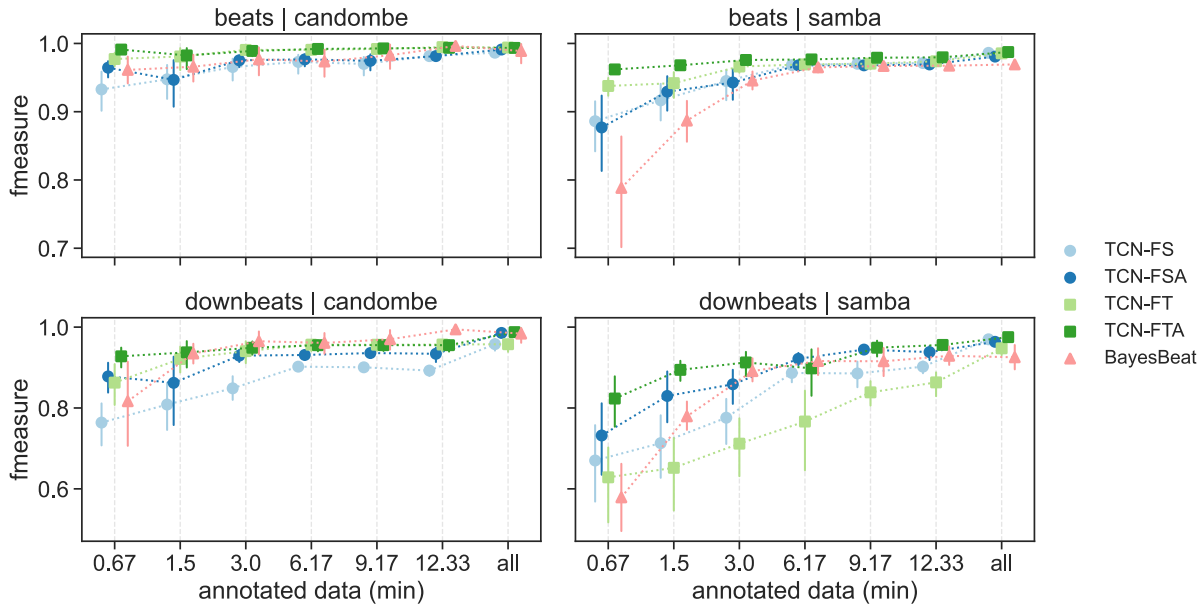


Figure 1: Performance of different model and training configurations. Label "all" indicates fully-annotated dataset.

2.5 Baselines

We include two types of baselines. Firstly, the BayesBeat statistical model [17] is used as reference to the adaptability and computational cost of the TCN. It has fewer parameters, thus training is faster. The second type of baselines are three off-the-shelf models—a Signal Processing technique, and two neural networks trained on Western music—; they illustrate the need for tailor-made solutions/adaptations in our context. Details presented below.

BayesBeat. This is based on the *dynamic bar pointer model* [30], and it simultaneously estimates beats, downbeats, tempo, meter, and rhythmic patterns, by expressing them as hidden variables in a hidden Markov model (HMM). An observation feature based on the spectral flux is computed from the audio signal and an observation model uses Gaussian mixture models (GMM) that are fitted during training to the feature values of each bin in a one–bar grid, so that rhythmic patterns are learned. Several patterns can be modeled, though one pattern is assumed to remain constant throughout the audio signal.

BayesBeat has a few hyperparameters that the user should choose depending on the music. Those are the number of rhythmic patterns, the type of feature to use (e.g., using only low, or low and high frequencies), and the feature grouping (e.g., how to compute the rhythmic pattern clusters), the tempo range, and whole note subdivisions. In [17], it is reported that using two separate frequency bands ($\gtrless$ 250 Hz) helps finding the correct metrical level and is beneficial for beat and downbeat tracking. But, considering more frequency bands did not improve the results [17]. According to [31], using one rhythmic pattern per rhythm class is usually enough to achieve a good performance and provides the best results in most cases. Following this, we use one rhythmic pattern and two frequency bands.

Off-the-shelf baselines. We use the joint beat and downbeat tracking model of Böck et al. [4] as per its implementation in *madmom v0.17.dev0* [29]. It consists of an LSTM-based model trained in ten datasets spanning Western genres, and Carnatic, Cretan and Turkish music excerpts. We also include the beat tracker from Ellis [32], which estimates a global tempo and then uses dynamic programming to find the best set of beats that reflect such tempo. As a final baseline, we include the TCN of Section 2.3 trained with the Western datasets from Section 2.1 (TCN-BL).

2.6 Evaluation metrics

We use as our main metric F-measure [33], along with the continuity-based metrics [34, 35] CMLt ("correct metrical level") which corresponds to the ratio between correct and annotated beats, and AMLt ("allowed metrical level") which accepts phase errors of half a beat period or octave errors in estimation. For the computational cost of the models, we simply report the time they take to train by using in-build timing functions in the code.

3. EXPERIMENTS AND RESULTS

3.1 Performance of models

Figure 1 shows the F-measure results for the TCN models trained for *candombe* and for *samba* with different amounts of data using each of the training strategies, as well as BayesBeat, computed as the bootstrapped results of ten experiments (95% confidence) with different random seeds for each combination of model and data amount.

A first striking observation is that for both beats and downbeats, the performance curve for most models has a small positive slope, which means it is indeed possible to nearly achieve best model performance (which would require training with full dataset) by just training with few samples. This is particularly true for the estimation of beat, for which models rapidly reach F-measure scores above 80% with less than a minute of data in both *candombe* and *samba* for almost all configurations. This is an interesting

Model	*Candombe*						*Samba*					
	Beat			Downbeat			Beat			Downbeat		
	CMLt	AMLt	F	CMLt	AMLt	F	CMLt	AMLt	F	CMLt	AMLt	F
BayesBeat_0.67	95.0	95.0	96.1	82.2	92.0	81.7	70.5	74.2	78.8	57.4	72.9	57.9
BayesBeat_12.33	99.6	99.6	99.6	99.8	99.8	99.4	93.5	96.0	96.7	92.5	94.9	92.9
BayesBeat_all	98.6	98.6	98.9	98.8	98.8	98.4	94.0	96.0	96.9	92.0	95.3	92.5
TCN-FSA_0.67	96.1	96.2	96.4	87.7	90.9	87.8	83.4	86.3	87.7	60.7	80.7	73.2
TCN-FSA_12.33	98.1	98.1	98.1	93.9	96.3	93.4	94.3	95.7	96.9	91.1	95.0	93.8
TCN-FSA_all	99.0	99.3	99.1	99.2	99.6	98.6	95.7	98.2	98.1	95.7	98.0	96.4
TCN-FTA_0.67	99.2	99.2	99.1	93.5	96.8	92.8	92.9	95.7	96.2	69.4	90.0	82.3
TCN-FTA_12.33	99.6	99.6	99.4	96.3	99.7	95.5	95.8	97.2	98.0	92.5	96.9	95.6
TCN-FTA_all	99.5	99.5	99.3	99.3	99.8	98.8	97.3	98.7	98.7	96.7	97.2	97.5
TCN-BL	11.1	18.7	15.9	14.9	31.9	4.1	46.5	65.6	60.0	5.9	52.5	9.6
Ellis [32]	34.8	38.1	38.0	-	-	-	82.3	87.6	87.1	-	-	-
Böck [4]	11.7	14.4	11.5	26.7	40.3	0.5	46.9	76.0	66.4	5.2	66.6	2.0

Table 1: Mean F-measure (F) and continuity scores (CMLt, AMLt) in beat and downbeat tracking tasks across both genres.

result, meaning that not much gain in performance is expected with the increase of annotations for such datasets. An end-user could annotate less that a minute of data and yet obtain decent performances. The same holds for downbeat in *candombe*, but not in *samba*. In the latter, there is a clear gain in adding more data, which has to do with the differences between the two rhythms, as discussed below.

Differences between *candombe* and *samba*. Observing the results in Figure 1, we see that the models tend to require more data to achieve better performance on *samba* than on *candombe*, and the uncertainty about the performance for *samba* is larger. Our intuition behind this result is that, as mentioned in Section 1.2, because *samba* has a bigger combination of timbres and pitches than candombe, the decision of what snippets to annotate (i.e., the sampling) might be more critical for the former than for the latter, e.g., ensuring timbre representation.

Best model configuration. The best performing configuration for beat and downbeat tracking in both music traditions is the fine-tuned TCN model with data augmentation (FTA). Particularly, data augmentation produced significant improvement in performance for downbeat tracking in samba. Interestingly, for the adaptive setting concerned in this work, the BayesBeat baseline is competitive with the TCN model, especially considering the computational cost (see Section 3.2).

Comparison with off-the-shelf benchmarks. Table 1 shows the performance of the TCN and the BayesBeat baseline for different data subsets, namely the smallest and largest subsets, and the full dataset. It also shows the performance of the three off-the-shelf baselines explained in Section 2.5. In alignment with previous works [12, 21], the models trained with Western music (TCN-BL and Böck [4]) perform very poorly in *candombe*, and reach only about 66% F-measure in *samba*, both significantly lower than the performance of the same models in Western music genres. The model of Ellis [32] scores considerably better, but is not consistent in both datasets. This shows the necessity of adapting meter tracking models to these music genres, as even the models trained with the smallest subsets of data (0.67 min) outperform the baselines.

3.2 How much time do the models take to train?

Our analysis is motivated by the adaptation of meter tracking models in real-world use cases. For this adaptation to make sense it has to be done quickly. In this regard, we estimate the time each model configuration takes in training, and contrast it with the BayesBeat baseline. Figure 2 shows how the train duration varies with the size of the train set for *samba* (very similar results were obtained for *candombe*). The TCN takes about the same time in both *samba* and *candombe*, with a minimum of about 100 s for the smallest subset. Among the TCN configurations, the most expensive ones use data augmentation. This makes sense given that more data is used for training. As expected, the BayesBeat trains significantly faster than the TCN, taking on average 1.62 s to train with 0.67 min of data, and being in the order of 50 to 150 times faster than the TCN when data augmentation is not used. This big gap in computing time, together with the results of Figure 1 and Table 1, makes BayesBeat an overall good alternative for adapting meter tracking to these Latin American music. We observed that all configurations take about the same inference time, around 25 s for the full test set.

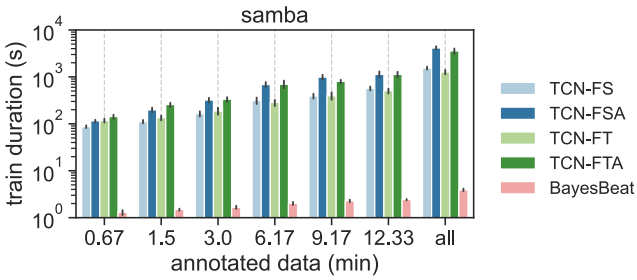


Figure 2: Training time for the different amounts of data.

3.3 When can we train with small data?

Our intuition is that the more variability in the data (in terms of meters, rhythmic patterns and instrumentation), the harder it is for a model to learn with small data. This aligns with our experiments in the adaptability of these methods to *samba* and *candombe*, and also agrees with the musicological insights of Section 1.2. To have a more

quantitative understanding of this, we derived a bar profile for each type of music. First we extract a feature map from each excerpt using the beat/downbeat annotations to time-quantize a locally normalized onset strength function [36] at the tatum scale — this was done with the *carat* [37] toolbox, considering the tatum duration as one quarter of the time-span between successive beats. Then, for each dataset, we summarize these feature maps across time, which results in a distribution of feature values per tatum. To allow an analysis of these profiles in different regions of the spectrum, we compute the onset strength in two frequency bands (20 Hz to 200 Hz; and > 200 Hz). We present these distributions as violin plots in Figure 3 for *candombe*, *samba*, and for the Ballroom dataset.

In Figure 3, we verify that for some tatums strength distributions are concentrated around 1 or 0, indicating a strong characteristic accent or lack thereof at that point of the bar respectively. High variance, in its turn, means "fuzzyness" in the rhythm pattern, which could justify the difficulty in learning that rhythm, specially with small data.

Samba, which has eight tatums per bar (2/4 meter), is known for having a strong metrical accent at beat 2, which we may readily identify in its low-frequency channel at tatum 5. The first beat also has a high median value but is less "deterministic" due to its high variance. In turn, the low-frequency profile of *candombe* displays a high-variance downbeat, no accent on beat 2, and strong accents on beats 3 and 4, but also a strong contrametric accent at tatum 4. These characteristics could help explain why the off-the-shelf beat tracking models, which expect beats to be accented, perform worst on *candombe*. Looking back at *samba*, we see that tatums 2 and 3 show small standard deviations and correspond to "off" tatums; together with beat 2, they make three out of eight tatums that exhibit very small variance in the low channel. In *candombe*, besides tatum 4, tatums 2, 3, 7, 8, 9, 14, and 16 also present small variance. This abundance of "anchor" points could justify why adaptation in *candombe* came with little data.

In Ballroom, we clearly see that beats are distinct for having high strength and low variance in both channels, whereas the rest of the tatums show no clear trend. Its few reference points could pose a challenge for learning models. Furthermore, beat patterns (the combination the four tatums in-between beats, including the beat itself) are also indistinguishable from one another, which could aggravate this matter. To test these observations, we trained a set of models from scratch for Ballroom using the same methodology that for *samba* and *candombe*. Results are depicted in Figure 4. The performance results correlate with the intuition that Ballroom is a more challenging dataset given that it comprises multiple genres, and also that for learning beat and downbeat more data would be needed.

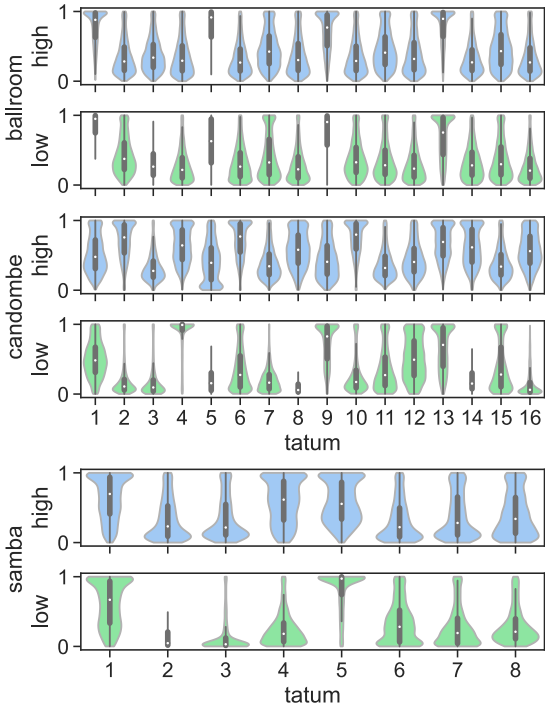


Figure 3: Tatum strength distribution per frequency band for Ballroom (just 4/4 tracks), *candombe*, and *samba*.

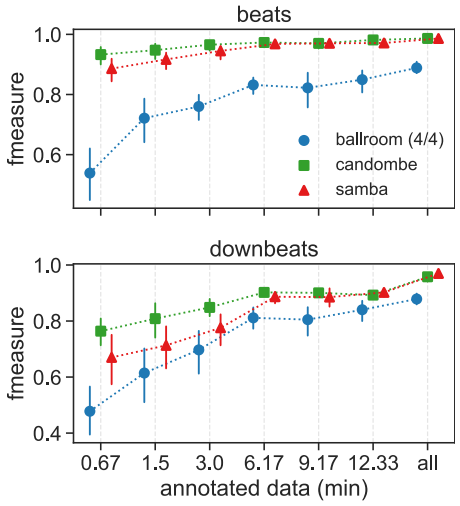


Figure 4: TCN-FS performance in Ballroom.

4. CONCLUSIONS AND FUTURE WORK

We adapted a meter tracking model using small quantities of data to work in particular Latin American music traditions, namely *samba* and *candombe*. We showed that, under certain homogeneity conditions, it is indeed possible to train such models with a few minutes of annotated data and training cycles, and obtain almost full-dataset performance. This result has promising consequences in real-world applications, as it opens the possibility of adapting such models to other music genres with modest labeling efforts. The most competitive model is a fine-tuned TCN with data augmentation, whereas BayesBeat is a good option under computational cost constraints. In the future, we will investigate rhythm complexity metrics that could serve to predict the amount of annotated data needed to adapt meter tracking models to particular music genres.

5. ACKNOWLEDGEMENTS

This study was partially funded by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) — Finance Code 001.

6. REFERENCES

[1] A. Klapuri, A. Eronen, and J. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Trans. Audio, Speech, Language Process.*, vol. 14, no. 1, pp. 342–355, 2006.

[2] B.-Y. Chen, W.-H. Hsu, W.-H. Liao, M. A. M. Ramírez, Y. Mitsufuji, and Y.-H. Yang, "Automatic DJ transitions with differentiable audio effects and generative adversarial networks," in *Proc. 2022 IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2022, pp. 466–470.

[3] A. Srinivasamurthy, A. Holzapfel, K. K. Ganguli, and X. Serra, "Aspects of tempo and rhythmic elaboration in Hindustani music: A corpus study," *Frontiers in Digital Humanities*, vol. 4, 2017. [Online]. Available: https://www.frontiersin.org/article/10.3389/fdigh.2017.00020

[4] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proc. 17th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, New York, USA, Aug. 2016, pp. 255–261.

[5] S. Durand, J. P. Bello, B. David, and G. Richard, "Robust downbeat tracking using an ensemble of convolutional networks," *IEEE Trans. Audio, Speech, Language Process.*, vol. 25, no. 1, pp. 76–89, Jan 2017.

[6] M. Fuentes, B. McFee, H. Crayencour, S. Essid, and J. Bello, "Analysis of common design choices in deep learning systems for downbeat tracking," in *Proc. 19th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, Paris, France, Sep. 2018, pp. 106–112.

[7] M. Heydari, F. Cwitkowitz, and Z. Duan, "Beatnet: CRNN and particle filtering for online joint beat downbeat and meter tracking," in *Proc. 22nd Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, Online, Nov. 2021, pp. 270–277.

[8] M. E. P. Davies, S. Böck, and M. Fuentes, *Tempo, Beat and Downbeat Estimation*. https://tempobeatdownbeat.github.io/tutorial/intro.html, Nov. 2021. [Online]. Available: https://tempobeatdownbeat.github.io/tutorial/intro.html

[9] C. N. Silla Jr., A. L. Koerich, and C. A. A. Kaestner, "The Latin music database," in *9th Int. Conf. Music Inf. Retrieval (ISMIR)*, Philadelphia, USA, Sep. 2008, pp. 451–456.

[10] E. Cano, F. Mora-Ángel, G. A. López Gil, J. R. Zapata, A. Escamilla, J. F. Alzate, and M. Betancur, "Sesquialtera in the Colombian bambuco: Perception and estimation of beat and meter," in *Proc. 21st Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, Montreal, Canada, Oct. 2020, pp. 409–415.

[11] G. M. Sarria M., J. Diaz, and C. Arce-Lopera, "Analyzing and extending the Salsa music dataset," in *Proc. XXII Symp. Image, Signal Process., Artif. Vision (STSIVA)*, Bucaramanga, Colombia, Apr. 2019, pp. 1–5.

[12] L. Nunes, M. Rocamora, L. Jure, and L. W. P. Biscainho, "Beat and downbeat tracking based on rhythmic patterns applied to the Uruguayan Candombe drumming," in *Proc. 16th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, Málaga, Spain, Oct. 2015, pp. 246–270.

[13] D. Fiocchi, M. Buccoli, M. Zanoni, F. Antonacci, and A. Sarti, "Beat tracking using recurrent neural network: a transfer learning approach," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Rome, Italy, Sep. 2018, pp. 1929–1933.

[14] A. S. Pinto, S. Böck, J. S. Cardoso, and M. E. P. Davies, "User-driven fine-tuning for beat tracking," *Electronics*, vol. 10, no. 13, Jun. 2021.

[15] K. Yamamoto, "Human-in-the-loop adaptation for interactive musical beat tracking," in *Proc. 22nd Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, Online, Nov. 2021, pp. 794–801.

[16] S. Böck and M. E. P. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," in *Proc. 21st Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, Montreal, Canada, Oct. 2020, pp. 574–582.

[17] F. Krebs, S. Böck, and G. Widmer, "Rhythmic pattern modelling for beat and downbeat tracking from musical audio," in *Proc. 14th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, Curitiba, Brazil, Nov. 2013, pp. 227–232.

[18] J. Fonseca, M. Fuentes, F. Bonini Baraldi, and M. E. P. Davies, "On the use of automatic onset detection for the analysis of Maracatu de baque solto," in *Perspectives on Music, Sound and Musicology: Research, Education and Practice*, L. C. Castilho, R. Dias, and J. F. Pinho, Eds. Cham, Switzerland: Springer, 2021, pp. 209–225.

[19] G. Gonçalves and O. Costa, *The Carioca Groove: The Rio de Janeiro's Samba Schools Drum Sections*. Rio de Janeiro, Brazil: Groove, 2000.

[20] M. Rocamora, L. Jure, B. Marenco, M. Fuentes, F. Lanzaro, and A. Gómez, "An audio-visual database of Candombe performances for computational musicological studies," in *Proc. II Congreso Int. de Ciencia y Tecnología Musical (CICTeM)*, Buenos Aires, Argentina, Sep. 2015, pp. 17–24.

[21] L. S. Maia, P. D. Tomaz Jr., M. Fuentes, M. Rocamora, L. W. P. Biscainho, M. V. M. Costa, and S. Cohen, "A novel dataset of Brazilian rhythmic instruments and some experiments in computational rhythm analysis," in *Proc. 2018 AES Lat. Am. Congr. Audio Eng. (AES LAC)*, Montevideo, Uruguay, Sep. 2018, pp. 53–60.

[22] F. Gouyon, A. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano, "An experimental comparison of audio tempo induction algorithms," *IEEE Trans. Audio, Speech, Language Process.*, vol. 14, no. 5, pp. 1832–1844, Sep. 2006.

[23] M. E. P. Davies, N. Degara, and M. D. Plumbley, "Evaluation metrics for musical audio beat tracking algorithms," Queen Mary University of London, London, UK, Tech. Report, Oct. 2009.

[24] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech, Audio Process.*, vol. 10, no. 5, pp. 293–302, Jul. 2002.

[25] U. Marchand and G. Peeters, "Swing ratio estimation," in *Proc. 18th Int. Conf. Digital Audio Effects (DAFx)*, Trondheim, Norway, Dec. 2015, pp. 423–428.

[26] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, Classical, and Jazz music databases," in *Proc. 3rd Int. Conf. Music Inf. Retrieval (ISMIR)*, Paris, France, Oct. 2002, pp. 287–288.

[27] M. Goto, "Development of the RWC music database," in *Proc. 18th Int. Congr. Acoust. (ICA)*, Kyoto, Japan, Apr. 2004, pp. I–553–556.

[28] R. M. Bittner, M. Fuentes, D. Rubinstein, A. Jansson, K. Choi, and T. Kell, "mirdata: Software for reproducible usage of datasets," in *Proc. 20th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, 2019.

[29] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proc. 24th ACM Int. Conf. Multimedia*, Amsterdam, The Netherlands, Oct. 2016, pp. 1174–1178.

[30] N. Whiteley, A. Cemgil, and S. Godsill, "Bayesian modelling of temporal structure in musical audio," in *Proc. 7th Int. Conf. Music Inf. Retrieval (ISMIR)*, Victoria, Canada, Oct. 2006, pp. 29–34.

[31] A. Holzapfel, F. Krebs, and A. Srinivasamurthy, "Tracking the "odd": Meter inference in a culturally diverse music corpus," in *Proc. 15th Int. Soc. Music Inf. Retrieval Conf. (ISMIR)*, Taipei, Taiwan, Oct. 2014, pp. 425–430.

[32] D. P. W. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, Mar. 2007.

[33] S. Dixon, "Evaluation of the audio beat tracking system BeatRoot," *J. New Music Res.*, vol. 36, no. 1, pp. 39–50, 2007.

[34] S. W. Hainsworth, "Techniques for the automated analysis of musical audio," Ph.D. dissertation, Department of Engineering, Cambridge University, 2003.

[35] A. P. Klapuri, A. J. Eronen, and J. T. Astola, "Analysis of the meter of acoustic musical signals," *IEEE Trans. Audio, Speech, Language Process.*, vol. 14, no. 1, pp. 342–355, Jan. 2006.

[36] M. Rocamora, L. Jure, and L. W. P. Biscainho, "Tools for detection and classification of piano drum patterns from Candombe recordings," in *Proc. 9th Conf. Interdisciplinary Musicology (CIM14)*, Berlin, Germany, Dec. 2014, pp. 382–387.

[37] M. Rocamora and L. Jure, "carat: Computer-Aided Rhythmic Analysis Toolbox." in *Proc. Analytical Approaches World Music (AAWM)*, Birmingham, UK, Jul. 2019.

CRITIQUING TASK- VERSUS GOAL-ORIENTED APPROACHES: A CASE FOR MAKAM RECOGNITION

Kaustuv Kanti Ganguli[1,3,†] **Sertan Şentürk**[2,3,†] **Carlos Guedes**[3]

[1] College of Interdisciplinary Studies, Zayed University, UAE
[2] Independent Researcher, UK
[3] Music and Sound Cultures Lab, New York University Abu Dhabi, UAE

Kaustuv.Ganguli@zu.ac.ae, sertan.senturk@gmail.com, carlos.guedes@nyu.edu

ABSTRACT

Computational Musicology and Music Information Retrieval (MIR) address the core musical question under study from a different perspective, often combining top-down vs. bottom-up approaches. However, the evaluation metrics for MIR tend to capture the model accuracy in terms of the goal. [1] For instance, mode recognition is implemented with a goal to evaluate and compare melodic analysis approaches, but it is worth investigating if at all it lends itself as one befitting proxy task. [2] This is particularly relevant in non-Eurogenetic music repertoires where the grammatical rules are rather prescriptive. We employ methodologies that combine domain knowledge and data-driven optimisations as a possible way for a comprehensive understanding of these relationships. This is tested on Makam, one of the understudied corpora in MIR. We evaluate an array of feature-engineering methods on the largest mode recognition dataset curated for Ottoman-Turkish makam music, composed of 1000 recordings in 50 makams. We also address (ethno)musicology-driven tasks with a view to gathering more profound insights into this music, such as tuning, intonation, and melodic similarity. We aim to propose avenues to extend the study to makam characterisation over the mere goal of recognizing the mode, to better understand the (dis)similarity space and other plausible musically interesting facets.

1. INTRODUCTION

The melodic framework in many music traditions is often governed by the system of modes. A mode can be viewed as falling somewhere between a scale and a tune in terms of its defining grammar, which includes the tonal material, tonal hierarchy, and characteristic melodic movements [1–3]. While the function and the understanding of these frameworks are distinct from a culture-specific perspective, in a broader sense, they may be considered as the modes of the studied music culture. Some music traditions that can be considered modal are Indian art (raga) music, the Turkish/Arabic makam/maqam traditions, and the Gregorian church modes. Concerning the relevance of musical mode in non-Eurogenetic music repertoires, there are two contrasting viewpoints on whether to rethink or reject modal structure. While the former advocates for adapting the concept of mode as an underlying framework to systematise musical patterns, the latter tends to nullify the syntactic jargon of using a foreign language grammar (e.g., mode) to interpret literature in another (e.g., makam).

Makam/maqam is the melodic framework that consists of a system of scales defined by successive intervals, habitual melodic phrases, modulation pathways, ornamentation techniques and aesthetic conventions. It is used in Turkish (and Arabic, including the Middle East and Western Indian Ocean) music, providing a complex set of rules for compositions and performance. A typical subset of the repertoire — Ottoman-Turkish makam music (OTMM) — is well-established as a classical music tradition. Historically, there are a few hundred makams, whereas in practice, most of the repertoire is composed in one of the top 20 makams [4, 5]. In OTMM, melodies typically revolve around an initial tone and a final tone [6], where the final tone is referred to as being synonymous with tonic. There is no definitive reference frequency to tune the tonic. Recognizing makam is in itself a much more difficult task due to various characteristics such as heterophony and high variability in interpretations by musicians [6]. Moreover, from the pedagogy and practice perspective, recognizing the underlying makam with a unary label may not be interesting enough. A knowledge seeker (e.g., from the perspective of an anthropologist or ethnomusicologist) would rather gain wisdom on the characterisation of a makam and its discriminatory aspects to differentiate it from seemingly similar-sounding neighbouring makams.

In the realm of music information retrieval (MIR), mode recognition as a task has been given considerable importance from the purview of this, lending itself as one befitting proxy to evaluate and compare melodic analy-

† Equal contribution; corresponding authors.
[1] We define goal as optimization of evaluation metrics, e.g., accuracy.
[2] We define task as what a model is intended to learn, e.g., modal features such as intervals, note sequence, relative salience, etc.

© K. K. Ganguli, S. Şentürk, and C. Guedes. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** K. K. Ganguli, S. Şentürk, and C. Guedes, "Critiquing Task- versus Goal-oriented Approaches: A Case for Makam Recognition", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

sis approaches. Such a list is lengthy, ranging from automatic transcription, tuning analysis, music discovery, music similarity and recommendation to computational (ethno)musicology applications [7, 8]. However, in most practical scenarios, mode recognition is not the central theme but just the first step evidencing the robustness of acoustic features and/or statistical models. In such a setup, the recognition accuracy tends to become just another optimisation function. This simply indicates that the ethos of mode recognition does not remain a task anymore but a 'goal'. In this work, we aim to critically address how such an approach can mislead the uninitiated audience by attributing the complexity of the musical characteristics to the shortcomings of computational models.

It is inevitable to have thorough interdisciplinary collaboration to address the specifics in their entirety. However, we will approach the first study in this direction from a corpus-based computational musicology paradigm. This means developing culture-aware music technologies combing data- and knowledge-driven methods. We aim to benchmark the reported state-of-the-art literature, improve baseline features and simulate intuitive models to contest them. Once the potential of our approach in achieving the 'goal' accuracy is established, we break away from supervised (classification) to unsupervised (clustering) learning to appreciate the nuances and facilitate a comprehensive understanding of the same realm, this time as a 'task'. The byproduct of such a work is to provide tools for several musicologically-relevant subtasks such as tuning and intonation analysis, temporal modelling, and so on. While musicological studies latch on qualitative and limited representative examples, empirical methods work at the level of larger corpora and are, therefore, particularly useful for information retrieval-based tasks where scalability and reproducibility are highly regarded, if not mandated. The extracted features facilitate the curation of large audio corpora not only by greatly reducing the time and effort spent on manual annotations but also by providing automatically extracted, reliable and reproducible information. It is to be noted that we are limiting the scope of this work to building on past work pertaining to aggregated feature representation disregarding any time information. Hence, sequence models or sophisticated deep learning models (e.g. [9]) are not included in the current discourse.

However, in such corpus-based studies that too are underrepresented in MIR, the features extracted from the data, [3] source code and the experimental results are not always shared, making it more inaccessible to reproduce or build on the literature. Thus the unavailability of public tools, datasets, and reproducible experimentation are major obstacles to computational music information research, especially if such relevant tasks have not been applied to studied music traditions earlier. The CompMusic project [4] contributed towards bridging this gap by creating open corpora and computational tools for several non-Eurogenetic music repertoires. This work builds on the dataset introduced in MORTY (MOde Recognition and Tonic Ydentification toolbox), which is the largest mode recognition dataset curated for OTMM, composed of 1000 recordings in 50 makams [10].

In this work, our contribution is two-fold. Firstly, we adapt a new feature called time-delayed melody surfaces (TDMS) from raga recognition to makam recognition that shows comparable results to that of the current state-of-the-art [11]. The second contribution is to establish a similarity space of makam melodic features that characterise the root cause of erroneous cases. The structure of the paper is as follows. Section 2 discusses relevant literature on mode recognition at large and a detailed review of makam recognition, more as a goal and not a task. Section 3 describes the methodological details on audio preprocessing, the dataset(s), and feature extraction/modeling. Next, the experimental details regarding the model architecture and evaluation strategies are discussed in Section 4.1. This essentially engulfs the 'goal'-oriented approaches and comparison with the state-of-the-art, followed by the discussion of an alternative paradigm of unsupervised learning. The latter aids in the 'task'-based approach and highlights the gained musicological insights from this study and possible avenues of extending to makam characterisation over merely recognizing the label. Finally, Section 5 summarises the contributions and poses the scope for further developments in the current study.

2. MODE RECOGNITION

There has been extensive interest in mode recognition in the last two decades; a good summary is presented in [12]. Most of this work focuses on culture-specific approaches for music traditions like OTMM [10, 11, 13, 14], Carnatic music [15–17], Hindustani music [18–20], Dastgah music [21–23] and medieval chants [24, 25]. A considerable portion of these studies is based on comparing pitch distributions [13, 15, 16, 18, 19, 26], which are shown to be reliable in the respective mode recognition task or, for that matter, goal per se. There also exist recent approaches that are based on characteristic melodic motif mining using network analysis [17, 20], aggregating note models using automatic transcription [27–30], or audio-score alignment [31, 32]. All of these methods have been designed specifically to address the studied music culture (with the exceptions of [20] and [10]), and they are not generalisable to other music cultures without considerable effort. Next, we present some of the specific literature on mode recognition that we base our analyses on.

Pitch Distributions (PD) and Pitch Class Distributions (PCD) have been the state-of-art feature for mode recognition tasks for a very long time [10, 12, 13, 19], irrespective of the fact that they completely disregard the temporal aspects of the melody, which are essential to a mode characterisation [33]. Karakurt et al. [10] applied a joint recognition of makam and tonic using PDs and PCDs. In the training phase, the authors used kNN classifiers with either single or multiple distributions per mode. Their best performing model achieved an accuracy of 71.8% on the OTMM

[3] Commercial audio recordings are generally difficult to be made public due to copyright issues.

[4] https://compmusic.upf.edu/

recognition dataset (explained in Section 3.1). Gulati et al. [20] proposed a novel feature for mode recognition in the context of Hindustani and Carnatic ragas, called the time-delayed melody surface (TDMS), which we will use in this study. Authors reported that TDMS-based models outperform PCD-based models of [10] in the raga recognition task.

Yeşiler et al. [14] used a Multilayer Perceptron (MLP) on pitch distribution of first and last sections together with overall distributions using a feature vector of 159 attributes (53-TET * 3 octaves). The highest accuracy reported is 75.6% on the OTMM recognition dataset with additional insights on relevant segments for better discriminability. Demirel et al. [11] advocate the advantage of using chroma features for the mode recognition task as it discards the need for automatic melody extraction of polyphonic audio, hence getting away with the imperfections thereof. Authors created makam templates from annotated data and used template matching using support vector machine (SVM) classifiers. The best performing model achieved an accuracy of 77% on the OTMM recognition dataset, which, to our knowledge, is the state-of-the-art in makam recognition applied to the OTMM corpus. Other works are not strictly mode recognition but use the framework of representation-cum-distance-measure for discriminating between allied raga-pairs [34, 35]. Section 4 discusses aspects we borrow from this methodology to quantify the classification errors from a clustering viewpoint.

3. METHODOLOGY

In [10], mode recognition is formally defined as classifying the mode of an audio fragment from a discrete set of modes. In the context of OTMM, the problem reduces to classifying the makam. Given that the mode recognition framework is already established and that we are benchmarking on literature applied to OTMM, we save some real estate assuming that the data (pre)processing and partial feature extractions will be exactly reproduced.

3.1 The OTMM Corpus and the Dataset

Considering the lack of open data sources for makam music, the CompMusic project gathered audio recordings, music scores and relevant metadata, and published in the public domain the *Dunya Ottoman-Turkish Makam Music Corpus* [5, 36], which is currently the most representative corpus for OTMM available for computational research purposes. From the corpus, [10] curated a test dataset of audio recordings with annotated makam and tonic, called the *Ottoman-Turkish makam music recognition dataset*. The dataset covers 20 commonly performed makams [5] composed of 1000 audio recordings. A single makam is performed in each recording (i.e. there are 50 recordings per makam). To the best of our knowledge, this dataset is the largest and the most comprehensive dataset for the evaluation of automatic makam recognition. Finally, the dataset has been used by other researchers [10, 11, 14] to demonstrate their methods, including the current state-of-the-art makam recognition approach [11].

We use the latest version of the dataset. [6] We use the pre-computed melody time-series (termed as "predominant melody" by [36]) provided in CompMusic Dunya. [7] The pitch is detected at a hop-size to sampling-rate ratio of 0.023 that translates to 23 ms intervals for 44.1 kHz sampled audio [37]. To compare across performances, it is crucial to normalise the melody with respect to the tonic frequency. For this study, we use manually curated tonic frequencies linked from the *Ottoman-Turkish tonic dataset*. [8] To avoid the effect of nonlinearity in the logarithmic Hz scale, we normalise the pitch time-series to a log-linear cents scale.

3.2 Feature Extraction and Modeling

The next step is to synthesise derived features from the raw predominant melody. These mid- or high-level features can be interpreted and mapped to musicological inferences. We follow an approach akin to Krumhansl's [38] to compute the histogram of pitch samples to construct the pitch-class distribution. The pitch values are octave-folded (0 — 1200 cents) and quantised into p bins of equal width. The bin centre is the arithmetic mean of the adjacent bin edges. The salience of each bin is proportional to the accumulated duration of the pitches within that bin. A probability distribution function is constructed where the area under the histogram sums to unity. Even though we use the equivalent of a PD method attributed to the high bin resolution, we converge to a PCD [18]. The PCD configuration is given in Section 4. The first row of Figure 1 shows PCDs computed from each Mahur, Rast, and Acemşiran in the dataset, with the average pitch at each bin drawn as a dashed line.

The next step is to construct a two-dimensional surface based on the concept of delay coordinates (also termed phase space embedding) [20]. The time-delayed melody surface (TDMS) is a compact representation that captures both the tonal and the temporal characteristics of melody, is robust to octave errors, also partially nullifies the relevance of melody transcription. We experiment with different parameters (See Section 4). The second row of Figure 1 shows TDMS averaged from the TDMS of all recordings in Mahur, Rast, and Acemşiran makams in the dataset. The horizontal and vertical trajectories indicate pitch transitions between the pitch classes. The isolated square shape-formation indicates a separation between the higher and lower tetrachords in the course of the melodic progression. In both PCD and TDMS features, makams Rast and Mahur are similar to a high degree. The PCD of makam Acemaşiran bears relatively small differences from the prior two; however, the TDMS representation manages to significantly differentiate itself via dif-

[5] Namely: Acemaşiran, Acemkürdi, Bestenigar, Beyati, Hicaz, Hicazkar, Hüseyni, Hüzzam, Karcığar, Kürdilihicazkar, Mahur, Muhayyer, Neva, Nihavent, Rast, Saba, Segah, Sultanıyegah, Suzinak, and Uşşak.

[6] **dlfm2016-fix1** — `https://zenodo.org/record/4883680`

[7] `https://dunya.compmusic.upf.edu`

[8] `https://zenodo.org/record/260038`

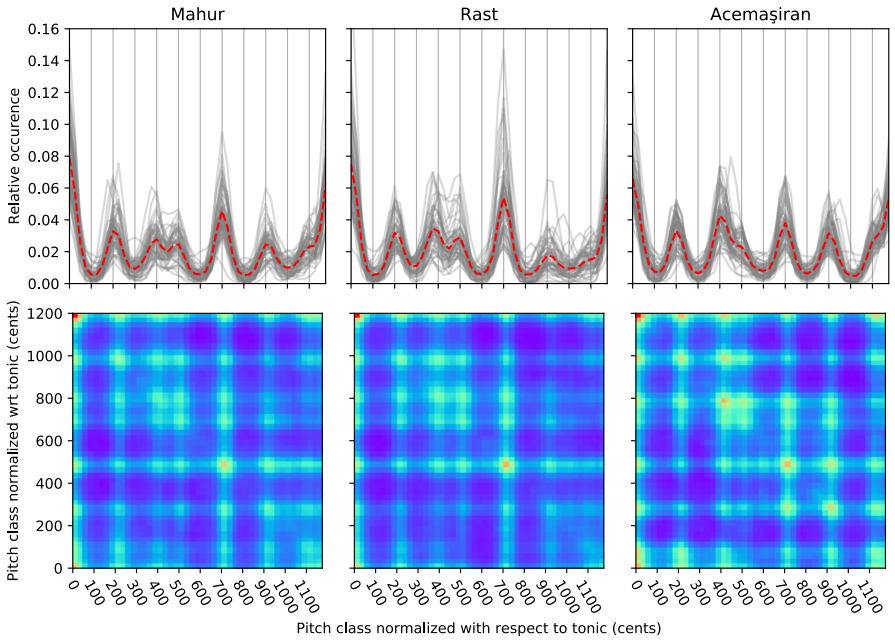


Figure 1. PCD (top row) and TDMS (bottom row) representations of Mahur (left column), Rast (middle column), and Acemaşiran (right column) makams.

ferences in melodic progression, captured through the delay coordinates. For example, in the TDMS representation, makams Mahur and Rast show clear transitions between 6^{th} (900 cents), 7^{th} scale degrees (1100 cents), and the tonic pitch class, whereas makam Acemşiran exhibits a progressions between 3^{rd} (400 cents) and 6^{th} scale degrees (900 cents).

4. EXPERIMENTS

Akin to the title of the paper, one of the main goals of the study is to treat makam recognition as a task and not a goal in itself. Our experiments are divided into two parts. The former addresses the 'goal'-oriented aspect, i.e., the best combination of features and classifiers in order to achieve the *optimal* accuracy. We also stress on intuitions why certain configuration of train-test partitioning would make more sense than others or why certain classifier is meant to 'learn' and not be totally data-proximity dependent. We report and discuss a subset of the results relevant to the optimal settings; the full experimental results are made available. [9]

For PCDs, we use the empirical "optimal" parameters for OTMM reported by [10], namely a bin resolution of 25 cents and Gaussian smoothing applied using a kernel width of 25 cents. We set the bin resolution of TDMS to 25 cents to compare with PCD, and grid-search time delay indices ($\in \{0.25, 0.5.1, 1.5, 2.5, 5\}$ seconds), compression exponents ($\in \{0.1, 0.25, 0.5, 0.75, 1\}$) and Gaussian smoothing kernel widths ($\in \{0, 12.5, 25, 50\}$ standard deviation in cents) in the classification experiments below to find the optimal configuration for OTMM.

[9] https://sertansenturk.com/work-research/ismir-2022-makam/

4.1 (Supervised) Classification

In line with the objective of supervised learning, i.e., to model the intra-class similarity and inter-class differences, the inherent 'goal' is to maximise classification accuracy. However, we carefully choose the feature set and classifier in order to suit the music theory. That is to say, we aim to incorporate knowledge constraints into mainstream data-driven computational models. We restrict ourselves to k-nearest neighbors [10, 12, 13, 19, 20], support vector machine [11, 12], multilayer perceptron [14] and logistic regression [12] that were extensively used in past mode recognition work.

Past studies used different cross-validation (CV) techniques such as leave-one-out CV [13], 10-fold CV [10, 19, 20] and nested k-fold CV [11, 14]. In our initial experiments, we compared nested 10-fold CV, 10-fold CV (without any unseen test set), and 10-times repeated shuffle split CV with 10% of the recordings reserved as test set for each repetition. We used stratified splits in all our experiments to keep the makam classes balanced and repeated each experiment 10 times. We report the mean & standard deviation of classification accuracy reported on the test set. We also compute a confusion matrix for each test set and aggregate it across all test sets in each repeated experiment per model. We report the results of the 10-times repeated shuffle split CV in the rest of the Section. Similar to [20], we observed that TDMS is robust in different time delay indices (between 0.5 and 2.5 seconds), kernel width (less than 25 standard deviations in cents) and compression exponent (above 0.25). For the rest of the experiments, we report results for TDMS with an "optimal" configuration of 1-second time-delay index, 12.5 cents of smoothing kernel width and a compression exponent of 0.5. Table 1

Model	PCD	TDMS
Support Vector Machine	$71.0 \mp 3.2\%$	$\mathbf{77.2 \mp 3.5}\%$
Multilayer Perceptron	$70.9 \mp 3.8\%$	$74.6 \mp 5.0\%$
k-nearest Neighbors	$68.2 \mp 3.4\%$	$70.2 \mp 3.1\%$
Logistic Regression	$66.8 \mp 3.9\%$	$75.5 \mp 4.1\%$

Table 1. Average $\mp$ standard deviation of classification accuracy for all feature-classifier combinations.

shows the mean and standard deviation in classification accuracy for PCD and TDMS using different models. In sum, TDMS with SVM works the best at par with the current state-of-the-art [11]. TDMS consistently performs better than PCD; statistical significance results are kept out of the scope of this work.

The associated confusion matrix for the optimal performing system is shown in Figure 2. Makams {Acemaşiran, Hicaz, Hüzzam} and {Bestenigar, Rast, Uşşak} are examples of highly discriminable and highly confused pairs respectively. The inferences from the confusion matrix are, however, limited to qualitative evaluation of the confused cases and a count of them. In the next Section, we propose a new approach to compute the pairwise distances in a clustering scenario which facilitates a quantitative evaluation of the proportion and magnitude of the confusions. This is, in a way, a manifestation of the recognition task wherein we model the inherent complexity in the data rather than the limitations of the method.

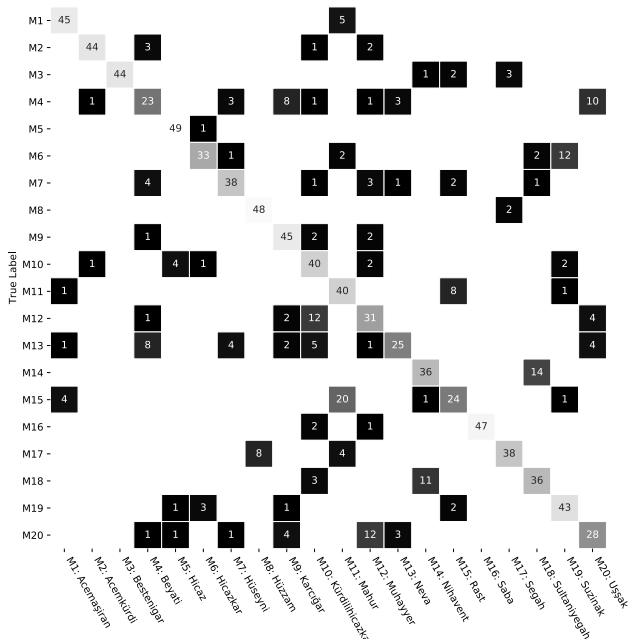


Figure 2. Aggregated confusions matrix for the optimal performing system: TDMS with SVM.

4.2 (Unsupervised) Clustering

To address the second half of the experiments, our focus moves to the 'task'-oriented results. This, we believe, is the highlight of the current contribution in terms of gaining/reconfirming musicological knowledge/concepts from the computational model that can feed back into the pedagogy and practices to further enrich the repertoire.

A typical mode recognition study would stop at this point after the goal accuracy is achieved [10, 11, 20]. Even though we report comparable results to that of the current state-of-the-art accuracy, we are keen on evaluating how much the representation-cum-distance-measure attribute to musicological insights. Over and beyond the error analysis, we stress validating the gap and reinforcing other possible avenues, so the complexity of the musical characteristics is not attributed to the shortcomings of computational models. One such way is to disregard the makam labels and study the melodic similarity space of relevant predictor features. Through these methods, we aim to verify whether the machine learning models indeed 'learn' what they are intended for. We present three complementary and supplementary retrieval scenarios to corroborate the 'task' details and bridge the gap that the best classification could achieve.

Hierarchical clustering: In the presence of theoretical grouping of makams, yet not having a prescription on the counts, it is practically impossible to set a k for a k-means clustering algorithm. However, hierarchical clustering seems to offer a dynamic solution in such scenarios. Here, each element is treated as distinct clusters at the lowest threshold, whereas there is a single giant cluster at the highest threshold. We present, in part of Figure 3, the dendrogram representation to capture the melodic similarity/grouping space across the 20 makams obtained from the hierarchical clustering of the TDMS features averaged from the 50 recordings per makam using Canberra distance. We use the Canberra distance to contrast with the hierarchical clustering reported in [14, see Figure 2] that was calculated from averaged pitch distributions, and keep the empirical experimentation of different distance metrics out of the scope. The groupings are shown in different colours, and the relative height where the tree elements merge is indicative of the normalised threshold. At the highest threshold, makam Saba isolates itself from the rest 19; this is indicative of the distinct nature of the pitch distribution captured through TDMS. Makam-pair (Rast, Mahur) show a very low distance, indicating high similarity in the feature space. A high distance between the (Hüzzam, Hicaz) pair, as shown in other figures, is also evident.

Cluster purity matrix: One supplementary way to capture all pairwise distances is through an unconventional method of computing a distance matrix with the salience function of cluster purity. This is broadly a homogeneity measure that evidences the quality of clustering. Any value close to 1 indicates perfect clustering, while 0.5 signifies random clustering. Out of the $\binom{20}{2} = 190$ distinct makam-pairs, 115 pairs show a cluster purity score ≥ 0.85, 73 other pairs bear cluster purity values in the range of (0.5,

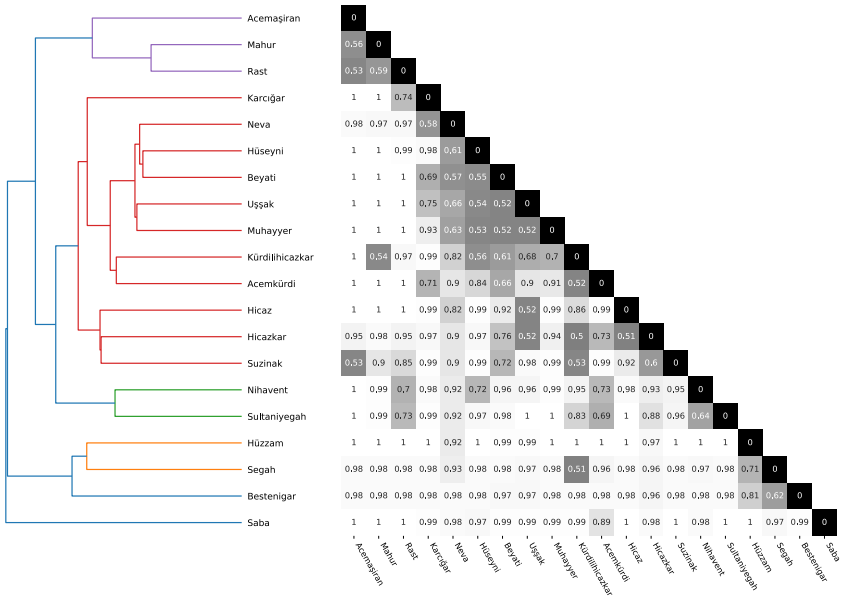


Figure 3. Left: dendrogram representation to capture the melodic similarity space across the 20 makams obtained from the hierarchical clustering of the TDMS. Right: the cluster purity matrix capturing the homogeneity of pairwise distances.

0.85), and none below a score of 0.5. We plot the pairwise purity scores partly in Figure 3, this representation has an intuitive inverse proportionality with the confusion matrix. The makam indices obtained from the dendrogram are aligned with that of the current matrix. It is intuitive to follow that the row corresponding to makam Saba (which is the most distinctive) has got the highest cluster purity (mode=1) with many other pairs, whereas the aforementioned confusable pairs yield a random clustering. This visualisation provides a complementary view of what is aggregated in the dendrogram.

Query retrieval score: The third scenario we present complementary to the clustering is the receiver-operated characteristics (ROC). We use the same two makam-pairs in the context of a query search for all possible matching and non-matching pairings out of each makams-pair. The ROCs in Figure 4 show the true positive rate versus the false positive rate achieved in the detection of non-matching makam pairs for the PCD (we consciously chose PCD over TDMS to introduce diversity) representations for four unique distance measures, inspired from [34]. The subplots correspond to makam-pairs (Hüzzam, Hicaz) and (Rast, Mahur), which are examples of highly discriminable and highly confused pairs, respectively. The ROC curves, the area under the curve (AUC) and equal error rate (EER) clearly indicate a better retrieval for the former, while the latter almost grazes the diagonal. We have inferences on why certain distance metric works better, but discussion on their relative performance is not directly related to the main narrative of this work and hence omitted [34, 39].

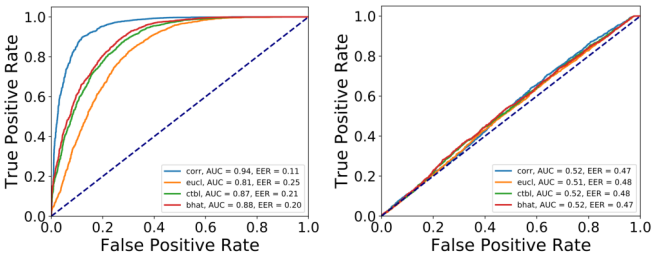


Figure 4. ROCs obtained using correlation (corr), euclidean (eucl), city-block (ctbl), and Bhattacharyya (bhat) distance from PCDs. Left: (Hüzzam, Hicaz) and right: (Rast, Mahur) makam-pairs.

5. CONCLUSION

We employed methodologies that combine domain knowledge and data-driven optimisations with a view to understanding the makam recognition 'task' in depth. We report comparable accuracy (77.2%) with the state-of-the-art [11] using the newly adapted TDMS feature with SVM. This achievement evidences the credential in our approach that provides us with a solid ground to argue the critique on goal- versus task-oriented approaches by comparing and contrasting. We have reported only the best-performing configuration in this paper [10] which will eventually be expanded to include temporal features and sequence models. In sum, we advocate that good supervised learning performance is a necessary but insufficient condition for a computational representation-cum-distance-measure to be considered informative for all purposes. As future work, we will incorporate convolutional networks and transformers to reproduce makam recognition on OTMM corpus to understand the potential of deep learning and the trade-off between goal and task involved. The application of such an approach aids in understanding music and other forms of sound culture and developing methodologies for cross-cultural mapping and comparing these materials.

[10] Array of alternate configurations: `https://sertansenturk.com/work-research/ismir-2022-makam/`

6. ACKNOWLEDGEMENTS

This research is part of project “Computationally engaged approaches to rhythm and musical heritage: Generation, analysis, and performance practice," funded through a grant from the Research Enhancement Fund at the New York University Abu Dhabi.

7. REFERENCES

[1] H. S. Powers and R. Widdess, *India, subcontinent of*, 2nd ed., ser. New Grove Dictionary of Music. Macmillan, London, 2001, ch. III: Theory and practice of classical music, contributions to S. Sadie (ed.).

[2] H. S. Powers, F. Wiering, J. Porter, J. Cowdery, R. Widdess, R. Davis, and A. Marett, “Mode. grove music online,” 2008.

[3] K. K. Ganguli, “How do we ’See’ & ’Say’ a raga: A Perspective Canvas,” *Samakalika Sangeetham*, vol. 4, no. 2, pp. 112–119, Oct. 2013.

[4] T. Çevikoğli, “Klasik Türk müziğinin bugünkü sorunları,” in *Proceedings of International Congress of Asian and North African Studies (ICANAS 38’)*, Ankara, 2007.

[5] B. Uyar, H. S. Atli, S. Şentürk, B. Bozkurt, and X. Serra, “A corpus for computational research of Turkish makam music,” in *Proceedings of the 1st International Workshop on Digital Libraries for Musicology (DLfM 2014)*, London, UK, 2014, pp. 1–7.

[6] B. Bozkurt, R. Ayangil, and A. Holzapfel, “Computational analysis of Turkish makam music: Review of state-of-the-art and challenges,” *Journal of New Music Research*, vol. 43, no. 1, pp. 3–23, 2014.

[7] K. K. Ganguli and P. Rao, “On the perception of raga motifs by trained musicians,” *The Journal of the Acoustical Society of America*, vol. 145, no. 4, pp. 2418–2434, 2019.

[8] ——, “A study of variability in raga motifs in performance contexts,” *Journal of New Music Research*, vol. 50, no. 1, pp. 102–116, 2021.

[9] S. T. Madhusudhan and G. Chowdhary, “DEEPSRGM-sequence classification and ranking in Indian classical music with deep learning,” in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR 2019)*, 2019, pp. 533–540.

[10] A. Karakurt, S. Şentürk, and X. Serra, “MORTY: A toolbox for mode recognition and tonic identification,” in *Proceedings of the 3rd International workshop on Digital Libraries for Musicology (DLfM 2016)*, New York, NY, 2016, pp. 9–16.

[11] E. Demirel, B. Bozkurt, and X. Serra, “Automatic makam recognition using chroma features,” in *Proceedings of the 8th International Workshop on Folk Music Analysis (FMA 2018)*. Thessaloniki, Greece: Aristotle University of Thessaloniki, 2018, pp. 19–24.

[12] G. K. Koduri, S. Gulati, P. Rao, and X. Serra, “Raga recognition based on pitch distribution methods,” *Journal of New Music Research*, vol. 41, no. 4, pp. 337–350, 2012.

[13] A. C. Gedik and B. Bozkurt, “Pitch-frequency histogram-based music information retrieval for Turkish music,” *Signal Processing*, vol. 90, no. 4, pp. 1049–1063, 2010.

[14] F. Yeşiler, B. Bozkurt, and X. Serra, “Makam recognition using extended pitch distribution features and multi-layer perceptrons,” in *Proceedings of the 15th Sound and Music Computing Conference (SMC 2018)*. Limassol, Cyprus: Cyprus University of Technology, 2018, pp. 249–253.

[15] P. Dighe, P. Agrawal, H. Karnick, S. Thota, and B. Raj, “Scale independent raga identification using chromagram patterns and swara based features,” in *Proceedings of IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, 2013, pp. 1–4.

[16] P. Dighe, H. Karnick, and B. Raj, “Swara histogram based structural analysis and identification of Indian classical ragas.” in *Proceedings of 14th International Society for Music Information Retrieval Conference (ISMIR 2013)*, 2013, pp. 35–40.

[17] S. Gulati, J. Serra, V. Ishwar, S. Şentürk, and X. Serra, “Phrase-based rāga recognition using vector space modeling,” in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2016)*. IEEE, 2016, pp. 66–70.

[18] P. Chordia and A. Rae, “Raag recognition using pitch-class and pitch-class dyad distributions.” in *Proceedings of 8th International Society for Music Information Retrieval Conference (ISMIR 2007)*, 2007, pp. 431–436.

[19] P. Chordia and S. Şentürk, “Joint recognition of raag and tonic in north Indian music,” *Computer Music Journal*, vol. 37, no. 3, pp. 82–98, 2013.

[20] S. Gulati, J. Serrà Julià, K. K. Ganguli, S. Sentürk, and X. Serra, “Time-delayed melody surfaces for rāga recognition,” in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, New York, NY, 2016, pp. 751–757.

[21] S. Abdoli, “Iranian traditional music dastgah classification.” in *Proceedings of 12th International Society for Music Information Retrieval Conference (ISMIR 2011)*, 2011, pp. 275–280.

[22] P. Heydarian and D. Bainbridge, “Dastgàh recognition in Iranian music: Different features and optimized parameters,” in *Proceedings of 6th International Conference on Digital Libraries for Musicology (DLfM 2019)*, New York, NY, USA, 2019, p. 53–57.

[23] L. Cimarone, B. Bozkurt, and X. Serra, "Automatic dastgah recognition using Markov models," in *Proceedings of 14th International Symposium on Computer Music Multidisciplinary Research (CMMR 2019)*, Marseille, France, 10 2019, pp. 51–58.

[24] D. Huron and J. Veltman, "A cognitive approach to medieval mode: Evidence for an historical antecedent to the major/minor system," *Empirical Musicology Review*, vol. 1, no. 1, pp. 33–55, 2006.

[25] B. Cornelissen, W. Zuidema, and J. A. Burgoyne, "Mode classification and natural units in plainchant," in *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR 2020)*, 2020, pp. 869–875.

[26] K. K. Ganguli, S. Gulati, X. Serra, and P. Rao, "Data-driven exploration of melodic structures in Hindustani music," in *Proc. of the International Society for Music Information Retrieval (ISMIR)*, Aug. 2016, pp. 605–611, New York, USA.

[27] G. K. Koduri, V. Ishwar, J. Serrà, and X. Serra, "Intonation analysis of rāgas in Carnatic music," *Journal of New Music Research*, vol. 43, no. 1, pp. 72–93, 2014.

[28] K. K. Ganguli, A. Rastogi, V. Pandit, P. Kantan, and P. Rao, "Efficient melodic query based audio search for Hindustani vocal compositions," in *Proc. of Int. Soc. for Music Information Retrieval (ISMIR)*, Oct. 2015.

[29] K. K. Ganguli, A. Lele, S. Pinjani, P. Rao, A. Srinivasamurthy, and S. Gulati, "Melodic shape stylization for robust and efficient motif detection in hindustani vocal music," in *National Conference on Communications (NCC)*, 2017.

[30] K. K. Ganguli and P. Rao, "Discrimination of melodic patterns in Indian classical music," in *Proc. of National Conference on Communications (NCC)*, Feb. 2015.

[31] S. Şentürk, G. K. Koduri, and X. Serra, "A score-informed computational description of svaras using a statistical model," in *Proceedings of 13th Sound and Music Computing Conference (SMC 2016)*, Hamburg, Germany, 2016, pp. 427–433.

[32] J. C. Ross, A. Mishra, K. K. Ganguli, P. Bhattacharyya, and P. Rao, "Identifying raga similarity through embeddings learned from compositions' notation." in *Proc. of the International Society for Music Information Retrieval (ISMIR)*, 2017.

[33] S. Gulati, J. Serra, K. K. Ganguli, and X. Serra, "Landmark detection in Hindustani music melodies," in *Proc. of Int. Computer Music, Sound and Music Computing*, 2014, pp. 1062–1068, Athens, Greece.

[34] K. K. Ganguli and P. Rao, "On the distributional representation of ragas: experiments with allied raga pairs," *Transactions of the International Society for Music Information Retrieval*, vol. 1, no. 1, 2018.

[35] ——, "Towards computational modeling of the ungrammatical in a raga performance." in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China, 2017, pp. 39–45.

[36] S. Şentürk, "Computational analysis of audio recordings and music scores for the description and discovery of Ottoman-Turkish makam music," Ph.D. dissertation, Universitat Pompeu Fabra, Barcelona, Spain, December 2016.

[37] H. S. Atlı, B. Uyar, S. Şentürk, B. Bozkurt, and X. Serra, "Audio feature extraction for exploring Turkish makam music," in *Proceedings of 3rd International Conference on Audio Technologies for Music and Media (ATMM 2014)*, Ankara, Turkey, 2014.

[38] C. L. Krumhansl, *Cognitive Foundations of Musical Pitch*. Oxford University Press, New York, 1990, ch. 4: A key-finding algorithm based on tonal hierarchies, pp. 77 – 110.

[39] K. K. Ganguli, "A corpus-based approach to computational modeling of melody in raga music," Ph.D. dissertation, Indian Institute of Technology Bombay, Mumbai, India, 2019.

A DATASET FOR GREEK TRADITIONAL AND FOLK MUSIC: Lyra

Charilaos Papaioannou[1], **Ioannis Valiantzas**[2], **Theodoros Giannakopoulos**[3], **Maximos Kaliakatsos-Papakostas**[4], **Alexandros Potamianos**[1]
[1] School of ECE, National Technical University of Athens, Greece
[2] Department of Music Studies, National and Kapodistrian University Of Athens, Greece
[3] National Center for Scientific Research - Demokritos, Greece
[4] Athena RC, Greece
cpapaioan@mail.ntua.gr

ABSTRACT

Studying under-represented music traditions under the MIR scope is crucial, not only for developing novel analysis tools, but also for unveiling musical functions that might prove useful in studying world musics. This paper presents a dataset for Greek Traditional and Folk music that includes 1570 pieces, summing in around 80 hours of data. The dataset incorporates YouTube timestamped links for retrieving audio and video, along with rich metadata information with regards to instrumentation, geography and genre, among others. The content has been collected from a Greek documentary series that is available online, where academics present music traditions of Greece with live music and dance performance during the show, along with discussions about social, cultural and musicological aspects of the presented music. Therefore, this procedure has resulted in a significant wealth of descriptions regarding a variety of aspects, such as musical genre, places of origin and musical instruments. In addition, the audio recordings were performed under strict production-level specifications, in terms of recording equipment, leading to very clean and homogeneous audio content. In this work, apart from presenting the dataset in detail, we propose a baseline deep-learning classification approach to recognize the involved musicological attributes. The dataset, the baseline classification methods and the models are provided in public repositories. Future directions for further refining the dataset are also discussed.

1. INTRODUCTION

Traditional music of Greece is under-represented in available datasets, despite the fact that this music offers unique perspectives that combine characteristics of Western and Eastern music. The development of a traditional Greek music dataset is interesting to study in its own right, while it is also expected to provide a more complete picture of the music in the Mediterranean and Europe. Computational methods have been extensively studied in the past decades for carrying out ethnomusicological studies, constituting the field of Computational Ethnomusicology. A review of such methods is presented in [1], where it is evident that there are many benefits in compiling datasets of traditional music that can readily be used in computational models.

Several such datasets have been developed. For instance, an enormous database of Dutch melodies and songs that allows studying multiple musicological aspects is presented in [2]. Similarly, a dataset of Indian art music was presented in [3], where various MIR-related tasks were adjusted and applied therein. The employment of standard MIR tools has been evaluated in Arab-Andalusian music [4] and Flamenco music with the COFLA dataset [5]. Iranian Dastgah classification has been studied with MIR tools in [6]. A dataset of Georgian vocal music from historic tape recordings of three-voice songs was presented in [7], where the aim was to preserve cultural heritage and also to apply and examine existing MIR methods (e.g., for pitch and onset detection) in data of this form. Such datasets also provide opportunities for studying idiosyncratic musical instruments, e.g., in the case of Chinese music [8], where many instruments are employed that are not related to the ones used in Western music.

In several cases, off-the-shelf MIR tools are not well-suited for tasks that include traditional musical types. For instance, studying melodic similarity of Turkish non equally-tempered makam phrases in symbolic format required the development of a novel representation based on MIDI [9]. Similarly, rhythmic attributes of makam music with new models that integrate note transition rules for this music had to be developed for lyrics-to-audio alignment [10]. In addition, a novel method, based on audio signal processing, was created for identifying asymmetric rhythms in Greek traditional music [11].

The "Lyra" dataset of traditional and folk Greek music aims to contribute to all the aforementioned fields, from data-driven (computational) ethnomusicology to shaping new directions of research for MIR tools. The majority of available datasets that concern Greek music are unsuitable

© C. Papaioannou, I. Valiantzas, T. Giannakopoulos, M. Kaliakatsos-Papakostas and A. Potamianos. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** C. Papaioannou, I. Valiantzas, T. Giannakopoulos, M. Kaliakatsos-Papakostas and A. Potamianos, "A Dataset for Greek Traditional and Folk Music: Lyra", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

for computational analysis due to their unstructured nature. An exception is the Greek Audio Dataset (GAD) [12], which includes 1000 pieces with audio content (YouTube links), lyrics and annotations for genre (coarse categories including traditional, pop, rock), mood (valence-arousal plane coordinates) and pre-computed audio features; this dataset was later expanded in the Greek Music Dataset (GMD) [13], consisting of 1400 audio pieces with the aforementioned data. However both the GMD and the GAD are not focused on traditional and folk music, while the quality of audio recordings is varying significantly between genres.

The Lyra dataset presented in this paper, in contrast to the GMD, includes mainly traditional and folk Greek music with fine-grained labeling, focusing on musicological aspects of interest. Additionally, the recording quality is homogeneous across all pieces. Musicological soundness and high quality content are ensured by the fact that data has been collected and annotated from a documentary series that was presented by academics in Greek television. Some baseline classification tasks that are of particular interest in this dataset are presented, namely classification of pieces in genres, instruments and places of origin. In all cases, the results show that computational analysis can provide useful insight about musicological relations and phenomena in traditional and folk music of Greece – and, possibly, of other places.

2. DATASET EXTRACTION AND DESCRIPTION

2.1 Challenges and Methods

Large amounts of clean data is fundamental for current AI models to achieve their full potential. In this paper, we walked all the way, from the "data in-the-wild" multimedia content of a TV show to a fully annotated dataset, through a combination of machine automation and human evaluation/annotation processes. The consistency of the dataset and the richness of information it provides are tested by developing and training models that perform three different classification tasks.

In the case of Greek traditional and folk music, there are few cases where metadata is combined with recordings in a structured manner. Additionally, there is a matter of quality of recordings as it is significantly affected by various factors, including the equipment used, the social occasion (e.g., during a festival or inside a studio) and the time period in which it took place, i.e., older recordings tend to be of lower quality.

An integration of dissimilar recordings, in terms of quality, can introduce significant deficiencies towards studying the musicological characteristics of world music with computational tools. In order to truncate the effect of the audio quality factor, we decided to incorporate the episodes from the Greek documentary series "To Alati tis Gis - Salt of the Earth" broadcasted by ERT (Hellenic Broadcasting Corporation), where primarily traditional and folk music is presented. The episodes were filmed during a 10 year period under strict production-level specifications, resulting to very clean and homogeneous audio content while significant wealth of information is provided by the presenter and the guests in the form of narrations between music performances.

The presented dataset consists of both the multimedia content and the annotations of interest. The multimedia content is provided as start and end timestamps that correspond to a single music piece, as parts of a longer episode, which is available online. Regarding the annotations, a taxonomy of labels is defined, based on the potential purposes of studies that might involve this dataset, considering also what metadata information can be retrieved either directly from the source or be integrated by volunteer annotators during the data collection process.

The study of Greek traditional and folk music involves knowledge about (i) the instrumentation, (ii) the genres, (iii) the places of origin and (iv) the way listeners perceive this music in terms of "danceability", among others. While musical instruments, genres and geography are semantically well-defined, the same can not be claimed for listeners' perception. Having at hand the multimedia content, i.e., audio and video, can be helpful to this end. Annotation about whether a music piece is being danced during its live performance can reveal cultural characteristics regarding the way this piece is perceived by the community, because body movements play an important role in music perception [14].

As a result, the taxonomy consists of (i) the musical instruments participating in the performance of each music piece (singing voice is considered an instrument), (ii) the musical genres and sub-genres that are identified by musicologists in Greek music, (iii) the places of origin and (iv) whether the music piece is being danced during its performance.

Volunteer annotators, students of the Department of Music Studies, undertook the task of separating each episode in music pieces and also labeling each one of them according to the specified taxonomy. A helper website was utilized where the respective category labels were added. An account was created for each annotator for the label assignment task. Every piece was labeled by two annotators and the final labels are the set of them where both annotators agree. At the end, the dataset that contains the aforementioned annotations along with the timestamps and the respective video id for each music piece was extracted from the database of the helper site.

2.2 Dataset description

Lyra dataset is organized into a single table where each row corresponds to a music piece while the columns include the various metadata information. Table 1 demonstrates the metadata categories.

Beginning with the simplest metadata categories, in terms of description, "id" is a unique identifier for each piece, generated by its title, replacing Greek with Latin characters and spaces with dashes. As expected, the number of unique values will be the same with the number of pieces, namely 1570. The same stands for "start-ts" and

name	# unique	multi-label	description
id	1570	No	unique identifier of each music piece
instruments	32	Yes	instruments participating in performance
genres	32	Yes	music style annotations
place	81	Yes (to contain regions)	full hierarchy of the place of origin
coordinates	81	No	latitude and longitude
is-danced	2	No	binary value (0 or 1)
youtube-id	74	No	id of the episode available online
start-ts	1570	No	start timestamp of the piece
end-ts	1570	No	end timestamp of the piece

Table 1. Metadata contained in the dataset.

"end-ts" that denote the exact time (second) that a song starts and ends in the corresponding video. The duration of the music pieces sums to approximately 80 hours.

The column "youtube-id" contains the id under which the video of the full episode is available online. The count of unique values are essentially the number of episodes that were used for the creation of the dataset. A typical duration of an episode is roughly a hundred minutes. The "is-danced" binary label informs about whether a music piece is being danced by the guests of the show. The music pieces annotated with "1" are approximately 51% while in the rest of them no dance performance occurs.

The classification of Greek music in "genres" is a work that requires one to take into account a number of socio-cultural and anthropo-geographical criteria. At an abstract level, we can distinguish the music of urban centers in contrast to the music of rural areas of Greece, with the former including rebetiko, laiko, urban-folk among others, while the latter, the music of rural areas, is what we generally call traditional music. Figure 1 shows the frequencies of the genres in the dataset, with "traditional" being the dominant one constituting almost 78% of the total. Depending on the place of origin, the style of a traditional music piece varies accordingly and, thus, several sub-genres flourish, such as Epirotic for the songs originated from Epirus. The 32 unique values in this metadata category are separated into 5 distinct genres and 27 sub-genres.

From a musicological perspective, Greek traditional music can be divided into two large geographic areas, i.e., the island and the mainland Greece. Each one creates a

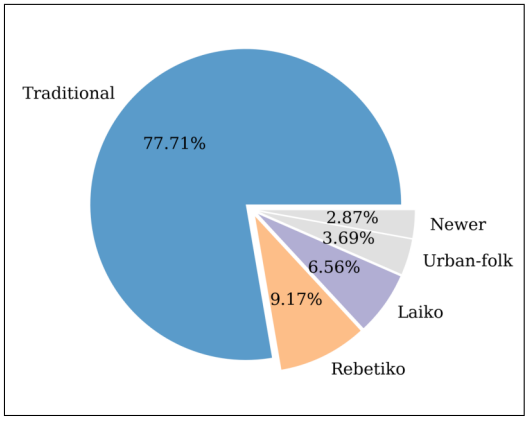


Figure 1. Relative frequencies of the music genres in the dataset.

distinctive musical feeling as there are large variations both on the rhythmic approach and the scales that are commonly used. For example, in islands we frequently come across music pieces with simple, fast rhythms while in the mainland more complex, slow rhythms are the norm.

The "place" (of origin) metadata category can be annotated with (i) a single label when the region from which a song derives is known, (ii) two labels when both region and a specific place are known and (iii) "None" denoting that there is not a specific place of origin for this piece. As an example, a music piece can be annotated with the region "Aegean sea" or with both "Aegean sea" and "Naxos", an island of the Aegean sea, if this knowledge is available. Specifically, from the 81 unique places in the dataset, 20 are regions and only half of them include the remaining 61. The most represented regions can be seen in Figure 2.

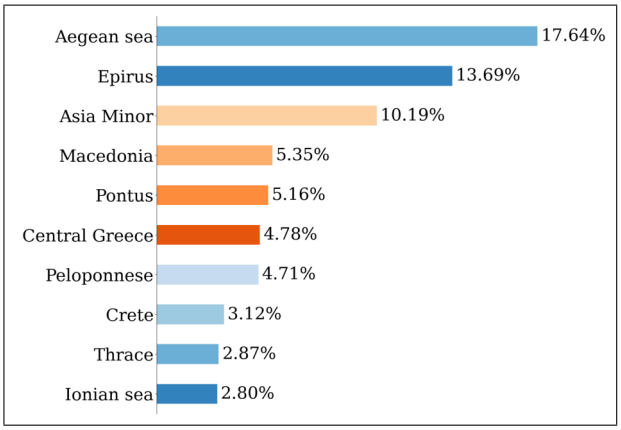


Figure 2. Relative frequencies of the most represented regions in the dataset.

The exact latitude and longitude of each place is also available at the "coordinates" column. The music pieces that do not have an explicit place of origin, such as the ones that belong to the "laiko" genre, are accounted for approximately 23% of the total. Figure 3 shows the location of the 81 places that exist in the dataset. We may notice the constant ability of music to excess the borders; places where Greek culture thrived in the past and neighboring countries

that share the same tradition, form a mosaic of people that communicated freely with each other in a musical way that has reached towards us.

Figure 3. Map of all the places of the dataset.

Analogous connections, like the ones between genres and places, one expects to be observed between places and instruments as well. Indeed, for over 100 years, the established music ensembles of Greek traditional music are generally two, namely (i) those with the violin as leading instrument (often substituted by lyra and santouri), which have a greater presence in island Greece and (ii) those with the klarino (Greek clarinet) as the leading instrument, which is dominant in the mainland.

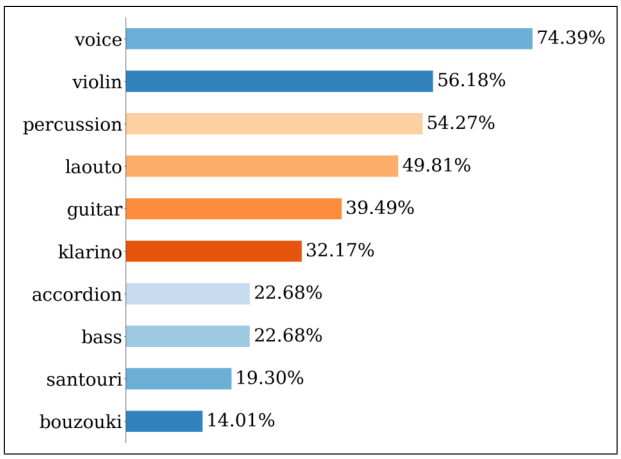


Figure 4. Relative frequencies of the most common musical instruments in the dataset.

In the popular and modern music domain, there is a great variety of instruments, but in most cases bouzouki, guitar, accordion and bass are common members of a laiko or rebetiko music ensemble. In the traditional music groups, the percussion and the laouto (Greek lute) are permanent companions of the leading instruments, offering melodic-harmonic background and rhythmic support. Of course, voice holds the main role in all kinds of performances.

In Figure 4 one can see the frequencies of the most popular instruments in the dataset. Singing voice is evident in almost 75% of it and instruments like violin, percussion and laouto, that have presence in both islands and mainland as well, are following.

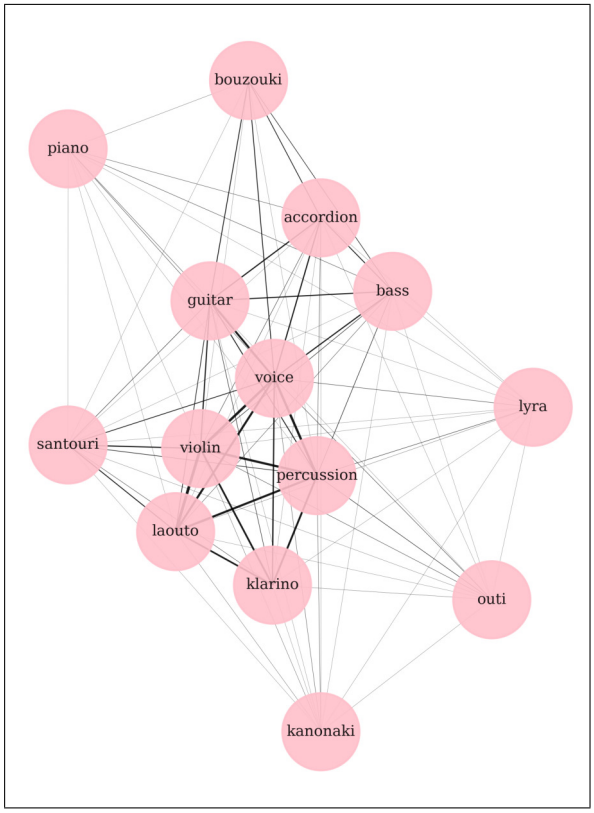


Figure 5. Graph that shows the co-occurrences of the fourteen most common musical instruments. The edge width is proportional to the number of music pieces a pair of instruments co-occur.

With regards to the music ensembles, it should be noted that 296 unique groups of instruments exist in the dataset, with the one constituted by voice, violin, percussion, laouto and klarino being by far the most popular by participating in the performance of around 12% of the music pieces. The co-occurrences of the most popular instruments can be seen in Figure 5 where the width of the graph edges is proportional to the number of pieces a pair of instruments co-occur in the dataset.

Sample rows of the dataset can be seen in Table 2. The dataset along with the baseline classification methods and the trained models are available online. [1]

The shared metadata should be considered as the version 1.0 of the dataset. In the next versions, it will be evolved towards two main directions, namely (i) the incorporation of more metadata categories such as annotations according to the content of the lyrics, the lyrics themselves as well as information about the types of the dances that occur and (ii) the addition of more music pieces by following the same process either for next episodes of the same documentary series or for other series that have a similar theme.

3. BASELINE CLASSIFICATION

The audio recording of each music piece is represented using a Mel-scaled Spectrogram (mel-spectrogram): this is

[1] https://github.com/pxaris/lyra-dataset

id	instruments	genres	place	coordinates	is-danced	youtube-id	start-ts	end-ts
alexandra	voice|violin| percussion| laouto| klarino	Traditional| Epirotic	Epirus| Zagori	39.8648| 20.9284	0	qrOwc1mLFUk	749	927
choros-tik	percussion| lyra	Traditional| Pontian	Pontus	40.9883| 39.7270	1	Aws0Y3aLaIs	1731	1886
agiothodo-ritissa	voice|violin| santouri| percussion| laouto|guitar	Rebetiko	None	None	1	0cj8BNcAhg4	2632	2853
einai-arga-poly-arga	voice|piano| guitar|bass| bouzouki| accordion	Laiko	None	None	0	zkoqg3VRVLA	2365	2614

Table 2. Sample rows of the dataset. Pipe "|" is used to separate values in a field.

a spectrogram whose frequencies are converted to the mel scale according to the equation:

$$m = 2595 \log_{10}(1 + \frac{f}{700}) = 1127 \ln(1 + \frac{f}{700}) \quad (1)$$

where **m** is the frequency in Mels and **f** is the frequency in Hz. Mel-spectrograms are calculated per fix-sized segment duration of 10 seconds. A non-overlapping window of 50 milliseconds has been applied, therefore the Mel-spectrogram size is 200 windows $\times$ 128 frequency bins.

As a baseline classification approach, each 10-second Mel-spectrogram is classified to the aforementioned tasks (genre, place and instruments) using a Convolutional Neural Network (CNN). CNNs have been widely used in general audio [15], speech [16, 17] and music [18] classification tasks. In particular, we have adopted the following architecture: 4 convolutional layers of 5 $\times$ 5 kernels, single stride, and max pooling of size 2. The number of convolutional kernels (channels) are for the first layer 32, for the second 64, for the third 128 and for the fourth 256. The final output of the convolutional layers is passed through 3 fully connected layers, with the first having an output dimension of 1024, the second 256 and the third equal to the number of classes.

Note that fix-sized duration of segments is necessary, since audio recordings do not share the same size. Adopting a much longer segment would require zero padding for several mel-spectrograms and probably more CNN parameters. In addition, splitting the song into non-overlapping segments achieves some type of data augmentation. For two of the adopted classification tasks (namely genre and place), we have trained the CNNs using a multiclass, single-label setup, while for the instrument task, which is multi-label, we have trained multiple binary CNNs, one for each instrument, which have been evaluated separately.

After the training and validation procedure of each of the aforementioned CNNs, final testing was applied on the respective test recordings. For the test set, to avoid spreading pieces from the same broadcast across data splits, we separate training and test data on an episode level. From the 74 unique episodes, we randomly split 20% of them and use all the music pieces they include, namely 330, to form the test data. Obviously, this final testing needs to be carried out on a "song-level", not a 10-sec segment level. Towards this end, a simple aggregation method was adopted, by just averaging the posteriors of the individual segment decisions of the CNNs. This aggregated estimate was used as the final prediction and evaluated in the final testing.

4. RESULTS

The performance results during the training of the baseline classifiers are shown in Table 3, computed on a validation subset that corresponds to 20% of the segments. For the multi-label task (instrument recognition), we show the F1 metric for each binary subtask separately, while for the single-label tasks (genre and place) we show the overall macro F1 for all classes. We remind that this evaluation is performed on the validation split of the 10-second data.

Classifiers type	Task	F1 (%)
Multiclass classifiers	Genre	82.3
	Place	85.5
Binary classifiers for Instruments	voice	75.8
	violin	86.3
	percussion	97.1
	laouto	96.7
	guitar	80.2
	klarino	95.0

Table 3. F1 scores of the classifiers on the 10-second segments of the training data using a 20% validation subset.

As soon as the 10-second classifiers are trained, they are applied on the whole recordings of the testing data, and a simple majority aggregation is performed to extract the final decision, as described in the previous Section. For the instrument classification task, we compute the Area Under the Curve (AUC) metric per label (binary classification subtask). The results are shown in Table 4.

Finally, the confusion matrices along with the respective F1 measures for the multiclass, single-label classification tasks of "genres" and "places" are shown in Figures 6 and 7. All genres have been taken into account for the respective classification, but not the sub-genres. On "places" task, the pieces have been classified to the 10 most common regions (including "None") plus the "other" category for the remaining. Genres classifier macro F1-score

Instrument	AUC (%)
voice	68.9
violin	85.2
percussion	95.1
laouto	93.8
guitar	73.5
klarino	90.9

Table 4. Area Under the Curve (AUC) scores of the instrument classifiers on the test data.

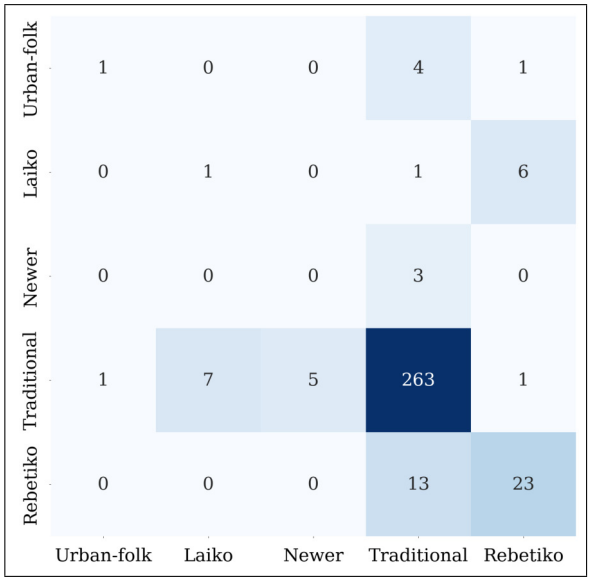


Figure 6. Confusion matrix for "genre" classes on the test data. **Macro F1-**score: **39.9**% and **Micro F1-**score: **87.2**%.

is 39.9% and micro F1-score is 87.2%, while for the places classifier the macro F1-score is 34.4% and the micro F1-score is 42.4%.

5. DISCUSSION

A reason that the "voice" classifier has lower performance compared to the other ones may be of a musicological character. Indeed, while the presence of the rest of the instruments can depend significantly on the music style of a piece, the same does not apply to "voice" as it is the dominant musical instrument in any genre. Given the fact that the binary classifier is trained to recognize an instrument (evident in a part of a music piece) in each of the 10-second segments, regardless if it is present on it or not, we expect to move towards a space with latent musical features such as the music style, where "voice" may not be as discriminative as the rest of the instruments are.

With regards to confusion matrices, the misclassifications can be either due to imbalance between classes or statistical correlations across them. Specifically, for the "places" task, the confusions between regions that are geographically near may be justifiable, while for "genres" task the imbalance between the classes seems to have significantly affected the performance of the model at the least represented ones.

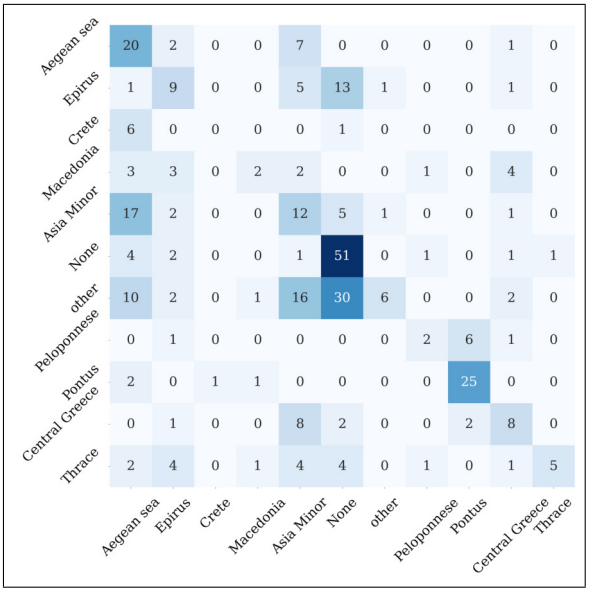


Figure 7. Confusion matrix for "place" classes on the test data. **Macro F1-**score: **34.4**% and **Micro F1-**score: **42.4**%.

We intend to improve models performance and conduct a more in-detail analysis in order to examine hypotheses such as the aforementioned ones in a follow-up study.

6. CONCLUSIONS

Greek traditional and folk music integrates components of Eastern and Western idioms, providing interesting research directions in the field of computational ethnomusicology. This paper presents "Lyra", a dataset of 1570 traditional and folk Greek music pieces that includes audio and video (timestamps and links to YouTube videos), along with annotations that describe aspects of particular interest for this dataset, including instrumentation, geographic information and labels of genre and subgenre, among others. The advantage of this dataset is that the entire content is harvested from web resources of a Greek documentary series that was produced by academics with specialization in this music and, therefore, includes high-quality and rich annotations extracted from the content of the shows. Additionally, the production of recordings and video material is professional-level, providing a common ground in terms of audio quality. Three baseline audio-based classification tasks are performed, namely instrument identification, place of origin and genre classification.

The presented results indicate that specialized tasks, that use the audio signal, can potentially provide valuable insight about several aspects of this music. The combination of video and audio signals allows possible experimentation on methods that process multimodal data. The Lyra dataset includes material that readily allows MIR tools to be employed for reaching valuable musicological results. This dataset can also, potentially, foster the expansion of the MIR methods altogether.

7. ACKNOWLEDGEMENTS

The authors would like to thank Professor Lambros Liavas, the producer, researcher and presenter of the documentary series "To Alati tis Gis - Salt of the Earth". His work not only provides a fruitful source of knowledge and art but also acts as an inspiration for new researchers. This work would not have been possible without the help of the volunteer annotators. A big thank to the students of the Department of Music Studies of National and Kapodistrian University of Athens. Finally, we would like to thank the reviewers for their valuable and constructive comments.

8. REFERENCES

[1] M. Panteli, E. Benetos, and S. Dixon, "A review of manual and computational approaches for the study of world music corpora," *Journal of New Music Research*, vol. 47, no. 2, pp. 176–189, 2018.

[2] P. Van Kranenburg, M. De Bruin, and A. Volk, "Documenting a song culture: The dutch song database as a resource for musicological research," *International Journal on Digital Libraries*, vol. 20, no. 1, pp. 13–23, 2019.

[3] A. Srinivasamurthy, G. K. Koduri, S. Gulati, V. Ishwar, and X. Serra, "Corpora for music information research in indian art music," in *Georgaki A, Kouroupetroglou G, eds. Proceedings of the 2014 International Computer Music Conference, ICMC/SMC; 2014 Sept 14-20; Athens, Greece*. Michigan Publishing, 2014.

[4] R. C. Repetto, N. Pretto, A. Chaachoo, B. Bozkurt, and X. Serra, "An open corpus for the computational research of arab-andalusian music," in *Proceedings of the 5th International Conference on Digital Libraries for Musicology*, 2018, pp. 78–86.

[5] N. Kroher, J.-M. Díaz-Báñez, J. Mora, and E. Gómez, "Corpus cofla: A research corpus for the computational study of flamenco music," *Journal on Computing and Cultural Heritage (JOCCH)*, vol. 9, no. 2, pp. 1–21, 2016.

[6] S. Abdoli, "Iranian traditional music dastgah classification." in *Proceedings of the 12th International Society for Music Information Retrieval Conference*, 2011, pp. 275–280.

[7] S. Rosenzweig, F. Scherbaum, D. Shugliashvili, V. Arifi-Müller, and M. Müller, "Erkomaishvili dataset: A curated corpus of traditional georgian vocal music for computational musicology," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.

[8] X. Gong, Y. Zhu, H. Zhu, and H. Wei, "Chmusic: A traditional chinese music dataset for evaluation of instrument recognition," in *2021 4th International Conference on Big Data Technologies*, 2021, pp. 184–189.

[9] M. K. Karaosmanoğlu, B. Bozkurt, A. Holzapfel, and N. D. Dişiaçık, "A symbolic dataset of turkish makam music phrases," in *Proceedings of Fourth International Workshop on Folk Music Analysis (FMA2014)*, 2014, pp. 10–14.

[10] G. B. Dzhambazov, A. Srinivasamurthy, S. Sentürk, and X. Serra, "On the use of note onsets for improved lyrics-to-audio alignment in turkish makam music," in *Devaney J, Mandel MI, Turnbull D, Tzanetakis G, editors. Proceedings of the 17th International Society for Music Information Retrieval Conference*, 2016.

[11] T. Fouloulis, A. Pikrakis, and E. Cambouropoulos, "Traditional asymmetric rhythms: A refined model of meter induction based on asymmetric meter templates," in *Proceedings of the third international workshop on folk music analysis*, 2013, pp. 28–32.

[12] D. Makris, K. L. Kermanidis, and I. Karydis, "The greek audio dataset," in *IFIP International Conference on Artificial Intelligence Applications and Innovations*. Springer, 2014, pp. 165–173.

[13] D. Makris, I. Karydis, and S. Sioutas, "The greek music dataset," in *Proceedings of the 16th International Conference on Engineering Applications of Neural Networks (INNS)*, 2015, pp. 1–7.

[14] M. Leman and P.-J. Maes, "The role of embodiment in the perception of music," *Empirical Musicology Review*, vol. 9, no. 3-4, pp. 236–246, 2015.

[15] M. Papakostas and T. Giannakopoulos, "Speech-music discrimination using deep visual feature extractors," *Expert Systems with Applications*, vol. 114, pp. 334–344, 2018.

[16] Z. Huang, M. Dong, Q. Mao, and Y. Zhan, "Speech emotion recognition using cnn," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 801–804.

[17] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Convolutional recurrent neural networks for music classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2392–2396.

[18] M. Ashraf, G. Geng, X. Wang, F. Ahmad, and F. Abid, "A globally regularized joint neural architecture for music classification," *IEEE Access*, vol. 8, pp. 220 980–220 989, 2020.

ANALYSIS AND DETECTION OF SINGING TECHNIQUES IN REPERTOIRES OF J-POP SOLO SINGERS

Yuya Yamamoto[1] **Juhan Nam**[2] **Hiroko Terasawa**[1]
[1] Doctoral Program in Informatics, University of Tsukuba, Japan
[2] Graduate School of Culture Technology, KAIST, South Korea
s2130507@s.tsukuba.ac.jp, juhan.nam@kaist.ac.kr, terasawa@slis.tsukuba.ac.jp

ABSTRACT

In this paper, we focus on singing techniques within the scope of music information retrieval research. We investigate how singers use singing techniques using real-world recordings of famous solo singers in Japanese popular music songs (J-POP). First, we built a new dataset of singing techniques. The dataset consists of 168 commercial J-POP songs, and each song is annotated using various singing techniques with timestamps and vocal pitch contours. We also present descriptive statistics of singing techniques on the dataset to clarify what and how often singing techniques appear. We further explored the difficulty of the automatic detection of singing techniques using previously proposed machine learning techniques. In the detection, we also investigate the effectiveness of auxiliary information (i.e., pitch and distribution of label duration), not only providing the baseline. The best result achieves 40.4% at macro-average F-measure on nine-way multi-class detection. We provide the annotation of the dataset and its detail on the appendix website [0] .

1. INTRODUCTION

A singing voice is one of the most essential elements of music. It provides impactful emotional expressions through melody and lyrics. In particular, in popular music, the role of the singing voice is more important, as the vocal quality and unique style of singers are critical in attracting people. Vocals mainly consist of singer individuality (i.e., vocal fold vibration and vocal tract resonance) and singing expressions (i.e., fine control of pitch, timbre, and loudness). Singing techniques, as the name suggests, are extended techniques in human singing. It characterizes singing voices as a component of expressions in vocal performances. In particular, many professional singers in popular music use singing techniques to make a performance more expressive, each in a different manner.

[0] https://yamathcy.github.io/ISMIR2022J-POP/

© Yuya Yamamoto, Juhan Nam, and Hiroko Terasawa. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Yuya Yamamoto, Juhan Nam, and Hiroko Terasawa, "Analysis and detection of singing techniques in repertoires of J-POP solo singers", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

The singing voice is an important subject in the field of music information retrieval (MIR), and there are many prior works that extract quantitative or objective information as in the task of singing transcription [1], lyric identification [2], and singer identification [2]. Computational analysis of singing techniques based on MIR can contribute to the clarification of singing styles and can be applied to music discovery, vocal training, consumer-generated media and so on. Singing techniques have been of keen interest, especially in J-POP, both among singers and music creators; the singing style of J-POP singers has a wide variety, and their singing techniques are diverse. In addition, singing techniques are one of the evaluation criteria in Japanese commercial karaoke systems with scoring systems. Meanwhile, as for music creators, *VOCALO* songs, whose singer's voice are created by a singing voice synthesis software such as VOCALOID [3], have been established as a music genre in Japan. Many VOCALOID creators manually manipulate the voice of VOCALOID to make it more expressive, sometimes while referring to how the actual singer produce the singing voice, like how singing techniques are applied. Therefore, J-POP is an attractive research target for singing technique analysis and computational singing technique analysis may bring benefits to real-world applications.

In this regard, we present a case study of singing technique analysis in J-POP. We first introduce a new dataset, including the annotation of singing techniques for J-POP recordings. We analyze the dataset using descriptive statistics to illustrate the technique distributions and singer styles. Finally, we conduct singing technique detection based on machine learning methods, which can be beneficial to analyze the one side of the singing voice automatically. We demonstrate the detection of various singing techniques (i.e., identifying the singing technique type with its starting time and duration) and report baseline results using existing methods. Finally, we investigate the effect of the identification model with auxiliary information derived from the statistics of the dataset and pitch information.

2. RELATED WORKS

2.1 Singing Technique Analysis

Analyzing the acoustic patterns of singing expressions has been a long-standing research topic. Research on vibrato

dates to the 1930s [4]. Later, computer-based methods allowed a more quantitative analysis of vibrato parameters [5,6]. In addition to vibrato, other singing techniques, such as growling [7], portamento [8], voice registers [9,10], and extreme vocal effects [11] have also been investigated. An analysis of singing techniques was also conducted for J-POP. Migita et al. investigated the vibrato parameters of the imitated singing voices of J-POP vocalists [12]. Yamamoto et al. conducted an exploratory analysis of singing techniques that imitate the singing style of J-POP singers [13]. They found 13 types of singing techniques from 48 tracks and analyzed them with annotations. Because singing techniques in J-POP cover wide repertoires [14], such exploratory analysis is also important for the purpose of singing performance analysis. As our interests are handling and detecting multiple singing techniques rather than specifying a single technique, we annotated the label as in the manner in [13] but on the original real-world tracks of J-POP. Therefore, our new dataset consists of labels for multiple singing techniques.

2.2 Singing Technique Datasets

There are a handful of datasets that focus on the analysis of singing techniques. The Phonation Mode dataset consisted of four vocal modes (neutral, pressed, breathy, and flow) of sustained sung vowels from four singers [15]. Although the dataset includes a wide range of pitches, they are discrete, and thus lack a melodic context. VocalSet [16] handles the issue by having voices sung in contexts of scales, arpeggios, long tones, and excerpts. In addition, it covers a broader range of singing techniques, such as vibrato, trill, vocal fry, and inhaled singing. Because audio samples in the two datasets were newly recorded, they were collected under controlled conditions. Therefore, their characteristics of singing techniques might be different from those appeared in songs.

However, several datasets have been annotated for real-world vocal music. The KVT dataset was originally built for vocal-related music-tagging tasks in K-POP music [17]. It contains 70 vocal tags, of which six are related to the singing technique (whisper/quiet, vibrato, shouty, falsetto, speech-like, and non-breathy). The annotation was conducted using crowdsourcing. The MVD dataset was built to analyze screams in heavy metal music, and has four different types of screams (high fry, mid fry, low fry, and layered) [18]. The regions and types of screams in the audio files were manually annotated. Similar to these studies, we are interested in versatile singing techniques in real-world music and thus we chose commercial music to build the dataset.

2.3 Singing/Playing Technique Identification on MIR research

Studies have been conducted on the automatic identification of singing techniques beyond computational analysis. There are two scenarios for identifying a singing techniques. One is classifying sung vocal audio into singing techniques. Several studies have conducted singing technique classification on VocalSet [19] and phonation modes [20, 21]. These works identify singing techniques but do not provide time-related information, such as start time and duration. The other method is to detect the singing technique in time. Miryala et al. [22] identified the singing expressions of raga, classical music in India. They created 35 recordings in eight ragas sung by six singers and showed a classification accuracy of 84.7% using a rule-based classifier. Yang et al. [8] proposed AVA, an interface for analyzing vibrato and portamento based on the filter diagonalization method (FDM) and hidden Markov model (HMM), respectively. They also provided a case study of the analysis of vibrato and portamento in the Beijing opera. Ikemiya et al. [23] provided rule-based parameterization of four singing expressions (vibrato, kobushi, gliss-up, gliss-down), extracted them from recorded vocal tracks to transfer to other vocal tracks, and demonstrated them on two sung excerpts. Kalbag et al. [18] provided scream detection for heavy metal music. They also built the MVD dataset and demonstrated classification, including non-scream singing and non-vocal music. Since singing techniques appear locally on sung vocals, we adapted these temporal detection strategies for identification.

3. DATASET

In this section, we describe the organization of our new dataset used for the analysis of this study. Due to the limit of the space, more details are described in the appendix site [0] .

3.1 Song Selection

To cover a wide range of singing techniques, the dataset should include various types of vocalists, considering gender, genre, tempo, and mood. We first listed 42 solo singers (21 males and 21 females), and four famous hit songs were selected from each singer to be as different as possible. Each song was performed as solo performance in Japanese. We collected audio tracks from commercial CD recordings of the J-POP songs. We trimmed the collected audio tracks and annotated only the first consecutive section (i.e., verse-A, verse-B and Chorus). Since we prioritized including various songs in our dataset rather than fully annotating multiple verses in a single song, we could collect a diverse set of singing styles.

When using popular commercial songs for academic research, copyright is always an issue. The works by Kim [17] on K-POP and Kalbag [18] on heavy metal music are good examples of developing datasets using commercial music, which distributes only labels about the performed music but not the audio signals or score information. We followed their solutions and made our dataset that consists of singing technique and pitch (frequency) contour labels.

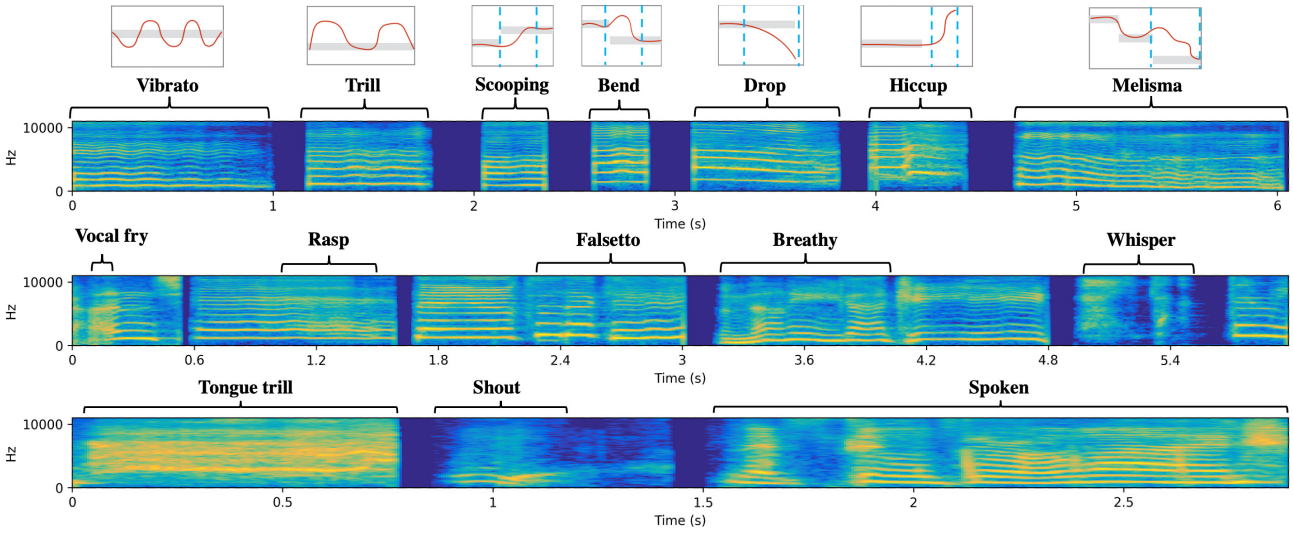


Figure 1. Singing techniques included in the dataset. (upper) Pitch techniques with a sketch of pitch contour. Gray is the target vocal note, the red line is the pitch contour and blue dotted lines are boundary of each technique. (middle) Timbral techniques. We also show the non-technique extracted from the same track. (lower) Miscellaneous techniques.

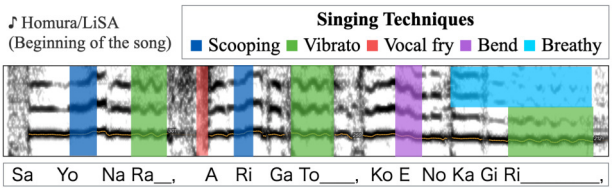


Figure 2. An overview of annotation. (middle) an excerpt spectrogram with singing technique (colored area) and vocal melody (orange curve).

3.2 Data Pre-processing

The dataset contains two types of annotations: vocal melody and singing techniques. We illustrate the annotation result in Figure 2. All annotations were conducted on isolated vocal tracks after vocal separation using Demucs v3 [24], which is a state-of-the-art model for musical source separation. During the annotation process, we confirmed that there was no dropout of vocal regions by comparing the original mixed track. Instead, we observed that the separated vocals tends to retain the back-chorus or sometimes instrumental sounds that are similar to the singer's voice (e.g., electric guitar, synthesizer).

3.3 Singing Technique Annotation

We thoroughly surveyed singing techniques based on instructional books [14, 25] and other conventional scientific research related to singing techniques [7, 13, 16, 17, 26–29] and defined the labels for the major techniques that appear in these references. We manually annotated songs using these labels.

Although many other singing techniques still exist, we considered the following 15 singing techniques in this study. We show the description of each singing technique in Table 1. The pitch contour sketches of these techniques and spectrogram examples are illustrated in Figure 1.

We note that these label names are not unique in the real-world (e.g., scooping is also called 'portamento,' 'glissando,' 'gliss-up,' 'shakuri' (in Japanese)). We made

Technique	description	type
vibrato	a periodic oscillation of pitch.	pitch
scooping	an upper continuous picth change	pitch
drop	a lower continuous picth change	pitch
bend	a short tremolo or U/inverted-U shaped pitch change	pitch
hiccup	a short hiccuping on attack/release of note	pitch
melisma	a musical arrangement in which several notes are applied to one syllable of a lyric.	pitch
trill	a continuous pitch change between two notes	pitch
falsetto	sung by falsetto register.	timbre
breathy	sung by breathy sound.	timbre
whisper	sung like whispering.	timbre
rasp	sung by a creaky voice with subharmonics.	timbre
vocal fry	sung by a creaky voice and pulse register phonation.	timbre
spoken	singing like rapping, *sprechgesang* [1], and some other styles like speaking.	misc.
shout	shouting.	misc.
tongue trill	a rolling tongue, occurred on [r] consonant.	misc.

Table 1. The description of each singing technique appeared in the dataset.

[1] A vocal style between singing and speaking.

[2] Although discarded in the analysis, we also annotate 'unknown' labels if the region seems to represent a singing technique but is difficult to classify them into any of the techniques above and found 24 unknown techniques in total. In the techniques, there are some sounds akin to coughing and pig squeals, for example.

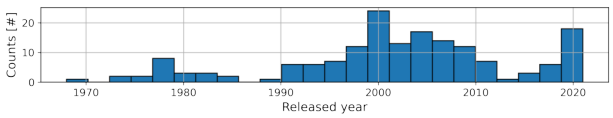


Figure 3. Distribution of song year of release.

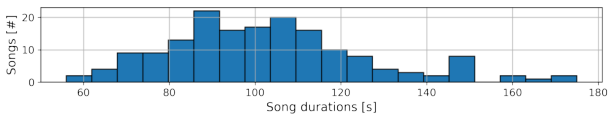


Figure 4. Distribution of song length in seconds.

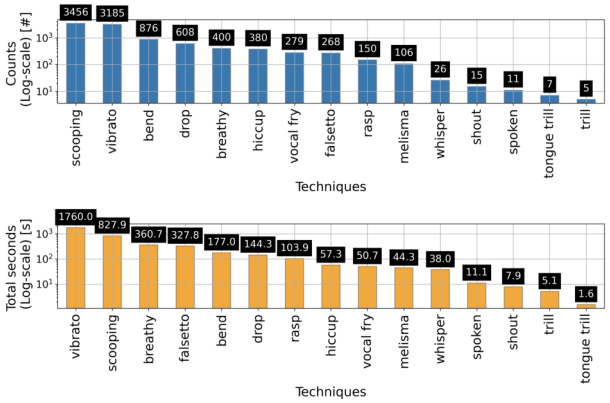


Figure 5. Statistics of the labels. upper: counts, lower: total duration.

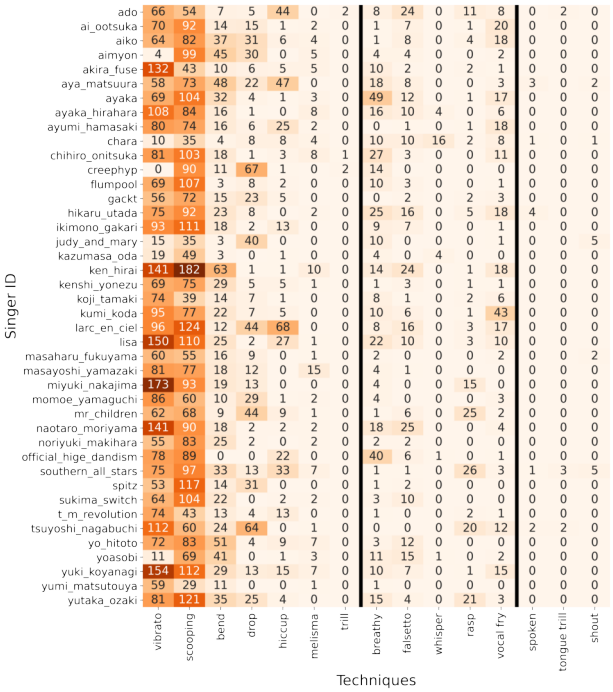


Figure 6. Occurrence distribution of singing techniques by singer. The vertical black line divides each category.

frame-level annotations on the audio tracks collected based on the vocabulary. Singing techniques were carefully annotated by an experienced vocalist (i.e., the first author of the paper) with the helps of sound playback and visualizing the spectrograms and pitchgrams. The annotation process is conducted on Sonic Visualizer [30].

3.4 Melody Annotation

Since pitch is an essential component of singing technique analysis, we further annotated melodic pitch using Tony [31], followed by manual correction such as removing the unvoiced parts and reverberation tails.

4. DESCRIPTIVE STATISTICS

4.1 Song Statistics

The distribution of songs selected for the dataset based on the year of release is shown in Figure 3. Songs can be collected from various eras, ranging from 1968 to 2021. The distribution of the songs in the dataset is shown in Figure 4. The overall length was 4h 47m 39s, and the average length of a song track was 1m 43s. The ratio of technique regions per song track was 22.8%.

4.2 Label Statistics

We present the statistics for the annotated labels as a histogram at the upper side of Figure 5. The most frequent technique is 'scooping'. It is followed by 'vibrato', 'bend', and 'drop'. This indicates that such techniques are relatively common in J-POP. These techniques are also used in Japanese commercial karaoke systems for vocal assessment [32]. 'scooping', 'bend' and 'drop' are portamento, whose frequent use is sometimes considered undesirable in classical singing [25]. It can be said that this frequent use of portamento is a characteristic of J-POP.

We show the distribution of techniques by each singer in Figure 6 and find that the occurrence frequency of the techniques is different for each singer. We also confirmed that scooping appears more than 29 times for every singer, whereas several singers tend to not use vibrato frequently (e.g., 'creephyp', 'aimyon', and 'yoasobi').

The lower side of Figure 5 and Figure 7 shows the total duration and distribution of each technique using a boxenplot, respectively. Figure 7 shows that most of the techniques are relatively short (that is, the range between 0.1s and 1s.), especially for 'drops', 'scooping', 'bend', 'hiccup', and 'vocal fry'. The average length of the singing techniques was 0.4s.

5. SINGING TECHNIQUE DETECTION

In this section, we describe the detection of the singing techniques. We conducted experiments on a multi-class detection scenario, which handles classification and localization simultaneously. The problem setting is the same as that in sound event detection (SED) [33]. Figure 8 illustrates the proposed method. We investigated the performance of the CRNN model in a singing technique detec-

[3] https://seaborn.pydata.org/generated/seaborn.boxenplot.html

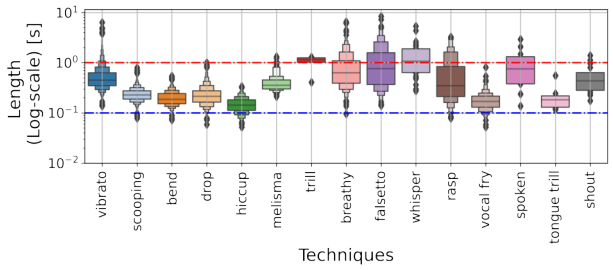


Figure 7. Distribution of each technique. Red and blue horizontal lines indicate 1 s and 0.1 s, respectively.

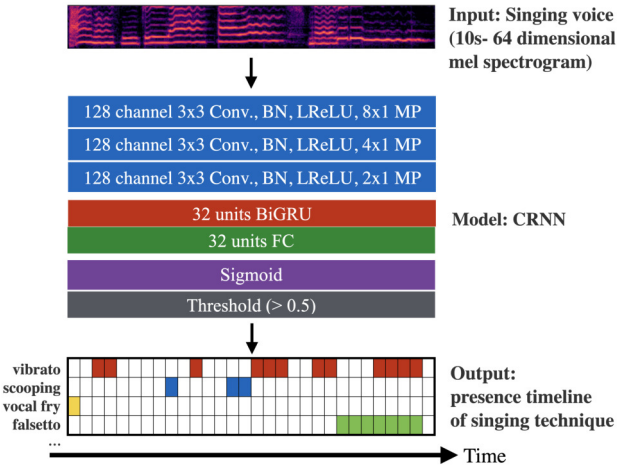


Figure 8. An overview of the singing technique detection. The configuration of the CRNN model is also shown.

tion task. In addition to running the simple setting, we also investigated how the considerations of the characteristics of the dataset improve the performance (i.e., label sparseness and pitch information). Thus, through the experiment, we attempted to answer for the following three research questions: 1) To what extent can we detect singing techniques using machine learning? 2) Can modification of the loss function treat the problem of label sparseness? 3) Can auxiliary pitch information help improve the detection performance?

5.1 Experimental Conditions

We performed singer-wise seven-fold cross validation during the experiment. We first split the singers into seven groups. Then, in each run, one group is left out for test set, another one group is for validation set , and the rest are used for training set. Owing to the label imbalances of techniques between singer, we used the 9 most common classes (i.e., 'bend,' 'breathy,' 'drop,' 'falsetto,' 'hiccup,' 'rasp,' 'scooping,' 'vibrato,' and 'vocal fry'), which appear on every fold. We divided the singer fold to balance the amount of each technique as much as possible.

We segmented the vocal tracks, which are separated by Demucs v3 [24] in advance, into 10s audio clips and non-overlapping parts at a sample rate of 44.1 kHz. Therefore, they were converted into 64-dimensional log-mel spectrograms. For experiments that involve the computation of short-time Fourier transform (STFT), we used a Hann window with 2048 samples to compute the discrete Fourier transform (DFT). The hop size was set to 10 ms in every experiment.

We used a CRNN model, which is widely used as a baseline system for SED. We adapted the settings of the model of Imoto et al. [34], which treated a similar issue (i.e, label sparseness) of ours in SED. The model configuration is shown in Figure 8. The model consists of three convolutional layers, one layer of bidirectional GRU cells, and one fully connected (FC) layers. The input was the aforementioned log-mel-spectrogram, and the output was a multi-hot vector representation of the singing technique for each time frame. All experiments were conducted using a batch size of 16, and the model parameters were optimized using an Adam optimizer with a learning rate of 10^{-4}. The model stopped training if the value of the loss function on the validation set did not improve over 10 epochs.

The model was optimized using the binary cross-entropy (BCE) loss. Performance was evaluated using segment-based recall (R), precision (P), macro-F-measure (Macro-F), and micro-F-measure (Micro-F) [35] using sed_eval. The macro-F-measure and micro-F-measure denote the class-wise and instance-wise average of the F-measure, respectively. We set the segment length to 50 ms.

5.2 Model Improvement

5.2.1 Label-frame sparsity

As mentioned in Section 4.2, most singing techniques have a short duration (i.e., shorter than 1 s). This can cause label imbalance between non-technique frames. To alleviate the problem, we investigated the effect of *focal loss* [36], which was originally proposed for image object detection. The focal loss can be represented as follows:

$$L_{fl}(p_t) = -\alpha(1-p_t)^{\gamma}\log(p_t) \quad (1)$$

where $\alpha \in [0,1]$ is a weighting factor for balancing the importance of positive and negative examples, and the term $(1-p_t)^{\gamma}$ is a modulating factor, with γ controlling the rate of dominant examples. We set $\alpha = 0.2$ and $\gamma = 2$ for all conditions in the work that used focal loss.

5.2.2 Pitch information

We demonstrated that some of the techniques, such as vibrato and scooping, have a pattern of the shape to certain extent. Therefore, it is possible that explicitly feeding the pitch helps in the detection. We further investigated the effect of the auxiliary information. Under this condition, we considered two ways of obtaining pitch in our experiment. One is the ground-truth (GT) pitch annotation mentioned in Subsection 3.4. However, when used in real-world applications, it is difficult to obtain correct pitch annotation. Hence, we also used pitch estimation predicted by CREPE [37] on a separated vocal track. As CREPE can compute pitch confidence, we adopted the pitch value

[4] https://tut-arg.github.io/sed_eval/index.html

	Macro-F	Micro-F	P	R
BCE	37.7%	56.6%	40.2%	41.7%
Focal	39.6%	57.7%	40.3%	42.6%
BCE-GT	39.1%	57.1%	41.4%	42.8%
BCE-CREPE	39.1%	57.1%	40.2%	41.7%
Focal-GT	**40.4%**	**58.3%**	**43.1%**	42.2%
Focal-CREPE	40.3%	57.9%	42.7%	**43.3%**

Table 2. The results of singing technique detection.

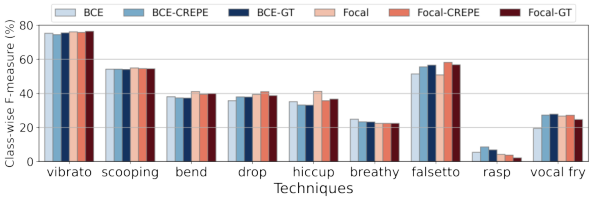


Figure 9. Class-wise F-measure

where the confidence value was higher than 0.5, whose overall accuracy was 78.0% evaluated by mir_eval [38]. We converted the pitch contour to a mel-band pitchgram that has the same frequency dimensions as the input mel-spectrogram, as in the work of singer identification [39]. Therefore, we stacked it on a mel-spectrogram along the channel axis.

5.3 Experimental Results

5.3.1 Baseline

The first row of Table 5.3 shows the results of the CRNN model compiled by BCE: 37.7% for Macro-F and 56.6% for Micro-F. The fact that Macro-F is much lower than Micro-F indicates that the model failed to identify minority rather than majority classes. Actually, the class-wise F-measure, as shown in Figure 9, indicates that detection of the minority techniques (e.g., 'rasp' and 'vocal fry') is more difficult than the majority techniques (e.g., 'vibrato' and 'scooping').

5.3.2 Effect of focal loss and pitch information

First, we show the results for the effect of *focal loss* in the middle part of Table 5.3. The detection performance improved by 1.9% in Macro-F. We further studied the class-wise F-measure, as shown in Figure 9, and confirmed that the performance of short techniques, such as 'bend' (38.0% $\rightarrow$ 41.1%), 'drop' (35.7% $\rightarrow$ 39.3%), and 'hiccup' (35.1% $\rightarrow$ 41.2%) are especially improved. This indicates that *focal loss* can adapt to the sparsity of the label.

Next, we showed the results of the effect of the auxiliary pitch information in the middle part of Table 5.3. A remarkable finding is that the F-measure of 'falsetto' was particularly improved (51.3% $\rightarrow$ 56.5%). We also confirmed that the recall value of 'falsetto' is raised more (64.7% $\rightarrow$ 74.1%) rather than the precision value (45.1% $\rightarrow$ 48.3%). This indicates that pitch information hint at the relationship between the sung pitch value and the occurrence of falsetto (i.e., falsetto can appear only at a higher pitch). We also confirmed that the input of CREPE pitch shows the similar tendency.

We further investigated the effects of combining these two types of auxiliary information. As shown in the bottom part of Table 5.3, the combination of GT pitch and focal loss is the best condition in terms of both macro- and micro-F values. We can confirm that both effects of each auxiliary information occur under this condition (red and dark-red bars in Figure 9). This indicates that the condition is robust to techniques that have a short duration or some relationships with pitch.

[5] The other metrics were following; 85.9% for voice recall, 29.9% for voicing false alarms, 94.5% for raw pitch accuracy.

5.3.3 Challenges

The remaining challenges in the task are detected unnatural region lengths and singers' identity. We confirmed that one of the common mis-detection cases is from the detection of too short or frequently switching regions. It might need to consider post-processing, such as temporal filtering or probabilistic model (e.g, hidden Marcov model, hidden semi-Marcov model, etc.). As for singer identity, it can affect the performance of the timbral techniques. We labeled timbral techniques when the sung voice transformed from the ordinary voice of the singer, and it caused the confusion. For example, the 'rasp' from a singer who has clean voice and ordinary voice from a singer who has raspy voice can be confusing. Therefore, there is still an issue of disentangling singers' identity and singing techniques.

6. CONCLUSION AND FUTURE WORK

This work presented an analysis and identification of the singing technique in J-POP music as a case study. In addition, we built a new dataset consisting of 168 J-POP songs with annotation of pitch contour and singing techniques. The contributions of this study are summarized as follows: 1) we constructed a dataset with frame-level annotations of singing techniques on 168 famous J-POP vocal songs, 2) conducted a descriptive-statistical analysis of the dataset, 3) proposed a CRNN-based singing technique detection model as a baseline, and confirmed that both auxiliary pitch information and focal loss help the detection performance.

As we showed the appearance of singing techniques relates to some other factors (e.g., difference of singing technique distribution by singer, pitch information helps the detection of falsetto, etc.), we are planning further investigations of relationships or joint identification between other musical factors (e.g. musical note, lyrics, beat and tempo, etc.) or higher level information (e.g. singer, songs' emotion, mood, etc.) with making more types of annotation. Although this work focuses only on J-POP, the knowledge can be applicable to other types of singing. Similar study on other style of vocal such as K-POP, western pops or non-popular vocal and comparison between them are other interesting topics for future work.

7. ACKNOWLEDGEMENTS

This work was supported by JST SPRING, Grant Number JPMJSP2124. Dr. Maiko Murai (Associate Prof. at the University of Tsukuba) gave advices on copyright law. Yoshihide Ishikawa, Sayori Takayama and Karolina Shi helped the preparation of the metadata. We are grateful to above all.

8. REFERENCES

[1] M. Ryynänen, *Singing Transcription*, A. Klapuri and M. Davy, Eds. Boston, MA: Springer US, 2006. [Online]. Available: https://doi.org/10.1007/0-387-32845-9_12

[2] E. J. Humphrey, S. Reddy, P. Seetharaman, A. Kumar, R. M. Bittner, A. Demetriou, S. Gulati, A. Jansson, T. Jehan, B. Lehner, A. Kruspe, and L. Yang, "An introduction to signal processing for singing-voice analysis: High notes in the effort to automate the understanding of vocals in music," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 82–94, 2018.

[3] H. Kenmochi and H. Ohshita, "Vocaloid - commercial singing synthesizer based on sample concatenation," in *Interspeech*, 2007.

[4] C. E. Seashore, "A musical ornament, the vibrato," *Proc. of Psychology of Music, 1938*, pp. 33–52, 1938.

[5] E. Prame, "Measurements of the vibrato rate of ten singers," *J. Acoust. Soc. Am.*, vol. 96, no. 4, pp. 1979–1984, 1994.

[6] J. Sundberg, "The perception of singing," in *The psychology of music*. Elsevier, 1999, pp. 171–214.

[7] K.-I. Sakakibara, L. Fuks, H. Imagawa, N. Tayama *et al.*, "Growl voice in ethnic and pop styles," in *Proceedings of International Symposium on Musical Acoustics*, 2004.

[8] L. Yang, S.-K. Rajab, and E. Chew, "Ava: an interactive system for visual and quantitative analyses of vibrato and portamento performance styles," in *In Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[9] K. Hirayama and K. Ito, "Discriminant analysis of the utterance state while singing," in *Proceedings of IEEE Symposium on Signal Processing and Information Technology (ISSPIT)*, 2012.

[10] Y. Lee, M. Oya, T. Kaburagi, S. Hidaka, and T. Nakagawa, "Differences among mixed, chest, and falsetto registers: A multiparametric study," *Journal of Voice (In Press, Corrected Proof)*, 2021.

[11] O. Nieto, "Voice transformations for extreme vocal effects," *Master thesis, Pompeu Fabra University*, 2008.

[12] N. Migita, M. Morise, and T. Nishiura, "A study of vibrato features to control singing voices," *Proceedings of International Congress on Acoustics*, pp. 23–27, 2010.

[13] Y. Yamamoto, T. Nakano, M. Goto, H. Terasawa, and Y. Hiraga, "Analysis of frequency, acoustic characteristics, and occurrence location of singing techniques using imitated j-pop singing voice," *The Special Interest Group Technical Report of IPSJ (MUS)*, no. 20, pp. 1–8, 2021, (in Japanese).

[14] AKIRA/Utana, *The Secret Singing Technique Used by Professional Singers 17 - The karaoke score improved from a 68 to a 92!-*. Tsuta-shobou, 2013, (in Japanese).

[15] P. Proutskova, C. Rhodes, T. Crawford, and G. Wiggins, "Breathy, resonant, pressed–automatic detection of phonation mode from audio recordings of singing," *Journal of New Music Research*, vol. 42, no. 2, pp. 171–186, 2013.

[16] J. Wilkins, P. Seetharaman, A. Wahl, and B. A. Pardo, "Vocalset: A singing voice dataset," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 468–474.

[17] K. L. Kim, J. Lee, S. Kum, C. L. Park, and J. Nam, "Semantic tagging of singing voices in popular music recordings," *IEEE/ACM TASLP*, vol. 28, pp. 1656–1668, 2020.

[18] V. Kalbag and A. Lerch, "Scream detection in heavy metal music," in *Proceedings of Sound and Music Computing (SMC)*, 2022.

[19] Y. Yamamoto, J. Nam, H. Terasawa, and Y. Hiraga, "Investigating time-frequency representations for audio feature extraction in singing technique classification," in *Proceedings of Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2021.

[20] J.-L. Rouas and L. Ioannidis, "Automatic classification of phonation modes in singing voice: towards singing style characterisation and application to ethnomusicological recordings," in *Interspeech*, 2016.

[21] D. Stoller, S. Dixon *et al.*, "Analysis and classification of phonation modes in singing," in *The 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.

[22] S. S. Miryala, K. Bali, R. Bhagwan, and M. Choudhury, "Automatically identifying vocal expressions for music transcription." in *In Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 2013.

[23] Y. Ikemiya, K. Yoshii, and K. Itoyama, "Transferring vocal expressions of a professional singer to unaccompanied singing signals," in *ISMIR-LBD 2014*, 2014.

[24] A. Défossez, "Hybrid spectrogram and waveform source separation," in *In Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[25] F. Husler and Y. Rodd-Marling, *Singing: The physical nature of the vocal organ: A guide to the unlocking of the singing voice*. Random House (UK), 1976.

[26] Y. Ikemiya, K. Itoyama, and H. G. Okuno, "Transferring vocal expression of f0 contour using singing voice synthesizer," in *Proceedings of International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems. (IEA/AIE)*, 2014, pp. 250–259.

[27] M. Panteli, R. Bittner, J. P. Bello, and S. Dixon, "Towards the characterization of singing styles in world music," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017, pp. 636–640.

[28] M. Nakazato, "How to sing ornamented melody in j-pop: Through the analysis of kobushi in ken hirai, keisuke kuwata, chemistry and dreams come true," *Japanese Journal of Music Education Practice*, vol. 5, no. 1, pp. 32–39, 2007, (in Japanese).

[29] M. Jones, "The rhythm and blues (r&b) protest songs of the civil rights movement: Outlining the natural alignment between the foundational r&b recordings artists and the african-american church during the movement," *Master Theses, Liberty University*, 2016.

[30] C. Cannam, C. Landone, and M. Sandler, "Sonic visualiser: An open source application for viewing, analysing, and annotating music audio files," in *Proc. ACMMM 2010*, 2010, pp. 1467–1468.

[31] M. Mauch, C. Cannam, R. Bittner, J. Bello *et al.*, "Computer-aided melody note transcription using the tony software: Accuracy and eciency," in *Proceedings of the First International Conference on Technologies for Music Notation and Representation (TENOR 2015)*, 2015.

[32] D. C. LTD., "Karaoke systems," *Japan Patent 2020-166142*, 2020, (in Japanese).

[33] A. Mesaros, T. Heittola, T. Virtanen, and M. D. Plumbley, "Sound event detection: A tutorial," *IEEE Signal Processing Magazine*, vol. 38, no. 5, pp. 67–83, 2021.

[34] K. Imoto, S. Mishima, Y. Arai, and R. Kondo, "Impact of sound duration and inactive frames on sound event detection performance," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2021, pp. 860–864.

[35] A. Mesaros, T. Heittola, and T. Virtanen, "Metrics for polyphonic sound event detection," *Applied Sciences*, vol. 6, no. 6, p. 162, 2016.

[36] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2017, pp. 2980–2988.

[37] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018, pp. 161–165.

[38] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[39] T.-H. Hsieh, K.-H. Cheng, Z.-C. Fan, Y.-C. Yang, and Y.-H. Yang, "Addressing the confounds of accompaniments in singer identification," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. IEEE, 2020, pp. 1–5.

Papers - Session IV

PERFORMANCE MIDI-TO-SCORE CONVERSION BY NEURAL BEAT TRACKING

Lele Liu[*1,2] **Qiuqiang Kong**[3] **Veronica Morfi**[1] **Emmanouil Benetos**[1,2]
[1] Centre for Digital Music, Queen Mary University of London, UK
[2] The Alan Turing Institute, UK [3] ByteDance Shanghai, China
lele.liu@qmul.ac.uk

ABSTRACT

Rhythm quantisation is an essential part of converting performance MIDI recordings into musical scores. Previous works on rhythm quantisation are limited to the use of probabilistic or statistical methods. In this paper, we propose a MIDI-to-score quantisation method using a convolutional-recurrent neural network (CRNN) trained on MIDI note sequences to predict whether notes are on beats. Then, we expand the CRNN model to predict the quantised times for all beat and non-beat notes. Furthermore, we enable the model to predict the key signatures, time signatures, and hand parts of all notes. Our proposed performance MIDI-to-score system achieves significantly better performance compared to commercial software evaluated on the MV2H metric. We release the toolbox for converting performance MIDI into MIDI scores at: https://github.com/cheriell/PM2S.

1. INTRODUCTION

Performance MIDI-to-Score (PM2S) conversion aims to convert a performance MIDI, usually recorded by electronic MIDI keyboards, into musical scores in computer-readable score format such as MIDI, MusicXML [1] and Lilypond [2]. PM2S covers a list of subtasks including rhythm quantisation, note value prediction, key estimation, voice separation, and possibly score typesetting such as beaming and playing techniques annotation. PM2S can be used in various scenarios such as music improvisation [3], complete music transcription [4–7] in combination with multi-pitch detection, music alignment [8], music education, music performance analysis [9], and building music archives.

Although research on the various subtasks on PM2S dates back to earlier decades [10–13], the first academic paper that fully formulated this task was published in 2016 [14]. The paper works on converting a performance MIDI recording into a LilyPond score. The authors firstly fixed spurious overlapping notes according to a defined note overlapping ratio, then applied a probabilistic model using Hidden Markov Models (HMMs) based on a tactus-root combination concept [15] for meter, harmony, and stream estimation. After that, the note onsets and offsets are quantised to beat subdivisions. Note spellings and staves are determined from the predicted harmony and streams.

Following the use of conventional methods for solving subtasks such as rhythm quantisation, recent years have seen some advances on statistical methods [5, 16–19]. Nakamura et al. [16] proposed an improvement for rhythm transcription using a merged-output HMM to solve the problem introduced by multiple voices. A note value prediction method using Markov random fields was proposed in [17]. The two methods were further improved for automatic music transcription by combining multi-pitch detection [5, 19]. Mcleod and Steedman [18] proposed an HMM-based meter detection method for aligning MIDI performances.

Over the past two years, researchers have published papers using deep learning methods on PM2S. Hiramatsu et al. [20] proposed to use a recurrent neural network for joint estimation of note values and voices from note pitches and onset times, which in combination with multi-pitch detection and rhythm quantisation, outperformed previous methods [14, 19] for automatic music transcription. The Score Transformer [21] provided a solution to generating human-readable scores from quantised MIDI files.

From previous works, we observe that the problem of PM2S is usually solved through a combination of different methods for subtasks. Moreover, although there are attempts on applying deep learning methods for the task, they do not cover the rhythm quantisation step.

To address the limitations mentioned above, we propose to use a deep learning method for PM2S. We develop a CRNN model that directly converts a performance MIDI into a MIDI score. We pay special attention to the rhythm quantisation step, and propose to solve the problem by tracking beats on a MIDI note sequence. We then expand the CRNN model to predict a compact output for generating a quantised MIDI score, including quantised onset times and note values, time signatures, key signatures and hand parts. To better capture our method's ability in modelling expressive performances, we train and evaluate our proposed model on a set of classical piano music datasets [22, 23]. In a comparison with two commercial software

* The author conducted this work as an intern at ByteDance.

© L. Liu, Q. Kong, V. Morfi, and E. Benetos. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** L. Liu, Q. Kong, V. Morfi, and E. Benetos, "Performance MIDI-to-score conversion by neural beat tracking", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

products MuseScore [24] and Finale [25], our proposed method achieved significantly better performance based on the MV2H metric [26].

2. METHODOLOGY

2.1 Problem definition

Our proposed method aims to predict a MIDI score from a performance MIDI using a deep learning model. Assume a performance MIDI note sequence $\mathbf{X}$ where notes are ordered firstly by time and secondly by pitch. Each note in the sequence is represented by a tuple:

$$\mathbf{X} = \{(p_n, o_n, d_n, v_n)\}_{n=1}^{N} \tag{1}$$

where p_n, o_n, d_n, v_n are the MIDI pitch number, onset and duration in seconds, and velocity for the n-th note, N is the number of notes. We aim to predict the transcribed MIDI score annotations including

$$\mathbf{Y_n} = \{(mo_n, nv_n, h_n)\}_{n=1}^{N} \tag{2}$$

$$\mathbf{Y_t} = \{(t_i, tn_i, td_i)\}_{i=1}^{T} \tag{3}$$

$$\mathbf{Y_k} = \{(t_i, k_i)\}_{i=1}^{K} \tag{4}$$

$$\mathbf{Y} = (\mathbf{Y_n}, \mathbf{Y_t}, \mathbf{Y_k}) \tag{5}$$

where $\mathbf{Y_n}$ covers the musical onset time mo_n, note value nv_n and hand part h_n for all notes; $\mathbf{Y_t}$ and $\mathbf{Y_k}$ are the time signature and key signature changes, where t_i is the time and tn_i, td_i and k_i are the time signature numerator, time signature denominator, and key signature respectively. T and K are the numbers of time signature changes and key signature changes. A MIDI score can be obtained by combining information from $\mathbf{X}$ and $\mathbf{Y}$.

2.2 From beat tracking to rhythm quantisation

Considering rhythm quantisation as a fine-grained tracking of beats and beat subdivisions, we propose to do rhythm quantisation by combining beat tracking and the prediction of musical note onset times in subdivisions within a beat. We use a deep learning model for both beat tracking and musical onset time prediction. Adding the beat tracking component allows the two components to learn from each other. It also allows us to use training data such as [23] that has beat-level annotations, but do not provide fine-grained alignment in subdivision level.

We define mo_n in Eq. (2) as a ratio of:

$$mo_n = s_n / S \tag{6}$$

where s_n is the musical note onset time in subdivisions within a beat, and S is the number of subdivisions per beat in rhythm quantisation.

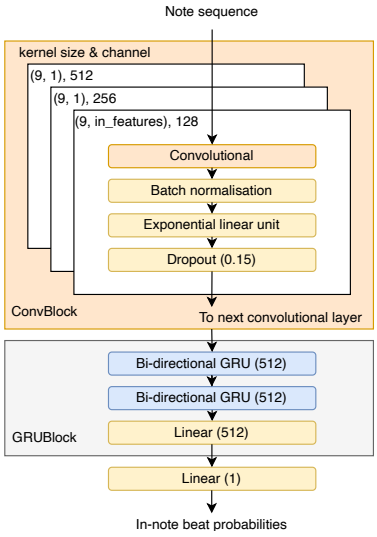


Figure 1. Model architecture for in-note beat prediction.

2.3 Neural beat tracking on note sequences

2.3.1 In-note and out-of-note beats

To match the model input data in note sequence format, we separate beats into two groups:

- *In-note*: beats concurrent with at least one note onset;
- *Out-of-note*: beats not concurrent with any note onset.

In our proposed approach, we first train a deep learning model to predict in-note beats in a binary classification task, and then use dynamic programming to infer out-of-note beats from the in-note ones.

2.3.2 In-note beat prediction

Let the ground truth in-note beat labels for each note in the note sequence be $\mathbf{B}_n \in \{0, 1\}$. We define a binary classification model that predicts the in-note beat probabilities for each note given the note sequence $\mathbf{X}$, that is:

$$P_n = P(\mathbf{B}_n | \mathbf{X}) \tag{7}$$

The model is trained to find the minimum binary cross-entropy loss defined as:

$$\mathcal{L} = -\frac{1}{N} \sum_{n=1}^{N} (\mathbf{B}_n * log(P_n) + (1 - \mathbf{B}_n) * log(1 - P_n)) \tag{8}$$

In order to acquire the beat and downbeat predictions, we use a CRNN with 3 convolutional layers and 2 bi-directional gated recurrent unit (GRU) layers. Figure 1 shows the model architecture. The predicted in-note beat probabilities are then converted to binary labels by dynamic thresholding. The dynamic threshold depends on the maximum predicted probability in a fixed segment length in seconds.

2.3.3 Out-of-note beat prediction

We assume there are in total N^i in-note beats and out-of-note beats are at subdivisions of their neighbouring in-note beats b_n^i and b_{n+1}^i where b_n^i means the n-th in-note beat

Algorithm 1 Out-of-note beat prediction

Input: List of in-note beats $\mathcal{B}^i$
Output: List of all beats $\mathcal{B}$ after adding out-of-note beats

$n \leftarrow 1$
for $K = 0, 1, 2, 3$ **do**
    Initialise objective function $\mathcal{O}_K \leftarrow 0$
    Initialise beat sequence $\mathcal{B}_K \leftarrow \{b_1\}$
end for
for $n = 1, 2, \ldots, N^i - 2$ **do**
    for $K_{cur} = 0, 1, 2, 3$ **do**
        Get out-of-note beats for current step by Eq. (9)
        if Tempo is beyond tempo range limits **then**
            Go to next K_{cur}
        end if
        for $K_{prev} = 0, 1, 2, 3$ **do**
            Update objective function by Eq. (11)
        end for
        Select the minimum objective among all K_{prev} values
        Add out-of-beats for current step to the beat sequence mapped to the selected K_{prev}
    end for
    for $K_{cur} = 0, 1, 2, 3$ **do**
        Update $\mathcal{O}_K$, $\mathcal{B}_K$ mapped with K_{cur}
    end for
end for
Return the beat sequence in $\mathcal{B}_K$ with the minimum objective function $\mathcal{O}_K$

and $n \in \{1, 2, \ldots, N^i\}$. We insert out-of-note beats b^o from candidates in:

$$b^o_{n,K} = \{b^i_n + \frac{k}{K+1}(b^i_{n+1} - b^i_n)\}^K_{k=1} \tag{9}$$

where $K \in \{0, 1, 2, 3\}$ is the number of out-of-note beats to insert for the current interval. We try to find a way that minimises the tempo change after adding out-of-note beats to the beat sequence. To describe the level of tempo change for a list of beats, we define an objective function as follows:

$$\mathcal{O}_1 = \sum_{n=1}^{N-2} \left| log\left(\frac{b_{n+2} - b_{n+1}}{b_{n+1} - b_n}\right) \right| \tag{10}$$

where N is the number of beats in the final beat sequence and b_n is the n-th beat. However, this function does not take into account adding too many out-of-note beats. We thus add a penalty associated with the number of out-of-note beats to the objective function, resulting in:

$$\mathcal{O} = \mathcal{O}_1 + \lambda \times N^o \tag{11}$$

where λ is the penalty coefficient applied to avoid adding too many out-of-note beats which is tuned based on experiments and N^o is the number of out-of-note beats added. In this way, we obtain an objective function $\mathcal{O}$ to be minimised in the out-of-note beat prediction process that encourages both a low level of tempo change and adds fewer out-of-note beats.

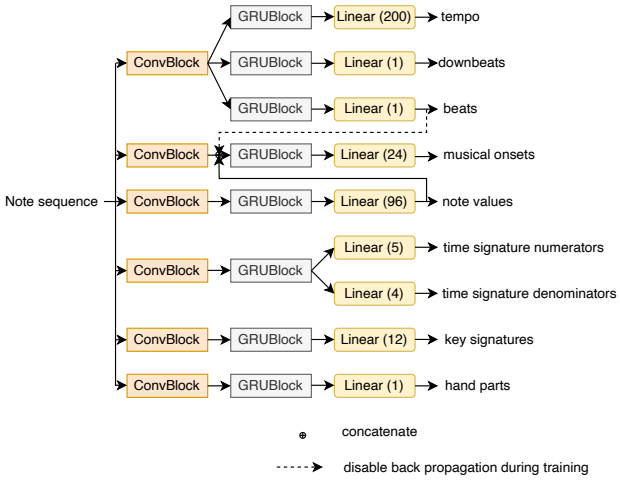


Figure 2. Model architecture for performance MIDI-to-score conversion.

We use a dynamic programming algorithm (defined in Algorithm 1) to find the optimal out-of-note beats while minimizing $\mathcal{O}$.

2.4 Performance MIDI-to-Score

Based on the problem definition in Section 2.1, we make some modifications to the prediction of time signature and key signature changes. In our proposed model, we define $\mathbf{Y_t}$ and $\mathbf{Y_k}$ in a way that the time signature and key signature values are mapped with each note by the note onsets. We also add beat b_n, downbeat db_n, and tempo tem_n prediction in note level. Beat probabilities are defined as in Eq. (7); similar definitions are used for downbeats. As a result, we define our model as $\mathbf{X} \rightarrow \mathbf{Y}'$ where:

$$\mathbf{Y}' = \{(mo_n, nv_n, h_n, tn_n, td_n, k_n, b_n, db_n, tem_n)\}^N_{n=1} \tag{12}$$

We then define a CRNN-based model that maps the input note sequence $\mathbf{X}$ with $\mathbf{Y}'$. Among the output elements, hand part h_n, beats b_n and downbeats db_n are defined as binary classification tasks; the others are defined as multi-class classification tasks, where

- Musical onset mo_n is defined as in Eq. (6), whose value is quantised by beat subdivisions;
- Note values nv_n are quantised by beat subdivisions;
- Time signature is defined to be $tn_n \in \{0, 2, 3, 4, 6\}$ and $td_n \in \{0, 2, 4, 8\}$, 0s indicate other values;
- Key signature k_n is defined to be in {C, Db, D, Eb, E, F, Gb, G, Ab, A, Bb, B};
- Tempo is quantised following a minimum inter-beat-interval of 0.01s.

Our proposed model structure is shown in Figure 2. The Convolutional blocks (ConvBlock) and GRU blocks (GRUBlock) have similar structure to the ones in Figure 1. Links between branches allow the output elements to learn from each other. The model is trained jointly using binary cross-entropy loss for binary classification output elements and negative log-likelihood loss for multi-task classification output elements.

A MIDI score can then be generated by combining the information given in the performance MIDI and the output

Statistics	Train	Valid	Test	Total
Distinct pieces	426	49	29	504
Performances	1324	157	29	1510
Duration (hour)	108.3	12.7	2.2	123.2
Notes (10^3)	3984.0	517.6	73.2	4574.7

Table 1. Dataset Statistics. Performances from the MAPS dataset and the CPM database for the same piece are not counted as different performances, since MAPS pieces are originally obtained from the CPM database.

predictions from our proposed model.

3. EXPERIMENTS

3.1 Data

3.1.1 Datasets

To better demonstrate the ability of our proposed method on quantising expressive performance, we use a collection of classical piano music pieces from 1) The MAPS dataset [22] and corresponding metrical annotations from the A-MAPS dataset [27]; 2) The Classical Piano-Midi (CPM) database [28]; and 3) The ASAP dataset [23].

In order to avoid piece overlaps between train, validation and test splits, we use the distinct music piece labels for all piano performances given in the ACPAS dataset [29]. We use a real recording subset in MAPS (the ENSTDkCl subset) as the test set as in [5, 20]. Other music pieces not included in the test set are randomly split into training and validation based on their distinct music piece labels. In this way, we get a train/validation/test setup with no overlapping music pieces among splits. Table 1 shows the dataset statistics.

3.1.2 Annotation

Annotations are provided in different formats in the datasets we use. The A-MAPS dataset and the CPM database provide fully annotated MIDI scores with tempo and metrical information. Thus, we extract the annotations we need from the MIDI scores directly. Performances from the ASAP dataset come with two sets of annotations, MIDI scores and annotations in .tsv files. However, the MIDI scores were written by non-professionals and are not a good source of ground truth. Thus, we follow the authors' suggestion to use the provided annotations in the .tsv files, in which beat, downbeat, time signature and key signature annotations are provided. Due to the fact that the .tsv annotations do not cover precise fine-grained metrical and hand part annotations, we mask the ASAP data on musical onset time, note value, and hand part prediction during training. Moreover, when matching note onset times to beats or non-beats in our proposed model, we set a tolerance of $\pm$50ms to beat matching to comprise short-time variances introduced by human performance.

3.1.3 Data augmentation

Given the note sequence information defined in Section 2, we consider the following data augmentation methods:

- *Pitch shift*: Shift MIDI pitch values up or down for the whole music performance. The shifted pitch is defined as: $p_s = p_0 + p_{shift}$ where p_0 is the original pitch value and pitch shift $p_{shift} \in \{0, \pm1, \pm2, ..., \pm12\}$.
- *Tempo change*: Change the tempo to a ratio of the original tempo. The new tempo $tem_c = r_t * tem_0$ where tem_0 is the original tempo and $r_t \in [0.8, 1.2]$ is the ratio.
- *Note removal*: For polyphonic music, there should be little influence to the metrical structure of the music piece when removing some concurrent notes. Thus, for each group of M concurrent notes, we randomly remove 0 to M-1 of them from the MIDI performance.
- *Note introduction*: Contrary to note removal, we randomly select 0-100% of notes from the MIDI performance, and add new notes that are concurrent with the selected ones. We keep the velocity and duration the same, so as to preserve the original music structure as much as possible. The new note pitches are ±12 semitones apart from the original note pitches.

3.2 Evaluation metrics

3.2.1 Beat tracking evaluation

We define a *note-level F-measure* for evaluating in-note beat tracking, and a *beat-level F-measure* which follows the benchmark F-measure for beat and downbeat tracking [30] with a time tolerance of $\pm$70ms. In both cases, a true positive means a predicted beat is in the ground truth; a false positive means a predicted beat is not in the ground truth; and a false negative means a ground truth beat is missing in prediction.

For both F-measures, we report the precision p, recall r and F-score f. Similar definitions are used for downbeats.

3.2.2 Performance MIDI-to-Score evaluation

We use the MV2H metric [26] to evaluate the system performance on PM2S conversion. The metric covers five sub-metrics including multi-pitch detection (F_p), voice separation (F_{vo}), metrical alignment (F_{me}), note value detection (F_{va}), and harmonic analysis (F_{ha}). The final accuracy F is the average of all the sub-metrics.

3.3 Comparative experiments

Among the subtasks in PM2S, rhythm quantisation is a crucial and difficult part. It is also highly related to the estimation of time signatures, musical onsets, and note values. In our proposed rhythm quantisation method which combines neural beat tracking and musical onset time estimation, we consider beat tracking as a more important step since it works on drawing the skeleton of the metrical grid.

Thus, for the first part of our experiments, we investigate different input data configurations and data augmentation methods for the beat tracking part of our proposed method. To get rid of influences from out-of-note beat prediction, we use the note-level F-measure in our comparison. After that, we validate our beat tracking method on

beat-level F-measure combining out-of-note beat prediction in comparison with a baseline beat tracking model.

3.3.1 Input data encoding

Given the input data in note sequences, we consider the following ways of encoding the input elements:

- Pitches:
  - M: 128-dimensional one-hot vectors in MIDI pitch numbers;
  - C: 12-dimensional one-hot vectors in octave values.
- Onset times:
  - AR: the raw value in seconds;
  - AO: one-hot vectors quantised by 10ms resolution;
  - SR: onset time shift in seconds compared to the previous note onset ($o_i^{shift} = o_i - o_{i-1}$ for $i > 0$, $o_0^{shift} = 0$);
  - SO: one-hot onset time-shift quantised by 10ms resolution (with a maximum onset time shift of 4s, larger values are trimmed to 4s).
- Durations:
  - R: the raw values in seconds;
  - O: one-hot vectors quantised by 10ms resolution (similar to onset shift, large values are trimmed to 4s).
- Velocities are always normalised to 0-1.

For all possible input encoding combinations, we train and evaluate the in-note beat prediction part of our proposed model (excluding downbeat). The model learning rate is set to be 0.001. Since different input encodings can cause changes in the model convergence speed, we do not add learning rate decay in this comparison. The model is trained using a batch size of 32 over 4 GPUs with a dropout rate of 0.15. For each combination, we take the best model checkpoint on the validation set during training, and evaluate it over the test set. The performances of the model on all different input data encoding combinations are reported in Table 2.

From the results, we can see that in terms of pitch encoding, using the MIDI pitches tends to outperform using the chroma groups most of the time. For onsets, onset shift leads to better results than absolute onset across all encoding combinations. Using one-hot encoding for onset is better than using the raw values for most cases. 6 out of 8 encoding combinations result in better model performance with onsets encoded in one-hot format. However, an opposite preference is discovered for duration encoding, where duration in raw values ends up with better results for more cases.

Among all the 16 input data encoding combinations tested, the one with MIDI pitch, one-hot onset shift, and duration in raw value shows the best model performance. We use this input data encoding combination in our subsequent experiments.

Input encodings			Note-level F-measure		
Pitch	Onset	Duration	p	r	f
M	AR	R	86.7	77.7	79.9
M	AR	O	89.9	57.2	65.2
M	AO	R	82.1	78.3	79.3
M	AO	O	83.3	72.2	76.0
M	SR	R	89.2	89.4	87.8
M	SR	O	88.8	91.9	89.5
M	SO	R	91.2	94.3	**91.3**
M	SO	O	91.1	90.3	89.1
C	AR	R	86.7	68.7	73.6
C	AR	O	80.7	64.4	67.2
C	AO	R	82.8	76.2	78.4
C	AO	O	82.7	71.9	76.0
C	SR	R	90.4	88.0	87.3
C	SR	O	90.2	88.9	87.5
C	SO	R	89.9	91.3	89.1
C	SO	O	88.8	90.6	88.0

Table 2. Model performance on different input data encoding combinations.

Input feature omitted	p	r	f
Pitch	90.4	93.7	90.6
Onset	84.6	72.8	76.4
Duration	89.5	93.1	90.1
Velocity	89.5	93.9	90.6
Use all features	91.2	94.3	**91.3**
Augmentation method omitted	p	r	f
Pitch shift	92.0	92.0	90.6
Tempo change	91.7	89.5	89.7
Note removal	91.5	90.9	90.4
Extra note	91.2	94.3	91.3
Use all methods	92.9	93.7	**92.2**
No data augmentation	89.7	95.2	90.9

Table 3. Note-level F-measure results on the ablation studies.

3.3.2 Ablation studies

Using the best input encoding combination observed in the previous comparison, we run an ablation study on the input features and different data augmentation techniques. We use all four input features when exploring different data augmentation methods.

Table 3 shows the ablation study results. From the table, we see that all four input features are helpful in beat tracking, among which the onset feature is the most beneficial one. This is consistent with the fact that onsets can be the feature to carry most metrical information in music performances. People can usually infer beat times from drum beats without knowing other information such as pitch or duration. Pitch and velocity are less important but still make a difference. That may be because pitch carries some harmony information that helps beat prediction. Velocity can provide useful information as well since beats are more probable to be concurrent with heavy notes. Finally, duration is of certain importance, suggesting it carries more metrical information than pitch and velocity.

Models (output data)	f_b	f_{db}
Baseline (b, db)	66.9	57.6
Proposed (b, db)	85.7	63.3
Proposed (d, db, t)	**86.2**	**69.8**

Table 4. Beat-level F-measure results on the baseline and proposed models. b: beat, db: downbeat, t: tempo.

Methods	F_p	F_{vo}	F_{me}	F_{va}	F_{ha}	F
Finale	82.2	54.6	9.9	92.2	86.2	65.0
MuseScore	10.0	65.0	15.3	95.0	84.5	54.0
Proposed	**99.8**	**87.0**	**61.7**	**99.9**	**91.1**	**87.9**

Table 5. MV2H evaluation on performance MIDI-to-score conversion.

Results on different data augmentation methods suggest that all four proposed data augmentation methods improve performance, among which tempo change is the most beneficial one. Not performing data augmentation does not result in the lowest note-level F-score. However, its low precision rate indicates a limitation on predicting more false positives, which is discouraged since we will be adding out-of-note beats based on the predicted in-note beats. It is possible that we can add the false negatives back when predicting out-of-note beats, but we cannot remove the false positives.

3.3.3 Proposed model vs. baseline model

Using the best configurations in previous experiment results, we combine the out-of-note beat prediction step and evaluate our proposed model performance on beat tracking using beat-level F-measure. We compare our method with a baseline model that is similar to a state-of-the-art audio beat tracking model [31]. We retrain the baseline model using a pianoroll input replacing the original audio spectrogram input, where the pianoroll is calculated from the performance MIDI, and let the model predict beat and downbeat probabilities. The probabilities are then passed to a dynamic Bayesian network [32, 33] to get beats and downbeats in seconds.

Table 4 shows the comparative results between the baseline and our proposed model. Results suggest our proposed model largely outperforms the baseline when jointly trained with beats and downbeats. This is possibly because that the baseline model is a general purpose system designed to operate on richer content than piano music alone, and that our proposed model can better handle tempo changes. By adding tempo to the output data, the performance of our proposed model can be further improved, which suggests a benefit of joint learning.

3.4 System performance evaluation

Using the best configurations found in the comparative experiments, we train and evaluate our proposed CRNN model defined in Section 2.4 on PM2S conversion. We report the model performance on the MV2H metric described in Section 3.2.2.

When looking for other methods for comparison, we realise that there are some difficulties in comparing our system with existing academic works. Cogliati et al. [14] evaluated their method in a subjective way by inviting five music theory students to rate the transcribed musical scores. Works such as [5, 19, 34] do provide a combination of methods to achieve PM2S conversion, however, the system performance was reported on audio-to-score transcription. Thus, we compared our proposed method with two commercial software products Finale v27 [25] and MuseScore v3 [24] that can do PM2S conversion.

Results in Table 5 suggest that our proposed model outperforms MuseScore and Finale across all MV2H submetrics. A significantly better performance can be observed in the metrical alignment sub-metric F_{me} which is highly related to the rhythm quantisation step. By checking outputs generated from MuseScore, we found that its low performance on F_p is caused by time shifts introduced when quantising notes according to a constant tempo estimated over the whole music piece. Constant tempo estimation also caused its low performance reported on F_{me}. A similar limitation can be found in output scores from Finale. On the contrary, our proposed method tracked tempo changes during rhythm quantisation and preserved the expressiveness of music performance as much as possible. This not only benefits metrical alignment, but also results in high accuracies on F_p and F_{va}. Still, the rhythm quantisation performance (F_{me}) is far from satisfactory. Some typical errors include double/half tempo error and errors introduced by missing/extra beat predictions.

To provide a better understanding of the performance of our proposed method, we provide some example outputs from our model together with their performance MIDI recordings [1] .

4. CONCLUSION

We described our proposed method for rhythm quantisation by using a CRNN beat tracking model that predicts whether notes are at a beat position or not. We explored different input data encoding and data augmentation methods on the beat tracking model. We found that note onset time is the most important input feature in beat prediction and it is best to encode it into a one-hot onset-shift vector. Tempo change benefits most among the data augmentation methods explored. We validated our model's beat tracking ability in comparison with a pianoroll-input baseline model. In the end, we report the performance of our proposed system on PM2S conversion in comparison with MuseScore and Finale.

Possible next steps include investigating more powerful model architectures such as the Transformer [35], expanding the output data to generate a machine-readable score [21], probing our system's ability in dealing with more genre and instrumentation [19, 36, 37], and exploring our method's potential in automatic music transcription by combining multi-pitch detection [38, 39].

[1] Example outputs: `https://cheriell.github.io/research/PM2S`

5. ACKNOWLEDGEMENTS

L. Liu is a research student at the UKRI Centre for Doctoral Training in Artificial Intelligence and Music, supported jointly by the China Scholarship Council and Queen Mary University of London. The work of L. Liu is supported by The Alan Turing Institute through an Enrichment Scheme.

We would like to thank Huan Zhang, Changhong Wang and the reviewers for their valuable feedback to improve our work.

6. REFERENCES

[1] M. Good, "MusicXML: An internet-friendly format for sheet music," *XML Conference and Expo*, pp. 1–12, 2001. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.118.5431&rep=rep1&type=pdf

[2] H.-W. Nienhuys and J. Nieuwenhuizen, "Lilypond, a System for Automated Music Engraving," in *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, 2003, pp. 167–171.

[3] S. D. Reeves and T. T. P. Walsh, *Creative jazz improvisation*. Taylor & Francis, 2022.

[4] R. G. C. Carvalho and P. Smaragdis, "Towards End-to-End Polyphonic Music Transcription: Transforming Music Audio Directly to A Score," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, vol. 2017-Octob, 2017, pp. 151–155.

[5] E. Nakamura, E. Benetos, K. Yoshii, and S. Dixon, "Towards Complete Polyphonic Music Transcription: Integrating Multi-Pitch Detection and Rhythm Quantization," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2018-April. IEEE, 2018, pp. 101–105.

[6] L. Liu and E. Benetos, "From Audio to Music Notation," in *Handbook of Artificial Intelligence for Music*, E. R. Miranda, Ed. Springer, 2021, pp. 693–714.

[7] L. Liu, V. Morfi, and E. Benetos, "Joint Multi-pitch Detection and Score Transcription for Polyphonic Piano Music," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021.

[8] J. Huang, E. Benetos, and S. Ewert, "Improving Lyrics Alignment through Joint Pitch Detection," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*. Institute of Electrical and Electronics Engineers (IEEE), 2 2022, pp. 451–455. [Online]. Available: https://arxiv.org/abs/2202.01646v1

[9] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a Multitrack Classical Music Performance Dataset for Multimodal Music Analysis: Challenges, Insights, and Applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, 2019.

[10] P. Desain and H. Honing, "The Quantization of Musical Time: A Connectionist Approach," *Computer Music Journal*, vol. 13, no. 3, pp. 56–66, 1989.

[11] D. Temperley and D. Sleator, "Modeling meter and harmony: A preference-rule approach," *Computer Music Journal*, vol. 23, no. 1, pp. 10–27, 1999.

[12] C. Raphael, "Automated Rhythm Transcription," in *ISMIR*, 2001, pp. 99–107.

[13] H. Takeda, N. Saito, T. Otsuki, M. Nakai, H. Shimodaira, and S. Sagayama, "Hidden Markov model for automatic transcription of MIDI signals," *Proceedings of 2002 IEEE Workshop on Multimedia Signal Processing, MMSP 2002*, pp. 428–431, 2002.

[14] A. Cogliati, D. Temperley, and Z. Duan, "Transcribing Human Piano Performance into Music Notation," in *ISMIR*, 2016.

[15] D. Temperley, "A unified probabilistic model for polyphonic music analysis," *Journal of New Music Research*, vol. 38, no. 1, pp. 3–18, 2009.

[16] E. Nakamura, K. Yoshii, and S. Sagayama, "Rhythm Transcription of Polyphonic Piano Music Based on Merged-Output HMM for Multiple Voices," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 25, no. 4, pp. 794–806, 2017.

[17] E. Nakamura, K. Yoshii, and S. Dixon, "Note Value Recognition for Piano Transcription Using Markov Random Fields," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 25, no. 9, pp. 1542–1554, 2017.

[18] A. McLeod and M. Steedman, "Meter detection and alignment of MIDI performance," *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, pp. 113–119, 2018.

[19] K. Shibata, E. Nakamura, and K. Yoshii, "Non-Local Musical Statistics as Guides for Audio-to-Score Piano Transcription," *arXiv preprint arXiv:2008.12710*, 2020. [Online]. Available: http://arxiv.org/abs/2008.12710

[20] Y. Hiramatsu, E. Nakamura, and K. Yoshii, "Joint Estimation of Note Values and Voices for Audio-to-Score Piano Transcription," in *ISMIR*. International Society for Music Information Retrieval Conference, 2021, pp. 278–284.

[21] M. Suzuki, "Score Transformer: Generating Musical Score from Note-level Representation," in *ACM International Conference Proceeding Series*, vol. 1, 2021.

[22] V. Emiya, R. Badeau, and B. David, "Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 18, no. 6, pp. 1643–1654, 2010.

[23] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP: A Dataset of Aligned Scores and Performances for Piano Transcription," in *ISMIR, International Society for Music Information Retrieval Conference*, 2020.

[24] "MuseScore." [Online]. Available: https://musescore.org

[25] "Finale music notation software." [Online]. Available: https://www.finalemusic.com

[26] A. Mcleod and M. Steedman, "Evaluating Automatic Polyphonic Music Transcription," in *ISMIR, International Society for Music Information Retrieval Conference*, 2018, pp. 42–49. [Online]. Available: https://www.github.com/apmcleod/MV2H.

[27] A. Ycart and E. Benetos, "A-MAPS: Augmented MAPS Dataset with Rhythm and Key Annotations," in *ISMIR, International Society for Music Information Retrieval Conference, Late-Breaking Demo*, 2018.

[28] "Classical Piano-Midi Database." [Online]. Available: http://www.piano-midi.de

[29] L. Liu, V. Morfi, and E. Benetos, "ACPAS: A Dataset of Aligned Classical Piano Audio And scores for Audio-To-Score Transcription," in *ISMIR Late-Breaking Demo*, 2021, pp. 2–4.

[30] M. E. P. Davies, N. Degara, and M. D. Plumbley, "Evaluation Methods for Musical Audio Beat Tracking Algorithms," *Centre for Digital Music, Queen Mary University of London, Tech. Rep.*, vol. C4DM-TR-09, no. October, p. 17, 2009.

[31] S. Böck and M. E. P. Davies, "Deconstruct, Analyse, Reconstruct: How To Improve Tempo, Beat, and Downbeat Estimation," in *ISMIR, International Society for Music Information Retrieval Conference*, 2020.

[32] S. Böck, F. Krebs, and G. Widmer, "A multi-model approach to beat tracking considering heterogeneous music styles," *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014*, pp. 603–608, 2014.

[33] F. Krebs, S. Böck, and G. Widmer, "An efficient state-space model for joint tempo and meter tracking," *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015*, pp. 72–78, 2015.

[34] A. Mcleod, "Evaluating Non-Aligned Musical Score Transcriptions with MV2H," in *ISMIR, International Society for Music Information Retrieval Conference, Late-Breaking Demo*, 2019.

[35] Y.-H. Chou, I.-C. Chen, C.-J. Chang, J. Ching, and Y.-H. Yang, "MidiBERT-Piano: Large-scale Pre-training for Symbolic Music Understanding," *arXiv preprint arXiv: 2107.05223*, 7 2021. [Online]. Available: https://arxiv.org/abs/2107.05223v1

[36] K. Tanaka, T. Nakatsuka, R. Nishikimi, K. Yoshii, and S. Morishima, "Multi-instrument Music Transcription Based on Deep Spherical Clustering of Spectrograms and Pitchgrams," in *ISMIR, International Society for Music Information Retrieval Conference*, 2020, pp. 327–334.

[37] Y. T. Wu, B. Chen, and L. Su, "Multi-Instrument Automatic Music Transcription with Self-Attention-Based Instance Segmentation," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 28, pp. 2796–2809, 2020.

[38] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution Piano Transcription with Pedals by Regressing Onsets and Offsets Times," *arXiv preprint arXiv:2010.01815*, pp. 1–10, 2020. [Online]. Available: http://arxiv.org/abs/2010.01815

[39] R. M. Bittner, J. J. Bosch, D. Rubinstein, G. Meseguer-Brocal, and S. Ewert, "A Lightweight Instrument-Agnostic Model for Polyphonic Note Transcription and Multipitch Estimation," in *ICASSP*, 2022, pp. 781–785.

SYMBOLIC MUSIC LOOP GENERATION WITH NEURAL DISCRETE REPRESENTATIONS

Sangjun Han[1], **Hyeongrae Ihm**[1], **Moontae Lee**[1,2], **Woohyung Lim**[1]
[1] LG AI Research, [2] University of Illinois at Chicago
{sj.han, hrim, moontae.lee, w.lim}@lgresearch.ai

ABSTRACT

Since most of music has repetitive structures from motifs to phrases, repeating musical ideas can be a basic operation for music composition. The basic block that we focus on is conceptualized as loops which are essential ingredients of music. Furthermore, meaningful note patterns can be formed in a finite space, so it is sufficient to represent them with combinations of discrete symbols as done in other domains. In this work, we propose symbolic music loop generation via learning discrete representations. We first extract loops from MIDI datasets using a loop detector and then learn an autoregressive model trained by discrete latent codes of the extracted loops. We show that our model outperforms well-known music generative models in terms of both fidelity and diversity, evaluating on random space. Our code and supplementary materials are available at https://github.com/sjhan91/Loop_VQVAE_Official.

1. INTRODUCTION

With the advance of generative models, many studies are trying to model sequential data such as language and speech. Music can also benefit from their previous works since it consists of a sequence of multiple notes to represent the composer's intention. Through the advancement, individuals can imitate the musical inspiration of artists without musical expertise.

Several works related to music generation have focused on generating long sequences by utilizing the expressive power of Transformer [1-4]. It is a promising approach because the Transformer with hierarchical layers can learn various types of repetitive structures on its self-attentions. However, it still has limitations, derived from the error accumulation and rhythmic irregularity, to achieve the ultimate goal which is to compose full-length music [1, 2]. To tackle that problem, we explicitly utilize recurrence properties in music, generating short and fixed-length music phrases that can be used as basic patterns of music.

Repeating musical ideas is a basic operation for music composition. This operation conceptualizes the loop, an essential ingredient for creating remixes or mash-ups [5]. With the concept, we can simplify the generation task into generating one distinctive pattern which consists only of a few bars. For loop extraction, previous works have attempted to detect autocorrelated peaks (it relies on an assumption that it is sufficient to detect the starting point of phrases) [5, 6], but we apply the knowledge of overall loop structures, obtained from a public audio dataset, to the MIDI domain.

Another intuition is that the musical ideas can be formed in combinations of finite symbols. It is known that learning discrete latent codes is sufficient to represent the continuous world since many modalities consist of sequences of symbols [7]. For example, objects in vision, words in language, and phonemes in speech may be candidates of symbols. Also, compressing raw data into discrete semantic units makes an autoregressive model easy to train by capturing long-range data dependency [8]. If we regard consecutive notes as symbols, it is natural to adopt discrete representation as the basis of our autoregressive generator. This process can emphasize pattern to pattern structures sacrificing the minimal loss of note details.

In contrast to using Inception Score (IS) and Fréchet Inception Distance (FID) for computer vision [9, 10], no consensus has been made for evaluating generated music. This is because the absence of pre-trained feature extractors prohibits the comparison of true and generated samples on feature space. Recently, Naeem has shown that random embedding can also be effective when the target distribution is far from pre-trained model statistics [11]. Using the random initialized networks, we evaluate our generative models on sample fidelity and diversity as firstly suggested in [12].

In this work, we propose symbolic music loop generation via learning discrete representations. It involves two main processes; 1) *loop extraction* from MIDI datasets using a loop detector and 2) *loop generation* from an autoregressive model trained by discrete latent codes of the extracted loops. The outputs of the generative model are loops consisting of 8 bars, which can be repeated seamlessly. Since we aim to generate polyphony and multitrack sequences, the bass and drum are chosen for our experiments, which are fundamental components of melody and rhythm. Additionally, we adopt an evaluation protocol from [11] to measure two-dimensional score which stands for fidelity and diversity. Our contributions are summarized as follows;

- We propose the framework of symbolic music loop generation, which involves loop extraction, loop generation, and its evaluations.

© S. Han, H. Ihm, M. Lee, and W. Lim. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: S. Han, H. Ihm, M. Lee, and W. Lim, "Symbolic Music Loop Generation with Neural Discrete Representations", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.*, Bengaluru, India,

- For loop extraction, we design a structure-aware loop detector trained by external audio sources to extract loops of 8 bars from MIDI.
- For loop generation, we verify that an autoregressive model combined with discrete representations can generate plausible loop phrases which can be repeated.
- With randomly initialized networks for embedding, we evaluate sample quality in terms of fidelity and diversity.

2. RELATED WORK

We introduce several works related to the history of loop extraction and music generation, the effectiveness of discrete representations, and the development of evaluating generative models.

2.1 Loop Extraction

For symbolic music generation, some researchers have prepared their dataset by sliding a window with a stride of 1 bar [13, 14]. Although they have achieved good generative performance, this process does not consider relative positions within music, generating ambiguous phrases.

Some works have imposed structural constraints on music generation models [15, 16], or directly detect novel segments which are repetitive in time series [17, 18]. In the audio domain, there have been attempts to extract loops explicitly by capturing repeated phrases [5, 6]. They extract harmonic features such as chroma vectors or mel-frequency cepstrum and catch autocorrelation peaks to determine the starting point of loops. They also require a heuristic process to decide which features should be more weighted. In contrast, our loop detector works on a data-driven approach, so it does not require a manual process such as feature extraction and weighting strategies.

A recent work for drum loop generation has informed the availability of a human-created loop dataset from Looperman (https://www.looperman.com/) [19]. Although it consists of audio sources with various instruments and genres, we suggest a promising approach to combine them with Lakh MIDI Dataset (LMD) [20]. Concretely, we extract domain-invariant loop structures from Looperman and train a loop detector using them to extract loops from LMD.

2.2 Symbolic Music Generation

To make MIDI available in machine learning, two representation methods are prevalent; event-based representation and time-grid based representation [21]. Although the former can represent high time-resolution with a few event vectors, we choose the latter one to benefit from fixed-length and repetitive structures for music. There have been several works to deal with polyphony multitrack representation [1, 13, 22]. Especially for time-grid based representation, MuseGAN [22] has stacked five instruments each of which consists of 4 bars and 84 pitches. We follow their method while restricting to two instruments, the bass and drum.

2.3 Discrete Representations

As opposed to VAE [23] with continuous prior, Oord has proved that a finite set of latent codes is sufficient to reconstruct while it prevents posterior collapse [7]. Powerful autoregressive priors with the latent codes have shown promising performance not only in image generation [8] but also text2image [24], and video generation [25]. Loop generation can also benefit from discrete data compression by expressing long-range structural patterns.

2.4 Evaluation of Generative Models

Proper evaluation metrics for generative models are important to reduce human evaluation labor. Previous works of music generation have inspected model metrics (*e.g.,* log-likelihood) or musical metrics on data space to evaluate how much true and generated samples are similar [3, 22, 26]. Comparison on data space, however, is vulnerable to pixel by pixel phase difference, ignoring data semantics. Without available pre-trained networks, it has been reported that random embeddings are more robust for evaluation rather than using models trained by other data domains [11]. In this respect, we take randomly initialized networks as our feature extractor and evaluate our music samples on feature space.

The commonly used metrics in computer vision are IS and FID which compute a one-dimensional score. To distinguish fidelity and diversity from the score, Sajjadi has suggested precision and recall evaluating overlapped ratio between true and generated distribution [12]. After that, some works have proposed different ways of constructing data distribution using k-nearest neighbors (KNN) [11, 27]. We adopt KNN based evaluation protocol since it is not affected by its initialization and is robust to outliers.

3. PROPOSED METHOD

Our work starts with collecting MIDI loops using a structure-aware loop detector. Since we design the detector to take not the music itself but bar-to-bar structures, we can train it with a loop-labeled dataset from audio domain. After obtaining the MIDI loop set, we design a loop generator in two-stage; compressing data into discrete space and building an autoregressive model with them. Lastly, generated samples are evaluated on qualitative and quantitative metrics.

3.1 Data Preparation

3.1.1 Looperman Dataset

Looperman Dataset from the audio domain is collected and transformed for training our loop detector. We collect 1,000 loops of 8 bars from Looperman, a website allowing to upload and download free music loops. Formally, we denote one loop that is 1-D audio as $x_{WAV} \in \mathcal{R}$. The process of data transformation for the loop detector will be described in section 3.2.1.

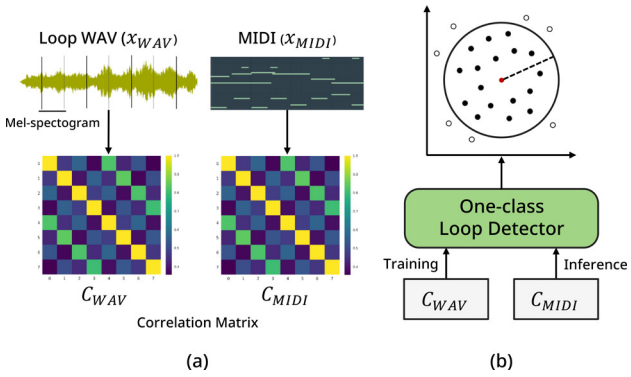


Figure 1. The process of loop extraction. (a) x_{WAV} and x_{MIDI} are transformed to each correlation matrix. (b) C_{WAV} are used to train the one-class loop detector and we extract loops from Lakh MIDI Dataset by forward passing C_{MIDI} to the detector.

3.1.2 Lakh MIDI Dataset

Lakh MIDI Dataset is a collection of MIDI files with various genres and tracks, so it is appropriate to conduct symbolic music experiments [20]. In this experiment, each note is quantized on the 16th note unit with binary representation so that 16 notes are placed in a bar. To verify the feasibility of multitrack polyphony generation, we extract two instruments; bass guitar (program=32~39) and drum (is_drum=True) which play crucial roles of melodic and rhythmic patterns in music. The bass pitches are clipped from C1 to B4 (48 pitches). If several pitches for the bass are played at the same time, we make only the lowest pitch alive for natural play. For the drum set, nine components (kick, snare, closed hi-hat, open hi-hat, low tom, mid tom, high tom, crash, and ride) are regarded as a standard set and the rest of the components are incorporated into the closest one or discarded. Formally, our pianoroll representation can be described as follows; $x_{MIDI} \in \{0,1\}^{(T\times B)\times P}$ where T is the number of time steps in a bar ($T = 16$), B is the number of bars ($B = 8$), and P is the number of pitches ($P = 57$). We collect 5,687,274 phrases of 8 bars by sliding a window with a stride of 1 bar, removing non-4/4 signature music. Using pretty_midi [28] and pypianoroll [29] in Python library, MIDI processing is conducted.

3.2 Loop Extraction

3.2.1 Data Transformation for the Loop Detector

We transform each the x_{WAV} and x_{MIDI} to $B \times B$ matrix indicating bar-to-bar correlation (Figure 1 (a)). For the x_{WAV}, we extract B mel-spectrograms each corresponding to B bars and compute a correlation matrix (C_{WAV}). For the x_{MIDI}, we compute normalized Hamming distance among bars and renormalize it to express correlation (C_{MIDI}). Only the upper triangle part of C is used. More details are described in Appendix B.3.

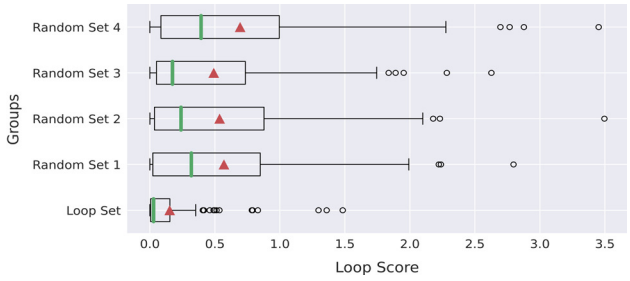


Figure 2. The evaluation of our loop detector. Green lines in the boxes indicate median values and red triangles for mean values.

3.2.2 Loop Extraction through the Loop Detector

Our loop detector is to classify loop and non-loop phrases, given only loop-labeled datasets. It is related to the problem of anomaly detection trained in an unsupervised way to construct normal data distribution. At inference time, outliers from the distribution are regarded as anomalous samples. Similarly, we treat C_{WAV} as normal samples (training set) and measure the likelihood of C_{MIDI} (test set) on C_{WAV} distribution. Among several ways, we choose One-Class Deep SVDD [30] as our loop detector, of which the training objective is

$$min\ \frac{1}{n}\sum_{i=1}^{n} ||f_w(x_i) - c||^2 + \lambda\Omega(W) \qquad (1)$$

where f_w is a neural network taking input x with learnable parameters W, n is the number of training samples, c is a center vector, and $\Omega(W)$ is a controllable regularizer. The objective can be thought as mapping all data samples close to center c, contracting a hypersphere. Initially, f_w is trained to reconstruct x with a decoder f_w^{-1} and center c is set to mean vectors acquired from f_w initial pass of the training data. f_w consists of 3 fully-connected layers with bias-off and LeakyReLU(0.1) to prevent hypersphere collapse as referred in [30]. During training, AdamW optimizer [31] is applied with cosine annealing from 1e-3 to 5e-6 for 1,000 epochs. At inference time, the loop score of C_{MIDI} can be obtained as

$$loop\ score = ||f_{w^*}(x) - c||^2 \qquad (2)$$

where w^* stands for optimized parameters of f (the process of loop extraction is illustrated on Figure 1 (b)). Samples with lower loop scores are considered close to the loop.

To evaluate the loop detector, we manually pick 100 loop samples from x_{MIDI} and compare them with 4 groups randomly picked (Figure 2). Although the randomized groups can contain subsets of loops, we can verify that the loop set group indicates the lowest loop score (0.153 ($\pm$0.282)). Also, paired t-test between the loop group and each other group shows the statistically significant difference with $p < 0.001$. When determining whether loop or not, we set a conservative threshold as positive one sigma of the loop score distribution (transformed by log to fit close to Gaussian distribution) from the training set. Consequently, we collect 751,935 x_{MIDI} to be used at loop generation stage.

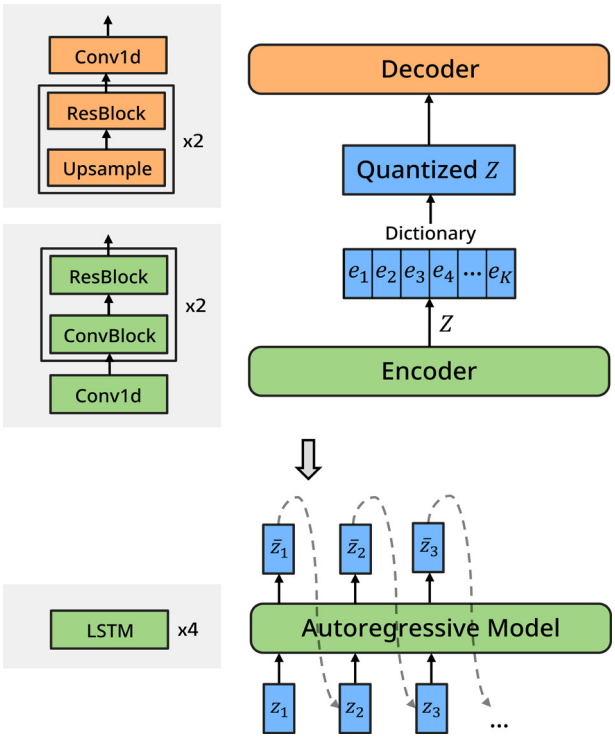


Figure 3. The process of loop generation. The upper denotes VQ-VAE and the bottom for LSTM based autoregressive model.

3.3 Loop Generation

3.3.1 Data Compression using VQ-VAE

VQ-VAE [7] maps a data sequence into discrete latent space and reconstructs it to the original data space. With an encoder $q_\phi$and a decoder p_θ, the objective becomes

$$max\ \mathbb{E}[log\ p_\theta(x|z)] - \beta||q_\phi(z|x) - sg[e]|| \quad (3)$$

where sg denotes a stop gradients operator for the dictionary embedding e. During forward pass, latent z from the encoder $q_\phi(z|x)$ is quantized to nearest embedding e. The second term above is responsible for the latent z not to diverge far from e. For every batch, the embedding dictionary is updated in the sense of centroids of K-means clustering.

Our VQ-VAE (Figure 3) encodes $\mathrm{x} = \{x_1, x_2, \cdots, x_{T\times B}\}$, $x_i \in \mathbb{R}^P$ into $\mathrm{z} = \{z_1, z_2, \cdots, z_S\}$, $z_i \in \mathbb{R}^D$ where S denotes the number of time steps ($S = 32$) in latent space and D denotes latent dimensions ($D = 16$). Starting from randomly initialized dictionary $\mathrm{e} = \{e_1, e_2, \cdots, e_K\}$, $e_i \in \mathbb{R}^D$ ($K = 512$), the latent z_i is mapped to the nearest embedding e_k where $k = argmin_j \|z_i - e_j\|$. After that, the embeddings are passed to the VQ-VAE decoder to reconstruct their original data. For the reconstruction objective, our data representation is regarded as multi-label for each time step (multiple 1s can exist on pitch P dimension at one time step), so cross-entropy loss with softmax is not suitable. Our objective for the reconstruction is described as

Model	Reconstruction Error
CNN-VAE	4.412e-3
VQ-VAE	6.643e-3

Table 1. The reconstruction errors. This is computed as the average of hamming distance between input and target samples of the validation set.

$$min\ -\frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} L\left(\sigma(o_{ij}), y_{ij}\right) \quad (4)$$

where L denotes binary cross-entropy, σ is sigmoid function, o is outputs of the VQ-VAE with m notes, and y for ground truth. When $\sigma(o_{ij}) \geq 0.5$, it predicts label as 1, otherwise 0.

The details of our VQ-VAE architectures (Figure 3) are described in Appendix B.4. Empirically, we have verified that a high compression ratio, especially along S dimension, severely deteriorates the reconstruction task of the VQ-VAE. The value of S has been determined to consider both the compression ratio and reconstruction quality. AdamW optimizer and cosine annealing from 1e-3 to 5e-6 is applied to the model training. As did in [32], random restarting is applied to less referenced e_i, replacing them with one of batch samples. After training, we forward pass the training set to obtain quantized embeddings z.

3.3.2 Generation through an Autoregressive Model

We design an autoregressive model $\prod_{i=0}^{S} p_\theta(z_i\,|z_{<i})$ over the quantized embeddings to generate unseen samples. The quantized indices k are used as inputs of 4-layers LSTM with an embedding layer. For each time step, softmax output predicts next step index k (validation accuracy 76.651% in teacher forcing mode). z_0 is sampled from multinomial distribution $p(z_0)$ of the training set. When sampling unseen data in full sampling mode, temperatures in softmax enable us to control sample diversity (temperature=0.7). We compare several sampling methods such as temperature sampling, top-k sampling [33], and nucleus sampling [34]. The generated indices are decoded by the VQ-VAE's decoder.

4. QUANTITATIVE EVALUATION

For quantitative evaluation of generated samples, we address three concepts; 1) *model metric* related to evaluating the capacity of generative models, 2) *musical style* related to measuring how much our intended properties of the loop are involved in generated samples, and 3) *similarity metrics* related to how much generated samples are involved in the training set on feature space (or vice versa).

4.1 Model Metric

Reconstruction Error: For VAE, the reconstruction error is part of the objective function that indicates how well the

Model	LS	UP	ND	P	R	D	C
Training Set	6.806e-3	5.769	14.383	-	-	-	-
CNN-VAE	3.496e-1	5.614	12.648	0.642±0.033	0.625±0.021	0.617±0.076	0.746 ±0.024
Music Transformer	7.200e-1	4.127	11.290	0.546±0.077	0.359±0.113	0.687±0.174	0.408±0.064
MuseGAN	2.307e-1	**5.790**	14.011	0.641±0.013	**0.689±0.012**	0.673±0.045	0.842±0.013
VQ-VAE+LSTM (temperature sampling)	2.275e-1	5.079	14.289	0.768±0.013	0.655±0.022	1.263±0.047	0.949±0.002
VQ-VAE+LSTM (top-k sampling=30)	**1.978e-1**	5.044	14.320	0.779±0.015	0.636±0.015	1.328±0.072	**0.952±0.004**
VQ-VAE+LSTM (top-p sampling=0.08)	2.037e-1	5.042	**14.341**	**0.783±0.017**	0.638±0.029	**1.337±0.075**	0.950±0.005

Table 2. The result table indicates all metrics explained at section 4.2 and 4.3. We compute P, R, D, and C from 10 different networks, so we denote mean values with standard deviations. Best values are marked in bold font.

Model	F_1 score (P & R)	F_1 score (D & C)
CNN-VAE	0.633	0.675
Music Transformer	0.433	0.512
MuseGAN	0.664	0.748
VQ-VAE+LSTM (temperature sampling)	**0.707**	1.084
VQ-VAE+LSTM (top-k sampling=30)	0.700	1.109
VQ-VAE+LSTM (top-p sampling=0.08)	0.703	**1.111**

Table 3. F1 score from Table 2 results.

model decodes its latent features to the original data. However, perfect satisfaction with the objective does not guarantee to generate high-quality samples (empirically, we have verified that original VAE has produced many noisy samples even after achieving the minimal reconstruction error when forcing Kullback-Leibler divergence term to 0).

4.2 Musical Style

The investigation of used harmonics and rhythm patterns indicates the musical style of our generated samples.

Loop Score (LS): Using the trained loop detector, we can evaluate how much generated samples are close to the loop.

Unique Pitch (UP): It computes the average number of used pitches per bar [22]. It reflects harmonic components on data space. It is desirable for generated samples to follow UP values of the training set.

Note Density (ND): It computes the average number of notes played per bar considering all instruments [26]. It reflects rhythmic components on data space. It is desirable for generated samples to follow ND values of the training set.

4.3 Similarity Metrics

Evaluating generated music samples on data space considers only musical features that humans can perceive. Here, we measure similarity of true and generated samples on latent space while indicating the sample fidelity and diversity.

Precision & Recall (P & R): The fidelity of generative models can be evaluated on precision which measures how much generated distributions are involved in true distributions. Likewise, the diversity can be realized by recall which measures how much true distributions are involved in generated distributions [12].

Kynkäänniemi have proposed improved P & R that count the presence of samples on the overlapped data manifold constructed by KNN [27]. In the case of precision, the data manifold is constructed from multiple spheres whose center is determined by true samples and whose radius is the distance between the true samples and their k-th nearest neighbors. All operations in the P & R should be conducted on latent space, so we use simple CNN networks initialized randomly for embedding [11]. For both P & R, we fix *k* of the KNN to 5 and get the average of the metrics from 10 different networks. To avoid extensive computation of the KNN, we use 10,000 samples for each network.

Density and Coverage (D & C): P & R are vulnerable to outliers overestimating data manifold. To remedy this, [11] have counted the average number of overlapped samples on a sphere. The concept and process of D & C are similar with P & R, except that the density can be greater than 1.

5. EXPERIMENTS

We evaluate our model by comparing it to 1) *training set*, 2) *CNN-VAE* similar structure with our VQ-VAE, 3) *Music Transformer* [3], and 4) *MuseGAN* [22]. All models have generated loop samples as many as the training set. Additionally, we apply several sampling methods for the VQ-VAE+LSTM and compare them. The experiment details of the baselines are explained in Appendix A.

5.1 Quantitative Evaluation Results

5.1.1 Model Metric Results

Due to the finite latent codes, our VQ-VAE is a little worse than the CNN-VAE for the reconstruction task (Table 1).

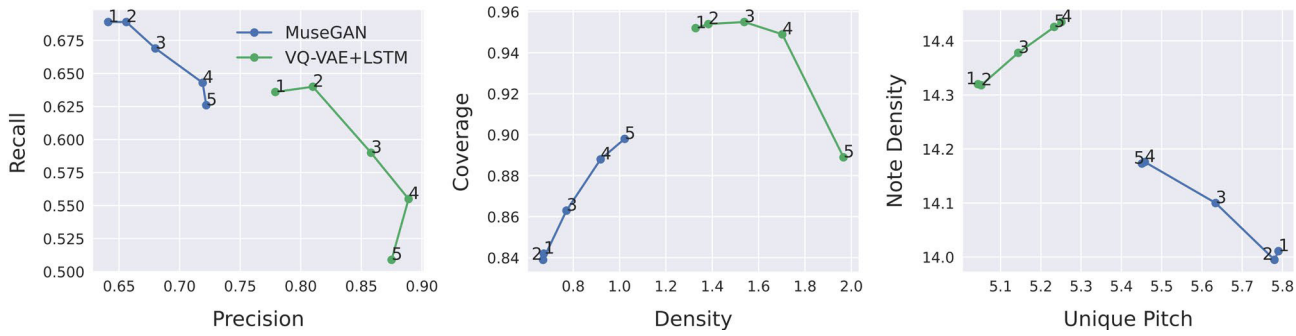


Figure 4. Rejection sampling results for the MuseGAN and VQ-VAE+LSTM. Each numbering markers indicates the various rejection rates (getting stricter from 1 to 5). The rates correspond to {no apply, 1, 0.1, 0.01, 0.001}.

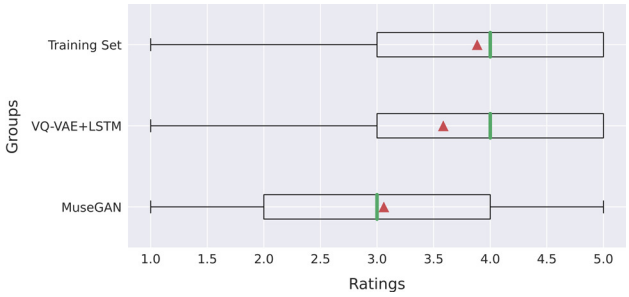


Figure 5. Human listening test. Green lines in the boxes indicate median values and red triangles for mean values.

As referred in [7], we have observed that β in the VQ-VAE objectives did not affect much to the task ($\beta = 0.25$).

5.1.2 Musical Metric Results

Our proposed model achieves the highest performance in terms of LS and ND (Table 2). The reason for poor performance in UP may be related to the VQ-VAE's reconstruction performance. Nevertheless, they can generate highly structural and rhythmic samples in a holistic view. Note that the Music Transformer fails to preserve the loop properties in its generated samples.

5.1.3 Similarity Metric Results

Except for the recall, our model achieves much better scores in terms of precision, density and coverage (Table 2). In MuseGAN, the recall may be overestimated by generated outliers. Depending on sampling methods and their parameters for full sampling mode, we can observe that there is a trade-off between fidelity and diversity. Even with the random embeddings, the all metrics show consistent performance (low standard deviations). A comprehensive evaluation (F_1 score) can be found in Table 3.

5.2 Rejection Sampling

It is promising that the loop detector can be used to control the trade-off between fidelity and diversity of generative models. This concept, rejection sampling, is to reject generated samples which do not meet our conditions. (loop scores in this context) [8]. If setting the rejection rate strictly, we can obtain music samples which are closer to the loop. For the experiments, we choose high-scored models that are MuseGAN and VQ-VAE+LSTM with top-k sampling. Figure 4 shows the P & R, D & C, and UP & ND results for various rejection rates. In P & R, the two models show similar trends while applying more strict rejection rates, but opposite trends for other metrics. Contrary to our assumption, both D & C of MuseGAN increase as we apply the stricter loop detector. It rather disproves that MuseGAN produces many outliers, so the loop detector may help to increase sample diversity close to the true distribution. In terms of ND, both models indicate minimal effect with the rejection sampling. However, they show contradiction about the aspect of UP changes.

5.3 Human Listening Test

We conduct a listening test for 20 people. We select the training set as a baseline and compare loop samples from two generative models (MuseGAN and VQ-VAE+LSTM with top-k sampling). Participants are asked to listen 10 samples for each group (total 30 samples) and evaluate how much the sample sounds like the loop music (which can be repeated seamlessly) on a Likert scale.

As Figure 5, the training set group achieves the highest ratings with an average rating of 3.885 ($\pm$1.045). For the generative models, VQ-VAE+LSTM achieves the highest average rating 3.585 ($\pm$1.141). It seems that the participants have felt them closer to the real music since loops from discrete representations are repetitive and structural (MuseGAN 3.060 ($\pm$1.172)). Additionally, we carry out Kruskal-Wallis H test which is non-parametric one-way ANOVA. The test shows a statistically significant difference among the test groups with $H = 49.811$, $p < 0.001$.

6. CONCLUSION

We leverage recurring nature of music by adopting the concept of the loop. To fulfill our objective, we address two processes; *loop extraction* and *loop generation*. First, we design a loop detector trained by a loop audio dataset to prepare loops from MIDI. Second, we adopt the two-stage generative approach, compressing data into discrete representations and designing an autoregressive model. Even without pre-trained feature extractors, we can evaluate our generative models on measuring fidelity and diversity. It is observed that our model outperforms well-known generative model for loop generation.

7. REFERENCES

[1] Yi Ren, Jinzheng He, Xu Tan, Tao Qin, Zhou Zhao, and Tie-Yan Liu, "PopMAG: Pop Music Accompaniment Generation," in *Proc. of the 28th ACM International Conference on Multimedia*, 2020, pp. 1198–1206.

[2] Yu-Siang Huang and Yi-Hsuan Yang, "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions," in *Proc. of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[3] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck, "Music Transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[4] Christine Payne, "MuseNet," *OpenAI*, openai.com/blog/musenet, 2019.

[5] Bee Suan Ong, and Sebastian Streich, "Music Loop Extraction from Digital Audio Signals," in *Proc. of the 2008 IEEE International Conference on Multimedia and Expo*, IEEE, 2008, pp. 681–684.

[6] Zhengshan Shi, and Gautham J. Mysore, "LoopMaker: Automatic Creation of Music Loops from Pre-recorded Music," in *Proc. of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–6.

[7] Aaron Van Den Oord, Oriol Vinyals, and Koray Kavukcuoglu, "Neural Discrete Representation Learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 6306–6315.

[8] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals, "Generating Diverse High-Fidelity Images with VQ-VAE-2," in *Advances in Neural Information Processing Systems*, 2019, pp. 7989–7999.

[9] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, "Improved Techniques for Training GANs," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[10] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter, "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6626–6637.

[11] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo, "Reliable Fidelity and Diversity Metrics for Generative Models," in *Proc. of the 37th International Conference on Machine Learning*, PMLR, 2020, pp. 7176–7185.

[12] Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly, "Assessing Generative Models via Precision and Recall," in *Advances in Neural Information Processing Systems*, 2018, pp. 5228–5237.

[13] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck, "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music," in *Proc. of the 35th International Conference on Machine Learning*, PMLR, 2018, pp. 4364–4373.

[14] Gautam Mittal, Jesse Engel, Curtis Hawthorne, and Ian Simon, "Symbolic Music Generation with Diffusion Models," *Proc. of the 22nd International Society for Music Information Retrieval Conference*, 2021, pp. 468–475.

[15] François Pachet, Alexandre Papadopoulos, and Pierre Roy, "Sampling Variations of Sequences for Structured Music Generation," in *Proc. of the 18th International Society for Music Information Retrieval Conference*, 2017, pp. 167–173.

[16] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever, "Generating Long Sequences with Sparse Transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[17] Jonathan Foote, "Automatic Audio Segmentation using a Measure of Audio Novelty," in *Proc. of the IEEE International Conference on Multimedia and Expo*, IEEE, 2000, pp. 452–455.

[18] Joan Serra, Meinard Müller, Peter Grosche, and Josep Lluis Arcos, "Unsupervised Detection of Music Boundaries by Time Series Structure Features," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2012, pp. 1613–1619.

[19] Tun-Min Hung, Bo-Yu Chen, Yen-Tung Yeh, and Yi-Hsuan Yang, "A Benchmarking Initiative for Audio-Domain Music Generation Using the Freesound Loop Dataset," in *Proc. of the 22nd International Society for Music Information Retrieval Conference*, 2021, pp 310–317.

[20] Colin Raffel, "Learning-based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching," Ph.D. dissertation, Columbia University, 2016.

[21] Shulei Ji, Jing Luo, and Xinyu Yang, "A Comprehensive Survey on Deep Music Generation: Multi-level Representations, Algorithms, Evaluations, and Future Directions," *arXiv preprint arXiv:2011.06801*, 2020.

[22] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang, "MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment," in *Proc. of the AAAI Conference on Artificial Intelligence*, 2018, pp. 34–41.

[23] Diederik P. Kingma, and Max Welling, “Auto-Encoding Variational Bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

[24] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever, "Zero-Shot Text-to-Image Generation," in *Proc. of the 38th International Conference on Machine Learning*, PMLR, 2021, pp. 8821–8831.

[25] Jacob Walker, Ali Razavi, and Aäron van den Oord, "Predicting Video with VQVAE," *arXiv preprint arXiv:2103.01950*, 2021.

[26] Kristy Choi, Curtis Hawthorne, Ian Simon, Monica Dinculescu, and Jesse Engel, “Encoding Musical Style with Transformer Autoencoders,” in *Proc. Of the 37th International Conference on Machine Learning*, PMLR, 2020, pp. 1899–1908.

[27] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila, “Improved Precision and Recall Metric for Assessing Generative Models,” in *Advances in Neural Information Processing Systems*, 2019, pp. 3927–3936.

[28] Colin Raffel and Daniel PW Ellis, “Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi,” in *Proc. of the 15th International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, 2014.

[29] Hao-Wen Dong, Wen-Yi Hsiao, and Yi-Hsuan Yang, “Pypianoroll: Open Source Python Package for Handling Multitrack Pianoroll,” in *Proc. of the 19th International Society for Music Information Retrieval Conference Late Breaking and Demo Papers*, 2018.

[30] Lukas Ruff, Robert A. Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib A. Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft, “Deep One-Class Classification,” in *Proc. of the 35th International Conference on Machine Learning*, PMLR, 2018, pp. 4393–4402.

[31] Ilya Loshchilov and Frank Hutter, “Decoupled Weight Decay Regularization,” *arXiv preprint arXiv:1711.05101*, 2017.

[32] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever, “Jukebox: A Generative Model for Music,” *arXiv preprint arXiv:2005.00341*, 2020.

[33] Angela Fan, Mike Lewis, and Yann Dauphin, “Hierarchical Neural Story Generation,” *arXiv preprint arXiv:1805.04833*, 2018.

[34] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi, “The Curious Case of Neural Text Degeneration,” *arXiv preprint arXiv:1904.09751*, 2019.

AUTOMATIC MUSIC MIXING WITH DEEP LEARNING AND OUT-OF-DOMAIN DATA

Marco A. Martínez-Ramírez[♭] **Wei-Hsiang Liao**[♭] **Giorgio Fabbro**[♯] **Stefan Uhlich**[♯]
Chihiro Nagashima[♭] **Yuki Mitsufuji**[♭]
[♭]Sony Group Corporation, Tokyo, Japan [♯]Sony Europe B.V., Stuttgart, Germany
marco.martinez@sony.com, yuhki.mitsufuji@sony.com

ABSTRACT

Music mixing traditionally involves recording instruments in the form of clean, individual tracks and blending them into a final mixture using audio effects and expert knowledge (e.g., a mixing engineer). The automation of music production tasks has become an emerging field in recent years, where rule-based methods and machine learning approaches have been explored. Nevertheless, the lack of dry or clean instrument recordings limits the performance of such models, which is still far from professional human-made mixes. We explore whether we can use out-of-domain data such as wet or processed multitrack music recordings and repurpose it to train supervised deep learning models that can bridge the current gap in automatic mixing quality. To achieve this we propose a novel data preprocessing method that allows the models to perform automatic music mixing. We also redesigned a listening test method for evaluating music mixing systems. We validate our results through such subjective tests using highly experienced mixing engineers as participants.

1. INTRODUCTION

Music mixing is a highly cross-adaptive transformation since the processing of an individual track depends on the content of all tracks involved. Apart from artistic considerations, it typically tries to solve the problem of unmasking by manipulating the dynamics, spatialisation, timbre or pitch of multitrack recordings [1]. This manipulation is achieved through a set of linear and nonlinear effects, which generally can be classified into five different classes: *gain, equalization (EQ), panning, dynamic range compression (DRC) and artificial reverberation* [2].

Several methods have been investigated to automatize this task [3]. For example, rule-based systems [4–6], cross-adaptive audio mixing effects [7, 8] and data-driven methods [9, 10]. A black-box approach is introduced in [9], where a *Wave-U-Net* is trained to perform automatic mixing of drums. Conversely, neural proxies of audio effects are used as a fixed signal processing path within a deep neural framework to perform automatic mixing of songs [10]. However, the lack of dry or unprocessed multitrack data has limited the performance of these deep learning approaches. Thus an unified approach has yet to be found that achieves results close to or superior to the quality of professional human-made mixes [11, 12]. It has been hypothesized that the bottleneck of performance can be resolved with a large enough dataset [10]. Nevertheless, collecting data is difficult, as it is unusual for musicians and record labels to provide multitrack dry recordings.

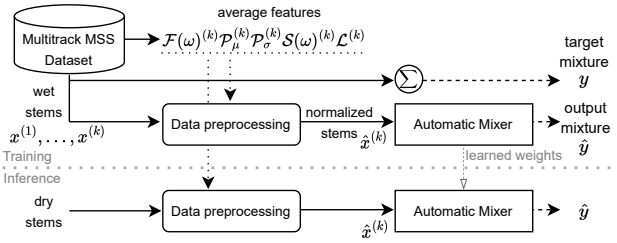


Figure 1: Our method uses data preprocessing by computing average features related to audio effects on an out-of-domain dataset (MSS data). This allows normalization of wet stems and supervised training of an automatic mixer. At inference, the same preprocessing is applied to dry data.

In this work, we consider the use of out-of-domain data in conjunction with a supervised deep learning approach to close such performance gap. To achieve this, we propose a novel method that performs a data normalization or augmentation procedure on each of the audio effect classes. Figure 1 depicts our method. We train deep neural networks to perform automatic loudness, EQ, panning, DRC and reverberation music mixing. Thus we present 1) a data preprocessing step that allows training with out-of-domain data, 2) a new deep learning architecture, 3) an exploration of stereo-invariant loss functions, 4) a design of a perceptual listening test targeting highly skilled professionals, and 5) listening test results showing that our approach is indistinguishable from professional human-made mixes. Audio samples and code can be found at https://marco-martinez-sony.github.io/FxNorm-automix/.

2. METHOD

2.1 Effect normalization and augmentation

One of the main challenges of deep learning models for automatic mixing is the lack of multitrack dry data. Since

© M. Martínez-Ramírez, W. Liao, G. Fabbro, S. Uhlich, C. Nagashima, and Y. Mitsufuji. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. Martínez-Ramírez, W. Liao, G. Fabbro, S. Uhlich, C. Nagashima, and Y. Mitsufuji, "Automatic music mixing with deep learning and out-of-domain data", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

collecting a large dry multitrack dataset is inherently difficult, we believe it is possible to reuse existing datasets, such as music source separation (MSS) data. Music source separation has been heavily researched in the last decade [13] and a significant effort has been put into collecting training data, such as the well-known MUSDB18 dataset [14]. However, direct application of such data is infeasible, since the mixture is a summation of the wet stems, that is, mixing effects have already been applied.

Several methods have been shown to be capable of reverse engineering the effect parameters [15–17], however such approaches require the original dry multitrack recordings and target mix. There are also data-driven methods for removing effects [18, 19], although they only apply to specific effects and are prone to adding sound artifacts.

Instead of removing audio effects, we propose to normalize each stem based on audio features related to each class of audio effects. In this way, different data features are scaled or transformed to make an equal contribution across the dataset [20]. We propose normalization schemes for loudness, EQ, panning, DRC and reverberation, thus ensuring that all stems have been normalized to the same range of audio features. During training, we expect the models to learn how to undo or denormalize the input stems and thus approximate the original mix. At inference, we also normalize the real multitrack dry data thus allowing the model to perform automatic music mixing. This Section introduces how each effect is normalized and Section 2.2 the full data preprocessing procedure.

Loudness—To normalize loudness, we independently compute the average loudness $\mathcal{L}^{(k)}$ for the stem type k as

$$\mathcal{L}^{(k)} = \frac{1}{N}\sum_{i=1}^{N} \text{LUFS}(x_i^{(k)}), \tag{1}$$

where LUFS is the integrated loudness level in dBFS in accordance to [21], x is the ith stem waveform of k type, e.g. k=vocals, and N is the total number of available songs. Then, based on $\mathcal{L}^{(k)}$, each stem is loudness normalized using [22].

Equalization—EQ normalization is based on the average frequency magnitude spectrum $\mathcal{F}^{(k)}$ as

$$\mathcal{F}(\omega)^{(k)} = \frac{1}{N}\sum_{i=1}^{N} \Gamma(\omega)_i^{(k)}, \quad \Gamma(\omega)^{(k)} = \frac{1}{M}\sum_{j=1}^{M} X(\omega)_j^{(k)}, \tag{2}$$

where $\Gamma^{(k)}$ corresponds to the individual stem mean frequency magnitude, ω is the frequency index, $X^{(k)}$ is the magnitude of the Short-Time Fourier Transform (STFT) of each $x^{(k)}$ and M its total number of frames.

We then proceed to normalize each stem by performing EQ matching with respect to $\mathcal{F}^{(k)}$. EQ matching typically involves finding the optimal filter settings by designing and applying a filter based on the difference between the target and input frequency magnitudes [23]. The differential frequency magnitude $\mathcal{F}(\omega)_{\text{diff}}^{(k)}$ is computed as

$$\mathcal{F}(\omega)_{\text{diff}}^{(k)} = 10^{\log_{10}(\mathcal{F}(\omega)^{(k)}) - \log_{10}(\Gamma(\omega)^{(k)})}. \tag{3}$$

$\mathcal{F}(\omega)_{\text{diff}}^{(k)}$ is further smoothed with a Savitzky-Golay filter [24]. An FIR filter is designed using the window method [25], and applied using forward-backward filtering [26] to have a zero-phase response. Since forward-back filtering squares the magnitude response, we apply a square root to $\mathcal{F}(\omega)_{\text{diff}}^{(k)}$ before designing the FIR filter.

Panning—The panning normalization is based on the Stereo Panning Spectrum [27, 28], where we use the average panning as a reference then re-pan accordingly.

We compute the left and right channel similarity measure $\Psi(\omega)$, to approximate the panning gains assigned during mixing. We focus only in amplitude panning thus we calculate $\Psi(\omega)$ using only frequency magnitude as

$$\Psi(\omega) = 2\frac{X(\omega)_{\text{L}} X(\omega)_{\text{R}}}{X(\omega)_{\text{L}}^2 + X(\omega)_{\text{R}}^2}, \tag{4}$$

where $X(\omega)_{\text{L}}$ and $X(\omega)_{\text{R}}$ correspond to the left and right channel STFT magnitudes. $\Psi(\omega)$ denotes whether a frequency bin ω_0 is panned to the center ($\Psi(\omega)_{\omega_0}$=1), or whether is panned to either side ($0 \leq \Psi(\omega)_{\omega_0} < 1$). The stereo side can be determined with the difference $\Delta(\omega)$ between partial similarities $\Psi(\omega)_{\text{L}}$ and $\Psi(\omega)_{\text{R}}$ as

$$\Psi(\omega)_{\text{L}} = \frac{X(\omega)_{\text{L}} X(\omega)_{\text{R}}}{X(\omega)_{\text{L}}^2}, \Psi(\omega)_{\text{R}} = \frac{X(\omega)_{\text{L}} X(\omega)_{\text{R}}}{X(\omega)_{\text{R}}^2}, \tag{5}$$

thus $\Delta(\omega) = \Psi(\omega)_{\text{L}} - \Psi(\omega)_{\text{R}}$, where $\Delta(\omega)>0$ and $\Delta(\omega)<0$ correspond to signals panned left or right, respectively. The panning gains, $\Phi(\omega)_{\text{L}}$ and $\Phi(\omega)_{\text{R}}$, then can be estimated by choosing a panning law to approximate the panning coefficient $\alpha(\omega)$. We empirically found that a linear panning law, $\Phi(\omega)_{\text{L}}$=1-$\alpha(\omega)$ and $\Phi(\omega)_{\text{R}}$=$\alpha(\omega)$, yields better approximations. Thus we approximate the panning gains based on $\Delta(\omega)$ and assuming $\Psi(\omega)$=2$\alpha(\omega)$. We compute the average similarity measure $\mathcal{S}(\omega)^{(k)}$ across all $\Psi(\omega)^{(k)}$ computed from N stems and their STFT frames M. $\mathcal{S}(\omega)^{(k)}$ is also further smoothed with [24].

For each individual stem, re-panning is implemented by 1) computing $\Psi(\omega)$ and $\Delta(\omega)$, 2) estimating $\Phi(\omega)_{\text{L}}$ and $\Phi(\omega)_{\text{R}}$, and 3) estimating the reference panning gains, $\hat{\Phi}(\omega)_{\text{L}}$ and $\hat{\Phi}(\omega)_{\text{R}}$, using $\mathcal{S}(\omega)^{(k)}$ and $\Delta(\omega)$. Finally, we calculate a gain factor based on the ratio of panning gains which we apply to $X(\omega)_{\text{L}}$ and $X(\omega)_{\text{R}}$ as

$$\hat{X}(\omega)_{\text{L}} = \frac{\hat{\Phi}(\omega)_{\text{L}}}{\Phi(\omega)_{\text{L}}} X(\omega)_{\text{L}}, \hat{X}(\omega)_{\text{R}} = \frac{\hat{\Phi}(\omega)_{\text{R}}}{\Phi(\omega)_{\text{R}}} X(\omega)_{\text{R}}. \tag{6}$$

Panning normalization is performed per frame and the normalized audio is obtained with the inverse STFT of $\hat{X}(\omega)_{\text{L}}$ and $\hat{X}(\omega)_{\text{R}}$ with their original channel phase.

Dynamic Range Compression—DRC usually alters the transients of the input [29], thus we base our DRC normalization on the onset peak levels which are linked to such transient modification [30].

We first perform onset detection based on the High Frequency Content (HFC) method [31, 32]. HFC emphasizes the energy variation that occurs in the upper part of the spectrum which is typical of onsets [33]. We then select the maximum peak for each of the detected onsets.

The average peak level $\mathcal{P}_{\mu}^{(k)}$ and peak level standard deviation $\mathcal{P}_{\sigma}^{(k)}$ are defined as $\frac{1}{N}\sum_{i}^{N}\mu_{i}^{(k)}$ and $\frac{1}{N}\sum_{i}^{N}\sigma_{i}^{(k)}$, respectively. Where $\mu^{(k)}$ and $\sigma^{(k)}$ are the channel mean peak level and standard deviation in dB, which for a robust estimate are calculated from the top 75th percentile of detected peaks.

DRC normalization consists in upper bounding the peak levels of the audio. Thus, if $\mu^{(k)} > (\mathcal{P}_{\mu}^{(k)} + \mathcal{P}_{\sigma}^{(k)})$, we apply a compressor to the input audio by performing an incremental grid search of the *ratio* and *threshold* parameters until $\mu^{(k)} < (\mathcal{P}_{\mu}^{(k)} + \mathcal{P}_{\sigma}^{(k)})$. *Threshold* is the level above which compression starts, and *ratio* determines the amount of compression [29]. This is done with fixed *attack* and *release* values, i.e. the start and stop timing settings.

Artificial Reverberation—Due to the inherent characteristics of reverberation [29], an attempt to similarly normalize this effect is not trivial. Blind estimation of reverberation features, such as reverberation time (RT) or direct-to-reverberant ratio, is an open research area in itself, and considering that most of the research has been done for speech signals [34–36], its application to music is beyond the scope of this paper.

We propose instead a data augmentation approach where we stochastically add reverberation to already reverberated stems. Reverberation is added using a mixing method called the 'Abbey Road reverb trick' [37], where a chain of EQ and reverb effects is applied as a send effect, i.e. a copy of the input is processed and added to the input. Naively adding reverberation directly to the input audio has the potential to clutter the low and low-mid frequencies. In this manner, the process of learning how much reverberation is required for a given mix is carried out by the network by learning to filter out the additional reverberation that is present in the input stems.

2.2 Data preprocessing

Since most mixing effects are interrelated, applying the normalization methods separately yields inaccurate results, e.g. EQ normalization modifies the dynamics of the audio, thus modifying DRC features. To minimize this, we perform effect preprocessing progressively and based in the following order: EQ, DRC, panning and loudness. Since EQ and DRC normalization are done on a per channel basis, panning normalization is applied after the above to avoid further modification of the stereo features.

Thus, for each stem type k, average feature computation corresponds to 1) calculate $\mathcal{F}(\omega)^{(k)}$ and EQ normalize, 2) calculate $\mathcal{P}_{\mu}^{(k)}$ and $\mathcal{P}_{\sigma}^{(k)}$ and DRC normalize, 3) calculate $\mathcal{S}(\omega)^{(k)}$ and panning normalize, and 4) calculate $\mathcal{L}^{(k)}$ and loudness normalize.

We apply our reverberation preprocessing method after all the average features have been calculated. Considering that this procedure is based on a stochastic data augmentation procedure, for consistency we treat the added reverberation as noise for the normalization pipeline. The final data preprocessing method corresponds to applying reverberation augmentation followed by EQ, DRC, panning, and loudness normalization methods.

2.3 Architecture

We propose a new architecture based on [38] and [39]. It operates in the time-domain and processes raw waveforms in a frame-wise manner. The model can be divided into three parts: adaptive front-end, latent-space mixer and synthesis back-end. The model is depicted in Figure 2 and its architecture is summarized in Table 1.

The **adaptive front-end** is exactly the same as in [38], with the only difference that now the input $\hat{x}$ corresponds to K stereo tracks of length A. In the front-end, time-domain convolutions are applied to the input audio in order to learn a latent representation $\boldsymbol{Z}$ and a filter bank which output feature map $\boldsymbol{X}_1$ corresponds to a frequency band decomposition of $\hat{x}$. The **latent-space mixer** corresponds to the temporal dilated convolutions (TCN) separator block of [39], and to improve long-term dependencies learning, the TCN is followed by stacked Bidirectional Long Short-Term Memory (BLSTM) layers. Contrary to [39], the objective of the mixer is not to learn a mask for source separation, but rather to learn a mixing mask $\hat{\boldsymbol{Z}}$.

The **synthesis back-end** uses $\hat{\boldsymbol{Z}}$ to modify $\boldsymbol{X}_1$ based on the given mixing task and its design is motivated by [38]. It upsamples the mixing mask using nearest neighbor interpolation and via an element-wise multiplication applies it to each filter-bank channel source of $\boldsymbol{X}_1$ at each time step. We hypothesize that this frequency-based transformation is akin to the model learning and applying a dynamic equalization effect [40] while also filtering out the extra reverberant content of the normalized input stems.

The resulting feature map $\boldsymbol{X}_4$ is further modified via a Squeeze-and-Excitation (SE) block [41] implemented as shown in [38]. The SE layers scale the channel-wise information of $\boldsymbol{X}_4$ by applying an adaptive gain $\boldsymbol{s}_c$, and consequently, learning a loudness gain and panning transformation for each filter bank channel. The modified frequency decomposition $\boldsymbol{X}_5$ is then reconstructed using a non-trainable transposed convolution as shown in [38]. Finally, the resulting $2K$ stereo tracks are channel-wise summed into a stereo output and a hyperbolic tangent is used to avoid clipping. All convolutions use a stride of 1 to avoid ringing and filtering artifacts [42]. BLSTMs and SE layers are applied to the filter dimension.

2.4 Loss function

Based on the stereo-invariant loss function introduced by [10], we explore two variations of such loss. First, due to the perceptual nature of the mixing task and motivated

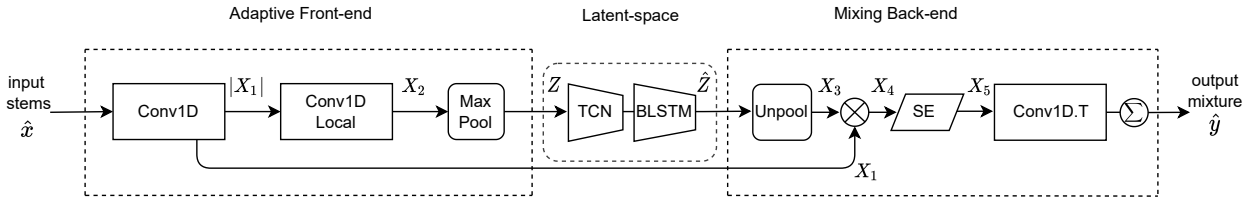


Figure 2: Block diagram of the proposed model

Table 1: Summarized architecture of the proposed model.

Layer	Output shape	Output
Input stems	$(2K, A)$	$\hat{\boldsymbol{x}}$
Conv1D	(N, A)	$\boldsymbol{X}_1$
Conv1D-Local	(N, A)	$\boldsymbol{X}_2$
MaxPooling	$(N, A/64)$	$\boldsymbol{Z}$
TCN	$(N, A/64)$	.
BLSTM	$(A/64, N)$	$\hat{\boldsymbol{Z}}$
Unpooling	(N, A)	$\boldsymbol{X}_3$
$\boldsymbol{X}_3 \times \boldsymbol{X}_1$	(N, A)	$\boldsymbol{X}_4$
SE (Abs)	(A, N)	.
SE (Global Avg)	$(1, N)$	.
SE (FC)	$(1, 16N)$	.
SE (FC)	$(1, N)$	$\boldsymbol{s}_c$
$\boldsymbol{X}_4 \times s_c$	(N, A)	$\boldsymbol{X}_5$
Conv1D.T	$(2K, A)$	.
Summation	$(2, A)$	$\hat{y}$

by [43], we apply A-weighting pre-emphasis and low-pass FIR filters (ρ) to the target and output audio frames y and $\hat{y}$, respectively. Then we compute the sum and difference signals y_{sum}=$\rho(y_{\text{L}})$+$\rho(y_{\text{R}})$ and y_{diff}=$\rho(y_{\text{L}})$-$\rho(y_{\text{R}})$.

The first loss follows closely [10] and is based on the Spectral Convergence (SC), magnitude-normalized Frobenius norm, and the L1-norm spectral log-magnitude (L1Log) as

$$\begin{aligned} L_{\text{a}} = l_{\text{SC}}(Y_{\text{sum}}, \hat{Y}_{\text{sum}}) + l_{\text{L1Log}}(Y_{\text{sum}}, \hat{Y}_{\text{sum}}) \\ + l_{\text{SC}}(Y_{\text{diff}}, \hat{Y}_{\text{diff}}) + l_{\text{L1Log}}(Y_{\text{diff}}, \hat{Y}_{\text{diff}}), \end{aligned} \tag{7}$$

where Y and $\hat{Y}$ are the respective 4096-Fast Fourier Transform (FFT) magnitude with a 25% hop size.

The second loss replaces the SC loss component with a less penalizing normalization such as the widely used L2-norm on the spectral magnitude (L2), defined as

$$\begin{aligned} L_{\text{b}} = l_{\text{L2}}(Y_{\text{sum}}, \hat{Y}_{\text{sum}}) + l_{\text{L1Log}}(Y_{\text{sum}}, \hat{Y}_{\text{sum}}) \\ + l_{\text{L2}}(Y_{\text{diff}}, \hat{Y}_{\text{diff}}) + l_{\text{L1Log}}(Y_{\text{diff}}, \hat{Y}_{\text{diff}}). \end{aligned} \tag{8}$$

We empirically found the perceptual-based pre-emphasis filter as vital when modeling a highly perceptual task such as music mixing. We also found an easier convergence during training when using a single frame-size loss rather than a multi-resolution magnitude loss.

3. EXPERIMENTS

3.1 Dataset

We first conduct experiments with a small dataset (S) which corresponds to the MUSDB18 dataset [14]. This dataset consists of wet stems for *vocals*, *drums*, *bass* and *other* (K=4), where the *mixture* is the summation of such stems. There are a total of 150 songs, of which 86 are used for training and 14 and 50 for validation and testing purposes, respectively.

We also use a private large dataset (L) for music separation which corresponds to 1,505 extra multitrack songs created in the same manner as MUSDB18. We incorporate the MUSDB18 training and validation sets into the large dataset, thus obtaining a total of 1,455+86=1,541 training songs and 50+14=64 validation songs. In both datasets, most of the songs correspond to mainstream western music, with rock and pop as the predominant genres.

To validate our method, we use a private set of 18 dry multitrack songs, where all songs have been produced by different musicians and mixing engineers. From this data we grouped the respective dry stems without applying any effect in the process.

3.2 Dataset preprocessing

We apply our preprocessing methods to both datasets independently. EQ preprocessing is computed with a 65,536 point STFT with hop size of 25% and a FIR filter of 1,001 taps. In order to avoid clipping, all audio stem channels are loudness normalized to -30 dBFS prior to the EQ feature computation and normalization.

For DRC preprocessing, to determine the timing settings of the compressor used during the normalization, we averaged the values found in mixing engineering best practices [8, 44]. *Attack* values correspond to 7.5, 10, 10 and 15 ms and *release* values to 400, 180, 500 and 666 ms, for *vocals*, *drums*, *bass* and *other*, respectively. For the incremental grid search we use *ratio* and *threshold* values within {4, 20} and {-40, -10} db respectively. For the HFC computation we use 128 mel bands of all stems with the exception of bass where we found better onset detection with 16 mel bands. Prior to the DRC preprocessing, all channels are peak normalized to -10 dB.

Panning preprocessing is done with a 2,048 point STFT with hop size of 50%. For all stems, frequency bins are re-panned until 16 kHz, with the exception of *drums*, which is re-panned until 16 kHz in order to avoid artifacts.

Reverberation augmentation is done by applying uniform random sampling on a set of 130 different impulse responses (IR) whose RT is within {2, 4} seconds (s). The

EQ used prior to the reverberation corresponds to a low-shelf and high-shelf with a fixed gain of -30 dB whose cut-off frequency is uniformly sampled within {500, 700} Hz and {7, 10} kHz, respectively. At inference, to simulate the reverberant characteristics of the training data, before applying the aforementioned augmentation, a "pre-reverb" is added in the same manner but from a different set of 400 IRs whose RT is within {1, 1.5}-*s*. A shorter RT is chosen, as high reverberation levels has been shown to have a more detrimental effect on subjective preference than low levels [45]. Reverberation augmentation is applied only to *vocals* and *other* stems. Reverberation and DRC effects are implemented using the Python package from [10].

3.3 Hyperparameters

The convolutional layers on the front-end have N=128 filters of size 64 and 128 respectively, and a 64-point window for max-pooling. In the mixer, the bottleneck layer has 256 channels, the skip connection paths have 64 channels, and the convolutional blocks have 128 channels of kernel size 3. We use 4 stacks of 6 convolutional layers with a dilation factor of 1,2,...,32. We stacked 3 BLSTMs with a hidden feature map of 64 channels. The FC layers in the SE block have 2048 ($16N$) and 128 channels respectively.

The input consists of $2K$ channels, each channel consisting of A=10-*s* audio frames at 44.1 kHz. The receptive field of the mixer is 505 samples, and taking into account the pooling operation, the overall network has a receptive field of 32,446 samples. Thus, the loss function is only calculated on a 8.52-*s* frame centered in the middle of the 10-*s* input. The network has in total 2.7M parameters. As a baseline, we use the modified *Wave-U-Net* (WUN) introduced by [9]. The same settings are used as in the original work which yields a network of 2.5M parameters.

3.4 Training

Regarding on-the-fly data augmentation, we apply a randomization of the order of the stereo channels, this is done consistently across the stems and the target mix. For the proposed network we use the pretraining from [38]. We train both models using a batch size of 4, an initial learning rate β=0.001, and the following learning rate schedule; β for 300 epochs, β/3 for 100 epochs, β/10 for 100 epochs, β/30 for 50 epochs, β/100 for 50 epochs and β/1000 for 25 epochs. An epoch consists of 1600 batches, L2 norm of the gradient is clipped by 0.2 and we use 10^{-7} for weight decay regularization. We select the model with the lowest validation loss.

4. RESULTS & ANALYSIS

We trained both networks with the preprocessed datasets S and L and the loss functions L_a and L_b, which yielded Ours-S-L$_a$, Ours-S-L$_b$, Ours-L-L$_a$ and Ours-L-L$_b$ for our proposed network, and WUN-S-L$_b$ and WUN-L-L$_b$ for Wave-U-Net. Convergence during training for WUN with L_a was unsuccessful, therefore it is excluded from the results.

	dry test set				MUSDB18 test set			
model	spectral	panning	dynamic	loudness	spectral	panning	dynamic	loudness
Normalized	<u>0.435</u>	0.593	0.168	0.633	<u>0.256</u>	0.783	0.109	0.643
WUN-S-L$_b$	0.527	0.215	0.082	0.094	**0.250**	0.250	<u>0.078</u>	0.097
Ours-S-L$_a$	0.547	0.201	0.062	<u>0.056</u>	0.299	<u>0.191</u>	0.086	0.095
Ours-S-L$_b$	**0.427**	0.207	0.063	0.061	0.276	0.212	0.085	0.084
WUN-L-L$_b$	0.551	<u>0.182</u>	0.066	**0.054**	0.279	0.195	**0.074**	**0.072**
Ours-L-L$_a$	0.590	0.191	**0.055**	0.091	0.312	0.593	0.168	0.105
Ours-L-L$_b$	0.519	**0.170**	<u>0.056</u>	0.061	0.276	**0.160**	0.084	<u>0.079</u>

Table 2: Objective metrics correspond to the average mean absolute percentage error by feature subgroup.

4.1 Quantitative evaluation

To measure how close the output mixes are to the reference mixes, we use the following audio features, spectral: centroid, bandwidth, contrast, flatness, and roll-off [46]; panning: the Panning Root Mean Square (RMS) [27]; dynamic: RMS level, dynamic spread and crest factor [47]; and LUFS loudness level [21]. All features are computed using a running mean of 0.5-*s* [27]. Table 2 shows the results for the dry test set and the MUSDB18 test set. As expected, the overall error values are larger for the dry test set. Also, the loudness, dynamics, and panning values show a closer match to the reference mix when compared to the Normalized mix, which is the sum of input stems after data preprocessing. Spectral values do not match in the same way, which could indicate that the generated mixes deviate in terms of EQ and reverberation.

4.2 Listening Test

We designed a the listening test using the Web Audio Evaluation Tool [48] and the APE interface [49]. The test is intended for professional mixing engineers only and we ask participants to rate the mixes based on Production Value, Clarity and Excitement as shown in [50]. Perceptual tests for mixing systems [9,10] often differ from MUSHRA [51] tests, as the reference sample is omitted in order to encourage a direct comparison between mixes. However, based on feedback from pilot tests, we decided to include the 4 dry stems in the test as references. In this way, the ratings may reflect the quality of transformation on each stem as applied by the mixing systems.

Fourteen participants with an average mixing engineering experience of 11.6 years took part in the test. In total there were six different songs and for each song six different mixes were presented. Each mix was 25-*s* taken from the chorus-to-verse transitions of the full mixes. The six mixes correspond to the Ours-S-L$_b$, Ours-L-L$_a$, Ours-L-L$_b$, WUN-S-L$_b$ and WUN-L-L$_b$ models plus a professional Human mix. Ours-S-L$_a$ was omitted from the test to allow more songs to be tested while avoiding listening fatigue [52]. A low-anchor was not used as it has been shown to compress the other ratings at the higher end [1]. All mixes were loudness normalized to -23 dBFS [53].

Figure 3 shows the results of the listening test and Table 3 shows the pairwise comparisons of mixes. For Production Value, there is no statistically significant difference between the Human mixtures and the models that were

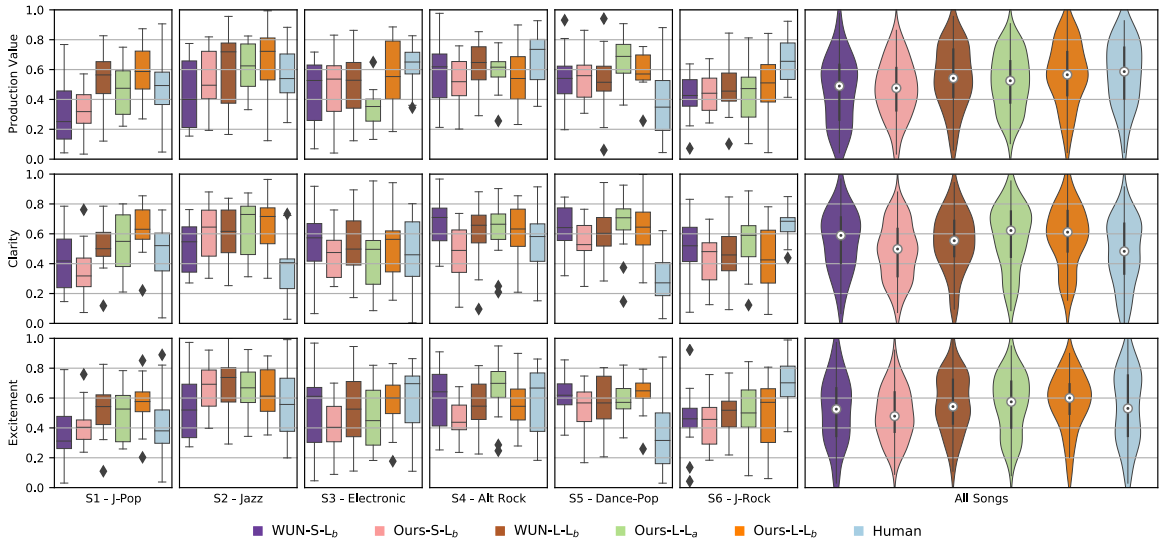


Figure 3: Listening test boxplots and violin plots for all individual six songs (S1 to S6) and all songs, respectively.

trained on the large dataset. For Clarity, Human mixes are rated lower than Ours-L-L$_a$, Ours-L-L$_b$ with a p-value of 0.008 and 0.013, respectively, indicating that our models generate mixtures with less masking. For Excitement, the null hypothesis is accepted, and therefore Human mixes are considered no different than any other mixture system. Although Ours-L-L$_b$ was among the best among all criteria, the null hypothesis is also accepted when compared to WUN-L-L$_b$. Thus, a further analysis is needed among both networks, e.g. in terms of long-term learned dependencies and ability to generate full-length coherent mixes.

In general, models trained with large datasets received the highest scores, which could confirm that lack of data has been the bottleneck of data-driven mixing systems. However, for Clarity and Excitement, the mixes by WUN-S-L$_b$ are not considered different from the rest, which might indicate that our data preprocessing method is both effective for small and large datasets.

It should be noted that, in general, none of the mixes is consistently considered as very good. This relates with other findings, where even commercial mixes made by renown engineers are not rated as high [1, 44]. This opens up a new research direction for evaluating mixing systems, as this type of test is too difficult for inexperienced participants, and when experienced participants do participate, they tend not to rate any mix as very good.

For individual songs, Human mixes have lower rates for Jazz and Dance-Pop. For the latter, low ratings on all criteria may be due to highly compressed drums. However, the high ratings in terms of Excitement and Production Value for the Human J-Rock mix might be due to hard-panned guitars. In contrast, the models are conservative when it comes to panning and are unlikely to hard-pan sources, however a further analysis is required. Furthermore, although we show the models successfully mixing multiple genres, an in-depth analysis of the genre distribution of the dataset with ratings by genre is needed.

Production Value	WUN-S-L$_b$	Ours-S-L$_b$	WUN-L-L$_b$	Ours-L-L$_a$	Ours-L-L$_b$	Human
WUN-S-L$_b$	·	o	$\underline{\star}$	o	$\underline{**}$	$\underline{\star}$
Ours-S-L$_b$	o	·	o	o	$\underline{**}$	$\underline{\star}$
WUN-L-L$_b$	$\overline{\star}$	o	·	o	o	o
Ours-L-L$_a$	o	o	o	·	o	o
Ours-L-L$_b$	$\overline{**}$	$\overline{**}$	o	o	·	o
Human	$\overline{\star}$	$\overline{\star}$	o	o	o	·
Clarity	**WUN-S-L$_b$**	**Ours-S-L$_b$**	**WUN-L-L$_b$**	**Ours-L-L$_a$**	**Ours-L-L$_b$**	**Human**
WUN-S-L$_b$	·	o	o	o	o	o
Ours-S-L$_b$	o	·	o	$\underline{\star}$	$\underline{**}$	o
WUN-L-L$_b$	o	o	·	o	o	o
Ours-L-L$_a$	o	$\overline{\star}$	o	·	o	$\overline{\star}$
Ours-L-L$_b$	o	$\overline{**}$	o	o	·	$\overline{\star}$
Human	o	o	o	$\underline{\star}$	$\underline{\star}$	·
Excitement	**WUN-S-L$_b$**	**Ours-S-L$_b$**	**WUN-L-L$_b$**	**Ours-L-L$_a$**	**Ours-L-L$_b$**	**Human**
WUN-S-L$_b$	·	o	o	o	o	o
Ours-S-L$_b$	o	·	$\underline{*}$	$\underline{\star}$	$\underline{*}$	o
WUN-L-L$_b$	o	$\overline{*}$	·	o	o	o
Ours-L-L$_a$	o	$\overline{\star}$	o	·	o	o
Ours-L-L$_b$	o	$\overline{*}$	o	o	·	o
Human	o	o	o	o	o	·

Table 3: Post hoc Mann-Whitney test results of pairwise comparison with Bonferroni Correction. o > 0.05, $\star < 0.05$, $* < 0.01$, $** < 0.001$. E.g. when y-axis is compared to x-axis, $\overline{*}$ or $\underline{*}$ indicate y-axis is significantly better or worse than x-axis for a p-value < 0.01, respectively.

5. CONCLUSION

We present a novel data preprocessing approach that allows us to train deep learning networks with existing wet or processed multitrack data by reusing them to perform an automatic mixing task. During inference, we apply the same data preprocesing to dry multitrack data and we tested the generated mixes via objective and subjective tests. We introduced a new deep learning architecture designed for the proposed task, a perceptual-based loss function along with a redesigned listening test aimed at experienced mixing engineers. The results indicate that our approach successfully achieves automatic loudness, EQ, DRC, panning, and reverberation music mixing. Resulting mixes compared to professional mixes scored higher in terms of Clarity and are indistinguishable in terms of Production Value and Excitement. We believe that the proposed preprocessing can be applied to other data-driven MIR tasks.

6. ACKNOWLEDGMENTS

We would like to express special thanks to K. Gokan and S. Masui for their valuable comments and all the mixing engineers from Sony Music Entertainment Japan, Sony Europe and Queen Mary University of London who participated in the listening test.

7. REFERENCES

[1] R. Stables, J. D. Reiss, and B. De Man, *Intelligent Music Production*. Focal Press, 2019.

[2] P. D. Pestana and J. D. Reiss, "Intelligent audio production strategies informed by best practices," in *53rd Conference on Semantic Audio: Audio Engineering Society*, January 2014.

[3] B. De Man, J. D. Reiss, and R. Stables, "Ten years of automatic mixing," in *3rd AES Workshop on Intelligent Music Production*, September 2017.

[4] G. Bocko, M. F. Bocko, D. Headlam, J. Lundberg, and G. Ren, "Automatic music production system employing probabilistic expert systems," in *Audio Engineering Society Convention 129*. Audio Engineering Society, November 2010.

[5] B. De Man and J. D. Reiss, "A knowledge-engineered autonomous mixing system," in *Audio Engineering Society Convention 135*. Audio Engineering Society, October 2013.

[6] F. Everardo, "Towards an automated multitrack mixing tool using answer set programming," in *Proceedings of the 14th Sound and Music Computing Conference*, Espoo, Finland, July 2017.

[7] J. D. Reiss, "Intelligent systems for mixing multichannel audio," in *17th International Conference on Digital Signal Processing*. IEEE, 2011, pp. 1–6.

[8] P. Pestana, "Automatic mixing systems using adaptive digital audio effects," Ph.D. dissertation, Universidade Católica Portuguesa, 2013.

[9] M. A. Martínez-Ramírez, D. Stoller, and D. Moffat, "A deep learning approach to intelligent drum mixing with the wave-u-net." Journal of the Audio Engineering Society, 2021.

[10] C. J. Steinmetz, J. Pons, S. Pascual, and J. Serrà, "Automatic multitrack mixing with a differentiable mixing console of neural audio effects," in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2021.

[11] D. Moffat and M. B. Sandler, "Approaches in intelligent music production," *Arts*, vol. 8, no. 5, p. 14, September 2019. [Online]. Available: https://doi.org/10.3390/arts8040125

[12] T. Wilmering, D. Moffat, A. Milo, and M. B. Sandler, "A history of audio effects," *Applied Sciences*, vol. 10, no. 3, p. 791, January 2020. [Online]. Available: https://doi.org/10.3390/app10030791

[13] Y. Mitsufuji, G. Fabbro, S. Uhlich, F.-R. Stöter, A. Défossez, M. Kim, W. Choi, C.-Y. Yu, and K.-W. Cheuk, "Music demixing challenge 2021," *Frontiers in Signal Processing*, vol. 1, 2022. [Online]. Available: https://doi.org/10.3389/frsip.2021.808395

[14] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[15] D. Barchiesi and J. Reiss, "Reverse engineering of a mix," *Journal of the Audio Engineering Society*, vol. 58, no. 7/8, pp. 563–576, 2010.

[16] D. Moffat and M. B. Sandler, "Automatic mixing level balancing enhanced through source interference identification," in *Audio Engineering Society Convention 146*, Dublin, Ireland, March 2019.

[17] J. T. Colonel and J. Reiss, "Reverse engineering of a recording mix with differentiable digital signal processing," *The Journal of the Acoustical Society of America*, vol. 150, no. 1, pp. 608–619, 2021.

[18] J. Imort, G. Fabbro, M. A. Martínez-Ramírez, S. Uhlich, Y. Koyama, and Y. Mitsufuji, "Distortion audio effects: Learning how to recover the clean signal," in *23rd International Society for Music Information Retrieval Conference (ISMIR)*, 2022.

[19] J. Koo, S. Paik, and K. Lee, "Reverb conversion of mixed vocal tracks using an end-to-end convolutional deep neural network," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[20] D. Singh and B. Singh, "Investigating the impact of data normalization on classification performance," *Applied Soft Computing*, vol. 97, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494619302947

[21] R. ITU-R, "ITU-R BS. 1770-2, algorithms to measure audio programme loudness and true-peak audio level," *International Telecommunications Union, Geneva*, 2011.

[22] C. J. Steinmetz and J. D. Reiss, "pyloudnorm: A simple yet flexible loudness meter in python," in *Audio Engineering Society Convention 150*, 2021.

[23] F. G. Germain, G. J. Mysore, and T. Fujioka, "Equalization matching of speech recordings in real-world environments," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[24] R. W. Schafer, "What is a savitzky-golay filter?" *IEEE Signal processing magazine*, vol. 28, no. 4, pp. 111–117, 2011.

[25] P. Gaydecki, *Foundations of digital signal processing: theory, algorithms and hardware design*. Iet, 2004, vol. 15.

[26] J. O. Smith III, *Introduction to Digital Filters: with Audio Applications*. W3K Publishing, 2007, vol. 2.

[27] G. Tzanetakis, R. Jones, and K. McNally, "Stereo panning features for classifying recording production style." in *8th International Society for Music Information Retrieval Conference (ISMIR)*, 2007.

[28] C. Avendano, "Frequency-domain source identification and manipulation in stereo mixes for enhancement, suppression and re-panning applications," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2003.

[29] U. Zölzer *et al.*, *DAFX-Digital audio effects*. John Wiley & Sons, 2002.

[30] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A tutorial on onset detection in music signals," *IEEE Transactions on speech and audio processing*, vol. 13, no. 5, pp. 1035–1047, 2005.

[31] P. Masri, "Computer modelling of sound for transformation and synthesis of musical signals," Ph.D. dissertation, University of Bristol, 1996.

[32] P. Brossier, Tintamar, E. Müller, N. Philippsen, T. Seaver, H. Fritz, cyclopsian, S. Alexander, J. Williams, J. Cowgill, and A. Cruz, "Aubio - a library for audio and music analysis," February 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2578765

[33] P. Brossier, "Automatic annotation of musical audio for interactive applications," Ph.D. dissertation, Queen Mary University of London, 2006.

[34] J. Eaton, N. D. Gaubitch, A. H. Moore, and P. A. Naylor, "Estimation of room acoustic parameters: The ace challenge," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 10, pp. 1681–1693, 2016.

[35] H. Löllmann, E. Yilmaz, M. Jeub, and P. Vary, "An improved algorithm for blind reverberation time estimation," in *Proceedings of International Workshop on Acoustic Echo and Noise Control (IWAENC)*, 2010, pp. 1–4.

[36] S. Duangpummet, J. Karnjana, W. Kongprawechnon, and M. Unoki, "Blind estimation of room acoustic parameters and speech transmission index using mtf-based cnns," in *29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 181–185.

[37] P. White, "Processing reverb," *Sound On Sound*, vol. November 2020 Edition, 2022. [Online]. Available: https://www.soundonsound.com/techniques/processing-reverb

[38] M. A. Martínez-Ramírez, E. Benetos, and J. D. Reiss, "Deep learning for black-box modeling of audio effects," *Applied Sciences*, vol. 10, no. 2, p. 638, 2020. [Online]. Available: https://doi.org/10.3390/app10020638

[39] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 27, no. 8, pp. 1256–1266, 2019.

[40] V. Välimäki and J. D. Reiss, "All about audio equalization: Solutions and frontiers," *Applied Sciences*, vol. 6, no. 5, p. 129, 2016.

[41] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[42] J. Pons, S. Pascual, G. Cengarle, and J. Serrà, "Upsampling artifacts in neural audio synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[43] A. Wright and V. Välimäki, "Perceptual loss function for neural modeling of audio systems," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[44] B. De Man, "Towards a better understanding of mix engineering," Ph.D. dissertation, Queen Mary University of London, 2017.

[45] B. De Man, K. McNally, and J. D. Reiss, "Perceptual evaluation and analysis of reverberation in multitrack music production," *Journal of the Audio Engineering Society*, 2017.

[46] G. Peeters, "A large set of audio features for sound description (similarity and classification) in the cuidado project," *Analysis/Synthesis Team. IRCAM, Paris, France*, vol. 54, no. 0, pp. 1–25, 2004.

[47] Z. Ma, B. De Man, P. D. Pestana, D. A. Black, and J. D. Reiss, "Intelligent multitrack dynamic range compression," *Journal of the Audio Engineering Society*, vol. 63, no. 6, pp. 412–426, 2015.

[48] N. Jillings, B. De Man, D. Moffat, J. D. Reiss, and R. Stables, "Web audio evaluation tool: A browser-based listening test environment," in *International Sound and Music Computing Conference*, Maynooth, Ireland, July 2015.

[49] B. De Man and J. D. Reiss, "APE: Audio perceptual evaluation toolbox for matlab," in *Audio Engineering Society Convention 136*. Audio Engineering Society, 2014.

[50] P. D. Pestana and J. D. Reiss, "A cross-adaptive dynamic spectral panning technique." in *17th International Conference on Digital Audio Effects (DAFx-14)*, 2014.

[51] ITU-R BS. 1534-1, "Method for the subjective assessment of intermediate sound quality (MUSHRA)," *International Telecommunications Union, Geneva*, 2001.

[52] R. Schatz, S. Egger, and K. Masuch, "The impact of test duration on user fatigue and reliability of subjective quality ratings," *Journal of the Audio Engineering Society*, vol. 60, no. 1/2, pp. 63–73, 2012.

[53] EBU, "EBU recommendation: Loudness normalisation and permitted maximum level of audio signals," EBU-R128: Saconnex, 2014.

MUSIC-STAR: A STYLE TRANSLATION SYSTEM FOR AUDIO-BASED RE-INSTRUMENTATION

Mahshid Alinoori **Vassilios Tzerpos**
Department of Electrical Engineering and Computer Science, York University, Canada
{mahshida, bil}@yorku.ca

ABSTRACT

Music style translation aims to generate variations of existing pieces of music by altering the style-related characteristics of the original piece while content, such as the melody, remains unchanged. These alterations could involve timbre translation, re-harmonization, or music rearrangement. Previous studies have achieved promising results utilizing time-frequency and symbolic music representations. Music style translation on raw audio has also been investigated and applied to single-instrument pieces. Although processing raw audio is more challenging, it provides richer information about timbres, dynamics, and articulations.

In this paper, we introduce Music-STAR, the first audio-based translation system that translates the existing instruments in a piece into a set of target instruments without using source separation. To conduct our experiments, we also present an audio dataset that contains two-track pieces performed by two instrument sets alongside their stems. We carry out subjective and objective evaluations to compare Music-STAR with a variety of baseline methods and demonstrate its superiority.

1. INTRODUCTION

Music style translation is defined as transforming the style-variant components of a music piece to create variations that preserve the content. This can take several forms depending on how "music style" is characterized. Dai et al. [1] classify style into three categories: composition, performance, and timbre. Recent works have mainly focused on timbre translation [2–7] and composition style translation [8–12].

Timbre translation aims to alter the timbre information, which typically results in a change in instrumentation. Timbre translation models mostly use time-frequency representations [2–6]. Some of these [4–6] treat the spectrograms as images and apply GAN-based models [13,14] designed for image-to-image translation [15]. Timbre translation has also been explored by integrating classic signal processing and deep learning methods [16, 17].

© M. Alinoori and V. Tzerpos. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. Alinoori and V. Tzerpos, "Music-STAR: a Style Translation system for Audio-based Re-instrumentation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

The universal translation network [7] is the only model that works with audio waveforms directly. Inspired by the WaveNet autoencoder [18], it translates an arbitrary source piece to several specific timbre domains. Although one universal encoder is used to encode the domain-independent features, every domain needs to have a specific conditional WaveNet [19] decoder.

On the other hand, composition style transfer attends to tasks such as re-harmonization and music rearrangement. [1] Almost all works on composition style translation exploit symbolic representations, i.e., MIDI and piano rolls. Some of these models focus on music rearrangement by altering the accompaniments [8–10] where the style is associated with the genre. MIDI-VAE [9] is among the few cases that do not overlook the dynamics and account for note velocities. Wang et al. [11] has introduced the only GAN-based model that operates on symbolic representation to perform genre transformation. Hung et al. [12] approach music rearrangement by modifying the instrumentation, where they establish a correlation between rearrangement and transforming multiple timbres in a musical piece.

In this paper, we explore music re-instrumentation of audio waveforms that contain more than one instrument. We present several baseline solutions and then propose Music-STAR, a system designed explicitly for audio-based translation, which is built upon the WaveNet autoencoder [18]. To conduct our experiments, we also present a dataset called StarNet, in which every piece of music is performed by two different sets of instruments. The source code, audio samples, and supplementary materials are available at https://mahshidaln.github.io/Music-STAR.

2. DATASET

In order to train and evaluate Music-STAR and the baseline models, we need an audio dataset that contains multi-track pieces played with different sets of instruments alongside their stems. For this purpose, we have created the StarNet dataset, in which every piece is composed of two instrument tracks from two domains: strings-piano and clarinet-vibraphone. In other words, for every piece, the dataset includes strings-piano and clarinet-vibraphone mixtures as well as their corresponding isolated tracks.

We have chosen piano ↔ vibraphone and strings ↔ clarinet translations to preserve the type of excitation in

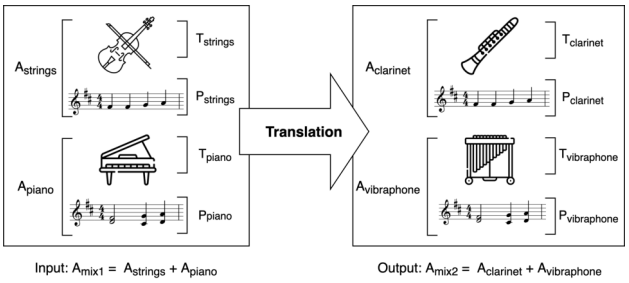


Figure 1. An example of multi-instrument translation where the timbre components of strings ($T_{strings}$) and piano (T_{piano}) are translated into clarinet ($T_{clarinet}$) and vibraphone ($T_{vibraphone}$), respectively, while retaining the corresponding pitch components ($P_{strings} = P_{clarinet}$ and $P_{piano} = P_{vibraphone}$).

the source/target instruments (discrete excitations in piano and vibraphone versus continuous excitations in clarinet and strings).

Accessing recorded music stems is often hard due to copyright limitations. Therefore, we select a variety of music pieces from MusicNet [20] and other freely available classical music MIDI collections. Instead of using their original instruments, we apply virtual instruments for the aforementioned combinations. The resulting dataset contains two domains, one for each instrument combination, adding up to roughly 11 hours of audio for each domain. The StarNet dataset is available at `https://zenodo.org/record/6917099`.

To conduct our experiments, we use two versions of StarNet:

1. Preprocessed StarNet: The preprocessed version of StarNet includes uncompressed stereo WAV files with a sample rate of 44.1 kHz and a bit depth of 16 bits. The preprocessing step includes detecting and removing the intervals where one or both instruments are silent to ensure their simultaneous presence for a considerable amount of time. After the silence removal, the dataset size shrinks to roughly 9 hours. Silence detection is done by indicating the minimum loudness and minimum silence duration.

2. Reduced StarNet: The reduced version of StarNet is obtained after resampling the preprocessed version at 16 kHz, merging the two audio channels and outputting mono audio, and finally quantizing the audio by 8-bit mu-law encoding.

3. METHODOLOGY

Pitch and timbre are among the fundamental acoustic properties of every audio track in a recorded mixture. As we address instrumentation changes in our work, timbre is regarded as the style component, while pitch constitutes the content we intend to preserve. Based on this definition, we introduce the following notation of a piano track and its style and content components:

$$A_{piano} := T_{piano} + P_{piano}$$

where A is an audio signal, T is the timbre component, and P is the set of pitch components and their embodied durations in that signal. Separating style and content components, also known as disentanglement, has been leveraged in previous music translation studies [7,9,12] and is mostly addressed by adversarial learning in encoder-decoder architectures. Note that T and P in the notations are conceptual terms, and the corresponding features will be extracted by the autoencoders and represented by the embeddings.

An example of single-instrument translation is shown below, where a piano track is translated to vibraphone in a way that the pitch component is retained, and the timbre component alters:

1. $Input = A_{piano} := T_{piano} + P_{piano}$
2. $Output = A_{vibraphone} := T_{vibraphone} + P_{piano}$

Our task is to tackle a more complex problem, i.e., dealing with multi-instrument pieces (Fig. 1). In the case of our dataset, the input audio signal will be one of the following:

1. $A_{mix1} = A_{strings} + A_{piano} :=$
   $T_{strings} + P_{strings} + T_{piano} + P_{piano}$
2. $A_{mix2} = A_{clarinet} + A_{vibraphone} :=$
   $T_{clarinet} + P_{clarinet} + T_{vibraphone} + P_{vibraphone}$

As a result, pitch-timbre disentanglement does not suffice here as we need to isolate two sets of timbre and pitch components.

In this section, we first introduce the baseline methods for performing multi-instrument translation, i.e., single-instrument translation pipeline and separation-based translation pipeline. Then we present our proposed methods, i.e., the embedding-supervised method and Music-STAR.

3.1 Single-instrument Translation Pipeline

The most simplistic approach to multi-instrument translation is to translate each instrument track separately and obtain the final output by mixing them. However, this is only possible when the music stems are available.

In order to implement this method, we employ an existing audio-based translation model [7], which is built upon the WaveNet autoencoder [18]. This model consists of a universal encoder and six decoders corresponding to six target domains, all collected from the MusicNet [20] dataset.

In this universal network, the temporal encoder maps the pitch components into an embedding space utilizing a WaveNet-like architecture. The decoders are conditioned on the pitch embeddings and generate audio samples that entail the specific timbre they have been trained for in an autoregressive manner. A domain confusion network is employed during training to ensure that no domain-specific information is included in the embeddings by imposing domain confusion loss [21]. The authors also emphasize the role of distorting the input by modulating the pitch locally

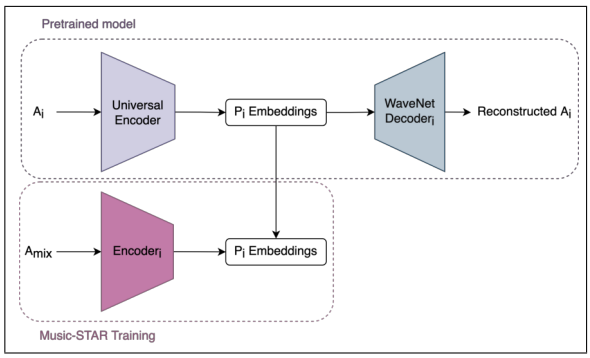


Figure 2. The embedding-supervised encoder is trained to generate the same embeddings as the universal encoder for every instrument *i* present in an audio mix (A_{mix}).

in every audio segment while training. This way, they ensure that the encoder does not memorize the content and drive it to capture meaningful features. In other words, the network is trained as a denoising autoencoder that learns to recover the original input by applying teacher forcing [22] while training the decoders. The teacher forcing technique is carried out by feeding the original input audio segment to the decoder instead of the generated samples.

3.2 Separation-based Translation Pipeline

The previous method does not apply to use cases where the instrument tracks are not available separately and where the system must take the audio mixture as input. One way to tackle this is to adopt a pipeline of audio source separation and single-instrument translation. The source separation module isolates each of the tracks so that a single-instrument translation model can transform each of them into a target instrument.

To perform source separation, we adopt Demucs [23], a state-of-the-art music source separation model that operates in the waveform domain, taking an audio mixture and outputting isolated audio tracks. These stems are then fed into the universal network described in Section 3.1 for the translation phase.

3.3 Embedding-supervised Method

Training source separation models to separate every instrument in a mixture is a demanding task, and thus we favor solutions that do not depend on it. We have investigated a potential solution that we refer to as the embedding-supervised method, which performs a semi-separation task through the encoding process. In this case, the embedding-supervised encoder learns to capture the pitch-related features of one of the two instruments in the mixture. In order to achieve this, we need a pre-trained encoder that can provide the desirable pitch embeddings for a single instrument and assist the embedding-supervised encoder in distinguishing between the instruments throughout the training step. Recall that the universal encoder in our baseline model learns to extract pitch-related information from the input. Thus the output of such an encoder can provide the embeddings we seek. Based on this, we can train the embedding-supervised encoder to mimic the output of the universal encoder when given a mixture as the input. (Fig. 2)

For instance, if we have the mixture as:

$$A_{mix} = A_{strings} + A_{piano}$$

we feed the piano track A_{piano}:

$$A_{piano} := T_{piano} + P_{piano}$$

into the universal encoder, and its output will represent piano track pitch information (P_{piano}). We then input the mixture A_{mix} to the embedding-supervised encoder and train it to output the same code, i.e., the embeddings for P_{piano}. Consequently, we have a piano-specific encoder to extract the piano pitch content from any mixture. Note that for such training, existence of the stems in the dataset is necessary. We can isolate the information corresponding to one instrument via this approach without applying actual source separation. We can train one encoder per instrument with the help of the universal encoder. The model strives to minimize the loss function below:

$$L(E^i_{es}(mix), E_u(x^i)) \quad (1)$$

where E_{es} is the embedding-supervised encoder, mix is the audio segment from the input mixture, E_u is the universal encoder, x^i is the fragment of the instrument track i that we want to extract from the mixture, and L is L1 loss.

During inference, a pre-trained WaveNet decoder can also be engaged to translate the code to an arbitrary target instrument.

3.4 Music-STAR

Our ultimate goal is to realize the idea of multi-instrument translation where we do not need to engage source separation modules or additional encoders, and in general remove the restrictions of the aforementioned techniques. To this end, we introduce Music-STAR, designed explicitly for audio-based multi-instrument translation as described below.

As mentioned in Section 3.1, the universal network is trained using a random local pitch modulation that distorts the input to ensure that the encoder does not memorize the input content. Since the encoder learns to extract pitch-related information, we expect the code to embody data from the distorted segments. However, the decoder is trained using the teacher forcing technique, where the original audio segment is fed into the decoder alongside the embeddings produced by the encoder. The benefits of this approach are two-fold. First, the decoder learns the timbre of the original segment that is not present in the latent space. Second, for the decoder to reconstruct the original audio with the right pitch, it forces the encoder to represent the original segment in the latent space by removing the distortion. We will employ a similar idea below.

To begin, we represent our input audio as:

$$A_{input} = A_{strings} + A_{piano} := \\ T_{strings} + P_{strings} + T_{piano} + P_{piano}$$

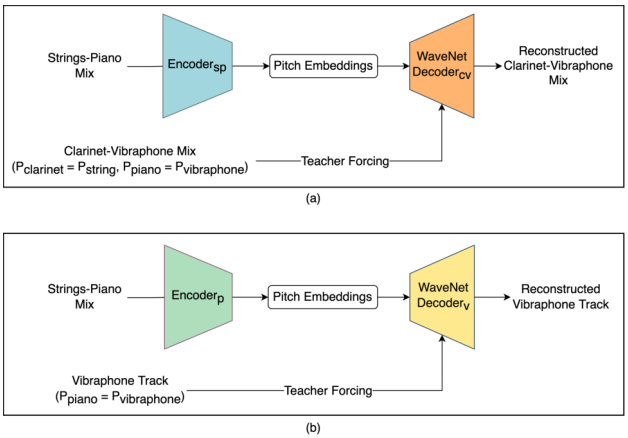


Figure 3. Training Music-STAR using (a) the mixture-supervised method to generated the target mixture, (b) the stem-supervised method to generate a single target stem. The modules subscripts *sp*, *cv*, *p*, and *v* corresponds to string-piano, clarinet-vibraphone, piano, and vibraphone, respectively.

The output we are looking for is:

$$A_{output} = A_{clarinet} + A_{vibraphone} := \\ T_{clarinet} + P_{clarinet} + T_{vibraphone} + P_{vibraphone}$$

where $P_{strings} = P_{clarinet}$ and $P_{piano} = P_{vibraphone}$.

To obtain this output, we should make sure that:

1. The encoder includes only the representations of P_{piano} and $P_{strings}$ in the embeddings, removing the information related to the other components of A_{input} (T_{piano} and $T_{strings}$), which has been proven possible when removing the distortion in [7].

2. The decoder generates the output signal by applying the target timbres that it has learned from the audio segments used for teacher forcing.

Accordingly, we conclude that the autoencoder will be able to extract the pitch-related features of the input mixture and translate it into target timbres if we train the model by:

1. Using the strings-piano mixture as the encoder's input, and

2. Using the clarinet-vibraphone counterpart of the input to apply teacher forcing to the decoder.

When the decoder is learning to translate the encoder's output to generate the clarinet-vibraphone segment, it inevitably guides the encoder to hand in helpful information for the task, which should be strings and piano pitch representations (see Fig. 3(a)). Since we train the model using the source and target audio mixtures, we name this approach the *mixture-supervised* method. The loss function for the mixture-supervised method is formulated as

$$\sum_{j} L(D(E(mix), t_j), t_j) \qquad (2)$$

where L is the cross-entropy loss, D is the decoder, E represents the encoder, mix is the input mixture segment, and t_j is the jth sample from the target mixture used for teacher forcing.

In our experiments, we also account for a variation of the mixture-supervised method where the target mixture (used in teacher forcing) is replaced by only one of its instrument tracks. We call the resulting method *stem-supervised*, in which we employ two autoencoders, each for translating one of the instruments in the mixture, and combine their outputs in the end to obtain the final mix (see Fig. 3(b)).

4. EXPERIMENTS

4.1 Single-instrument Translation Pipeline

We employ the universal network [7] to perform single-instrument translation. The network is originally trained on six classical music domains from the MusicNet dataset that includes mono audio segments with a sample rate of 16 kHz, which are later quantized by 8-bit mu-law encoding. The input files are randomly selected, and then a 0.75-second audio chunk is randomly segmented out of the file for every training input. A duration between 0.25 to 0.5 seconds of that segment is then selected for applying distortion by modulating the pitch.

In order to use this architecture as a baseline, we need to make sure that the decoders can generate the timbres included in the StarNet dataset. Therefore, we use the pre-trained universal encoder and finetune the decoders on the reduced StarNet dataset. Note that the data in reduced StarNet has the same properties as the data used for training the original model in terms of sample rate, bit depth, and the number of channels. Since the universal encoder has been trained using substantial computational resources on six domains and captures domain-agnostic features, it is powerful enough to successfully encode inputs from other domains. We finetuned the decoders with a batch size of 16, a learning rate of $1\mathrm{e}{-3}$, and a decay factor of 0.98 for 100 epochs.

4.2 Separation-based Translation Pipeline

We adopt Demucs [23] as the music source separation model, which is trained originally on MuseDB [24]. It is vital for our experiment that the network separates the two instruments present in the mixtures. Therefore, we train Demucs on preprocessed StarNet from scratch. We use the original learning rate of $3\mathrm{e}{-4}$ and Adam optimizer with a batch size of 32 for 250 epochs. The resulting model can take an arbitrary length of an input mixture and successfully separate the strings track from the piano or the clarinet track from the vibraphone.

The separated audio tracks are then post-processed to comply with the data configuration in reduced StarNet and fed into the finetuned models from Section 4.1 that correspond to their target instruments.

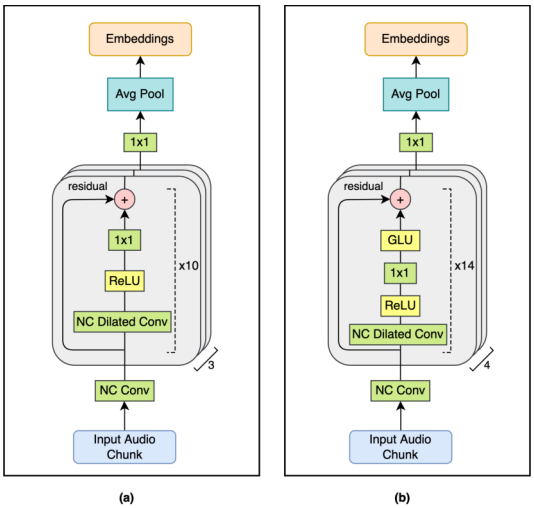


Figure 4. (a) The architecture of the universal encoder. (b) The architecture of the encoders used in the embedding-supervised and Music-STAR models.

4.3 Embedding-supervised Method

An autoencoder architecture builds the backbone of the model. This method focuses on training the embedding-supervised encoder with the help of the universal encoder. The architecture of the universal encoder is depicted in Fig. 4(a). Since the embedding-supervised encoder should learn to extract the same information as the universal encoder from a mixture rather than a stem, one would expect that a more complex architecture would be required for it. This was indeed confirmed by our pilot experiments.

The architecture we use as the embedding-supervised encoder is shown in Fig. 4(b). Similar to the universal encoder, it starts with a non-causal non-dilated convolution. The encoder consists of four blocks of 14-layer residual networks composed of non-causal dilated convolutions with a kernel size of 3, followed by ReLU nonlinearity and a 1x1 convolution. A GLU activation function follows the 1x1 convolution, and the output is summed with the input to form the residual connection. The result of this connection is then fed into the next layer of the encoder. The dilation increase factor of 2 and the 128 channels are identical to the universal encoder.

Unlike the baseline model that trains a universal encoder for all the domains in the training set, we need to train a single encoder for each instrument to extract content information from a mixture. The embedding-supervised encoders are trained on 0.75-second audio chunks randomly segmented out of the mixture files from reduced StarNet, and the same segments are extracted from the instrument tracks to be fed into the universal encoder. We adopt Adam optimization and exponential learning decay with a learning rate of $3e{-}4$, a decay factor of 0.98, and a batch size of 16 for a total of 100 epochs.

The finetuned WaveNet decoders described in Section 4.1 are then attached to the embedding-supervised encoders during inference to translate the code into the target instruments.

4.4 Music-STAR

We employ the WaveNet autoencoder architecture in the mixture-supervised method as well. The encoder we use for this method has the same architecture as the one used in the embedding-supervised method (Fig. 4(b)). The decoder's architecture is identical to the WaveNet decoders described in [7].

We train the models on the reduced version of StarNet. We pick random one-second audio segments of the input mixture and extract the same segment in the target mixture to apply teacher forcing to the decoder. Training is done using the Adam optimizer and exponential learning decay, where a learning rate of $3e{-}4$ and a decay rate of 0.99 are applied. The model is trained for a total of 100 epochs with a batch size of 16.

We use the same training configuration as above for the stem-supervised setting, except that two autoencoders are involved, each for transforming the source mixture into one of the target instruments.

5. EVALUATION

We conduct subjective and objective evaluations of the obtained audio and compare the re-instrumentation methods on three criteria:

1. Content preservation: how much of the pitch content of the input is retained in the output.
2. Style fit: How well the output presents the target timbres.
3. Audio quality: How clean and distortion-free the generated audio is.

5.1 Subjective Evaluation

The subjective evaluation provides a qualitative assessment of our models based on how users perceive the generated outputs. We carry out the subjective evaluation by distributing a survey among 30 participants, some of whom have a musical background. Each survey contains two different pieces, one from each domain (strings-piano and clarinet-vibraphone) selected out of 10 pieces in total. The target mixture is provided for each piece to demonstrate the gold standard, followed by corresponding translation outputs resulting from the five methods. The order of outputs is different for the two pieces, and the participants have no prior knowledge of the order.

Every piece in the survey is evaluated through three questions asking the participants to rank the five outputs based on:

1. How well they preserve the target musical content, which accounts for content preservation.
2. How well they present the instruments' tone colors (timbre) of the target piece, corresponding to style fit.
3. How clean and distortion-free they are, presenting the audio quality.

Method	Subjective			Objective	
	Content	Style	Quality	Content (Jaccard)	Style (Cosine)
Single-instrument	166	150	153	0.371	0.483
Separation-based	165	147	159	0.392	0.474
Embedding-supervised	161	157	149	0.350	0.472
Stem-supervised	154	190	183	0.323	**0.699**
Mixture-supervised	**254**	**256**	**256**	**0.426**	**0.698**

Table 1. Evaluation results on the three criteria of content preservation, style fit, and audio quality. The results from the subjective evaluation show the scores for each model according to the rankings provided by the surveys. Objective evaluation reports Jaccard similarity and cosine similarity for the first two criteria, respectively.

After collecting the surveys, we concluded the rankings for each question by scoring the models. For each criterion, the methods receive a score of 5 if one of their generated outputs is ranked first, a score of 4 if ranked second, a score of 3 if ranked third, a score of 2 if ranked fourth, and a score of 1 if ranked last. The aggregate scores are shown in Table 1. More detailed analysis is available online [1] .

5.2 Objective Evaluation

We aim to provide a quantitative quality assessment on content preservation and style fit using techniques employed in previous studies.

5.2.1 Content Preservation

Following the work by Cífka et al. [3], we assess the models on content preservation by calculating the Jaccard similarity between the pitch contours of the outputs and their corresponding gold standards. Since we are addressing multi-instrument music in our study, we extract the pitch contours using the multi-pitch Melodia algorithm [25] which provides the existing pitch frequencies in Hz. We round the frequency values to the nearest semitone. Then we express the similarity of the pitch sets of each time step in terms of the Jaccard index. Higher Jaccard similarity between the output and the gold standard signifies better content preservation.

5.2.2 Style Fit

We evaluate the style fit factor using the deep metric triplet network offered by Lee et al. [26]. The network consists of a backbone model, which provides embeddings for three inputs, and a triplet model that outputs a similarity score between those embeddings. The three inputs are called *the anchor*, *the positive*, and *the negative*. The network is trained to generate the embeddings in a way that the positive is closer to the anchor than the negative. During inference, a similarity score between the anchor and the other two will be reported.

Following Cífka et al. [3], we use MFCCs as the input features as they provide timbre-related information. We train the triplet network using the mixtures in the preprocessed StarNet dataset. For instance, the MFCCs of eight-second clarinet-vibraphone audio segments are used as both the anchors and the positive inputs in the training phase. At the same time, their counterparts from the strings-piano domain are regarded as negative inputs.

During inference, we presented the translation outputs from the five methods as the anchors, their corresponding gold standard as the positive, and the performance from the other domain as the negative. We report the average cosine similarity between the outputs of each translation method and their corresponding gold standards. Higher cosine similarity denotes more likeness to the target timbre and a better style fit.

5.3 Discussion

All evaluation results are presented in Table 1. The scores reported by the subjective evaluation are the total sum of the models' scores on both Clarinet-Vibraphone $\leftrightarrow$ Strings-Piano translations. Objective evaluation reports the average performance of models in translating the two domains. Both objective and subjective evaluations conclude that mixture-supervised Music-STAR is the predominant model in performing multi-instrument music translation. The mixture-supervised method outperforms the other methods in all the assessments except for objective style fit, where it reaches an almost equal cosine similarity with the stem-supervised method. An advantage that it has over the other methods is that the presence of the stems is not necessary for training the model. Mixture-supervised Music-STAR is, to the best of our knowledge, the only model capable of performing timbre translation on multiple instruments in a mixed signal.

The embedding-supervised method is, on average, the worst-performing model. The encoder used in this model is trained based on the universal encoder's outputs to extract the pitch information of one instrument out of a mixture. An imperfect ground truth places the model at the disadvantage of even more unsatisfactory performance. Also, the performance of single-instrument and separation-based translation models on different criteria are very similar.

6. CONCLUSION

This paper introduced Music-STAR, the first audio-based multi-instrument music translation system. Music-STAR tackles multi-instrument translation without applying explicit source separation to the input mixtures. We also introduce the StarNet dataset that includes two-instrument pieces performed in two domains alongside their stems. We explored a variety of possible solutions based on the WaveNet autoencoder, and finally reached a successful mixture-supervised method capable of performing simultaneous source separation and pitch-timbre disentanglement for two instruments.

Future work will target increasing the number of instrument tracks in the mixtures, and adding to the variety of instrument combinations. We also plan to develop a more ornate dataset in terms of the details on articulations and dynamics.

[1] `https://mahshidaln.github.io/Music-STAR`

7. REFERENCES

[1] S. Dai, Z. Zhang, and G. G. Xia, "Music style transfer: A position paper," *arXiv preprint arXiv:1803.06841*, 2018.

[2] A. Bitton, P. Esling, and A. Chemla-Romeu-Santos, "Modulated variational auto-encoders for many-to-many musical timbre transfer," *arXiv preprint arXiv:1810.00222*, 2018.

[3] O. Cífka, A. Ozerov, U. Şimşekli, and G. Richard, "Self-supervised VQ-VAE for one-shot music style transfer," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 96–100.

[4] S. Huang, Q. Li, C. Anil, X. Bao, S. Oore, and R. B. Grosse, "TimbreTron: A WaveNet (CycleGAN (CQT (audio))) pipeline for musical timbre transfer," *arXiv preprint arXiv:1811.09620*, 2018.

[5] D. K. Jain, A. Kumar, L. Cai, S. Singhal, and V. Kumar, "ATT: Attention-based timbre transfer," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–6.

[6] C.-Y. Lu, M.-X. Xue, C.-C. Chang, C.-R. Lee, and L. Su, "Play as you like: Timbre-enhanced multi-modal music style transfer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1061–1068.

[7] N. Mor, L. Wolf, A. Polyak, and Y. Taigman, "A universal music translation network," *arXiv preprint arXiv:1805.07848*, 2018.

[8] W. T. Lu and L. Su, "Transferring the style of homophonic music using recurrent neural networks and autoregressive model," in *ISMIR*, 2018, pp. 740–746.

[9] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer," *arXiv preprint arXiv:1809.07600*, 2018.

[10] O. Cífka, U. Şimşekli, and G. Richard, "Groove2Groove: One-shot music style transfer with supervision from synthetic data," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2638–2650, 2020.

[11] J. Wang, C. Jin, W. Zhao, S. Liu, and X. Lv, "An unsupervised methodology for musical style translation," in *2019 15th International Conference on Computational Intelligence and Security (CIS)*. IEEE, 2019, pp. 216–220.

[12] Y.-N. Hung, I. Chiang, Y.-A. Chen, and Y.-H. Yang, "Musical composition style transfer via disentangled timbre representations," *arXiv preprint arXiv:1905.13567*, 2019.

[13] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[14] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard GAN," *arXiv preprint arXiv:1807.00734*, 2018.

[15] Y. Pang, J. Lin, T. Qin, and Z. Chen, "Image-to-Image translation: Methods and applications," *arXiv preprint arXiv:2101.08629*, 2021.

[16] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," *arXiv preprint arXiv:2001.04643*, 2020.

[17] F. Ganis, E. F. Knudesn, S. V. Lyster, R. Otterbein, D. Südholt, and C. Erkut, "Real-time timbre transfer and sound synthesis using DDSP," *arXiv preprint arXiv:2103.07220*, 2021.

[18] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1068–1077.

[19] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[20] J. Thickstun, Z. Harchaoui, and S. M. Kakade, "Learning features of music from scratch," in *International Conference on Learning Representations (ICLR)*, 2017.

[21] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The journal of machine learning research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[22] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[23] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music source separation in the waveform domain," *arXiv preprint arXiv:1911.13254*, 2019.

[24] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[25] J. Salamon and E. Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 6, pp. 1759–1770, 2012.

[26] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Disentangled multidimensional metric learning for music similarity," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 6–10.

LEARNING UNSUPERVISED HIERARCHIES OF AUDIO CONCEPTS

Darius Afchar[1,2] **Romain Hennequin**[1] **Vincent Guigue**[2]
[1]Deezer Research, France [2]MLIA, ISIR - Sorbonne Université - CNRS, France
research@deezer.com

ABSTRACT

Music signals are difficult to interpret from their low-level features, perhaps even more than images: e.g. highlighting part of a spectrogram or an image is often insufficient to convey high-level ideas that are genuinely relevant to humans. In computer vision, concept learning was therein proposed to adjust explanations to the right abstraction level (e.g. detect clinical concepts from radiographs). These methods have yet to be used for MIR.

In this paper, we adapt concept learning to the realm of music, with its particularities. For instance, music concepts are typically non-independent and of mixed nature (e.g. genre, instruments, mood), unlike previous work that assumed disentangled concepts. We propose a method to learn numerous music concepts from audio and then automatically hierarchise them to expose their mutual relationships. We conduct experiments on datasets of playlists from a music streaming service, serving as a few annotated examples for diverse concepts. Evaluations show that the mined hierarchies are aligned with both ground-truth hierarchies of concepts – when available – and with proxy sources of concept similarity in the general case.

1. INTRODUCTION

Music signals are challenging to interpret [1]. For instance, inspecting a spectrogram by parts – *e.g.* think of attention maps [2] – does not convey much meaning with respect to the high-level descriptions that are useful to humans – *e.g.* the mood of a song. Fortunately, a solution was proposed in a field that shares similar problems: computer vision. As pixels of images are too low-level to be understandable, *concept learning* was introduced to rationalise the abstractions steps involved in a feed-forward model (*e.g.* detection of colours, shapes, patterns, ...), which in turn can be used to describe the content of an image directly, and to align models with human reasonings better. Indeed, this field has had successful applications in explainability (XAI) – dissecting a model's predictions into human-grounded concepts [3–5], interactive learning [6] or knowledge transfer between tasks [7]. However, these methods have yet to be used for Music Information Retrieval (MIR).

© D. Afchar, R. Hennequin, and V. Guigue. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** D. Afchar, R. Hennequin, and V. Guigue, "Learning Unsupervised Hierarchies of Audio Concepts", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

Music has particularities that are not captured by the current formulation of concept learning. For instance, let us consider genres as common musical concepts we would like to detect from spectrograms. Genres are typically blended as they influence or result from other genres (*e.g. Punk Rock*). By contrast, most literature on concept learning has considered sets of few identified and disentangled concepts, which does not apply to genres. Moreover, one trend in MIR is to incorporate multiple types of descriptors into a shared space (*e.g.* mood, genre, instruments [8, 9]), thus requiring handling even more entanglements.

We thus propose to study a novel problem for concept learning that is relevant for music: **learning hierarchies of concepts**. Indeed, a hierarchy enables navigating thousands of non-independent concepts without cognitive overload by uncovering their mutual relationships with a structure, thus rationalising which concepts share the same class or are derived from one another. In addition, the use of taxonomy – *i.e.* hierarchy – is common in musicology and feels like a natural representation in this context [10, 11].

The problem we study is interesting for *music representation learning* because research is often limited by data labelling, which is labour-intensive and can be ambiguous or tied to a specific dataset. Meanwhile, concepts can be learned with few shots, which open the doors to new sources of music descriptors. In particular, we propose to leverage a dataset of playlists from a music streaming service as a folksonomy of concepts. Playlists are often built with a specific concept; there exists one for any genre, instrument, mood, and many more niche ideas. Yet, there is no overall ground-truth organisation of playlists, making their use cumbersome. That is why we propose to solve this task by first adapting a method from concept learning research to learn music concepts from playlists in a few-shot manner and then, building on top of this method, automatically derive a hierarchy from the learned concepts.

The paper proceeds as follows: we provide background on concept and hierarchy learning (section 2); we adapt the literature of concept-based explanations to the realm of MIR (3); we present a novel way to organise the detected concepts into a hierarchy in an unsupervised manner (4); finally, we experiment and discuss our attempt to provide a new audio concepts hierarchy with mixed types (5).

2. PRELIMINARIES AND RELATED WORK

We first present the literature on concept learning and hierarchy mining, then frame our problem through XAI to uncover common pitfalls to avoid when interpreting data.

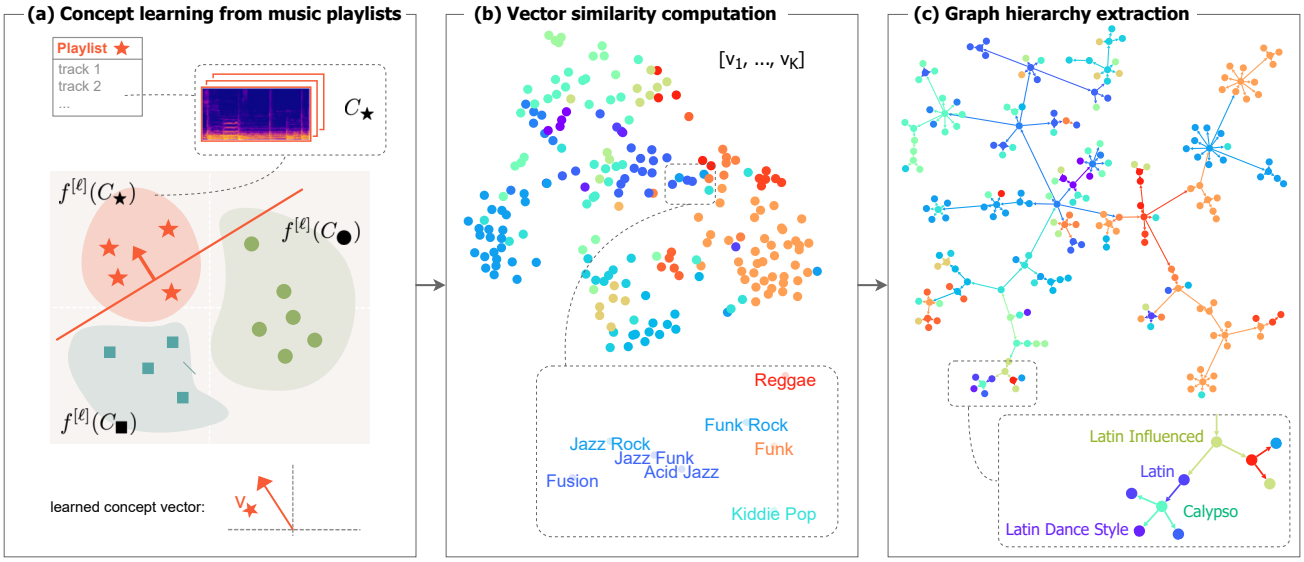


Figure 1. **Overview of our method** *(a)* CAVs learning (section 3.1); *(b)* t-SNE projection of concepts $\mathcal{V}_{\text{genre}}$ learned on a small dataset with 219 music genres; *(c)* resulting extracted hierarchy $H(S(\mathcal{V}_{\text{genre}}))$. Nodes colours denote ground-truth clusters. Experiments on a bigger concept dataset are conducted in section 5.

2.1 Concept learning

Concept learning has witnessed growing interest in recent years [4]. The introduction of concepts stems from the inadequacy of usual *explanation spaces* with human mental models of problems: tabular features, pixels, and words in sentences are sometimes too low-level to build relevant explanations or to interact with. This idea is far from novel from a purely technical standpoint: a stream of papers aim to learn a set of sub-attributes in a fully supervised manner (*e.g.* facial features [12], or animal attributes [13]), that are then combined to predict some end-task labels, typically through a linear regression to track the importance of each attribute. This idea has been revamped recently for *concept bottleneck* learning [14] and *prototypical part learning* [15]. In the same vein, *disentanglement techniques* [16] have sought to realign latent representations to interpretable attributes. All those works rely on the assumption that a somewhat complete set of useful concepts exists, well-defined and rather disentangled, and that a corresponding labelled dataset is readily available to train on.

This is rarely the case in practice. Kim et al. introduced CAVs for images [3] and demonstrated that concepts could be learned reliably from few annotated examples – technical details are provided in section 3.1. Going beyond the previously few datasets annotated for concepts, this work has led to new and diverse applications: skin lesion interpretations [17], emotion recognition [18], interactivity [6, 19], automatic concept learning [20], batch normalisation [21], sufficiency of explanations with shapley values [22, 23], causal inference [24]. To our knowledge, there exists no application in the music domain.

2.2 Unsupervised Hierarchy Mining

Using knowledge graphs and taxonomies is frequent leverage in MIR [25]. Unsupervised learning of hierarchy is, however, less so. Part of it lies in the difficulty of adequately defining the meaning of the hierarchical relations, learning the hierarchy in a tractable way, and evaluating the found hierarchy without ground-truth.

Fortunately, there is active literature coming from *natural language processing (NLP)* research. Many work proposed variants of *topic modelling* [26] to enable sampling hierarchised structures [27, 28]. However, these methods get computationally intractable as the number of nodes, topics, and hierarchical levels grows. Many successes thus came from scalable greedy methods such as hierarchical clustering [29, 30], rule-mining [31, 32] and graph-based analyses [33, 34]. Outside of NLP, hierarchies have been learned based on the latent manifold geometry [35], predefined structure [36], and for images [37, 38].

To our knowledge, these techniques have never been combined with concept learning, as we propose.

2.3 Explainability

Our end goal is to interpret music signals, which is linked to explainability (XAI). Before jumping into our method to learn a hierarchy of concepts, we need to acknowledge that many works have debated the sanity of explanation methods [39–43] and the fact that these methods largely disagree with one another [44]. XAI has vast boundaries, which has also percolated to its music sub-field for which there exists many explanation paradigms and evaluations [45]. Explanations for MIR are thus also prone to exhibiting pathologies that hamper knowledge mining: *e.g.* shortcut learning [46], lack of clear definitions [39, 47], flawed evaluations [48], out-of-distribution artifacts [49, 50] or misalignment with human reasonings [51–54].

Uncovering these failure cases sharpens our understanding of how models make decisions, and shows that we have to go beyond evasive considerations on explain-

ability to make them relevant [41] since trustworthy AI is not achieved automatically with XAI [55]. One taxonomic distinction we would like to underline is whether methods aim to explain models or data [56]. Our method is intended to belong to the latter category for which the purpose is not to explain how models work (*e.g.* troubleshooting learned filters) but rather to use them as proxies to gain new insights about reality (*e.g.* causal inference [57]).

With all these aspects in mind and based on previous literature, we formalise our three explanation goals:

1. **Attribution.** Identify musical concepts associated with a given track's audio signal.
2. **Transferability.** Detected concepts should be applicable to various settings and tasks.
3. **Generality.** The hierarchy should be independent of the signals and make sense for various settings.

Our first goal is linked to the traditional explanation literature supporting *transparency* [45, 47] and is common in concept-learning. We have defined our second and third goals to stress that the finality of our work is to be as universal as possible.

3. LEARNING MUSIC CONCEPTS

In this section, we first adopt the few-shot supervision setting of concept learning to learn the music concept $\vec{v}_C$ of each playlist, viewed as a set C of audio signals examples.

3.1 Background on CAVs

We review some essential notions from the concept learning framework we use to learn playlist concepts [3].

We denote $\vec{f}^{[\ell]}$ the output of the l-th layer of a trained neural network f taking as input some samples $x \sim X$. A "concept activation vector" $\vec{v}_C$ is defined for a set of positive examples C as the normal to a separating hyperplane between $\{\vec{f}^{[\ell]}(x) \mid x \in C\}$ versus negative examples – usually $\{\vec{f}^{[\ell]}(x) \mid x \notin C\}$, we provide an illustration in Figure 1(a). In practice, the hyperplane is learned through a logistic regression on a subset of C, while holding out the remaining samples for validation and testing [1] .

As additional insights, this setting is "sound" for few-shot learning because it uses a pretrained model f with prior knowledge on the domain – relating to transfer learning [58] – and learns a linear mapping on its space, hence reducing the risk of over-parametrisation and justifying why more complex mappings are not considered.

3.2 Sets of examples C

The work of Kim et al. [3] lends itself naturally to learning concepts from playlists. Indeed, playlists can be seen as small sets of positive music examples built around a particular musical idea. As it will be further detailed in our experiments, we can use data from music streaming services where playlists are abundant. We will denote $\mathcal{C} = \{C_i\}_i^K$ the set of K sets of music tracks that will enable to learn a CAV set $\mathcal{V} = \{\vec{v}_i\}_i^K$ discriminating $\mathcal{C}$ in the $\vec{f}^{[\ell]}$ space.

It must be noted that playlists are more than a set of tracks, we usually have access to textual data with a title and a description that help rationalise the learnt concepts.

3.3 Backbone model $\vec{f}^{[\ell]}$

As often in few-shot learning, CAVs learning makes use of a pretrained model f – referred to as *backbone* – to embed samples into a latent space with a higher level of abstraction. In this paper, our working hypothesis is that **music concepts can be described solely through audio signals**. We wanted to fully explore the potential of audio signals, which is a known challenging setting in MIR and recommendation [59], and leave its combination with other sources of features for future work. In addition, having audio inputs enables using our model with any music track from any dataset, hence supporting goal 2, while relying on features such as collaborative filtering or other usage data would have tied our model to a given dataset.

After having experimented with several models, as the recent self-supervised model *CLMR* [60], we have chosen to use *MusiCNN* [61] as the backbone since it had demonstrated consistent performances across various musical tasks, which we deemed on par with our goal 2 and 3. MusiCNN was trained on the *Million Song Dataset* [62].

3.4 Preliminary results

We provide some experimental observations to fix ideas and prepare the next section. If we train CAVs as in the original paper [3] – further detailed in section 5.2, we obtain a set of vectors $\mathcal{V}$ that allows to detect a set of K concepts given any music input spectrogram. An example result of learned concepts is given in Figure 1(b) for one of the studied datasets. Unsurprisingly, we find that many associated concepts end up having close vector weights [2] . If we were to use our concept detectors directly, we would have many concepts activated simultaneously, which could make predictions hard to read when K is very big. On the positive side, note that **vector proximity could be used as a means to quantify concept similarities**. With this idea, we go a step further and next present our method to build a hierarchy to help rationalise concept dependencies.

4. HIERARCHISING MUSIC CONCEPTS

Having learned numerous non-independent concepts $\mathcal{V} = \{\vec{v}_i\}_i^K$, we next elaborate on how to organise them with a structure to make them usable (goals 2, 3).

Many classes of structures are possible, and it would be a mistake to select one with purely theoretical considerations since cognitive and social sciences [63, 64] have underlined that explanatory preferences are shaped by diverse factors that cannot be reduced to a set of so-called golden explanation principles. Therefore, from an empirical standpoint, taxonomies seem to be frequent with music

[1] By definition, $\vec{v}_C$ is the learned parameter of a logistic regression.

[2] In the figure, proximity is defined as the cosine similarity.

concepts (*e.g.* genres [8], instruments [25]). We thus argue that this class of graph is rather natural for humans and chose to consider it only.

4.1 Similarity graph computation

Before mining a hierarchy, it is necessary to determine how concepts – now acting as graph nodes – relate to one another. Essentially, we are trying to define a similarity measure between concepts.

When sampling uniformly from a music dataset, concept activations are rare signals overshadowed by uninformative negative detection. The straightforward solution of estimating the empirical covariance of concept activations is thus not a well-behaved measure of similarity. Fortunately, there is another reliable, interpretable (and faster) way to do it. We propose to consider the positive examples of each concept playlist – embedded in $\vec{f}^{[\ell]}$, and to check on what side of other concept hyperplanes their centroid lies. Formally, given $\mathcal{V} = \{\vec{v}_i\}_1^K$, we compute the matrices of concept similarities S and adjacency A (*i.e.* A is a binarised matrix representing graph edges):

$$S_{i,j}(\mathcal{V}) = \mathbb{E}_{x\sim\mathcal{C}_i}\left[\sigma(\langle\vec{v}_j, \vec{f}^{[\ell]}(x)\rangle)\right] \quad (1)$$

$$A_{i,j}(\mathcal{V}) = \left[S_{i,j}(\mathcal{V}) \geq \frac{1}{2}\right] \quad (2)$$

where σ denotes the logistic sigmoid function and $\mathcal{C}_i$ a random variable sampling spectrogram excerpts from tracks of the playlist C_i. Bias is integrated in $\vec{v}_j$ for simplicity. The threshold at $\frac{1}{2}$ for A translates having each playlist centroid on the positive side of the concept hyperplanes. Because $\mathcal{V}$ is learned through logistic regressions that return well-calibrated probabilities, S and A return similarities with well-defined interpretations.

4.2 Hierarchy extraction

Having defined similarities between concepts, we next simplify S and A to a hierarchy that highlights what concepts are similar to many others and what others represent unique and niche ideas.

For our problem and its scale, we propose to adapt the greedy hierarchy construction of Heymann et Garcia-Molina [33] that makes use of the notion of **betweenness graph centrality** [65]. In this method, given an unweighted graph of similarities, a tree is greedily grown by picking each node in decreasing order of centrality in the similarity graph and linking it to the most similar and already-added node in the tree. We denote by $H(S(\mathcal{V}))$ the result of this algorithm on the similarity graph given by the adjacency matrix $A(\mathcal{V})$ and using $S(\mathcal{V})$ as similarity.

Though this algorithm was originally applied to word tags, we argue that the inductive bias of the targeted hierarchy is applicable to our problem. In particular, the use of centrality is justified by the existence of noisy links in the similarity graph, said links are assumed to be more frequent higher up in the hierarchy (*general-general assumption*). For us, this means that general concepts – *e.g. Pop* and *Rock* – should be more likely to be similar than niche concepts – *Bedroom Pop* and *Goth Rock*. This sounds reasonable for music content since we expect niche concepts to relate to a specific musical idea, style, or instrument.

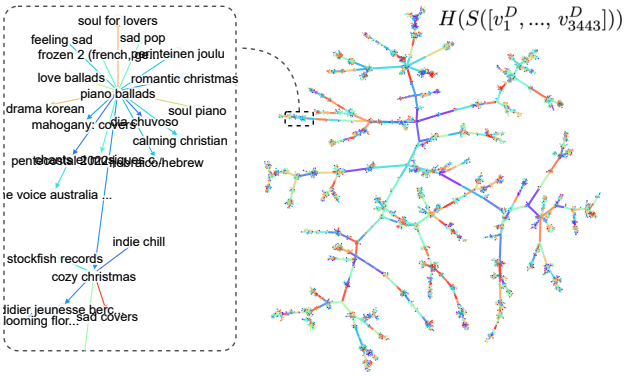


Figure 2. Obtained hierarchy on the Deezer dataset. An interactive plot of this figure, as well as many others, is available at `research.deezer.com/concept_hierarchy/`.

5. EXPERIMENTS

We first detail our experimental setups, then validate that concepts can be reliably learned from audio, and finally show that our proposed structuring method leads to consistent hierarchies for MIR applications.

5.1 Datasets

The set of playlists $\mathcal{C}_D$ we use was exported through the API of the streaming platform Deezer [3]. We have filtered **3635 playlists of mixed types** to be used as concepts: all 1498 editorial playlists available – considered thoughtfully curated but biased by popularity, and 2137 public user playlists with the lower artist popularity – to improve diversity. Playlists have a mean length of 75 tracks and a minimum of 40. This dataset includes the playlist ids, titles, descriptions, and public 30s audio preview for each track. There are 245074 unique tracks from 116497 unique artists: playlists overlap is thus very limited.

In order to compare our computed hierarchy to existing taxonomies, we make use of the APM Music dataset [66]. This is a more traditionally tagged dataset, but its relevance for our task is to include a ground-truth two-level hierarchy of the *music genres* (219) and *moods* (165) tag types. We aggregate sets of positively tagged examples as $\mathcal{C}_{\text{genre}}$ for each genre tags and respectively $\mathcal{C}_{\text{mood}}$ for moods.

5.2 Validating concept learning

We first present experimental evidence related to learning sets of concepts (section 3), to validate goal 1.

5.2.1 Experimental setup

Concept examples are split in a 70% (*train*) / 10% (*validation*) / 20% (*test*) fashion. According to the requirements of our backbone [61], we convert audio inputs to

[3] `https://developers.deezer.com/api`

Source	Audio (↓)	CF (↓)	BERT (↑)	W2V$_1$ (↑)	W2V$_2$ (↑)
$H(S(\mathcal{V}_D))$	**2.449 ± 0.022**	0.845 ± 0.013	0.345 ± 0.007	0.286 ± 0.009	0.542 ± 0.007
$H(S_{\text{CF}})$	**2.413 ± 0.021**	**0.345 ± 0.007**	0.416 ± 0.008	0.336 ± 0.010	0.601 ± 0.008
$H(S_{\text{BERT}})$	2.858 ± 0.028	0.868 ± 0.013	**0.726 ± 0.005**	0.505 ± 0.011	0.652 ± 0.008
$H(S_{\text{W2V-1}})$	2.952 ± 0.028	0.932 ± 0.012	0.523 ± 0.008	**0.804 ± 0.005**	0.721 ± 0.007
$H(S_{\text{W2V-2}})$	2.843 ± 0.026	0.847 ± 0.012	0.531 ± 0.008	0.596 ± 0.009	**0.836 ± 0.004**
Random	3.388 ± 0.027	1.104 ± 0.006	0.239 ± 0.004	0.142 ± 0.006	0.452 ± 0.006

Table 1. Evaluation of the full hierarchy on audio, user-logs and title semantic similarities with 95% confidence. *Audio* and *CF* source are endowed with an **Euclidean distance**, the last three semantic sources use **cosine similarities**.

mel-spectrograms with log-magnitudes. Representations are extracted from the penultimate layer of the backbone, denoted as $f^{[\ell]}$ – which gave us the best performances for concept learning overall. Our complete training code, with further details and exact parameters, is provided for reproducibility at `github.com/deezer/concept_hierarchy`.

5.2.2 *Learning metrics evaluation*

On the Deezer dataset $\mathcal{C}_D$, the mean balanced accuracy of learning concepts is **83.8% ± 0.3**, with a 95% confidence interval. In detail, we display a histogram of test balanced accuracies in Figure 3, showing that a **majority of playlist concepts can be learned reliably from audio**.

As a rationalisation of failure cases, the hypothesis we made in section 3.4 to detect concepts from audio may not always hold. For instance, it fails when playlist concepts rely on factors extraneous to audio (*e.g.* playlist of a movie soundtrack) or when the backbone space is not expressive enough to discriminate concepts. This actually happens in these experiments with concepts similar in all respects but for their singing languages: *e.g.* the model makes no difference between Japanese and British jazz. We filter concepts v_i below a given test accuracy threshold to account for this effect. Fixed at 70%, 192 concepts are filtered out, leading to $K = 3443$ remaining concepts. This threshold was set empirically by checking that those left-out concepts were indeed unclear from audio (*e.g.* OST, new releases, charts). We denote the resulting concept set $\mathcal{V}_D$.

The two smaller APM datasets lead to similar results with a **78.3% ± 0.8** for genres, denoted $\mathcal{V}_{\text{genre}}$, and a **71.3% ± 0.6** accuracy for mood tags, denoted $\mathcal{V}_{\text{mood}}$.

As a wrap-up of this section, goal 1 of detecting concepts from audio seems satisfied in our experiments.

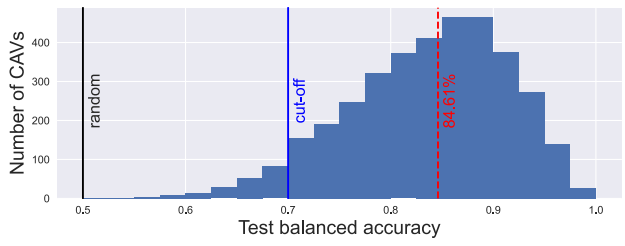


Figure 3. Concept learning performances. The middle line indicates the cut-off at 70% accuracy, the right line is the mean accuracy of the remaining CAVs.

5.3 Hierarchy Mining

We conduct a three-fold evaluation of our proposed hierarchy extraction method (section 4) in order to evaluate goal 3: structure; ground-truth comparison; alignment with other sources of concept similarities.

5.3.1 *Structure priors*

We first provide evidence supporting our choice to extract a hierarchical tree graph over other choices of structures. Though we do not have access to a ground-truth graph of the relations between concepts in $\mathcal{C}_D$, we have some priors on what a satisfying result should be like. For instance, we know that musical concepts are often blended (*e.g.* genres). It is thus safe to assume that a good graph of concepts should be *connected*. Extending this property, isolated nodes should strongly be discouraged. As another prior assumption on our target graph, we know that we can usually find several similar playlists to a given one, but that very strongly connected nodes should be infrequent, *sparsity* should thus also be valued.

With these two priors in mind, we inspect various graph baselines that could straightforwardly be obtained from $\mathcal{V}_D$ without our hierarchy extraction step: the similarity graph $A(\mathcal{V}_D)$ from equation (2); a similarity graph obtained from $S(\mathcal{V}_D)$ with an adjusted threshold to match the sparsity of $H(S(\mathcal{V}_D))$ (*Sparse* A); a top-1 most similar neighbours graph of each node; reference random graphs. Their respective sparsity and connectivity is shown in Table 2. It appears that baseline graphs become disconnected as soon as sparsity drops. We also underline that random baselines provide better-behaved graphs than their counterparts, indicating that similarity links are not evenly distributed among nodes, thus pointing to the existence of communities in the graph that justifies the use of betweenness [67]. Using a hierarchy extraction algorithm helps maintain a low sparsity while connecting every node in the graph.

Graph	#Edges (↓)	#CC (↓)	#IN (↓)
H **(proposed)**	3442 (0.03%)	1	0
$A(\mathcal{V}_D)$	1309163 (11%)	1	0
Sparse A	3443	2766	2735
Top-1	3443	134	0
Random Sparse A	3443	516	430
Random Top-1	3443	7	0

Table 2. Structure evaluation. We count the number of edges in each graph, of connected components (*CC*), and of isolated nodes (*IN*). Lower is better.

This study does not prevent using sparse and connected graphs other than hierarchies. We believe that other relevant structures may exist. Nonetheless, as additional benefits of using hierarchies, we note that tree-like graphs are always planar, which eases their visualisation and navigability by having only one most-relevant parent per node.

5.3.2 Ground-truth evaluation

To validate our generated concept hierarchy, we compare $H(S(\mathcal{V}_{\text{mood}}))$ and $H(S(\mathcal{V}_{\text{genre}}))$ to the expert tag hierarchy provided with the APM dataset – which is unavailable for $\mathcal{V}_D$. As those hierarchies are two-levelled – *i.e.* clustering of tags, we evaluate the accuracy of the mined edges at linking neighbours from the same cluster and compute a silhouette score [68]. Judging from the results given in Table 3, the generation is promising but far from perfect.

The estimated hierarchy of genres is illustrated in Figure 1(c) to help interpret the results. As a typical failure case, the orange cluster ("Electronica") ends up in two separate branches, which can happen because of the greedy construction of our hierarchy. In other cases, however, the ground-truth may be questioned: "Latin influence", "Latin", "Ethnic Dance" and "Calypso" form a link in our hierarchy, but belong to four different clusters in the ground-truth – due to non-musical taxonomic considerations – despite being musically similar. It is not easy to quantify the performance upper-bound we have between our musical concept detection and the practical expert taxonomies we compare to. Nevertheless, seeing that our generated hierarchies roughly align with ground-truth structures is a good sanity check.

Tags	Accuracy (%)	Silhouette
H_{mood}	45.1	-0.09 $\pm$ 0.05
H_{genre}	49.1	-0.17 $\pm$ 0.04

Table 3. Evaluation of our unsupervised hierarchy against ground-truth clustering of moods and genres tags.

5.3.3 Alignment to various sources

Coming back to the difficult setting of $\mathcal{V}_D$, we finally evaluate our hierarchy of concepts with mixed types, illustrated in Figure 2. As we do not have a ground-truth in this general case, we evaluate whether the edges of our hierarchy – built from $S(\mathcal{V})$ – make sense on other sources of similarity. Specifically, since our data is collected from a streaming service, collaborative filtering embeddings [69] based on listening logs are available to estimate playlist similarities (S_{CF}). Playlists similar in concepts could indeed be expected to be co-listened by users, though popularity biases are also at play. We also leverage playlist titles and expect neighbours to be rather close semantically. To that end, we use a large language model [70] that can embed any text prompt (S_{BERT}), and two music-specialised word embedders for which we average representations of in-vocabulary words of each concept names: $S_{\text{W2V-1}}$ [9] and $S_{\text{W2V-2}}$ [71]. For reference, we generate hierarchies from each domain-specific source, include a random hierarchy, and a *Audio* measure based on CAVs weights, following our observations made in section 3.4.

We compute the average similarity on each hierarchy's edges given those several sources of similarities. Results are provided in Table 1. By construction, each hierarchy maximises performance on the source it was built on. We rather inspect how one source of knowledge transfers to other sources. Collaborative filtering being the canonical way of estimating similarities in the streaming industry our dataset is extracted from and in general [45], we underline that our hierarchy – solely based on audio – closely matches its transfer performances, which is a good result [4]. We have thus shown that our hierarchy transfers to other sources of knowledge (goal 3).

5.4 Qualitative discussion

We retrieve many associations that make sense for humans in $H(S(\mathcal{V}_D))$: blues and jazz; rock and pop; motivation, dancing, running, and party are close to one another – which aligns with previous work [72]. More interestingly, we observe that *"LSD Trip"* is the parent of *"Surf"*, *"Summer of love"*, and *"Stoner Rock"*, effectively associating an activity to a sport, an historic phenomenon, and a music genre. Yet, we also find some association that make sense for the model but not for humans: *e.g. "Rock Christmas"* is also a child of *"LSD Trip"*. This phenomenon is hard to avoid with fully unsupervised methods [53, 73]. We can find many more interesting associations, but we have to beware of confirmation biases, as often in XAI [51, 52, 74]. We provide online visualisations of the results [5].

As observable limitations, $\mathcal{V}_D$ is biased by the streaming platform usage towards French, English, and Brazilian content, which could be addressed in future work with importance sampling. Then, some concept titles are misleading, *e.g.* playlists "City sounds: <city>" are confounded by the taste of their curator, resulting in city concepts being close to the 60's genre, despite this not being obvious solely judging from the title. Finally, as already mentioned, the backbone fails to discriminate linguistic information: *e.g. "Queer Pop"* and *"[Current] Pop"* differ by their lyric theme but are neighbours in the hierarchy.

6. CONCLUSION

Spectrograms are hard to interpret. We propose to extend concept learning to learn hierarchies of music concepts from playlists, with greater flexibility than usual music taggers. Our results should be viewed as complementary to expert music ontologies, as a means to witness how music is organically described by users and editors, and thus to capture new or evolving salient aspects of music.

This topic is novel and further steps are necessary in order to overcome cultural biases of our data, and to discover causal relations for our structure. Future work include leveraging the found hierarchy for dynamic music recommendation – e.g. exploring and switching branches.

[4] Why not use CF embeddings then? As a reminder, they are tied to the dataset, and thus fail goal 2. They are only used for comparison.

[5] `research.deezer.com/concept_hierarchy/`

7. REFERENCES

[1] A. Liberman, F. S. Cooper, D. P. Shankweiler, and M. Studdert-Kennedy, "Why are speech spectrograms hard to read?" *American Annals of the Deaf*, pp. 127–133, 1968.

[2] M. Won, S. Chun, and X. Serra, "Visualizing and understanding self-attention based music tagging," *Machine Learning for Music Discovery Workshop, ICML*, 2019.

[3] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, and R. sayres, "Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV)," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 2668–2677.

[4] C.-K. Yeh, B. Kim, and P. Ravikumar, "Human-centered concept explanations for neural networks," in *Neuro-Symbolic Artificial Intelligence: The State of the Art*, vol. 342, 2022, pp. 337–352.

[5] A. Ghandeharioun, B. Kim, C.-L. Li, B. Jou, B. Eoff, and R. W. Picard, "Dissect: Disentangled simultaneous explanations via concept traversals," *International Conference on Learning Representations (ICLR)*, 2022.

[6] C. J. Cai, E. Reif, N. Hegde, J. Hipp, B. Kim, D. Smilkov, M. Wattenberg, F. Viegas, G. S. Corrado, M. C. Stumpe *et al.*, "Human-centered tools for coping with imperfect algorithms during medical decision-making," in *CHI conference on human factors in computing systems*, 2019, pp. 1–14.

[7] P. Arendsen, D. Marcos, and D. Tuia, "Concept discovery for the interpretation of landscape scenicness," *Machine Learning and Knowledge Extraction*, vol. 2, no. 4, pp. 397–413, 2020.

[8] R. Hennequin, J. Royo-Letelier, and M. Moussallam, "Audio based disambiguation of music genre tags," *International Society of Music Information Retrieval Conference (ISMIR)*, 2018.

[9] S. Doh, J. Lee, T. H. Park, and J. Nam, "Musical word embedding: Bridging the gap between listening contexts and music," *ICML Workshop: Machine Learning for Music Discovery*, 2020.

[10] E. M. Von Hornbostel and C. Sachs, "Classification of musical instruments: Translated from the original german by anthony baines and klaus p. wachsmann," *The Galpin Society Journal*, pp. 3–29, 1961.

[11] M. Antović, "From expectation to concepts: Toward multilevel grounding in musical semantics," *Cognitive Semiotics*, vol. 9, no. 2, pp. 105–138, 2016.

[12] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, "Attribute and simile classifiers for face verification," in *12th International Conference on Computer Vision (ICCV)*. IEEE, 2009, pp. 365–372.

[13] C. H. Lampert, H. Nickisch, and S. Harmeling, "Learning to detect unseen object classes by between-class attribute transfer," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 951–958.

[14] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang, "Concept bottleneck models," in *International Conference on Machine Learning (ICML)*. PMLR, 2020, pp. 5338–5348.

[15] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su, "This looks like that: deep learning for interpretable image recognition," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[16] B. Zhou, Y. Sun, D. Bau, and A. Torralba, "Interpretable basis decomposition for visual explanation," in *European Conference on Computer Vision (ECCV)*, 2018, pp. 119–134.

[17] A. Lucieri, M. N. Bajwa, S. A. Braun, M. I. Malik, A. Dengel, and S. Ahmed, "On interpretability of deep learning based skin lesion classifiers using concept activation vectors," in *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–10.

[18] A. R. Asokan, N. Kumar, A. V. Ragam, and S. S. Sharath, "Interpretability for multimodal emotion recognition using concept activation vectors," *arXiv preprint arXiv:2202.01072*, 2022.

[19] C. Göpfert, Y. Chow, C.-w. Hsu, I. Vendrov, T. Lu, D. Ramachandran, and C. Boutilier, "Discovering personalized semantics for soft attributes in recommender systems using concept activation vectors," *The Web Conference (WWW)*, 2022.

[20] A. Ghorbani, J. Wexler, J. Y. Zou, and B. Kim, "Towards automatic concept-based explanations," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[21] Z. Chen, Y. Bei, and C. Rudin, "Concept whitening for interpretable image recognition," *Nature Machine Intelligence*, vol. 2, no. 12, pp. 772–782, 2020.

[22] E. Strumbelj and I. Kononenko, "An efficient explanation of individual classifications using game theory," *The Journal of Machine Learning Research*, vol. 11, pp. 1–18, 2010.

[23] C.-K. Yeh, B. Kim, S. Arik, C.-L. Li, T. Pfister, and P. Ravikumar, "On completeness-aware concept-based explanations in deep neural networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 20 554–20 565, 2020.

[24] Y. Goyal, A. Feder, U. Shalit, and B. Kim, "Explaining classifiers with causal concept effect (cace)," *arXiv preprint arXiv:1907.07165*, 2019.

[25] H. F. Garcia, A. Aguilar, E. Manilow, and B. Pardo, "Leveraging hierarchical structures for few-shot musical instrument recognition," *International Society of Music Information Retrieval Conference (ISMIR)*, 2021.

[26] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[27] T. Griffiths, M. Jordan, J. Tenenbaum, and D. Blei, "Hierarchical topic models and the nested chinese restaurant process," *Advances in Neural Information Processing Systems (NIPS)*, vol. 16, 2003.

[28] V.-A. Nguyen, J. L. Ying, P. Resnik, and J. Chang, "Learning a concept hierarchy from multi-labeled documents," *Advances in Neural Information Processing Systems (NIPS)*, vol. 27, 2014.

[29] G. N. Lance and W. T. Williams, "A general theory of classificatory sorting strategies: 1. hierarchical systems," *The computer journal*, vol. 9, no. 4, pp. 373–380, 1967.

[30] X. Liu, Y. Song, S. Liu, and H. Wang, "Automatic taxonomy construction from keywords," in *SIGKDD international conference on Knowledge discovery and data mining*, 2012, pp. 1433–1441.

[31] J. Völker and M. Niepert, "Statistical schema induction," in *Extended Semantic Web Conference*. Springer, 2011, pp. 124–138.

[32] V. Anoop, S. Asharaf, and P. Deepak, "Unsupervised concept hierarchy learning: a topic modeling guided approach," *Procedia computer science*, vol. 89, pp. 386–394, 2016.

[33] P. Heymann and H. Garcia-Molina, "Collaborative creation of communal hierarchical taxonomies in social tagging systems," Stanford, Tech. Rep., 2006.

[34] D. Benz, A. Hotho, and G. Stumme, "Semantics made by you and me: Self-emerging ontologies can capture the diversity of shared knowledge," in *Web Science Conference (WebSci09)*, 2010.

[35] A. Ross and F. Doshi-Velez, "Benchmarks, algorithms, and metrics for hierarchical disentanglement," in *International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 9084–9094.

[36] R. Jenatton, J. Mairal, G. Obozinski, and F. R. Bach, "Proximal methods for sparse hierarchical dictionary learning," in *International Conference on Machine Learning (ICML)*, 2010.

[37] E. Bart, I. Porteous, P. Perona, and M. Welling, "Unsupervised learning of visual taxonomies," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2008, pp. 1–8.

[38] P. Hase, C. Chen, O. Li, and C. Rudin, "Interpretable image recognition with hierarchical prototypes," in *AAAI Conference on Human Computation and Crowdsourcing*, vol. 7, 2019, pp. 32–40.

[39] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[40] J. Adebayo, J. Gilmer, M. Muelly, I. Goodfellow, M. Hardt, and B. Kim, "Sanity checks for saliency maps," *Advances in neural information processing systems (NeurIPS)*, 2018.

[41] Z. C. Lipton, "The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery." *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[42] D. Janzing, L. Minorics, and P. Blöbaum, "Feature relevance quantification in explainable ai: A causal problem," in *International Conference on artificial intelligence and statistics*. PMLR, 2020, pp. 2907–2916.

[43] A. Mahinpei, J. Clark, I. Lage, F. Doshi-Velez, and P. Weiwei, "Promises and pitfalls of black-box concept learning models," *ICML Workshop on Theoretic Foundation, Criticism, and Application Trend of Explainable AI*, 2021.

[44] S. Krishna, T. Han, A. Gu, J. Pombra, S. Jabbari, S. Wu, and H. Lakkaraju, "The disagreement problem in explainable machine learning: A practitioner's perspective," *arXiv preprint arXiv:2202.01602*, 2022.

[45] D. Afchar, A. B Melchiorre, M. Schedl, R. Hennequin, E. V. Epure, and M. Moussallam, "Explainability in music recommender systems," in *AI Magazine*, 2022.

[46] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *Nature Machine Intelligence*, vol. 2, no. 11, pp. 665–673, 2020.

[47] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins *et al.*, "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[48] D. Afchar, V. Guigue, and R. Hennequin, "Towards rigorous interpretations: a formalisation of feature attribution," in *International Conference on Machine Learning (ICML)*. PMLR, 2021, pp. 76–86.

[49] P. Hase, H. Xie, and M. Bansal, "The out-of-distribution problem in explainability and search methods for feature importance explanations," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 34, 2021.

[50] D. Slack, S. Hilgard, E. Jia, S. Singh, and H. Lakkaraju, "Fooling lime and shap: Adversarial attacks on post hoc explanation methods," in *AAAI/ACM Conference on AI, Ethics, and Society*, 2020, pp. 180–186.

[51] H. Kaur, H. Nori, S. Jenkins, R. Caruana, H. Wallach, and J. Wortman Vaughan, "Interpreting interpretability: understanding data scientists' use of interpretability tools for machine learning," in *CHI conference on human factors in computing systems*, 2020, pp. 1–14.

[52] J. Dinu, J. Bigham, and J. Z. Kolter, "Challenging common interpretability assumptions in feature attribution explanations," *NeurIPS Workshop: ML Retrospectives, Surveys & Meta-Analyses (ML-RSA)*, 2020.

[53] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 4114–4124.

[54] L. Sixt, M. Schuessler, O.-I. Popescu, P. Weiß, and T. Landgraf, "Do users benefit from interpretable vision? a user study, baseline, and dataset," in *International Conference on Learning Representations (ICLR)*, 2022.

[55] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong, "Interpretable machine learning: Fundamental principles and 10 grand challenges," *Statistics Surveys*, 2021.

[56] H. Chen, J. D. Janizek, S. Lundberg, and S.-I. Lee, "True to the model or true to the data?" *ICML Workshop on Human Interpretability in Machine Learning*, 2020.

[57] J. Pearl, "Causal inference in statistics: An overview," *Statistics Surveys*, vol. 3, pp. 96 – 146, 2009.

[58] S. Bozinovski and A. Fulgosi, "The influence of pattern similarity and transfer of learning upon training of a base perceptron b2.(original in croatian: Utjecaj slicnosti likova i transfera ucenja na obucavanje baznog perceptrona b2)," in *Proc. Symp. Informatica*, 1976, pp. 3–121.

[59] O. Celma, P. Herrera, and X. Serra, "Bridging the music semantic gap," *International Semantic Web Conference*, 2006.

[60] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," *International Society of Music Information Retrieval Conference (ISMIR)*, 2021.

[61] J. Pons and X. Serra, "musicnn: Pre-trained convolutional neural networks for music audio tagging," *Late breaking/demo session of the International Society for Music Information Retrieval Conference (LBD-ISMIR)*, 2019.

[62] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *International Conference on Music Information Retrieval (ISMIR)*, 2011.

[63] T. Lombrozo, "Explanatory preferences shape learning and inference," *Trends in Cognitive Sciences*, vol. 20, no. 10, pp. 748–759, 2016.

[64] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[65] S. P. Borgatti and M. G. Everett, "A graph-theoretic perspective on centrality," *Social networks*, vol. 28, no. 4, pp. 466–484, 2006.

[66] B. Gao, E. Dellandréa, and L. Chen, "Music sparse decomposition onto a midi dictionary of musical words and its application to music mood classification," in *International Workshop on Content-Based Multimedia Indexing (CBMI)*. IEEE, 2012, pp. 1–6.

[67] M. Girvan and M. E. Newman, "Community structure in social and biological networks," *Proceedings of the national academy of sciences*, vol. 99, no. 12, pp. 7821–7826, 2002.

[68] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[69] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *International Conference on Data Mining*. IEEE, 2008, pp. 263–272.

[70] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Conference on Empirical Methods in Natural Language Processing*. ACL, 2019.

[71] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, "Multimodal metric learning for tag-based music retrieval," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 591–595.

[72] K. M. Ibrahim, J. Royo-Letelier, E. V. Epure, G. Peeters, and G. Richard, "Audio-based auto-tagging with contextual tags for music," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 16–20.

[73] F. Dalvi, A. R. Khan, F. Alam, N. Durrani, J. Xu, and H. Sajjad, "Discovering latent concepts learned in bert," in *International Conference on Learning Representations (ICLR)*, 2022.

[74] R. S. Nickerson, "Confirmation bias: A ubiquitous phenomenon in many guises," *Review of General Psychology*, vol. 2, no. 2, pp. 175–220, 1998.

MULTI-OBJECTIVE HYPER-PARAMETER OPTIMIZATION OF BEHAVIORAL SONG EMBEDDINGS

Massimo Quadrana
Apple
mquadrana@apple.com

Antoine Larreche-Mouly
Apple
alarreche@apple.com

Matthias Mauch
Apple
mmauch@apple.com

ABSTRACT

Song embeddings are a key component of most music recommendation engines. In this work, we study the hyper-parameter optimization of behavioral song embeddings based on Word2Vec on a selection of downstream tasks, namely next-song recommendation, false neighbor rejection, and artist and genre clustering. We present new optimization objectives and metrics to monitor the effects of hyper-parameter optimization. We show that single-objective optimization can cause side effects on the non optimized metrics and propose a simple multi-objective optimization to mitigate these effects. We find that next-song recommendation quality of Word2Vec is anti-correlated with song popularity, and we show how song embedding optimization can balance performance across different popularity levels. We then show potential positive downstream effects on the task of play prediction. Finally, we provide useful insights on the effects of training dataset scale by testing hyper-parameter optimization on an industry-scale dataset.

1. INTRODUCTION

Modern Recommendation Systems (RS) rely on embedding vectors to represent the latent user and item factors [1]. They can be employed for several downstream applications, ranging from song recommendation in radio stations or playlists [2–7], search [8, 9], tagging [10], to the generation of artist and genre representations for annotation and recommendation tasks [11–13]. Embeddings are usually generated in the early stages of complex RS pipelines, often through self-supervised methods like Word2Vec [14], which come with default hyper-parameters tuned on non-RS tasks (e.g., NLP).

Practitioners and researchers in the field hence need feasible optimization objectives and reliable metrics to compare against when optimizing embeddings. Recent research shows that hyper-parameter optimization can significantly improve recommendation quality [15,16]. While prior work mainly focuses on recommendation tasks, it is worth considering other tasks like false positive rejection and clustering during optimization.

Additionally, embedding spaces are used as a source of knowledge and interpretation either through visualization tools [17, 18], or by using relationships between items from a statistical standpoint or by leveraging information outside the input data [19–22]. We believe that music-RS practitioners and researchers could benefit from a deeper understanding of the behavior of song embedding optimization in relation to important factors such as song popularity. This could assist them in handling some emerging algorithmic biases like the popularity bias [23–25].

1.1 Contributions

In this work, we provide a strong framework within which it is possible to monitor the performance of embedding models and evaluate the potential of the embedding model to adapt to new tasks. Our work presents a general methodology that can be applied to any embedding system and not only to Word2Vec. The main contributions are:

- We define metrics and optimization objectives for three relevant tasks: next-song recommendation, false neighbor rejection, and genre and artist clustering.
- We demonstrate experimentally that single-objective optimization can have negative side effects on the non optimized metrics and propose a multi-objective optimization approach to combine recommendation and clustering objectives effectively.
- We show that next-song recommendation quality and song popularity are anti-correlated and reveal that song embedding optimization can balance performance across different popularity levels.
- We show the potential positive downstream effects on the task of play prediction revealing that the benefits of song embedding optimization extend beyond the tasks considered during optimization.
- Finally, we study embedding optimization at scale on an internal dataset of billions of listening events and show that increasing training dataset size allows for better configurations at the expense of longer optimization times.

2. METHOD

We consider song embeddings based on Word2Vec. This model was originally created to represent words in an

© M. Quadrana, A. Larreche-Mouly, and M. Mauch. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. Quadrana, A. Larreche-Mouly, and M. Mauch, "Multi-objective Hyper-parameter Optimization of Behavioral Song Embeddings", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

English corpus through a self-supervised shallow neural network trained to learn dense vector representations of the words from sentences based on the surrounding words [14]. The same principle is now commonly used in recommender systems to compute item embeddings from user interaction sequences like site sessions or playlists [25–28].

We study song embedding optimization with respect to four main tasks: next-song prediction, false neighbor rejection, artist clustering, and genre clustering. We define in this section the task and metrics that we used both for embedding optimization and evaluation of its effects on the resulting embedding space.

2.1 Next-Song Prediction

We choose next-song prediction as our primary target task. Next-item prediction is a common recommendation task whose goal is to predict the next item the user will interact with given some context [29]. In music recommenders, this often translates to predicting the next song given the history of songs played by the user [30]. To ensure the contextual relevance of recommendations, the past play history is often limited to the past few played songs or sessions, although more flexible solutions that look beyond a fixed horizon exist [31].

Since our main goal is not to build the most accurate next-song predictor, but rather to measure effects of optimization on the embedding space as directly as possible, we simplified next-song prediction to the extreme case of predicting the next played song based only on the *immediately preceding* song. For each song in the evaluation set (the *target* song henceforth), we consider the song the user played before it as the *query* song. For each (query, target) pair, we simply retrieve the top-100 exact nearest neighbors of the query and compute the average HitRate and Normalized Discounted Cumulative Gain (NDCG) on the top-100 ranked neighbors. HitRate is the fraction of times the target song is contained in the nearest neighbor of the query, while NDCG also accounts for the rank of the correct next-song within the predictions [32].

In order to ensure that improvements are due to true generalisation and not to overfitting, we split the evaluation into in-set and out-of-set. In *in-set evaluation*, we mask the last song in every training sequence and use second-to-last song as the query to compute next-song prediction metrics. In *out-of-set evaluation*, we hold-out the whole play sequences in the validation and test sets, and use every ordered pair of songs therein contained to compute the next-song prediction metrics. We use the average in-set and out-of-set HitRate and NDCG values in our experiments. The precise definition of a sequence varies between our experimental datasets and will be discussed in Section 3.

2.2 False Neighbor Rejection

Since high quality nearest neighbors are essential to providing a good recommendations, we are interested in evaluating the effectiveness of filtering out *spurious* song neighbors, i.e., songs that are in the closest neighborhood of another song merely by chance and not due to real behavioral or metadata similarity.

Query song	Hard Negative Neighbors
Paradise - Coldplay	Snowman - Sia Immigrant Song - Led Zeppelin Basket Case - Green Day
Smells Like Teen Spirit - Nirvana	Power - Kanye West Ob-La-Di, Ob-La-Da - The Beatles Rock Your Body - Justin Timberlake
Carry On Wayward Son - Kansas	Natural - Imagine Dragons Rap God - Eminen It Ain't Me - Kygo & Selena Gomez

Table 1: Examples of hard negative neighbors in Stream.

To this end we define the task of False Neighbor Rejection. We used a chi-squared X^2 test to identify false neighbors in the play sequence dataset [33]. The X^2 test compares the observed co-occurrence frequencies of every pair of songs in the listening sequences against the expected co-occurrence frequencies in case of independence. It is thus suitable to detect pairs of songs that appear in sequence by chance, and not because they are strongly related due to behavioral factors or metadata. In practice, we compute the X^2 coefficient for each unordered bigram of songs in the play sequence dataset. We consider as a *hard negative neighbor* of a song any other song having high co-occurrence but chi-squared statistic below a significance threshold [1] .

One limitation of this approach is that it requires a large number of events to be able to find frequently co-occurring song pairs by chance. We were therefore able to run this analysis only on the Stream dataset, from which we retrieved 5M hard-negative pairs, with an average of 92 hard-negatives per song. Some examples of hard negative neighbors are shown in Table 1. We define the HardNeg metric as the proportion of hard negatives for the top-100 nearest neighbors of every song (the smaller the better). We will use it as a safeguard metric agains undesired side effects of song embedding optimization.

2.3 Artist and Genre Clustering

We are interested in measuring how strongly classes such as artists and genres cluster together in a given embedding space. We introduce here the concept of Local Genre Coherence of the embedding space as the average fraction of songs in nearest neighbors set having the same primary genre as the query song. We similarly define the Local Artist Coherence as the average fraction of nearest neighbors belonging to the same artist as the query song. For example, an embedding space has Local Genre Coherence $= 0.5$ if on average 50% of the nearest neighbors of each song have its same primary genre.

Computing Local Coherence metrics is a costly operation since it requires us to inspect the nearest neighbors of every embedded song. We instead propose to use as *proxy metric* during optimization the much cheaper Caliński-

[1] We empirically chose 10^{-7} times the sum of all song occurrences in the dataset as threshold.

Dataset	Stream	Stream-1%	LFM-1b
Songs	66M	17M	31M
Events	255B	2.6B	1B
Sequences	0.9B	9.3M	120K
Sequence length: mean	277	276	9044
Sequence length: 25th percentile	16	16	1138
Sequence length: median	29	27	3930
Sequence length: 75th percentile	58	57	16155

Table 2: The statistics of the datasets.

Parameter	Default	Range	Description
d	100	$[25, 200]$	Embedding vector dimension
L	5	$[1, 40]$	Sliding window max length
α	0.75	$[-1.0, 1.0]$	Negative sampling exponent
N	5	$[1, 100]$	Number of negative samples
λ	0.025	$[0.001, 0.1]$	Initial learning rate

Table 3: Default Word2Vec hyper-parameters and their respective optimization ranges.

Harabasz index or Variance Ratio Criterion (VRC) [34] over artist and genre clusters in the embedding space. The VRC is the ratio of the sum of between-clusters dispersion and of within-cluster dispersion for all genre clusters, where dispersion is the sum of squared distances. Higher VRC values correspond to better separation among clusters, which is a desirable property because it reduces the chances of uncontrolled, cross-artist and cross-genre pollution in recommendations.

We study the optimization of embedding spaces with respect to clustering metrics and their relation to recommendation quality in Section 4.3 and 4.4.

3. DATASETS

We run our experiments on two datasets. The first is a large-scale proprietary dataset of anonymized streaming listening sequences and playlists. We call this dataset Stream. The second is the LFM-1b dataset which contains 1B time-stamped listening events collected from Last.fm [35].

The main statistics of each dataset are shown in Table 2. The sequence length distribution differs significantly between the two datasets. Stream contains large numbers of both long listening suquences and short playlists (the median length is 58, but mean is 277), while LFM-1b contains mainly long listening sequences (the median length is 3930). This will have an impact on the optimal hyper-parameters discovered for each dataset.

In order to compute artist and genre clustering metrics we need song-level artist and genre annotations. For the Stream dataset, each song in is mapped to the corresponding primary genre and artist through an internal auto-tagging pipeline. LFM-1b already comes with song to artist mappings, however song-level genre annotations are not directly available. We hence mapped each song to the *first* Freebase artist genre from the LFM-1b User Genre Profile dataset [36]. We are aware of the noise introduced by this approximation, yet we believe it provides a useful contribution towards the reproducibility of our experiments.

We partition both datasets by randomly sampling sequences without replacement with proportions 98/1/1 for training, validation and test respectively. However, Stream still contains 7500 times more sequences than LFM-1b, which makes optimization at full-scale impractical. We hence downsample the sequences in the dataset using a 1% rate. We will refer to this subsampled dataset to as Stream-1%. We will investigate how to scale the optimization up to the full dataset in more detail in Section 4.7.

4. EXPERIMENTS

We report here the optimization of song embeddings for the tasks defined in the previous section. We optimize the hyper-parameters of skip-gram Word2Vec by running Bayesian Hyper-Parameter Optimization (HPO) [37], initialized with 10 iterations of Random Search [38] before running Bayesian Search until convergence. Similarly to previous work, we constrained all training times to be approximately equal to the default Word2Vec configuration [16]. This ensures a fair comparison among trials, and prevents the optimizer from discovering configurations that are impractical to train at large scales.

4.1 Background on Word2Vec

Both variants of Word2Vec, Skipgram and the Continuous Bag of Words (CBOW), are self-supervised shallow neural network models trained by minimizing the categorical cross-entropy loss with approximation softmax [14]. Here we consider Skipgram with negative sampling for its superior computational efficiency [39].

The hyper-parameters of Word2Vec, their defaults and optimization ranges are detailed in Table 3. In short, d is the embedding size, L is the maximum window length, α controls the negative sampling (uniform sampling for $\alpha = 0$, popularity sampling $\alpha > 0$, inverse popularity sampling for $\alpha < 0$), N is the number of negative samples used to approximate the softmax and λ is the learning rate.

4.2 Optimizing for Next-Song Recommendation

We first analyze the optimization of next-song recommendation quality by running HPO with the HitRate objective. In line with previous works, we observe significant improvements with respect to the default configurations on both the tested datasets (second line of Table 4). On Stream-1%, HitRate improves by 5% and NDCG by 7%, while HardNeg reduces drastically by 40%. The reduction in HardNeg ensures that the improvement in recommendation accuracy does not comes at the expense of more false positive neighbors. On LFM-1b we similarly observe +19% HitRate and +31% NDCG. The results on this task are in line with previous findings [15, 16].

We are now also able to monitor the effects of optimization on Genre and Artist Clustering metrics. While, $\text{VRC}_{\text{Genre}}$ and $\text{VRC}_{\text{Artist}}$ slightly increase on Stream, they both reduce on LFM-1b. This suggests that recommendation and clustering metrics may be slightly anti-correlated

		Stream-1%					LFM-1b			
Opt. Type	Objective	HitRate	NDCG	HardNeg	$\text{VRC}_{\text{Genre}}$	$\text{VRC}_{\text{Artist}}$	HitRate	NDCG	$\text{VRC}_{\text{Genre}}$	$\text{VRC}_{\text{Artist}}$
N/A	N/A	0.3538	0.1378	0.0180	927	3.373	0.3771	0.1562	520	4.141
Single-obj	HitRate	$0.3725^{\uparrow}$	$0.1482^{\uparrow}$	$0.0108^{\uparrow}$	1033	4.437	$0.4492^{\uparrow}$	$0.2050^{\uparrow}$	382	3.457
	$\text{VRC}_{\text{Genre}}$	0.3000	0.1218	0.0146	2006	5.521	0.3776	0.1609	1293	8.603
	$\text{VRC}_{\text{Artist}}$	0.3402	0.1336	0.0175	1397	**34.347**	0.3532	0.1488	**1444**	**94.476**
Multi-obj	$\lambda_{\text{Genre}}(0.01)$	$\mathbf{0.3772}^{\uparrow}$	$\mathbf{0.1586}^{\Uparrow}$	$0.0084^{\uparrow}$	1495	5.216	$0.4428^{\uparrow}$	$\mathbf{0.2092}^{\uparrow}$	623	5.955
	$\lambda_{\text{Genre}}(0.1)$	$0.3742^{\uparrow}$	$0.1515^{\Uparrow}$	$0.0096^{\uparrow}$	1617	5.771	$0.4067^{\uparrow}$	$0.1698^{\uparrow}$	997	7.298
	$\lambda_{\text{Artist}}(0.01)$	$0.3699^{\uparrow}$	$0.1520^{\Uparrow}$	$\mathbf{0.0057}^{\Uparrow}$	1166	4.580	$0.4458^{\uparrow}$	$0.1883^{\uparrow}$	487	5.469
	$\lambda_{\text{Artist}}(0.1)$	0.3331	0.1332	$0.0093^{\uparrow}$	**2233**	7.422	$\mathbf{0.4537}^{\uparrow}$	$0.1998^{\uparrow}$	619	6.022

Table 4: Results of hyper-parameter optimization on both datasets. The first line reports the metrics for the default configuration. Best results are in **bold**. For HR/NDCG/HardNeg only: $^{\uparrow}$ and $^{\Uparrow}$ denote stat. sig. improvement over the default and the best single-objective configurations respectively (paired t-test at $p < 0.01$ with Bonferroni correction).

for this dataset. We will investigate this effect further in Section 4.3 and 4.4.

4.3 Optimising for Genre and Artist Clustering

We now analyze the optimization of embeddings with respect to the Local Genre and Artist Coherence by using the Variance Ratio Criterion as a proxy objective.

We first show that VRC is a suitable proxy by comparing it against Local Coherence metrics on Stream-1%. We computed Local Genre Coherence on 500 songs with at least 10k plays selected using stratified sampling across the top-10 played genres and using 50 nearest neighbors per song. Figure 1 (left) shows strong positive correlation between average Local Genre Coherence and $\text{VRC}_{\text{Genre}}$ of 5 embedding spaces generated with different hyper-parameters. We similarly computed Local Artist Coherence by sampling 125 artists having at least 25 songs using stratified sampling by artist popularity to account for popularity biases. We then average artist coherence score over 5 songs per artist and 50 nearest neighbors per song. Figure 1 (right) shows positive correlation between Local Artist Coherence and $\text{VRC}_{\text{Artist}}$ of 5 embedding spaces generated with different hyper-parameters.

Table 4 shows that the optimal configuration for genre clustering on Stream-1% doubles the $\text{VRC}_{\text{Genre}}$ score compared to the default hyper-parameters, but with a significant reduction in terms of HitRate and NDCG. The next-song recommendation quality is thus badly affected by the myopic optimization of genre coherence. This effect is slightly less evident for artist clustering optimization, in which we were able to obtain 10 fold greater $\text{VRC}_{\text{Artist}}$ with negligible decrease in HitRate and NDCG. In both cases, HardNeg is not significantly affected.

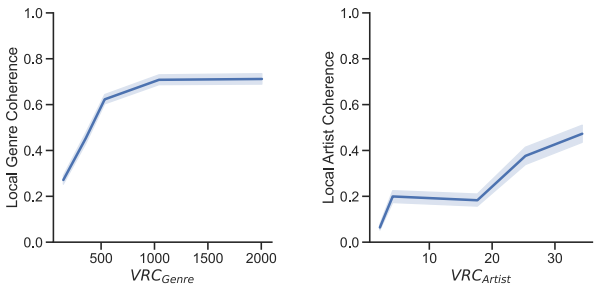


Figure 1: Local Coherence for genres (left) and artists (right) on Stream-1%. Means and 95% Confidence Intervals are shown.

We observe similar effects on LFM-1b, where significant improvements in $\text{VRC}_{\text{Genre}}$ and $\text{VRC}_{\text{Artist}}$ correspond to non significant improvement (genre clustering) or significant deterioration (artist clustering) of next-song recommendation metrics.

For both datasets, the optimal configurations have significantly worse next-song recommendation quality than the optimal one found by single-objective next-song recommendation. This suggests that optimizing embeddings for clustering alone can seriously harm the recommendation quality. In the next section we tackle the problem of optimizing both objectives simultaneously.

4.4 Multi-objective Optimization

High quality next-song recommendation and genre/artist clustering are both desirable properties of the embedding space but, as we have just observed, optimizing for a single objective can harm others. We investigate here the simultaneous optimization of both objectives through Multi-Objective Hyper-parameter Optimization (MOHPO).

The simplest approach to MOHPO is scalarization, which transforms a multi-objective goal into a single-objective one. Examples of scalarization are the weighted sum, Tchebycheff or ϵ-constraint approaches [40]. Advanced MOHPO techniques extend evolutionary algorithms and Bayesian Optimization to explicitly handle different trade-offs between the multiple objectives [40]. The choice of the optimal MOHPO method is beyond the scope of this work, hence we opted for Bayesian Optimization with scalarization since it fits easily into our existing optimization framework.

We focus on the joint optimization of HitRate and $\text{VRC}_{\text{Genre}}$. The same reasoning holds for HitRate and $\text{VRC}_{\text{Artist}}$ and it is omitted for space reasons. Since these metrics have widely different scales, we first define the relative improvement in $\text{VRC}_{\text{Genre}}$ of a new hyper-parameter

configuration t with respect to the default configuration:

$$\text{rVRC}_{\text{Genre}}{}^{(t)} = \frac{\text{VRC}_{\text{Genre}}{}^{(t)} - \text{VRC}_{\text{Genre}}{}^{(def)}}{\text{VRC}_{\text{Genre}}{}^{(def)}} \quad (1)$$

where $\text{VRC}_{\text{Genre}}{}^{(def)}$ is the $\text{VRC}_{\text{Genre}}$ for the default configuration. The new scalarized objective is the simple convex combination of the two objectives:

$$\lambda_{\text{Genre}}(\alpha) = \alpha\text{HitRate}^{(t)} + (1-\alpha)\text{rVRC}_{\text{Genre}}{}^{(t)} \quad (2)$$

where α controls the relative weight of next-song recommendation and genre clustering objectives in the optimization.

We tried values of $\alpha = 0.01$ and 0.1. Results shown in Table 4 highlight the effectiveness of this approach on Stream-1%. While there is not a single solution that performs best on all metrics, the configuration found with objective $\lambda_{\text{Genre}}(0.01)$ *dominates* the best solution discovered by next-song optimization on all metrics, with statistically better NDCG (+7%). The configuration found with objective $\lambda_{\text{Artist}}(0.01)$ also achieves statistically better HardNeg, with a reduction of 68%, at the expense of lower clustering scores. In both cases increasing the value of α to 0.1 we can effectively trade some next-song prediction accuracy for better genre and artist clustering quality. The optimal setup depends on the final application.

Similarly, on LFM-1b we observe that the best next-song recommendation strategies are found through multi-objective optimization, although no solution entirely dominates any single-objective this time. This fact can be partly attributed to the noisy genre annotations we have available for this dataset.

These results highlight the effectiveness of combining recommendation and clustering objectives, which can mutually benefit from each other if combined properly.

4.5 Insights on Song Popularity

The effects of popularity on recommendations are subject of intense research activity within the research community [23–25]. We contribute here by studying the effects of next-song recommendation HPO on (query, target) song pairs belonging to various buckets of popularity.

We first categorize songs into buckets of popularity, each of which comprises 20% of the total listening events in the dataset, being 0 the smallest bucket with most popular songs and 4 the largest bucket containing the least popular ones [2]. We randomly sample 1k songs per bucket pair and the aggregate HitRate based on query *and* target bucket. For the sake of space, we analyze the Stream dataset only here. The results on LFM-1b are available in the Supplementary Material.

Figure 2a shows the values of HitRate by bucket pair for the default configuration of Word2Vec. We first observe that the recommendation quality is localized to the same or nearest popularity bucket to which the query song belongs. As the query and target begin to differ in popularity, HitRate drops. We clearly see that HitRate is anti-correlated with popularity. One possible explanation is that popular songs by definition give less information about the user tastes and hence have larger sets of plausible "next-songs". On the other hand, the less popular songs often belong to taste "niches" and are thus easier to model.

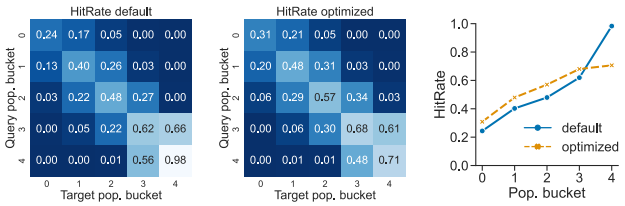


Figure 2: Query-target popularity analysis for Stream: HitRate by popularity bucket for the default configuration (a) and the optimized one (b), and HitRate for song pairs in the same popularity bucket (c). Bucket 0 contains the most popular songs.

Figure 2b shows the values of HitRate by bucket pair for embeddings for the optimal $\lambda_{\text{Genre}}(0.01)$ multi-objective configuration. The behavior of localized performance and anti-correlation observed using the default configuration is still visible. However, we notice a significant difference between the two configurations across the main diagonal. We have highlighted this difference in Figure 2c. We can see that tuning brings significant gains in HitRate at all buckets except for the least popular songs in 4. On both datasets [3] optimization seems to balance recommendation accuracy more across popularity buckets, which is a desirable effect.

4.6 Insights on Play Prediction

As song embeddings are usually used as inputs to other pipelines, we wanted to explore whether embedding optimization had beneficial effects on a different task from the one it was originally run on. For the sake of this experiment, we considered the problem of Play Prediction in seeded radio and autoplay. In seeded radio, the user selects a *seed*, e.g., an artist or a song, and the recommender generates an endless sequence of songs that are related to that seed. Similarly, autoplay generates a stream of music starting from the last played song in an album or a playlist. Seeded radio and autoplay are two prominent product features that allow users to generate streams of songs that are fully machine-learning driven. In both cases, the user organically selects only the seed or the collection from which to start the stream of music, and everything else is left to the recommendation algorithm to decide.

In this context, we study the reaction that users have on the *first* recommended song, since it is where the transition from organic to algorithmic selection happens. A bad listening experience at this point could easily induce users to stop listening entirely. We thus consider the task of pre-

[2] The songs that account for the top-20% of all plays end in bucket 0, the ones for the top-20% to 40% end in bucket 1, etc.

[3] Figures for LFM-1b omitted to fit within space limits.

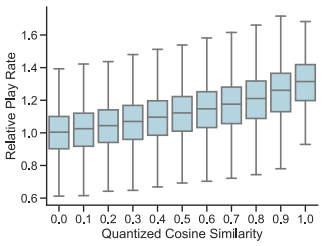


Figure 3: Visualization of the positive correlation between Relative Play Rate and cosine similarity on default embeddings (similarities quantized to the first decimal). Play Rates are scaled relative to cosine similarity $= 0.0$ to preserve business sensitive values.

	Default	Single-obj	Multi-obj
All pairs	0.2055	0.2140	0.2304
Frequent pairs	0.3335	0.3480	0.3717

Table 5: Pearson's correlation between Play Rate and cosine similarity for default and optimized embeddings.

dicting whether the first song generates a play event that lasts at least 30 seconds. We call this task Play Prediction.

Building an accurate predictor is beyond the scope of this paper. We instead measure the correlation between the Play Rate, i.e., the average number of times one organic play is followed by a successful algorithmic play, and the cosine similarity between the embeddings of the recommended and the seed songs. We use Word2Vec embeddings trained on the full Stream dataset. We compare default hyper-parameter configuration to the single-objective and multi-objective configurations having the highest HitRate on this dataset. We compute Play Rates on a proprietary dataset composed of 4.2M song pairs extracted from 184M listening sessions.

Figure 3 shows that there is a strong positive correlation between Play Rate and cosine similarity between embeddings. We observe positive correlation for both the default and optimized embedding spaces (Table 5). The correlation is stronger for the most frequent pairs (≥ 100 occurrences). However, the correlation is stronger for the optimized embeddings than the default ones, $+4\%$ for the best single-objective configuration and $+12\%$ for the best multi-objective configuration. This suggests that embedding optimization can have beneficial effects on tasks different from the one it was initially designed to address. Furthermore, it provides additional evidence on the superiority multi-objective optimization over single-objective one.

4.7 Optimization at Scale

To the best of our knowledge, there exists little literature on hyper-parameter optimization over massive datasets with billions of sequences and events. Previous work on the topic either optimize directly on the full dataset, which is unfeasible at scales like our Stream dataset, or consider a *fixed* subsample rate [16]. However, it is unclear which subsample rate would lead to results that are representative of performance at full-scale.

Sampling	Opt. time	HitRate	NDCG	HardNeg	$\text{VRC}_{\text{Genre}}$	$\text{VRC}_{\text{Artist}}$
N/A	N/A	0.3079	0.1217	0.0126	7229	3.266
1%	28h	0.3432	0.1378	0.0109	7367	3.780
2%	47h	0.3499	0.1421	0.0063	6886	3.374
5%	83h	0.3729	0.1506	0.0167	10410	3.992

Table 6: Results of multiple scale HPO on Stream, with the total optimization time and the best metrics obtained full scale (first line refers to the default configuration). Better performance is obtained at the largest subsample rates at the expense of longer optimization times. All pairwise comparisons on HitRate, NDCG and HardNeg are stat. sig. at $p < 0.01$ using Bonferroni correction, except for HardNeg between default and 1% sampling.

We explore this aspect by running Hyper-Parameter Optimization at increasing training dataset scales. The best hyper-parameters found at each data scale are then used to train the model at full-scale. We limit training time to stay within 25% more the time of training runs with the default hyper-parameters. This allows us to identify the best representative subsample rate while keeping optimization times within reasonable limits.

We split the Stream dataset into 4 overlapping training sets containing 1%, 2%, 5% and 98% of randomly selected sequences respectively, We run Bayesian hyper-parameter search on the first 3 splits and use the best hyper-parameters from each run to train Word2Vec at full 98% scale. For simplicity, we consider just single-objective HitRate optimization. Table 6 shows that there is significant correspondence between the sampling rates used during optimization and the final performance on the full scale dataset. The optimal configuration found at 5% scale is by far the best when evaluated at full-scale on all metrics. However, the total optimization time increases significantly when larger subsamples are used.

5. CONCLUSIONS

In this paper we analyzed the offline optimization of song embeddings through the lens of the tasks they are often employed on downstream. We proposed an effective way to optimize Word2Vec hyper-parameters on recommendation and clustering tasks jointly, with substantial benefits over their respective single-objective optimization variants. We investigated the interactions between next-song recommendation and song popularity. Our results suggest that careful optimization has the desirable property of balancing algorithm performance across popularity buckets. We showed the potential positive effects of optimization on the downstream task of Play Prediction, which provided further evidence on the superiority of multi-objective optimization over single-objective one. Finally, we investigated the effects of optimization at scale, which is particularly relevant for industrial applications.

As future work, we plan to validate these insights with online A/B testing. Our approach can be extended to other tasks, like diversity and fairness, and to the latest findings in Multi-Objective Hyper-parameter Optimization [40,41].

6. AKNOWLEDGEMENTS

We would like to thank Matt Jockers and all the members of the Music Machine Learning team for their help in improving and proofreading this work.

7. REFERENCES

[1] A. Pal, C. Eksombatchai, Y. Zhou, B. Zhao, C. Rosenberg, and J. Leskovec, "Pinnersage: Multi-modal user embedding framework for recommendations at pinterest," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 2311–2320. [Online]. Available: https://doi.org/10.1145/3394486.3403280

[2] G. Bonnin and D. Jannach, "Automated generation of music playlists: Survey and experiments," *ACM Computing Surveys (CSUR)*, vol. 47, no. 2, pp. 1–35, 2014.

[3] C.-W. Chen, P. Lamere, M. Schedl, and H. Zamani, "Recsys challenge 2018: Automatic music playlist continuation," in *Proceedings of the 12th ACM Conference on Recommender Systems*, ser. RecSys '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 527–528. [Online]. Available: https://doi.org/10.1145/3240323.3240342

[4] A. Patwari, N. Kong, J. Wang, U. Gargi, M. Covell, and A. Jansen, "Semantically meaningful attributes from co-listen embeddings for playlist exploration and expansion," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020, Montreal, Canada, October 11-16, 2020*, J. Cumming, J. H. Lee, B. McFee, M. Schedl, J. Devaney, C. McKay, E. Zangerle, and T. de Reuse, Eds., 2020, pp. 527–533. [Online]. Available: http://archives.ismir.net/ismir2020/paper/000125.pdf

[5] R. T. Irene, C. Borrelli, M. Zanoni, M. Buccoli, and A. Sarti, "Automatic playlist generation using convolutional neural networks and recurrent neural networks," in *2019 27th European Signal Processing Conference (EUSIPCO)*, 2019, pp. 1–5.

[6] C. Hansen, R. Mehrotra, C. Hansen, B. Brost, L. Maystre, and M. Lalmas, "Shifting consumption towards diverse content on music streaming platforms," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, ser. WSDM '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 238–246. [Online]. Available: https://doi.org/10.1145/3437963.3441775

[7] J. L. Moore, S. Chen, T. Joachims, and D. Turnbull, "Learning to embed songs and tags for playlist prediction," in *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, F. Gouyon, P. Herrera, L. G. Martins, and M. Müller, Eds. FEUP Edições, 2012, pp. 349–354. [Online]. Available: http://ismir2012.ismir.net/event/papers/349-ismir-2012.pdf

[8] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, "Multimodal metric learning for tag-based music retrieval," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*. IEEE, 2021, pp. 591–595. [Online]. Available: https://doi.org/10.1109/ICASSP39728.2021.9413514

[9] S. Doh, J. Lee, and J. Nam, "Million song search: Web interface for semantic music search using musical word embedding," in *International Society for Music Information Retrieval Conference, ISMIR 2021*. International Society for Music Information Retrieval, 2021.

[10] A. Ferraro, X. Favory, K. Drossos, Y. Kim, and D. Bogdanov, "Enriched music representations with multiple cross-modal contrastive learning," *IEEE Signal Processing Letters*, vol. 28, pp. 733–737, 2021.

[11] K. Chen, B. Liang, X. Ma, and M. Gu, "Learning audio embeddings with user listening data for content-based music recommendation," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3015–3019.

[12] F. Korzeniowski, S. Oramas, and F. Gouyon, "Artist similarity using graph neural networks," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, J. H. L. 0001, A. L. 0001, Z. Duan, J. Nam, P. Rao, P. van Kranenburg, and A. Srinivasamurthy, Eds., 2021, pp. 350–357. [Online]. Available: https://archives.ismir.net/ismir2021/paper/000043.pdf

[13] E. V. Epure, G. Salha, and R. Hennequin, "Multilingual music genre embeddings for effective cross-lingual music item annotation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*. Montreal, Canada: ISMIR, Oct. 2020, pp. 803–810. [Online]. Available: https://doi.org/10.5281/zenodo.4245556

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[15] H. Caselles-Dupré, F. Lesaint, and J. Royo-Letelier, "Word2vec applied to recommendation: Hyperparameters matter," in *Proceedings of the 12th ACM Conference on Recommender Systems*, ser. RecSys

'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 352–356. [Online]. Available: https://doi.org/10.1145/3240323.3240377

[16] B. P. Chamberlain, E. Rossi, D. Shiebler, S. Sedhain, and M. M. Bronstein, "Tuning word2vec for large scale recommendation systems," *Fourteenth ACM Conference on Recommender Systems*, Sep 2020. [Online]. Available: http://dx.doi.org/10.1145/3383313.3418486

[17] D. Smilkov, N. Thorat, C. Nicholson, E. Reif, F. B. Viégas, and M. Wattenberg, "Embedding projector: Interactive visualization and interpretation of embeddings," 2016. [Online]. Available: https://arxiv.org/abs/1611.05469

[18] P. Tovstogan, X. Serra, and D. Bogdanov, "Web interface for exploration of latent and tag spaces in music auto-tagging," in *Proceedings of the 37 th International Conference on Machine Learning; 2020 Jul 13-18; Vienna, Austria.[Vienna]: ICML; 2020.[3 p.]*. ICML, 2020.

[19] M. Baroni, G. Dinu, and G. Kruszewski, "Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 238–247. [Online]. Available: https://aclanthology.org/P14-1023

[20] J. Andreas and D. Klein, "How much do word embeddings encode about syntax?" in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Baltimore, Maryland: Association for Computational Linguistics, Jun. 2014, pp. 822–827. [Online]. Available: https://aclanthology.org/P14-2133

[21] Y. Chen, B. Perozzi, R. Al-rfou, and S. Skiena, "The expressive power of word embeddings," in *In: ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*, 2013.

[22] Z. Akata, S. Reed, D. Walter, H. Lee, and B. Schiele, "Evaluation of output embeddings for fine-grained image classification," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015. [Online]. Available: https://doi.org/10.1109%2Fcvpr.2015.7298911

[23] H. Abdollahpouri, R. Burke, and B. Mobasher, "Managing popularity bias in recommender systems with personalized re-ranking," in *The thirty-second international flairs conference*, 2019.

[24] D. Kowald, M. Schedl, and E. Lex, "The unfairness of popularity bias in music recommendation: A reproducibility study," in *Advances in Information Retrieval*, J. M. Jose, E. Yilmaz, J. Magalhães, P. Castells, N. Ferro, M. J. Silva, and F. Martins, Eds. Cham: Springer International Publishing, 2020, pp. 35–42.

[25] A. Vall, M. Quadrana, M. Schedl, and G. Widmer, "Order, context and popularity bias in next-song recommendations," *International Journal of Multimedia Information Retrieval*, vol. 8, no. 2, pp. 101–113, 2019.

[26] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2016, pp. 1–6.

[27] M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp, "E-commerce in your inbox: Product recommendations at scale," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1809–1818. [Online]. Available: https://doi.org/10.1145/2783258.2788627

[28] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *Proceedings of the Eleventh ACM Conference on Recommender Systems*, ser. RecSys '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 130–137. [Online]. Available: https://doi.org/10.1145/3109859.3109896

[29] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *ACM Comput. Surv.*, vol. 51, no. 4, jul 2018. [Online]. Available: https://doi.org/10.1145/3190616

[30] M. Schedl, "Deep learning in music recommendation systems," *Frontiers in Applied Mathematics and Statistics*, vol. 5, 2019. [Online]. Available: https://www.frontiersin.org/article/10.3389/fams.2019.00044

[31] C. Hansen, C. Hansen, L. Maystre, R. Mehrotra, B. Brost, F. Tomasi, and M. Lalmas, *Contextual and Sequential User Embeddings for Large-Scale Music Recommendation*. New York, NY, USA: Association for Computing Machinery, 2020, p. 53–62. [Online]. Available: https://doi.org/10.1145/3383313.3412248

[32] E. M. Voorhees and D. K. Harman, *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing)*. The MIT Press, 2005.

[33] C. Manning and H. Schutze, *Foundations of statistical natural language processing*. MIT press, 1999.

[34] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1–27, 1974. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/03610927408827101

[35] M. Schedl, "The lfm-1b dataset for music retrieval and recommendation," in *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*, ser. ICMR '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 103–110. [Online]. Available: https://doi.org/10.1145/2911996.2912004

[36] M. Schedl and B. Ferwerda, "Large-scale analysis of group-specific music genre taste from collaborative tags," in *2017 IEEE International Symposium on Multimedia (ISM)*, 2017, pp. 479–482.

[37] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf

[38] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger, Eds., vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

[39] A. Mnih and Y. W. Teh, "A fast and simple algorithm for training neural probabilistic language models," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ser. ICML'12. Madison, WI, USA: Omnipress, 2012, p. 419–426.

[40] F. Karl, T. Pielok, J. Moosbauer, F. Pfisterer, S. Coors, M. Binder, L. Schneider, J. Thomas, J. Richter, M. Lang, E. C. Garrido-Merchán, J. Branke, and B. Bischl, "Multi-objective hyperparameter optimization – an overview," 2022. [Online]. Available: https://arxiv.org/abs/2206.07438

[41] M. Abdolshah, A. Shilton, S. Rana, S. Gupta, and S. Venkatesh, "Multi-objective bayesian optimisation with preferences over objectives," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/a7b7e4b27722574c611fe91476a50238-Paper.pdf

ATEPP: A DATASET OF AUTOMATICALLY TRANSCRIBED EXPRESSIVE PIANO PERFORMANCE

Huan Zhang* **Jingjing Tang*** **Syed Rifat Mahmud Rafee*** **Simon Dixon** **György Fazekas**
School of Electronic Engineering and Computer Science, Queen Mary University of London
huan.zhang@qmul.ac.uk, jingjing.tang@qmul.ac.uk, s.rafee@qmul.ac.uk

ABSTRACT

Computational models of expressive piano performance rely on attributes like tempo, timing, dynamics and pedalling. Despite some promising models for performance assessment and performance rendering, results are limited by the scale, breadth and uniformity of existing datasets. In this paper, we present ATEPP, a dataset that contains 1000 hours of performances of standard piano repertoire by 49 world-renowned pianists, organized and aligned by compositions and movements for comparative studies. Scores in MusicXML format are also available for around half of the tracks. We first evaluate and verify the use of transcribed MIDI for representing expressive performance with a listening evaluation that involves recent transcription models. Then, the process of sourcing and curating the dataset is outlined, including composition entity resolution and a pipeline for audio matching and solo filtering. Finally, we conduct baseline experiments for performer identification and performance rendering on our dataset, demonstrating its potential in generalizing expressive features of individual performing style.

1. INTRODUCTION

Expressive piano performance has long been explored using data-driven approaches for performance analysis and generation. Recently, more attention has been paid to data-hungry, deep learning techniques, for expressive performance rendering and assessment [1, 2]. Large-scale datasets of expressive piano performances that vary across composition, performers, genres, etc. are demanded by researchers who intend to build comprehensive models and compare different architectures.

Most of the current work that studies expressive piano performances [3–5] uses MIDI rather than audio [6, 7], as MIDI provides easier access to performance attributes including tempo, timing, dynamics, and pedalling. However, datasets that consist of recorded MIDI files from computer-controlled pianos are limited in size and variety. Although promising approaches applied such datasets to train models for rendering human-like piano performances from scores [3,4], researchers were unable to explore performer-specific expressiveness or different schools of playing with deep learning models due to data limitations. Few pay attention to applying deep learning techniques to performer-related tasks such as performer identification [8, 9], style-specific performance rendering [4, 10], and performance style transfer.

Our contribution is three-fold: First, we performed an error analysis for piano performance transcription, comparing state-of-the-art models and verifying the reliability of transcribed performances with listening tests in Section 3. Second, we focus on Western classical piano music and release a dataset with sufficient richness and variety for studying expressiveness and styles across different performers. Our released dataset [1] is a performer-oriented dataset that consists of 11742 virtuoso recordings with 1007 hours of music. Instead of recording MIDI files by computer-controlled piano, we collected our dataset by applying state-of-art piano transcription models such as those by Kong et al. [11] and Hawthorne et al. [12] to transcribe the existing audio recordings of piano performances into MIDI files. More details of our dataset and a reproduction pipeline are presented in Sections 3 and 4. Finally, we demonstrate the application of our dataset to the tasks of performer identification and performance rendering in Section 5. Besides these two tasks, ATEPP can also be utilized in analyzing performance attributes [13–15], comparative study of performances and styles [16], as well as performance visualization [17].

2. RELATED WORK

2.1 Dataset Requirements for Piano Performance Research

In order to create a comprehensive dataset addressing tasks like assessment and rendering, we discuss the following requirements, with a comparison of existing datasets in Table 1.

- **Multiple performances of the same music:** With the goal of capturing expressive details and common per-

* Equal contributions

© H. Zhang, J. Tang, S. Rafee, S. Dixon and G. Fazekas. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** H. Zhang, J. Tang, S. Rafee, S. Dixon and G. Fazekas, "ATEPP: A Dataset of Automatically Transcribed Expressive Piano Performance", in *Proc. of the 23nd Int. Society for Music Information Retrieval Conf.,* India, Bengaluru, 2022.

[1] Released dataset and supplementary material (Appendix): https://github.com/BetsyTang/ATEPP. The dataset is made available under Creative Commons Attribution 4.0 International Public License (CC BY 4.0).

Dataset	Size			Artist		Modality		
	Performances	Hours	Compositions	Composers	Performers	Perf. MIDI	Score	CER
SUPRA [18]	478	52	408	111	153	✓	×	×
SMD [19]	50	4.7	50	11	unknown	✓	×	×
MazurkaBL [20]	2000	110	44	1	135	×	100%	✓
Maestro v3.0 [21]	1276	172	864	60	205⋆	✓	×	×
CrestMusePEDB [22]	443	unknown	35	14	12	✓	✓	×
GP† Curated [23]	7236	875	7236	1787	unknown	✓	×	×
ASAP [24]	1068	92	222	15	unknown	✓	100%	×
ATEPP	11742	1007	1580	25	49	✓	43%	✓

Table 1. Overview of major symbolic piano datasets. CER: composition entity resolution. ⋆Number obtained from crawling the Piano-e-Competition website for performer names and aligning with Maestro data. †GP stands for GiantMIDI-Piano.

formance idioms, comparative study of performance requires multiple versions of the same piece of music, ideally by multiple performers. In the past, datasets with very limited numbers of pieces were recorded and organized by researchers, such as the Mozart [25, 26], Schumann Träumerei [27] and Schubert [8] datasets. The non-trivial task of Composition Entity Resolution **(CER)**, involving the process of automatically aligning the complex naming schemes of classical music, is the major challenge of obtaining multiple performances of the same music at a larger scale. We will detail our CER process in Section 4.1. Among the existing datasets, only the CHARM Mazurka dataset [2] offers CER.

- **Representation:** While the audio recording most faithfully documents a performance, complex processing is needed to extract the expressive attributes from the waveform [28]. MazurkaBL [20] contains many precalculated features that are provided for the Mazurka dataset. Meanwhile, MIDI can serve as a mid-level, piano-roll like representation of piano performance actions. The SUPRA [18] dataset contains expressive MIDI digitised from pneumatic piano rolls, while SMD [19], Maestro [21] and CrestMusePEDB [22] all contain MIDI recorded from Yamaha Disklaviers.
- **Repertoire and diversity:** Given that piano performance traditions are largely associated with the Western classical music paradigm, SMD [19], Maestro [21] and CrestMusePEDB [22] all include standard repertoire from Baroque to Late-Romantic era, while GiantMidi-Piano [23] includes non-standard pieces that span 1.7k composers. The CHARM Mazurka dataset is a great example allowing for multiple-performance comparison, however its repertoire consists only of 49 mazurkas by Chopin.
- **Symbolic score:** A high-level representation of the composition score is typically needed in tasks such as performance rendering [3]. Expressive deviations can be observed by comparing with the quantized, dead-pan score. MazurkaBL [20] and ASAP [24] contain symbolic scores in MusicXML format.
- **Size:** Large datasets are essential for training deep neural networks. Among existing datasets, only GiantMidi-Piano [23] has more than 200 hours of piano music.

[2] `http://www.charm.rhul.ac.uk/index.html`, accessed 12 May 2022.

2.2 Automatic Piano Transcription

Empowered by deep learning models, recent automatic piano transcription systems can aid expressive performance research by outputting precise measurements of dynamics, timing and tempo at the note-level. The Onsets and Frames transcription model [29] combined framewise pitch detection with onset detection, to produce a full piano roll with velocity. The High-Resolution model [11] improved precision by regressing the exact timestamp of each note. A recently proposed generic encoder-decoder architecture [12] that exploits language-like modeling achieved model simplicity while retaining performance.

3. TRANSCRIPTION AND POST-PROCESSING FOR EXPRESSIVE PERFORMANCE

3.1 Common Errors Introduced by Transcription

We categorize common transcription errors into the following three rough categories: harmonic error, segmented note, and mis-touched short note. These errors are obtained by transcribing the performances from the Mazurka dataset using the High-Resolution model [11], and then aligning with its symbolic score in MusicXML using the algorithm by Nakamura et al. [30]. This algorithm aligns two signals, reference and performance, using hidden Markov models (HMMs), detects performance errors from the first alignment result, and then employs a merged-output HMM [31] to correct the errors.

1. **HE**: Harmonic errors (fifths and octaves): The most common type of transcription error is falsely detecting or failing to detect notes that are harmonically related to other played or detected notes. Usually these are missing or extra octaves or fifths, and they result from the overlap of the harmonic series of the pitches.
2. **SN**: Segmented notes: One continuous note being transcribed into two segments with a small (<10ms) gap between offset and onset. This error might come from amplitude modulation [32].

Model	HE	SN	MS	Other
High-Resolution [11]	3.2%	1.5%	1.2%	5.6%
OnsetsFrames [29]	4.2%	2.4%	0.1%	6.7%
Seq2Seq [12]	8.1%	2.9%	0.3%	7.9%

Table 2. Transcription note error rate (aligned with symbolic score) on the Mazurka dataset.

3. **MS**: Mis-touched short notes: The spurious, short notes (<16ms) that appear randomly in transcription.

In Table 2, we quantitatively evaluate the presence of these error types on the Mazurka dataset. Given that no performance ground truth exists for this data, we rely on the MusicXML score and the assumption that the performances match the same score version. Among the three most recent deep-learning based transcription models [11, 12, 29], the High-Resolution model [11] makes the fewest errors overall, but generates more short notes compared to the other models.

Besides notewise errors, another factor of concern for transcribed MIDI is the note duration. From the inference output [11], the notes' durations are elongated to achieve a sustain effect that is usually implemented by sustain pedal in reality. Whilst such elongation doesn't make an audible difference in MIDI rendering software, accurate end positions of each note are required in piano performance analysis.

3.2 Joint Note-Pedal Training

With the goal of reconciling the sustain effect from both pedal and keys, as well as achieving more accurate note offsets, we modify the original High-Resolution model [11] with joint note-pedal training. As the first piano transcription model that incorporated the sustain pedal into training, the High-Resolution model trained separate networks for key activity and sustain pedal (with binary velocity), while extending the note offset to match the pedal off timestamp. In joint note-pedal training, 88 keys and 3 pedal channels are combined to output 91 prediction classes with velocity for each channel, and the extension of note offsets is removed. In this case, during training the sustained effect would be conditioned on both key-down duration as well as pedal controls.

As shown in Figure 1, what we model is the key action from the pianist instead of the string damping time of the note (that can either come from the sustain pedal or key action), which deviates from the traditional transcription task. With other training parameters unchanged, the onset F1 score ($tol = 50\,ms$) achieved is 92.1% after 300k iterations, and onsets and offsets evaluation achieved 68.2%. Note that the evaluation results are much lower than the original results, as we are attempting to learn patterns of behaviour that are not present in the audio.

3.3 Score-Alignment and Correction

As described in Section 3.1, a score-performance alignment algorithm [30] is employed to automatically find

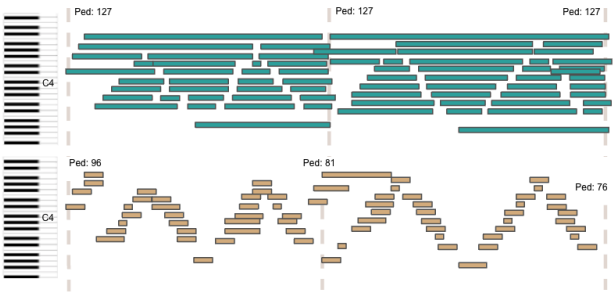


Figure 1. Output pianoroll comparison of the original High-Resolution model (top) and joint note-pedal version (bottom). Dashed lines represent pedal-on messages with velocity.

transcription errors with reference to a score. As a post-processing step, we correct the differences according to the alignment. Extra notes (those in transcription but not in score) are deleted, mismatching notes (aligned with pitch error) are corrected to the pitch given in the score, and missing notes (those in score but not in transcription) are interpolated and written back to MIDI according to the following rule:

$$g(i) = g(i - \Delta_p) + (g(i + \Delta_n) - g(i - \Delta_p))\frac{\Delta_p}{\Delta_p + \Delta_n}, \quad (1)$$

where $g(i)$ is the onset or offset timestamp of the missing note at beat i, and Δ_p, Δ_n represent the beat distances between the missing note and the previous or next existing notes, respectively.

3.4 Listening Evaluation

In order to evaluate the perceptual quality of transcribed piano music, we perform a subjective listening test where participants rate the similarity of reproduced MIDI and the reference recording. In the test, we compare the ground truth with 4 transcribed MIDI renderings: the original High-Resolution transcription system [11] (**C1**), the joint note-pedal model described in Section 3.2 (**C2**), a score-corrected version of C2 as described in Section 3.3 (**C3**), and the language-model transcription system [12] (**C4**). All MIDI performances are rendered using a KAWAI CA49 electric piano and recorded using Zoom H4n Pro Recorder. The recordings are then processed with basic noise-reduction in Audacity.

Participants in the listening test are presented with five *20 s* classical piano excerpts with varying style (Q1-Liszt, Q2-Debussy, Q3-Bach, Q4-Rachmaninov, Q5-Mozart; see appendix for music passages). The test is conducted using the MUSHRA protocol [33], each with 5 recordings (reference plus 4 stimuli). Participants are asked which transcribed stimulus sounded closer to the reference on a 100-point scale. During the test, we explicitly ask participants to ignore the timbral or acoustic differences but make judgements based on the expressive differences between the stimuli such as dynamics and timing.

We collected 1075 ratings from 43 listeners. Half of

	Q1	Q2	Q3	Q4	Q5	Overall
Reference	4.42±0.24	4.17±0.29	4.24±0.31	4.28±0.31	4.46±0.27	4.30±0.12
C1	**4.12±0.38**	3.52±0.37	3.88±0.32	3.60±0.41	3.88±0.4	3.81±0.16
C2	3.83±0.42	**3.86±0.39**	**4.28±0.31**	**3.97±0.42**	**4.06±0.45**	**4.01±0.17**
C3	3.44±0.37	2.96±0.36	3.44±0.35	3.32 ±0.42	3.76±0.41	3.38±0.17
C4	3.88±0.34	2.32±0.47	3.60±0.29	3.84±0.33	3.68±0.37	3.46±0.18

Table 3. Results of listening test. The mean opinion scores (MOS) and 95% confidence intervals are reported.

our listeners reported over 5 years of piano performing experience. Table 3 shows the mean opinion scores (converted to a 5-point scale) from the ratings. According to the Wilcoxon signed-rank test ($p < 0.05$), all stimulus groups differ significantly from the reference. Among the stimuli, C2 is preferred by the listeners, while C3 and C4 have significantly lower ratings compared to the other two groups of stimuli. In free text responses, the score-corrected transcription received negative comments such as *unnatural* and *abrupt*. Consequently, we use the C2, note-pedal jointly trained model for transcribing our dataset. The result also shows a perceptual difference of transcription quality across music styles, demonstrating the bias of transcription models: transcribed fast, arpeggio-heavy passages are rated with lower perceptual quality. But for slow, sparse textures, good transcriptions sound much closer to the reference.

4. DATASET OVERVIEW

4.1 Data Collection and Curation

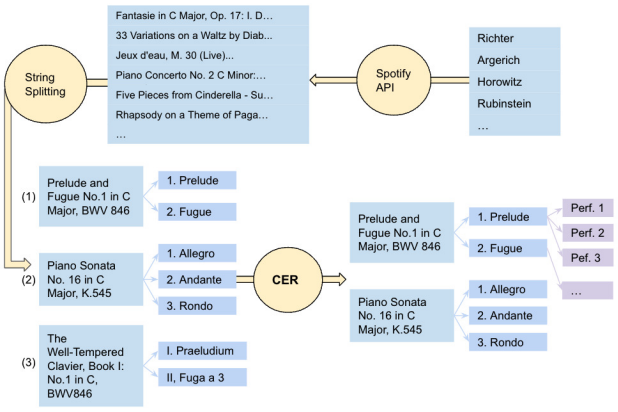


Figure 2. Data curation pipeline.

Our data collection pipeline is presented in Figure 2. Starting with 49 world-renowned pianists, metadata (including composer, performer, album, title and track duration) of their discography was obtained using the Spotify API [3]. After filtering out non-solo keywords such as *concerto* or *trio*, a composition-movement hierarchy was built.

As discussed in Section 2.1, the next challenging step was achieving *Composition Entity Resolution (CER)*, defined as finding out which tracks correspond to the same piece of music, given the variety of naming conventions in classical music. For example, compositions (1) and (3) in Figure 2 actually correspond to the same work, while their title differs so much that a simple string similarity match is not sufficient to resolve them as identical.

We address CER using three steps: **1)** Language-specific mapping. We manually compile a dictionary for interchangeable terms, such as *Prelude* ↔ *Praeludium*. **2)** Unique identifier extraction. Unique information such as key and catalogue number (*Opus*, *BWV*, *K.*, *D.*, etc.) are extracted from the title string. **3)** Fuzzy string matching. For both composition title and movement title, we use normalized Levenshtein distance [34] to compute similarity scores. Note that such string matching is not always reliable, as generic names like *Piano Sonata* are extremely frequent in our discography. Combining the three steps, our composition entity resolution is described in Algorithm 1, where inputs include composition title C and movement title M as well as duration D. Based on the organized metadata, we download each track from a corresponding open source audio at YouTube Music, while the online metadata is again validated by the same CER algorithm.

Algorithm 1 Composition Entity Resolution

```
# UniqueInfo extracts canonical key and composer-specific catalogue number.
for k1, k2 in UniqueInfo(C1, C2) do
    if k1 ≠ k2 then return False
    end if
end for
S_c ← 1 − (Levenshtein(C1, C2))/ max(|C1|, |C2|)
S_m ← 1 − (Levenshtein(M1, M2))/ max(|M1|, |M2|)
S_d ← abs(D1 − D2) / max(D1, D2)
S ← (S_c + S_m)/2 − S_d
return S ≥ 0.6
```

4.2 Audio Matching by Chroma Features

Besides metadata linking using CER, we also match tracks by downloaded audio content to ensure the same piece of music is being performed. Within each group of performances, we apply Chen et al.'s cover song detection algorithm [35] to compare each performance with a reference. We first extract the Harmonic Pitch Class Profile (HPCP) [36] from the reference and the target performance

[3] https://developer.spotify.com/documentation/web-api/

audio. Next, we use the Q_{max} measure, which represents the maximum value of a cumulative matrix computed from the HPCP descriptors of two performances [37]. The similarity between two performances is defined by Eq. 2. We only retain performances whose similarity is larger than 0.9 to build the ATEPP dataset.

$$Sim = 1 - Q_{max}, \quad 0 < Q_{max} < 1 \tag{2}$$

4.3 Applause Filtering with CNN

We also need to filter out any sound that might not be part of a solo piano performance. The most prominent ones are applause and speech from live recordings, which would be transcribed as random pitch. We train a deep learning model based on the Musicnn [38] architecture to filter out any non-solo-piano segment. For each 1 s segment, the probability of non-piano sound is predicted using a binary classifier. In training, a subset of AudioSet [39] with various environmental sounds is used as negative examples, and solo piano recordings are used as positive examples. With a binary tag-gram inferred from the audio, our post-processing step searches for the timestamp of the longest continuous non-solo segment at the beginning and the end to remove from the audio file.

Among the 11742 tracks, 567 of them are detected with starting or ending applause, and were subsequently cleaned. This was followed by a manual verification process by listening to ensure the audio split was accurate.

4.4 MusicXML Score

Given that the musical score is also important for music research, we collect scores in MusicXML format that correspond to our performance data. 228 files are drawn from the ASAP dataset [24], and 90 files from the MuseScore [4] online library, crowd-sourced by the users of MuseScore software. This results in a total of 319 movements, corresponding to 5124 tracks in our dataset (43% of all tracks). The score-performance correspondence is first determined automatically by name matching followed by manual correction.

4.5 Content and Statistics

The ATEPP dataset contains 11742 tracks of 1580 movements. The tracks overlap only 0.2% of the GiantMIDI-Piano dataset [23]. Figure 3 shows a breakdown of the movement-performance distribution. Among 1580 movements, 44% have more than 5 performances, providing us with rich data for studying different interpretations of the same piece of music.

In addition, we show a distribution of the top 25 pianists in our dataset in Figure 4, where Sviatoslav Richter contributes the most data in ATEPP. Composer-wise, solo piano works from 25 Western classical composers are included in our dataset, ranging from the Baroque to the Modern era (a full composer breakdown is given in the appendix).

[4] https://musescore.com/sheetmusic

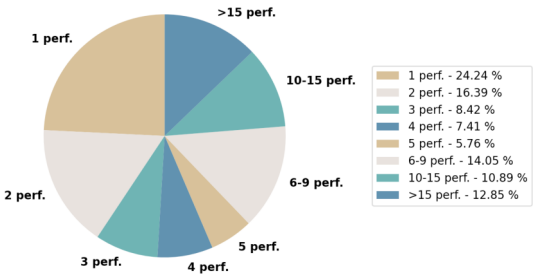


Figure 3. Distribution of movements by number of performances. E.g. 12% of our data have more than 15 performances.

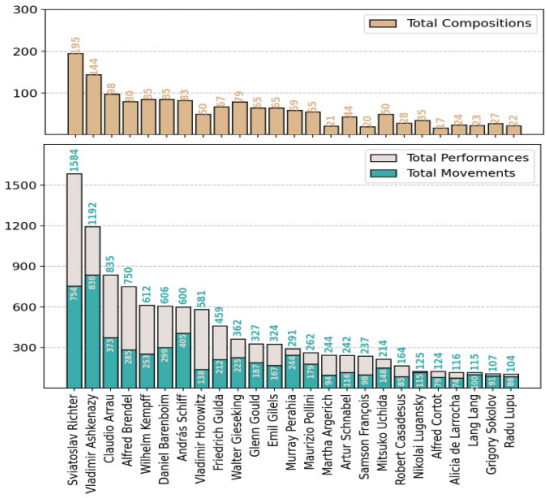


Figure 4. Distribution of the top 25 pianists' performances in the ATEPP dataset.

5. DATASET APPLICATIONS

5.1 Performer Identification

Distinguishing virtuoso performers using computational models has long been studied by researchers who focus on expressive parameters of music performances. Data-driven approaches such as traditional machine learning methods and feature distribution comparison have been applied to this task [8, 9, 40]. However, none of the existing studies have applied deep learning methods to performer identification, due to the lack of large-scale datasets with overlapping performances by different performers.

With the ATEPP dataset, we are now able to train deep neural networks to identify different performers. We choose four subsets from the ATEPP dataset, only considering performers with over 100 performances. For the Mixture subset, we only consider compositions that have more than 15 different performances. We also create three composer-specific subsets (L. Beethoven, F. Chopin, and J. S. Bach) to remove the bias of performer-composer correlation. The train-validate-test split is 8:1:1.

We extracted note-level features including onset and offset time, velocity, and pitch number from the MIDI files without any expression-related preprocessing. We stacked the sequences of those features and input them into a 1D

Subset	Pianists	Size	Acc.	F1-score
Mixture	16	4676	0.47	0.45
Beethoven	12	3078	0.48	0.46
Chopin	5	973	0.55	0.54
Bach	5	1019	0.59	0.55

Table 4. Performer identification results.

convolution neural network (see Appendix for the network details). The model was trained on four subsets at a learning rate of 10^{-4} with a decay weight of 10^{-8}.

With all subsets achieving over 0.45 F1 score in Table 4, our baseline model demonstrates the capability of learning individual performing style if given enough data, as well as generalizing across different compositions.

Moreover, the results from composer-specific datasets show that we can achieve comparable results even when performer-style correlation (e.g. Horowitz's repertoire concentrates on Romantic styles while Gould almost exclusively plays Bach) was removed. Confusion matrices are provided in the appendix as well as precision, recall, and F1-score for each performer.

5.2 Performance Rendering with VirtuosoNet

There has been a growing research interest in quantifying and modelling expressive performance using computational models [1], including understanding of how humans perform [41], automatically generating expressive performances [42], and rendering expressive performances from a score [3]. Again, ATEPP contributes to such tasks by providing expressive performance data on a large scale.

In this study, we show the utility of our dataset using the performance rendering model VirtuosoNet [3]. The model consists of an RNN with a hierarchical attention network to model note, beat and measure level hierarchy of music and a conditional variational autoencoder (CVAE) to model the expressive performance. We selected two pianists' Beethoven performances with over 300 tracks from the ATEPP dataset, and altered the model slightly to render a performance in the style of a particular performer by concatenating a performer label vector to the latent vector. During training, we use the same hyper-parameter setting as the original paper.

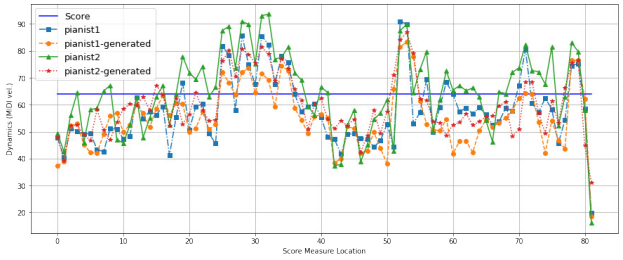


Figure 5. Measure level dynamic variation of Beethoven's Piano sonata No. 3 in C, Op. 2 No. 3, Mvt. II, as performed and generated in the styles of pianists 1 and 2. The flat line depicts the default dynamic level provided in the score file.

The trained model takes a series of note-level score features extracted from MusicXML and a performer name as input, and predicts the corresponding note-level performance features of that performer. Figure 5 shows the real and predicted (pianist1-generated & pianist2-generated) performances of an out-of-sample music, Beethoven's Piano Sonata No. 3 in C Major, Op. 2 No. 3, Mvt. II, by pianist 1 and 2 in terms of dynamics. The flat line represents the default dynamic level of a non-expressive, mechanical rendition of the score.

We can observe that both actual performances and generated performances (in the style of pianist1 and pianist2) tend to deviate from the default interpretation in a similar way; this can be a demonstration of common performance practice. For individual interpretations, we calculated the cross-correlation of the dynamic variation between the actual performance and the generated version using Pearson's correlation coefficient (r). We found that both actual performances from pianist1 and pianist2 are highly correlated ($r > 0.75$) with their generated counterparts, respectively. In addition, we computed the correlation coefficient between the actual performance of pianist1 and the generated performance of pianist2 and vice-versa. Both of them provide a lower correlation coefficient ($r<0.6$). This demonstrates that deep learning architectures are capable of learning some of the expressive techniques of each individual pianist from respective training data.

6. CONCLUSION AND FUTURE WORK

This paper presents ATEPP, a large-scale dataset of 11,742 expressive piano performances by 49 virtuoso pianists. Nearly half of the compositions are provided with scores in MusicXML format. All of the performances were transcribed by a piano transcription model which was trained jointly with pedals and keys. We performed a listening test to evaluate the reliability of the transcription algorithm. To our knowledge, our dataset is the largest dataset of expressive piano performance MIDI with robust metadata of classical music, derived via Composition Entity Resolution (CER). We presented our baseline experiments for performer identification and performer-oriented expressive performance rendering. The results demonstrate that the ATEPP dataset enables us to study expressive features of individual performing styles with deep learning methods.

In the future, we will consider a hierarchical database system that comprehensively links performances with performer, composer, and composition entities through an automatic CER process. A more balanced dataset is planned as well as an extended version with more variety across the skill level of performers.

7. ACKNOWLEDGMENTS

This work is supported by the UKRI Centre for Doctoral Training in Artificial Intelligence and Music, funded by UK Research and Innovation [grant number EP/S022694/1]. Jingjing is also funded by China Scholarship Council (CSC) [grant number 202008440382].

8. REFERENCES

[1] C. E. Cancino-Chacón, M. Grachten, W. Goebl, and G. Widmer, "Computational models of expressive music performance: A comprehensive and critical review," *Frontiers in Digital Humanities*, vol. 5, no. October, pp. 1–23, 2018.

[2] A. Lerch, C. Arthur, A. Pati, and S. Gururani, "Music performance analysis: A survey," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[3] D. Jeong, T. Kwon, Y. Kim, K. Lee, and J. Nam, "Virtuosonet: A hierarchical RNN-based system for modeling expressive piano performance," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 908–915.

[4] A. Maezawa, K. Yamamoto, and T. Fujishima, "Rendering music performance with interpretation variations using conditional variational RNN," *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 855–861, 2019.

[5] P. Ramoneda, M. Miron, and X. Serra, "Piano fingering with reinforcement learning," 2021.

[6] H.-W. Dong, C. Zhou, T. Berg-Kirkpatrick, and J. Mcauley, "Deep performer: Score-to-audio music performance synthesis," in *Proceeding of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[7] Y. Wu, E. Manilow, Y. Deng, R. Swavely, K. Kastner, T. Cooijmans, A. Courville, C.-Z. A. Huang, and J. Engel, "MIDI-DDSP: Detailed control of musical performance via hierarchical modeling," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=UseMOjWENv

[8] S. R. M. Rafee, G. Fazekas, and G. A. Wiggins, "Performer identification from symbolic representation of music using statistical models," in *Proceedings of the International Computer Music Conference (ICMC)*, 2021.

[9] E. Stamatatos and G. Widmer, "Automatic identification of music performers with learning ensembles," *Artificial Intelligence*, vol. 165, pp. 37–56, 2005.

[10] A. Tobudic and G. Widmer, "Learning to play like the great pianists," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 871–876.

[11] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution piano transcription with pedals by regressing onset and offset times," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 29, pp. 3707–3717, 2021.

[12] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, "Sequence-to-sequence piano transcription with transformers," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[13] M. Bernays and C. Traube, "Expressive production of piano timbre: Touch and playing techniques for timbre control in piano performance," *Proceedings of the Sound and Music Computing Conference*, no. August 2013, pp. 341–346, 2013.

[14] A. Robertson, "Decoding tempo and timing variations in music recordings from beat annotations," in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, 2012.

[15] B. H. Repp, "Acoustics, perception, and production of legato articulation on a digital piano," *The Journal of the Acoustical Society of America*, vol. 97, 1995.

[16] C. S. Sapp, "Comparative analysis of multiple performances," in *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, 2007.

[17] M. Grachten, W. Goebl, S. Flossmann, and G. Widmer, "Phase-plane representation and visualization of gestural structure in expressive timing," *Journal of New Music Research*, vol. 38, no. 2, pp. 183–195, Jun. 2009.

[18] Z. Shi, C. S. Sapp, K. Arul, J. McBride, and J. O. Smith, "SUPRA: Digitaizing the stanford university piano roll archive," in *Proceeding of the 20th International Society on Music Information Retrieval (ISMIR)*, 2019.

[19] V. Konz, W. Bogler, and V. Arifi-M, "Saarland music data," *Late-Breaking and Demo Session of the International Conference on Music Information Retrieval (ISMIR)*, 2011.

[20] K. Kosta, O. F. Bandtlow, and E. Chew, "MazurkaBL: Score-aligned loudness, beat, and expressive markings data for 2000 chopin mazurka recordings," in *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'18*, 2018, pp. 85–94.

[21] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C. Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the Maestro dataset," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019, pp. 1–12.

[22] M. Hashida, E. Nakamura, and H. Katayose, "CrestMusePEDB 2nd edition: Music performance database with phrase information," in *Proceedings of the 15th Sound and Music Computing (SMC) Conference*, 2018.

[23] Q. Kong, B. Li, J. Chen, and Y. Wang, "GiantMIDI-Piano : A large-scale midi dataset for classical piano music," *Transactions of the International Society for*

Music Information Retrieval, vol. 5, no. 1, pp. 87–98, 2022.

[24] F. Foscarin, A. Mcleod, P. Rigaux, F. Jacquemard, and M. Sakai, "ASAP : a dataset of aligned scores and performances for piano transcription," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[25] W. Goebl and S. Dixon, "Analysis of tempo classes in performances of mozart sonatas," in *Proceedings of the VII International Symposium on Systematic and Comparative Musicology and III International Conference on Cognitive Musicology*, 2001, pp. 65–76.

[26] A. Tobudic and G. Widmer, "Relational IBL in classical music," *Machine Learning*, vol. 64, no. 1-3, pp. 5–24, 2006.

[27] B. H. Repp, "The dynamics of expressive piano performance: Schumann's "Traumerei" revisited," *The Journal of the Acoustical Society of America*, pp. 641–50, 1996.

[28] G. Widmer, S. Dixon, W. Goebl, E. Pampalk, and A. Tobudic, "In search of the Horowitz factor," *AI Magazine*, vol. 24, no. 3, pp. 111–130, 2003.

[29] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 50–57, 2018.

[30] E. Nakamura, K. Yoshii, and H. Katayose, "Performance error detection and post-processing for fast and accurate symbolic music alignment," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[31] E. Nakamura, N. Ono, Y. Saito, and S. Sagayama, "Merged-output hidden markov model for score following of midi performance with ornaments, desynchronized voices, repeats and skips," in *Proceedings of the 11st Sound and Music Computing Conference 2017, SMC*, 2017.

[32] T. Cheng, S. Dixon, and M. Mauch, "Modelling the decay of piano sounds," in *Proceedings of International Conference on Acoustic, Speech and Signal Processsing (ICASSP)*. IEEE, 2015.

[33] B. Series, "Method for the subjective assessment of intermediate quality level of audio systems," *International Telecommunication Union Radiocommunication Assembly*, 2014.

[34] W. Heeringa, "Measuring dialect pronunciation differences using levenshtein distance," Ph.D. dissertation, University of Groningen, 2005.

[35] N. Chen, W. Li, and H. Xiao, "Fusing similarity functions for cover song identification," *Multimedia Tools and Applications*, vol. 77, no. 2, pp. 2629–2652, 2018.

[36] E. Gomez, "Tonal description of music audio signals," Ph.D. dissertation, University of Pompeu Fabra, 2006.

[37] J. Serrà, X. Serra, and R. G. Andrzejak, "Cross recurrence quantification for cover song identification," *New Journal of Physics*, vol. 11, 2009.

[38] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 637–644, 2018.

[39] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. Channing Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[40] R. Ramirez, E. Maestre, and X. Serra, "Automatic performer identification in commercial monophonic jazz performances," *Pattern Recognition Letters*, vol. 31, no. 12, pp. 1514–1523, 2010.

[41] C. E. Cancino-Chacón, T. Gadermaier, G. Widmer, and M. Grachten, "An evaluation of linear and non-linear models of expressive dynamics in classical piano and symphonic music," *Machine Learning*, vol. 106, no. 6, pp. 887–909, 2017.

[42] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: learning expressive musical performance," in *Neural Computing and Applications*, vol. 32, no. 4, 2018, pp. 955–967.

PDAUGMENT: DATA AUGMENTATION BY PITCH AND DURATION ADJUSTMENTS FOR AUTOMATIC LYRICS TRANSCRIPTION

Chen Zhang[1] **Jiaxing Yu**[1] **LuChin Chang**[1] **Xu Tan**[2] **Jiawei Chen**[3] **Tao Qin**[2] **Kejun Zhang**[1]
[1] Department of Computer Science, Zhejiang University, China
[2] Microsoft Research Asia
[3] South China University of Technology
zc99@zju.edu.cn, yujxzju@gmail.com, changluchin@gmail.com, xuta@microsoft.com,
csjiaweichen@mail.scut.edu.cn, taoqin@microsoft.com, zhangkejun@zju.edu.cn

ABSTRACT

Automatic lyrics transcription (ALT), which can be regarded as automatic speech recognition (ASR) on singing voice, is an interesting and practical topic in academia and industry. ALT has not been well developed mainly due to the dearth of the paired singing voice and lyrics datasets for model training. Considering that there is a large amount of ASR training data, a straightforward method is to leverage ASR data to enhance ALT training. However, the improvement is marginal when training the ALT system directly with ASR data, because of the gap between the singing voice and standard speech data which is rooted in music-specific acoustic characteristics in singing voice. In this paper, we propose PDAugment, a data augmentation method that adjusts pitch and duration of speech at syllable level under the guidance of music scores to help ALT training. Specifically, we adjust the pitch and duration of each syllable in natural speech to those of the corresponding note extracted from music scores, to narrow the gap between natural speech and singing voice. Experiments on DSing30 and Dali corpus show that the ALT system equipped with our PDAugment outperforms previous state-of-the-art systems by 5.9% and 18.1% WERs respectively, demonstrating the effectiveness of PDAugment for ALT.

1. INTRODUCTION

Automatic lyrics transcription (ALT), which recognizes lyrics from singing voice, is useful in many applications, such as lyrics-to-music alignment, query-by-singing, karaoke performance evaluation, keyword spotting, and so on. ALT on singing voice can be regarded as a counterpart of automatic speech recognition (ASR) on natural speech. Although ASR has witnessed rapid progress and brought convenience to people in daily life in recent years [1], there is not an ALT system that has the same level of high accuracy and robustness as the current ASR systems. The main challenge of developing a robust ALT system is the scarcity of available paired singing voice and lyrics datasets that can be used for the ALT model training. Though a straightforward method is to use speech data to enhance the training data of ALT, the performance gain is marginal because there are significant differences between speech and singing voice. For example, the singing voice has some music-specific acoustic characteristics [2,3] – the large variation of syllable duration and highly flexible pitch contours are very common in singing, but rarely seen in speech [4].

Previous works have already made some attempts to artificially generate "song-like" data from speech for model training. Kruspe et al. [5] applied time stretching and pitch shifting to natural speech in a random manner, which enriches the distribution of pitch and duration in "songified" speech data to a certain extent. Nonetheless, the adjustments are random, so there is still a gap between the patterns of "songified" speech data and those of real singing voices. Compared to the work of Kruspe et al. [5], Basak et al. [6] further took advantage of real singing voice data. It transferred natural speech to singing voice domain with the guidance of real opera data. But it only took the pitch contours into account, ignoring duration, another key characteristic. Besides, it directly replaced the pitch contours with those of the real opera data, without considering the alignment of the note and syllable, which may result in the low quality of synthesized audio.

In this paper, we propose PDAugment, a syllable-level data augmentation method by adjusting pitch and duration under the guidance of music scores to generate more consistent training data for ALT training. To narrow the gap between the adjusted speech and singing voice, we adjust the speech at the syllable level to make it more in line with the characteristics of the singing voice. We try to make the speech more closely fit the music score, to achieve the effect of "singing out" the speech content. PDAugment adjusts natural speech data by the following steps: 1) extracts note information from music scores to get the pitch and duration patterns of music; 2) aligns the speech and notes at


© C Zhang, J Yu, LC Chang, X Tan, J Chen, T Qin, K Zhang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** C Zhang, J Yu, LC Chang, X Tan, J Chen, T Qin, K Zhang, "PDAugment: Data Augmentation by Pitch and Duration Adjustments for Automatic Lyrics Transcription", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

syllable level; 3) adjusts the pitch and duration of syllables in natural speech to match those in aligned notes.

Our contributions can be summarized as follows:

- We develop PDAugment, a data augmentation method to enhance the training data for ALT training, by adjusting the pitch and duration of natural speech at syllable level with note information extracted from real music scores.
- We conduct experiments in two singing voice datasets: DSing30 dataset and Dali corpus. The adjusted LibriSpeech corpus is combined with the singing voice corpus for ALT training. The ALT system with PDAugment outperforms the previous state-of-the-art system and Random Aug by 5.9% and 7.8% WERs respectively in DSing30 dataset and 24.9% and 18.1% WERs respectively in Dali corpus. Compared to adding ASR data directly into the training data, our PDAugment has 10.7% and 21.7% WERs reduction in two datasets.
- We analyze the adjusted speech by statistics and visualization and find that PDAugment can significantly compensate the gap between speech and singing voice as well as synthesize relatively high-quality audio.

2. BACKGROUND

2.1 Automatic Lyrics Transcription

The lyrics of a song provide the textual information of singing voice and are important when contributing to the emotional perception of listeners. Automatic lyrics transcription (ALT) aims to recognize lyrics from singing voice, which is of great importance in music information retrieval and analysis. However, ALT is more challenging than ASR – not like speech, in singing voice, the same lyrics with different melodies will have different pitches and durations, which results in the sparsity of training data.

Some works took advantage of the characteristics of music itself: Gupta et al. [7] extended the length of pronounced vowels in output sequences by increasing the probability of a frame with the same phoneme after a certain vowel frame. Kruspe et al. [2] boosted the ALT system by using the newly generated alignment of singing and lyrics. Gupta et al. [8] tried to make use of the background music as extra information to improve recognition accuracy. Ahlback et al. [9] proposed to tag recordings with non-vocal silence and music labels to improve the recognition rate. However, they just designed some hard constraints or added extra information from the music domain, and still did not solve the problem of data scarcity.

Considering the lack of singing voice database, some work aimed at providing a relatively large singing voice dataset: Dabike et al. [10] collected DSing dataset from real-world user information. Demirel et al. [11] built a cascade pipeline with convolutional time-delay neural networks with self-attention based on DSing30 dataset and provided a baseline for ALT task. Some other works leveraged natural speech data for ALT training: they based on pre-trained automatic speech recognition models and then made some adaptations to improve the performance of singing voice: Fujihara et al. [12] built a language model with lyrics but only used the semantic information and ignored the acoustic properties of singing voice. Mesaros et al. [3] used speaker adaptation technique by shifting the GMM components only with global statistics but did not consider the local information.

However, singing voice has some music-specific acoustic characteristics which are not in speech, limiting the performance when training the ALT system directly with natural speech data. Some work tried to synthesize "song-like" data from natural speech to make up for this gap: Kruspe et al. [5] generated "songified" speech data by time stretching, pitch shifting and adding vibrato. However, the degrees of these adjustments are randomly selected within a range, without using the patterns in real music. Basak et al. [6] took use of the F0 contours in real opera data and converted the speech to singing voice through style transfer. Specifically, they decomposed the F0 contours from the real opera data, obtained the spectral envelope and the aperiodic parameter from the natural speech, and then used these parameters to synthesize the singing voice version of the original speech. Nonetheless, in real singing voice, the note and the syllable are often aligned, but they [6] did not perform any alignment of the F0 contours of singing voice with the speech signal. This misalignment may lead to the change of pitch within a consonant phoneme (in normal circumstances, the pitch only changes between two phonemes or in vowels), which further causes distortion of the synthesized audio and limits the performance of ALT system. Besides, they only adjusted the F0 contours, which is not enough to narrow the gap between speech and singing voice.

Some previous works [13, 14] investigated how to convert speech to singing, however, they not only required speech-singing paired data but also a model training process, which made these methods inappropriate for real-time data augmentation.

In this paper, we propose PDAugment, which improves the above adjustment methods by using real music scores and syllable-level alignment to adjust the pitch and duration of natural speech, to solve the problem of insufficient singing voice data.

2.2 Speech vs. Singing Voice

In a sense, singing voice can be considered a special form of speech, but there are still a lot of discrepancies between them [15, 16]. These discrepancies make it inappropriate to transcribe the lyrics from singing voice using a speech recognition model trained on ASR data directly. In order to demonstrate the discrepancies, we randomly select 10K sentences from LibriSpeech [17] for speech corpus and Dali [18] for singing voice dataset, and make some statistics on them. The natural speech and singing voice mainly differ in the following aspects and the analysis results are

listed in Table 1.

2.2.1 Pitch

We extract the pitch contours from singing voice and speech, and compare the range and stability of the pitch. Here we use semitone as the unit of pitch.

Pitch Range Generally speaking, the range of the pitch in singing voice is larger than that in speech. Loscos et al. [15] have pointed out that the frequency range in singing voice can be much larger compared to that in speech. For each sentence, we calculate the pitch range (the maximum pitch value minus the minimum pitch value in this sentence). After averaging the pitch range of the overall 10K sentences in the corpus, the average values are listed in *Pitch Range* in Table 1.

Pitch Stability The pitch of each frame in a certain syllable (when it is corresponding to a note) in singing voice remains almost constant whereas in speech the pitch changes freely along with the audio frames in a syllable. We call the characteristic of maintaining local stability within a syllable as *Pitch Stability*. Specifically, we calculate the pitch difference between every two adjacent frames in a sentence and average it across the entire corpus of 10K sentences. The larger the value of *Pitch Stability*, the more stable the pitch. The results can be seen in *Pitch Stability* of Table 1.

2.2.2 Duration

We also analyze and compare the range and stability of the syllable duration in singing voice and speech. The duration of each syllable varies a lot along with the melody in singing voice. While in speech, it depends on the pronunciation habits of the certain speaker.

Duration Range For each sentence, we calculate the difference between the duration of the longest syllable and the shortest syllable as the duration range. The average values of the duration ranges in the entire corpus are shown as *Duration Range* in Table 1.

Duration Variance We calculate the variance of the duration of syllables in each sentence and average the variances of all sentences in the whole corpus. The results are listed as *Duration Variance* in Table 1 to reflect the flexibility of duration in singing voice.

Property	*Speech*	*Singing Voice*
Pitch Range (semitone)	12.71	14.61
Pitch Stability	0.93	0.84
Duration Range (s)	0.44	2.40
Duration Variance	0.01	0.11

Table 1: The differences of acoustic properties between speech and singing voice.

Besides the differences in the characteristics we mentioned above, sometimes singers may add vibrato in some long vowels or make artistic modifications to the pronunciation of some words to make them sound more melodious, though it will result in a loss of intelligibility. Considering that some characteristics are hard to be quantified, in this work, we start with pitch and duration to build a prototype and propose PDAugment to augment ALT training data by adjusting pitch and duration of speech at syllable level according to music scores.

3. METHOD

3.1 Pipeline Overview

For automatic lyrics transcription, we follow the practice of existing automatic speech recognition systems and choose Conformer encoder [19] and Transformer decoder [20] as our basic model architecture. Different from standard ASR systems, we add a PDAugment module in front of the encoder as shown in Figure 1, to apply syllable-level adjustments to the pitch and duration of the input natural speech according to the information of aligned notes. When the input of ALT system is singing voice, we just do not enable PDAugment module. When the input is speech, PDAugment module takes the note information extracted from music scores as extra input to adjust the pitch and duration of speech, and then adds the adjusted speech into training data to enhance the ALT model. The loss function

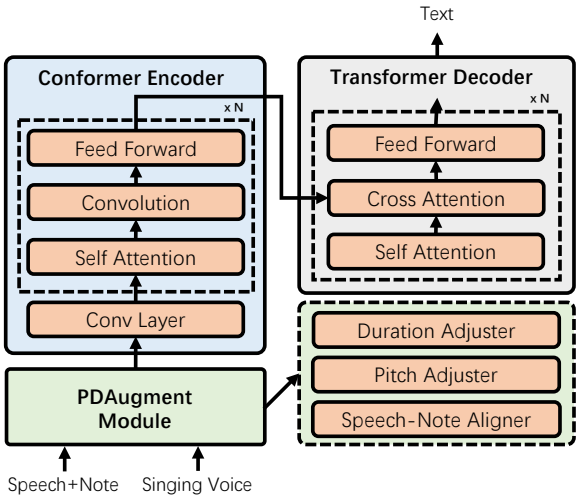


Figure 1: The overall pipeline of the ALT system equipped with PDAugment.

of the ALT model consists of decoder loss $\mathcal{L}_{dec}$, and ctc loss (on top of the encoder) $\mathcal{L}_{ctc}$: $\mathcal{L} = (1-\lambda)\mathcal{L}_{dec}+\lambda\mathcal{L}_{ctc}$, where λ is a hyperparameter to trade off the two loss terms. Considering that lyrics may contain more musical-specific expressions which are rarely seen in natural speech, the probability distributions of lyrics and standard text are quite different. We train the language model with in-domain lyrics data and then fuse it with ALT model in the beam search of the decoding stage.

We try to make the speech fit the patterns of singing voice more naturally as well as to achieve the effect of "singing out" the speech by PDAugment, so we adjust the pitch and duration of speech at syllable level according to those of corresponding notes in music scores instead of applying random adjustments. To do so, we propose PDAugment module, which consists of three key components:

1) speech-note aligner, which generates the syllable-level alignment to decide what the corresponding note of a certain syllable is for subsequent adjusters; 2) pitch adjuster, which adjusts the pitch of each syllable in speech according to that of aligned notes; and 3) duration adjuster, which adjusts the duration of each syllable in speech to be in line with the duration of the corresponding notes. We introduce each part in the following subsections.

3.2 Speech-Note Aligner

According to linguistic and musical knowledge, in singing voice, syllables can be viewed as the smallest textual unit corresponding to notes in the melodies. PDAugment adjusts the pitch and duration of natural speech at syllable level under the guidance of note information obtained from the music scores. In order to apply the syllable-level adjustments, we propose a speech-note aligner, which aims to align the speech and note (in melody) at syllable level.

The textual content can serve as the bridge of aligning notes of melody with the speech. Specifically, our speech-note aligner aligns the speech with notes (in melody) in the following steps:

1) In order to obtain the syllable-level alignment of text and speech, we first convert the text to phoneme using phonemizer [1] and then align the text with speech audio using the Montreal forced aligner (MFA) [21] tool [2] at phoneme level. Next, we group several phonemes into a syllable according to the linguistic rules [22] and get the syllable-level alignment of text and speech.

2) For the syllable-to-note mappings, we set one syllable to correspond to one note by default, because in most cases of singing voice, one syllable is aligned with one note [23]. Only when the time length ratio of the syllable in speech and the note in melody exceeds the predefined thresholds (the lower bound of the ratio is set to 0.5 as the upper bound to 2 from the perspective of preventing severe distortion of the sound), we generate one-to-many or many-to-one mappings to prevent audio distortion after adjustments.

3) We aggregate the syllable-level alignment of text and speech, and the syllable-to-note mappings to generate the syllable-level alignment of speech and note (in melody) as the input of the pitch and duration adjusters.

3.3 Pitch Adjuster

Pitch adjuster adjusts the pitch of input speech at syllable level according to the aligned notes. Specifically, we use WORLD [24], a fast and high-quality vocoder-based speech synthesis system to implement the adjustment. The WORLD system [3] parameterizes speech into three components: fundamental frequency (F0), aperiodicity, and spectral envelope and can reconstruct the speech with only estimated parameters. We use WORLD to estimate the three parameters of natural speech and only adjust the F0 contours according to that of corresponding notes. Then we synthesize speech with adjusted F0 accompanied by original aperiodicity and spectral envelope. Figure 2 shows the F0 contours before and after the pitch adjuster.

Figure 2: The change of F0 contour after pitch adjuster. The content of this example is “opening his door”.

Pitch adjuster calculates the pitch difference between speech and note with syllable-level alignment and adjusts F0 contours of speech accordingly. Some details are as follows: 1) Considering that the quality of synthesized speech will drop sharply when the range of adjustment is too large, we need to keep it within a reasonable threshold. Specifically, we calculate the average pitch of the speech and the corresponding melody respectively. When the average pitch of speech is too different from that of the corresponding melody (e.g., exceeding a threshold, which is 5 semitones in our experiment), we shift the pitch of the entire note sequence to make the difference within the threshold and use the shifted note for adjustment, otherwise, keep the pitch of the original note unchanged; 2) To maintain smooth transitions in synthesized speech and prevent speech from being interrupted, we perform pitch interpolation for the frames between two syllables. 3) When a syllable is mapped to multiple notes, we segment the speech of this syllable in proportion to the duration of notes and adjust the pitch of each segment according to the corresponding note.

3.4 Duration Adjuster

Duration adjuster changes the duration of input speech to align with the duration of the corresponding note. As shown in Figure 3, instead of scaling the whole syllable, we only scale the length of vowels and keep the length of consonants unchanged, because the duration of consonants in singing voice is not significantly longer than that in speech, while long vowels are common in singing voice [5]. There are one-to-many mappings and many-to-one mappings in the syllable-level alignment. For the first case, we calculate the total length of the syllables and adjust the length of all vowels in these syllables in proportion. For the second case, we adjust the length of the vowel, so that the total length of this syllable is equal to the total length of these notes.

[1] https://github.com/bootphon/phonemizer
[2] https://github.com/MontrealCorpusTools/Montreal-Forced-Aligner
[3] https://github.com/JeremyCCHsu/Python-Wrapper-for-World-Vocoder

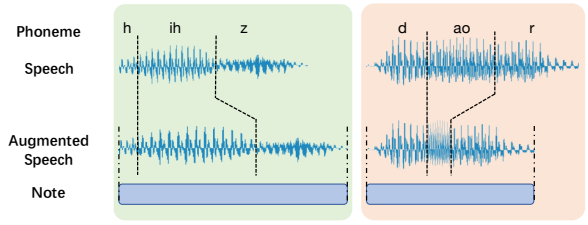


Figure 3: The change of duration after duration adjuster. The content of this example is "his door". The left block shows the case of lengthening the duration of speech and the right shows the case of shortening the duration.

4. EXPERIMENTS AND RESULTS

In this section, we first describe the experimental settings, including datasets, model configuration, and the details of training, inference, and evaluation. Then we report the experiment results, visualize the effect of PDAugment, and further conduct more analyses to verify the effectiveness of PDAugment.

4.1 Experimental Settings

4.1.1 Datasets

Singing Voice Datasets We conduct experiments on two singing voice datasets to verify the effectiveness of our PDAugment: DSing30 dataset [10] and Dali corpus (v2.0) [18, 25]. DSing30 dataset consists of about 4K monophonic Karaoke recordings of English pop songs with nearly 80K utterances, performed by 3,205 singers. We use the partition provided by [10] to make a fair comparison with them. Dali corpus is another large dataset of synchronized audio, lyrics, and notes. It consists of 1200 English polyphonic songs (with background music) for a total duration of 70 hours. Following [6], we use the sentence-level annotation of the dataset provided by [18] [4] and divide the dataset into training, development, and test with a proportion of 8:1:1 without any singers overlapping in each partition. We convert all the singing voice waveforms in our experiments into mel-spectrogram following [26] with a frame size of 25 ms and hop size of 10 ms.

Natural Speech Dataset Following the common practice in previous ASR work [19], we choose the widely used LibriSpeech [17] corpus as the natural speech dataset in our experiments and use the official training partition. The LibriSpeech corpus contains 960 hours of speech sampled at 16 kHz with 1129 female speakers and 1210 male speakers. Similar to singing voice, we convert the speech into mel-spectrogram with the same setting [27].

Music Score Dataset In this work, we choose the pop music subset of FreeMidi dataset [5] because almost all of the songs in our singing voice datasets are pop music. The pop music subset of FreeMidi has about 4000 MIDI files, which are used to provide note information for PDAugment module.

[4] https://github.com/gabolsgabs/DALI
[5] https://freemidi.org/genre-pop

Lyrics Corpus In order to construct a language model with more in-domain knowledge, we deliberately collected a large amount of lyrics data to build our lyrics corpus for language model training. Besides the training text of DSing30 dataset and Dali corpus, we collect lyrics of English pop songs from the Web. We crawl about 46M lines of lyrics and obtain nearly 17M sentences after removing duplicates including DSing30 dataset and Dali corpus. We attach a subset of the collected lyrics corpus in the supplementary material.

4.1.2 Model Configuration

We choose Conformer encoder [19] and Transformer decoder [20] as the basic ALT model architecture in our experiments since the effectiveness of the structure has been proved in ASR. Our language model is based on Transformer encoder. More details about the model configuration, training/inference settings, and codes are attached in the supplementary materials.

4.2 Results

In this subsection, we report the experimental results of the ALT system equipped with PDAugment in two singing voice datasets. We compare our results with several basic settings as baselines: 1) *Naive ALT*, the ALT model trained with only singing voice dataset; 2) *ASR Augmented*, the ALT model trained with the combination of singing voice dataset and ASR data directly; 3) *Random Aug* [5], the ALT model trained with the combination of singing voice dataset and randomly adjusted ASR data. The pitch is adjusted ranging from -6 to 6 semitones randomly and the duration ratio of speech before and after adjustment is randomly selected from 0.5 to 1.2. All 1), 2), and 3) are using the same model architecture as *PDAugment*. Besides the above three baselines, we compare our PDAugment with the previous systems which reported the best results in two datasets respectively. We compare our PDAugment with [11] using RNNLM for DSing30 dataset and compare the results with [6] for Dali corpus, All the results are listed in Table 2.

Method	DSing30		Dali	
	Dev	*Test*	*Dev*	*Test*
Naive ALT	28.2	27.4	80.9	86.3
ASR Augmented	20.8	20.5	75.5	75.7
Random Aug [5]	17.9	17.6	69.8	72.1
Previous Work [11]	17.7	15.7	-	-
Previous Work [6]	-	-	75.2	78.9
PDAugment	**10.1**	**9.8**	**53.4**	**54.0**

Table 2: The WERs (%) of DSing30 and Dali dataset. The audios of Dali corpus contain background music.

As can be seen, *Naive ALT* performs not well and gets high WERs in both DSing30 dataset and Dali corpus, which demonstrates the difficulty of the ALT task. After adding ASR data for ALT training, the performances of *ASR Augmented* setting in both of the two datasets have

been improved slightly compared to *Naive ALT*, but still with relatively high WERs, which indicates the limitation of using ASR training data directly.

When applying the adjustments, a question is if we adjust the pitch and duration with random ranges without note information from music scores, how well will the ALT system perform? The results of *Random Aug* can perfectly answer this question. As the results show, *Random Aug* can slightly improve the performance compared with *ASR Augmented*, demonstrating that increasing the volatility of pitch and duration in natural speech helps ALT system training, which is the same as what [5] claimed. *PDAugment* is significantly better than *Random Aug*, which indicates that adjusted speech can better help the ALT training with the guidance of music scores.

Besides, it is obvious that *PDAugment* greatly outperforms the previous work in both datasets. [11] performs worse than *PDAugment* because of not taking advantage of the massive ASR training data. Compared with [6] that replaced F0 contours of speech directly, *PDAugment* can narrow the gap between natural speech and singing voice in a more reasonable manner and achieve the lowest WERs among all the above methods. The results in both datasets show the effectiveness of PDAugment for ALT task and reflect the superiority of adding music-specific acoustic characteristics into natural speech.

4.3 Ablation Studies

We conduct more experimental analyses to deeply explore our PDAugment and verify the necessity of some design details. More ablation studies (different language models) can be found in the supplementary materials. The ablation studies are carried out on DSing30 dataset in this section.

4.3.1 Augmentation Types

In this subsection, we explore the effects of different kinds of augmentations (only adjust pitch, only adjust duration, and adjust pitch & duration) on increasing the performance of ALT system. We generate three types of adjusted speech by enabling different adjusters of PDAugment module and conduct the experiments on DSing30. The results are shown in Table 3.

Setting	*DSing30 Dev*	*DSing30 Test*
PDAugment	**10.1**	**9.8**
Disable Pitch Adjuster	13.6	13.4
Disable Duration Adjuster	13.8	13.8
Disable both Adjusters	20.8	20.5

Table 3: The WERs (%) of different types of augmentation of DSing30 dataset. All of the settings are trained on DSing30 and the original or adjusted LibriSpeech data.

As we can see, when we enable the whole PDAugment module, the ALT system can achieve the best performance, indicating the effectiveness of the pitch and duration adjusters. When we disable the pitch adjuster, the WER on DSing30 is 3.6% higher than *PDAugment*. The same thing happens when we disable the duration adjuster, the WER is 4.0% higher than *PDAugment*. And if both the pitch and duration are not adjusted, which means using the speech data directly for ALT training, the WER is the worst among all settings. The results demonstrate that both pitch and duration adjusters are necessary and can help with improving the recognition accuracy of the ALT system.

4.4 Adjusted Speech Analyses

Following Section 2.2, we analyze the acoustic properties of the original natural speech and the adjusted speech by random and PDAugment, and list the results in Table 4.

Property	*Original*	*Random*	*PDAugment*
Pitch Range (semitone)	12.71	13.87	14.19
Pitch Stability	0.93	0.59	0.69
Duration Range (s)	0.44	0.42	0.59
Duration Variance	0.01	0.01	0.05

Table 4: The differences of acoustic properties between original speech and adjusted speech.

Combining the information of Table 1 and Table 4, we can clearly find that the distribution pattern of acoustic properties (pitch and duration) after PDAugment is closer to singing voice compared with the original speech, which indicates that our PDAugment can change the patterns of pitch and duration in original speech and effectively narrow the gap between natural speech and singing voice. To avoid the distortion of adjusted speech, we limit the adjustment degree within a reasonable range, so the statistics of adjusted speech can not completely match that of singing voice. Nonetheless, adjusted speech is still good enough for ALT model to capture some music-specific characteristics.

In order to visually demonstrate the effect of PDAugment module, we plot the spectrograms of speech to compare the acoustic characteristics before and after different types of adjustments. The visualization of an example adjusted speech is attached in the supplementary materials.

5. CONCLUSION

In this paper, we proposed PDAugment, a data augmentation method by adjusting pitch and duration to make better use of natural speech for ALT training. PDAugment transfers natural speech into singing voice domain by adjusting pitch and duration at syllable level under the instruction of music scores. PDAugment module consists of speech-note aligner to align the speech with notes, and two adjusters to adjust pitch and duration respectively. Experiments on two singing voice datasets show that PDAugment can significantly reduce the WERs of ALT task. We also explore different types of augmentation, and further verify the effectiveness of PDAugment. In the future, we will consider narrowing the gap between natural speech and singing voice from more aspects such as vibrato, and try to add some music-specific constraints in the decoding stage.

6. ACKNOWLEDGEMENT

This work was supported by the Key Project of Natural Science Foundation of Zhejiang Province (No.LZ19F020002), the Key R&D Program of Zhejiang Province (No.2022C03126), and Project of Key Laboratory of Intelligent Processing Technology for Digital Music (Zhejiang Conservatory of Music), Ministry of Culture and Tourism (No.2022DMKLB001). Thanks to Future Design Laboratory of Zhejiang University for the help in dataset collection. Thanks are also due to Shen Zhou for bringing insights in music.

7. REFERENCES

[1] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[2] A. M. Kruspe, "Bootstrapping a system for phoneme recognition and keyword spotting in unaccompanied singing." in *ISMIR*, 2016, pp. 358–364.

[3] A. Mesaros and T. Virtanen, "Adaptation of a speech recognizer for singing voice," in *2009 17th European Signal Processing Conference*. IEEE, 2009, pp. 1779–1783.

[4] C.-P. Tsai, Y.-L. Tuan, and L.-s. Lee, "Transcribing lyrics from commercial song audio: the first step towards singing content processing," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5749–5753.

[5] A. M. Kruspe, "Training phoneme models for singing with" songified" speech data." in *ISMIR*, 2015, pp. 336–342.

[6] S. Basak, S. Agarwal, S. Ganapathy, and N. Takahashi, "End-to-end lyrics recognition with voice to singing style transfer," *arXiv preprint arXiv:2102.08575*, 2021.

[7] C. Gupta, H. Li, and Y. Wang, "Automatic pronunciation evaluation of singing." in *Interspeech*, 2018, pp. 1507–1511.

[8] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics alignment and transcription in polyphonic music: Does background music help?" in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 496–500.

[9] S. Ahlbäck, "Mstre-net: Multistreaming acoustic modeling for automatic lyrics transcription," in *ISMIR2021, International Society for Music Information Retrieval*, 2021.

[10] G. R. Dabike and J. Barker, "Automatic lyric transcription from karaoke vocal tracks: Resources and a baseline system." in *INTERSPEECH*, 2019, pp. 579–583.

[11] E. Demirel, S. Ahlbäck, and S. Dixon, "Automatic lyrics transcription using dilated convolutional neural networks with self-attention," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[12] H. Fujihara, M. Goto, J. Ogata, K. Komatani, T. Ogata, and H. G. Okuno, "Automatic synchronization between lyrics and music cd recordings based on viterbi alignment of segregated vocal signals," in *Eighth IEEE International Symposium on Multimedia (ISM'06)*. IEEE, 2006, pp. 257–264.

[13] J. Parekh, P. Rao, and Y.-H. Yang, "Speech-to-singing conversion in an encoder-decoder framework," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 261–265.

[14] D.-Y. Wu and Y.-H. Yang, "Speech-to-singing conversion based on boundary equilibrium gan," in *InterSpeech*, 2020.

[15] A. Loscos, P. Cano, and J. Bonada, "Low-delay singing voice alignment to text," in *ICMC*, vol. 11, 1999, pp. 27–61.

[16] A. Mesaros, "Singing voice identification and lyrics transcription for music information retrieval invited paper," in *2013 7th Conference on Speech Technology and Human-Computer Dialogue (SpeD)*. IEEE, 2013, pp. 1–10.

[17] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2015, pp. 5206–5210.

[18] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Dali: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm," in *Proceedings of the 19th International Society for Music Information Retrieval Conference*, 2018, pp. 431–437.

[19] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," *arXiv preprint arXiv:2005.08100*, 2020.

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[21] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi." in *Interspeech*, vol. 2017, 2017, pp. 498–502.

[22] D. M. Kearns, "Does english have useful syllable division patterns?" *Reading Research Quarterly*, vol. 55, pp. S145–S160, 2020.

[23] E. Nichols, D. Morris, S. Basu, and C. Raphael, "Relationships between lyrics and melody in popular music," in *ISMIR 2009-Proceedings of the 11th International Society for Music Information Retrieval Conference*, 2009, pp. 471–476.

[24] M. Morise, F. Yokomori, and K. Ozawa, "World: a vocoder-based high-quality speech synthesis system for real-time applications," *IEICE TRANSACTIONS on Information and Systems*, vol. 99, no. 7, pp. 1877–1884, 2016.

[25] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Creating dali, a large dataset of synchronized audio, lyrics, and notes," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.

[26] C. Zhang, X. Tan, Y. Ren, T. Qin, K. Zhang, and T.-Y. Liu, "Uwspeech: Speech to speech translation for unwritten languages," *arXiv preprint arXiv:2006.07926*, 2020.

[27] C. Zhang, Y. Ren, X. Tan, J. Liu, K. Zhang, T. Qin, S. Zhao, and T.-Y. Liu, "Denoispeech: Denoising text to speech with frame-level noise modeling," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 7063–7067.

PARAMETER SENSITIVITY OF DEEP-FEATURE BASED EVALUATION METRICS FOR AUDIO TEXTURES

Chitralekha Gupta **Yize Wei** **Zequn Gong**
Purnima Kamath **Zhuoyao Li** **Lonce Wyse**
National University of Singapore
chitralekha@u.nus.edu, yize.wei@u.nus.edu, zequn.gong@u.nus.edu
purnima.kamath@u.nus.edu, zhuoyaoli@u.nus.edu, lonce.wyse@nus.edu.sg

ABSTRACT

Standard evaluation metrics such as the Inception score and Fréchet Audio Distance provide a general audio quality distance metric between the synthesized audio and reference clean audio. However, the sensitivity of these metrics to variations in the statistical parameters that define an audio texture is not well studied. In this work, we provide a systematic study of the sensitivity of some of the existing audio quality evaluation metrics to parameter variations in audio textures. Furthermore, we also study three more potentially parameter-sensitive metrics for audio texture synthesis, (a) a Gram matrix based distance, (b) an Accumulated Gram metric using a summarized version of the Gram matrices, and (c) a cochlear-model based statistical features metric. These metrics use deep features that summarize the statistics of any given audio texture, thus being inherently sensitive to variations in the statistical parameters that define an audio texture. We study and evaluate the sensitivity of existing standard metrics as well as Gram matrix and cochlear-model based metrics in response to control-parameter variations for audio textures across a wide range of texture and parameter types, and validate with subjective evaluation. We find that each of the metrics is sensitive to different sets of texture-parameter types. This is the first step towards investigating objective metrics for assessing parameter sensitivity in audio textures.

1. INTRODUCTION

Audio textures are rich and varied sounds that are produced by a superposition of multiple acoustic events. Unlike the sound of an individual event, such as a footstep, or the complex spectrotemporal structures and sequences of speech or music, an audio texture is defined by properties or parameters that remain constant over time [1] despite statistical variation in the sound over time or between instances. For parametric audio texture synthesis, the goal is to generate novel sounds with descriptive parameters that match those of a target texture. Standard objective evaluation metrics for audio texture synthesis include diversity and quality based measures such as the Inception score [2], and Fréchet Audio Distance (FAD) [3]. However, the existing metrics have not been studied for their sensitivity to systematic variation in the parameter values that define the synthesized audio textures.

Various audio texture synthesis algorithms have been applied to modeling a wide range of natural audio texture types including rhythmic (eg. drums, tapping), pitched (eg. windchimes, churchbells) and other natural sounds (eg. rain, wind), but with different degrees of success. Evaluation of parametric variation textures has typically been through human listening experiments [4, 5].

"Deep features" (activation patterns across layers of neural networks) have been explored for various evaluation tasks such as out-of-distribution detection in audio classification [6], perceptual image similarity evaluation [7], audio distortion assessment [3] and audio texture synthesis [1, 5, 8]. In this work we study several existing objective metrics and also explore deep features and cochlear channel model statistics for potential evaluation metrics sensitive to parameter variations. We make use of a controlled audio texture dataset [9] to study these metrics for a wide range of audio textures - rhythmic, pitched, and non-rhythmic non-pitched under systematic parametric variation. We present a comparative study of the parameter sensitivity of the metrics and validate their reliability through human listening tests.

2. RELATED WORK

2.1 Why is a parameter sensitive metric needed?

McDermott and Simoncelli [1] developed a set of statistics based on a cochlear model to describe the perceptually relevant aspects of a given audio texture. To synthesize a new audio texture, a random input is iteratively perturbed until its statistics match those of a target. This algorithm produces convincing audio for many natural textures such as insect swarms, a stream, or applause. However, human listeners give low scores on realism for resynthesized pitched and rhythmic textures, such as wind chimes, drum break, walking on gravel, and church bells. Besides the audio quality, the statistical parameters associated with these

© C. Gupta, Y. Wei, Z. Gong, P. Kamath, Z. Li, and L. Wyse. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** C. Gupta, Y. Wei, Z. Gong, P. Kamath, Z. Li, and L. Wyse, "Parameter Sensitivity of Deep-Feature based Evaluation Metrics for Audio Textures", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

sounds can fail to be faithfully preserved in the synthesized sounds [4]. Moments derived from the time-averaged scattering coefficients when used for resynthesis of audio textures have also shown similar performance but using fewer coefficients [10, 11]. An objective metric for predicting these human evaluations of texture synthesis failures would be valuable.

Gatys et al. [12] did seminal work on image texture synthesis that replaced hand-crafted statistics with Gram matrix statistics computed as the correlation between hidden feature activations from layers of a trained convolutional neural network (CNN). Iteratively perturbing a random input to match Gram matrices produces compelling and novel image textures. Similarly, Ulyanov et al. [8] improved the quality of synthetic textures based on the dataset from McDermott and Simoncelli [1]. Antognini et al. [5] extended Ulyanov's work by modifying the architecture with 6 parallel single-layered untrained CNNs each with a different convolutional kernel size, and computed autocorrelation and diversity losses in addition to the original Gram matrix loss. They demonstrated some improvements, but found failure modes similar to that of McDermott and Simoncelli [1] using a combination of objective and subjective evaluation. Caracalla and Roebel [13] also relied on human listening tests to show the improvement in sound quality achieved in this kind of iterative texture synthesis using a complex spectrogram audio representation.

The key take-away from these previous studies is that there is a need for evaluating the preservation of statistical parameters in synthesised audio texture but a lack of an objective metric for that purpose.

2.2 The existing audio synthesis evaluation metrics

The evaluation of generative models in terms of perceptual realism is challenging. However, various objective metrics have been formulated for assessing the quality of synthesized audio textures. For example, L2 distance, cosine distance, and signal-to-distortion ratio use a clean reference signal to compare with the modified or enhanced synthetic signal. Such signal-level metrics are agnostic to the type of audio that is being enhanced. However, these metrics may not always correlate with human perception. FAD [3] takes a different approach as a reference-independent metric that, instead of looking at individual clips, computes the distance between the distribution of embedding statistics generated on a large set of clips.

The Inception score [2, 14, 15], passes generated examples through a pre-trained classifier. The mean KL divergence between the conditional output class probabilities and the marginal distribution of the same are then calculated. The Inception score penalizes models whose examples are not easily classified into a single class, as well as models whose examples collectively belong to only a few of the possible classes. However, the Inception score is not the right tool for evaluating a model's sensitivity to parameter differences between sounds within a class.

The goal of most of these audio quality assessment metrics has been to quantify the degradation of an enhanced/modified audio signal. However, to the best of our knowledge, there is a lack of a systematic study of the sensitivity of these metrics to parameter variations in synthesized audio. Such a study is particularly challenging for audio texture synthesis because of the inherent variations that correspond to a particular parametric description.

3. METRICS

We study two standard metrics and three Gram matrix metrics for sensitivity to parametric changes to audio textures.

3.1 Existing Metrics

3.1.1 Fréchet Audio Distance (FAD)

FAD [3] is a reference-independent metric for audio quality assessment that computes the Fréchet distance [16] between the multi-variate Gaussian distributions of the embeddings of train and test set audio data. These 128 dimensional embeddings are extracted from a VGG-ish model [17] pre-trained on clean audio data for classification.

$$FAD = ||\mu_b - \mu_t||^2 + tr(\Sigma_b + \Sigma_t - 2\sqrt{\Sigma_b \cdot \Sigma_t}) \quad (1)$$

where the training and test data embeddings are assumed to have multivariate Gaussian distributions $\mathcal{N}(\mu_b, \Sigma_b)$ and $\mathcal{N}(\mu_t, \Sigma_t)$, respectively. We used the open-source model and FAD computation code [3].

3.1.2 L2 Distance

As another standard metric for our study, we compute the Euclidean distance between the two matrices X_A and X_B, representing the spectrograms of the two input audio signals to be compared, x_A and x_B respectively.

3.2 Gram matrix based Metrics

3.2.1 Gram matrix Metric (GM)

Following Ulyanov and Lebedev [8] and Antognini et al. [5], we use 2D spectrogram representations of audio for iterative updates through the CNNs used to compute a summary statistic in the form of a Gram matrix. Two audio textures sound similar if the computed Gram matrices are similar. Here, we leverage on this property of Gram matrix to develop a metric designed to be sensitive to parameter variations. We define the Gram matrix metric as the mean squared error between the Gram matrices of two textures. We hypothesize that this metric would be sensitive to parametric variations in the statistics of the same texture type such as different rates of flowing water.

Formally, we define the Gram matrix metric GM as,

$$GM = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{D}||G_A^n - G_B^n||_2^2 \quad (2)$$

where, G_A^n and G_B^n are flattened Gram matrices derived from audio clips A and B respectively for the n^{th} CNN in an ensemble of N CNNs, and D is the total number of elements in the Gram matrix. In this work, we adopted the

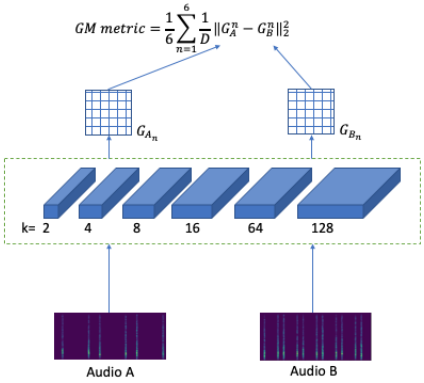


Figure 1. Gram matrix param-sensing metric computation block diagram consisting of an ensemble of six 1 layer 1d-CNNs, with different kernel sizes, k. Each CNN has 512 filters, and the weights are randomly initialized.

architecture used by Antognini et al. [5] for the Gram matrix computation, as shown in Figure 1. It consists of an ensemble of 6 (N=6) single-layered CNNs. We used spectrograms of the audio texture clips as the input to the network computed with 512 FFT bins and hop-size of 128 samples. Unlike images, we consider the spectrograms as one-dimensional input features with the frequency bins as the number of channels. We therefore use a one-dimensional convolution, where each CNN has a convolutional kernel k_n with a different width, in particular 2, 4, 8, 16, 64, 128. The different kernel sizes capture statistics over multiple time scales. We used $F = 512$ filters randomly initialized from a normal distribution (with no training) thus producing six 512×512 dimensional Gram matrices. Details are available in Supplementary Material [1] .

3.2.2 Gram matrix Cos Metric (GMcos)

We compute the cosine distance between the Gram matrices (instead of mean square error) as,

$$GMcos = 1 - \frac{G_A^n \cdot {G_B^n}^T}{||G_A^n||_2 ||G_B^n||_2} \tag{3}$$

where, G_A^n and G_B^n are flattened Gram matrices of audio clips A and B respectively for the n^{th} CNN in an ensemble of N CNNs.

3.2.3 Accumulated-Gram Metric (AGM)

Gram matrices for audio textures are usually sparse (see Supplementary Material (Section 1.) for more info). Neto et al. [18] used a compact version of the Gram matrices to predict the class label of a given sample. Similarly, for a metric that captures the essence of the Gram matrices generated from an audio texture clip, we compute a *Gram vector* corresponding to the clip by accumulating the values from its six Gram matrices, and then aggregating the rows over all the Gram matrices to get a compact one dimensional summarizing vector of length 128. For more details, please refer to Supplementary Material.

[1] https://animatedsound.com/ismir2022/metrics/supplementary/main.html

We define Accumulated-Gram metric (AGM) as the dot product of the difference of the two Gram vectors g_A and g_B corresponding to the two audio clips A and B as

$$AGM = (g_A - g_B) \cdot (g_A - g_B)^T \tag{4}$$

3.3 Cochlear Param-Metric (CPM)

McDermott and Simoncelli [1] use a 3-step texture model to generate a set of statistics of an audio texture as synthesis parameters. A filterbank with a set of cochlear filters is first applied on the incoming sound to decompose the sound to many sub-bands. Then, sub-band envelopes are computed and compressed. And finally a filterbank with a set of modulation filters is applied on each compressed subband envelope. They use seven sets of time-average statistics of nonlinear functions of either the envelopes or the modulation bands as a representation of textures, which includes marginal statistics, variance, and correlations. During synthesis, a white noise signal is iteratively modified to minimize the distance between the synthesised and target sound statistics. Our experiments are based on McWalter's implementation [4] of the [1] algorithm. An overview of the method, implementation details, and statistical parameters are summarized in Supplementary Material (Section 2.).

We use the statistical parameters calculated by this algorithm as a representation for the Cochlear Param-Metric (CPM). For each sound x, we calculate the seven sets of statistics S^i where $i = [1...7]$, and compute CPM as

$$CPM = \sum_{i=1}^{7} D(S_A^i, S_B^i) \tag{5}$$

where S_A^i is a flattened vector of the i^{th} statistics set of sound clip A, and $D(x1, x2)$ calculates the cosine distance between two vectors.

4. EXPERIMENTAL SETUP

4.1 Dataset

We use a subset of audio textures from [9] as summarized in Table 1. There are 13 texture types, each of which has a collection of examples that are determined by a set of variable control parameters. For example, various *windchimes* sounds are determined by parameters *chimesize* and *strength*. For our experiments only one control parameter varies at a time. For example, for the set of audio textures *windchimes-strength*, we generate 11 audio files with a varying strength parameter, with the first file having the lowest strength parameter value and the 11th file having the highest strength value. We generate 10 files for each these 11 setting, where all the 10 versions have the same parameter settings but are different instances. All the audio files being used are between 1.5 and 2 seconds long.

The *pitched* sound group consists of the frequency modulation, windchimes, chimes (without the wind sound), and feedback noise (FB noise). FB noise has the parameter 'pitchedness' that changes the texture from a noisy

Group	Texture Type	Parameters
Pitched	FM	modulation frequency (FM-mf) carrier frequency (FM-cf) modulation index (FM-mi)
	Windchimes	chimesize (windchimes-size) strength (windchimes-strength)
	Chimes	chimesize (chimes-size) strength (chimes-strength)
	FB Noise	pitchedness (fbnoise-pitchedness)
	Nsynth brass	pitch (nsynth-pitch)
Rhythmic	Pops	rate (pops-rate) center freq (pops-cf) irregularity (pops-irreg)
	Chirps	rate (chirps-rate) center freq (chirps-cf) irregularity (chirps-irreg)
	Tapping	rate (tapping-rate) relative phase (tapping-relphase)
	Drum break	tempo (drum-tempo) reverb (drum-rev)
Others	Wind	gustiness (wind-gust) howliness (wind-howl) strength (wind-strength)
	Waterfill	fill-level (water-fill)
	Bees	center frequency (bees-cf) busy-body (bees-busy)
	Applause	rate (applause-rate) no. of clappers (applause-clappers)

Table 1. Dataset overview

signal to a pitched sound. Under the *rhythmic* group are pops, chirps, tapping, and drum-break. The tapping sound is similar to the "tapping 1-2" sound in the collection of sounds in [1], and the drum break sound set is recorded and then processed with varying tempo and reverb parameters. The *others* group includes wind, water filling a container, buzzing bees, and applause. Most sounds are synthesized [2], except the waterfill, Nsynth brass, and drum-break textures, which are recorded or are manipulated versions of a real recording.

4.2 Subjective Evaluation

Listening tests were conducted to quantify listeners' sensitivity to control parameter changes based on their ability to identify the order and proximity of audio samples w.r.t two selected reference samples as well as each other. This evaluation was done to provide a perceptual baseline for our objective metrics comparison. Each audio trial consisted of 7 audio samples - 2 references and 5 test samples. The 2 references were selected for each texture with the control parameters (as in the Parameters column in table 1) set to 0 (left endpoint) and 1 (right endpoint). The 5 test samples are from the same texture selected with parameters between 0 and 1. A total of 9 such texture-trials were created, each with a different combination of control parameter settings for the 5 test audio samples. This was done to ensure that all the 9 control parameters values between 0 and 1 are included and are uniformly distributed across trials. The references, test samples, and the sequence of the individual texture-trials were randomized while conducting the test.

An interface was developed specifically for this test with the goal to intuitively convey the task details to the listeners. First they listened to the two references and then to the individual samples in the test. Then they were asked to create an arrangement by positioning thumbs on a slider corresponding to each test sample depending on how they perceived the order and distance of each sample w.r.t. the two references. Adjustments could be made by listening to their arrangement until they were satisfied. The listening test interface can be viewed on our webpage [3]. Amazon Mechanical Turk (AMT) was used to collect 30 responses per trial from a total of 348 unique participants.

[2] Dataset Appendix: https://animatedsound.com/ismir2022/metrics/appendix_dataset/index.html

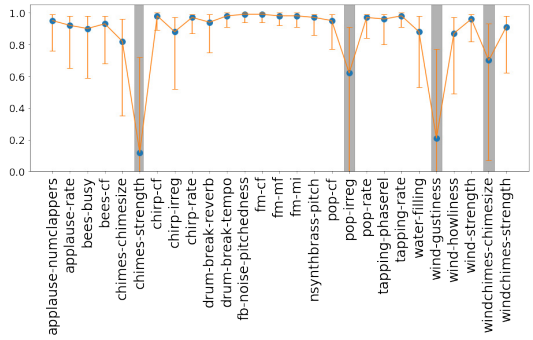


Figure 2. Correlation of perceptual distances with control parameters used to generate the textures. Grey bars indicate textures with no significant correlation.

5. EXPERIMENTS AND RESULTS

In this study, we experimentally address two main questions: 1) Are the objective metrics consistent for texture instances generated with the same parameters (Section 5.2)? 2) Are the objective metrics sensitive to parameter variations (Section 5.3)? We evaluate each metric by comparing them with the subjective responses.

5.1 Subjective tests

Figure 2 shows the correlation coefficients for the subjective perceptual distance captured w.r.t the synthesis control parameters. Parametric variations for chimes-strength, pop-irreg, wind-gustiness and windchimes-chimesize did not result in significant correlations primarily due to wide variance in human ratings, and are excluded from further comparison with objective metrics. Based on the distance measures, we derive rank ordering for the audio samples along the parametric dimension to compare with metrics. See the Supplementary Material for summary rank-order plots obtained from the listening tests for all textures under consideration in this paper. A selection of sounds used in our listening tests can be auditioned on our webpage.

5.2 Metric Consistency

Metric consistency means that the distance between two texture instances with the same parameter settings is small. We analyse consistency in terms of relative mean for three texture types - FM, pops, and water filling. The metric values are computed over one hundred comparisons between two parametrically identical textures. Relative mean is the single parameter mean with respect to the maximum average mean value of comparisons across different parameter settings for the texture as computed in Section 5.3.

[3] https://animatedsound.com/ismir2022/metrics/

Text-Param	L2	FAD	CPM	GM	GMcos	AGM
FM-cf	54.04	5.69	8.05	**0.07**	**0.02**	0.12
pops-rate	21.36	5.27	**3.44**	6.81	**2.46**	4.28
water-fill	61.09	40.60	23.09	16.19	**7.83**	**12.17**

Table 2. Metric Consistency in terms of relative mean in % for three texture types - FM, pops, and water filling. Lower is better. The best two in every row are highlighted in bold.

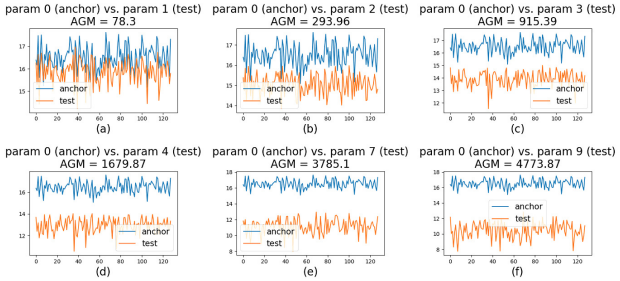


Figure 3. Gram vectors and AGM values of anchor for applause sounds with an increasing number of clappers from (a) to (f) compared to the anchor sound of a single clapper.

The results are shown in Table 2. The L2 metric shows a high relative mean value for all three texture types which confirms the understanding that a spectrogram-based distance measurements does not capture the statistics that define a texture. It is too sensitive to the "irrelevant" variations between similarly parameterized textures.

Although the relative mean of FAD is not as high as that of L2, it is higher than the Gram matrix based metrics, because the embeddings extracted from VGGish provide a holistic representation of the audio including both the overall statistics and temporal structure. The effect of the exact temporal structure of the sound on the metric is reduced but still present. The statistical metrics (CPM and GM-based) exhibit low relative mean values (good consistency), especially GMcos. The waterfill texture shows higher relative mean values across all metrics. This is at least in part because for real recordings, even for short durations, the fill-level is never constant while filling a container.

5.3 Metric Sensitivity to Parameter Variation

To further investigate the sensitivity of the objective metrics to parameter variations, we fix one texture example to a low parameter value (0), called the *anchor*, and compare it with nine *test* clips of the same texture-type but increasing parameter values (1-9). We compute this over 10 versions of the same anchor and test parameter settings, and average over them for the metrics L2, CPM, GM, GMcos, and AGM. However, to calculate FAD, we compute Fréchet distance between normal distributions of embeddings of the 10 versions, instead of between the embeddings of individual audio files (Eq. 1). An example of the sensitivity of the 1×128 dimensional AGM Gram vector to parameter variations is shown in Figure 3. The Gram vectors of a reference (anchor) applause audio texture clip that has the number of clappers parameter set to 1, is compared with six test audio clips of applause textures with increasing number of clappers. The divergence between the Gram vectors of the anchor and test sounds grows as the number of clappers is increased.

Texture-Param	L2	FAD	CPM	GM	GMcos	AGM
Pitched						
FM-mf	**0.99**	0.94	**0.99**	**1.00**	**0.99**	0.64
FM-cf	**0.99**	0.71	**0.99**	**1.00**	**0.99**	0.97
FM-mi	**0.99**	0.75	0.94	**0.99**	**1.00**	**0.99**
windchimes-strength	**0.97**	**0.97**	0.82	**0.95**	**0.95**	0.62
chimes-size	0.28	**0.88**	**0.92**	0.20	0.20	0.07
fbnoise-pitchedness	**1.00**	0.75	0.98	**1.00**	**0.99**	**1.00**
nsynth-pitch	0.09	-0.45	**0.73**	**0.23**	0.09	-0.50
Rhythmic						
pops-rate	**0.98**	0.85	0.80	0.79	0.89	**0.94**
pops-cf	**0.98**	0.96	**0.99**	**0.99**	**0.99**	**0.98**
chirps-rate	**0.97**	**0.94**	**0.94**	0.84	0.88	0.89
chirps-cf	**0.99**	**0.98**	**0.98**	**0.98**	**0.98**	0.97
chirps-irreg	0.81	**0.95**	**0.94**	0.90	0.92	0.90
tapping-rate	**1.00**	0.90	0.91	0.64	**0.94**	0.76
tapping-relphase	0.88	**0.96**	**0.98**	0.94	0.95	0.76
drum-tempo	0.08	**0.82**	**0.97**	**0.97**	0.63	0.81
drum-rev	**0.93**	0.85	0.81	0.90	0.81	**0.94**
Others						
wind-howl	**0.92**	**0.92**	**0.92**	0.80	**0.92**	0.86
wind-strength	**0.96**	0.86	**0.88**	0.87	**0.88**	0.56
water-fill	0.39	0.32	0.41	**0.75**	**0.73**	-0.27
bees-cf	**0.97**	0.90	**0.98**	**0.97**	**0.98**	0.80
bees-busy	-0.21	**0.89**	**0.89**	-0.34	-0.43	**0.92**
applause-rate	0.66	0.83	**0.86**	0.66	0.78	**0.90**
applause-clappers	**0.97**	0.94	0.93	**0.99**	**0.99**	**0.99**

Table 3. Pearson's Correlation between the avg human rank orders and the distance from the anchor computed by the objective metrics. The best two values in every row are in bold. Correlation values >=0.5 are green, <0.5 orange.

Figure 4 shows the plots of all the metric values for three textures compared with human responses in terms of average rank order. The Pearson's correlation between the objective metrics and the subjective responses are presented for all textures in Table 3, and their corresponding plots are available in the Supplementary Material.

5.3.1 Metric Performances

The L2 metric is again a poor performer. This is particularly clear for drumbreak-tempo textures as can be observed in the L2 column in Figure 4(b). FAD performs better than Gram matrix based measures for chimes-size, but shows worse performance for FM-cf, FM-mi, FBnoise-pitchedness, and waterfill (Table 3). FAD performs well for most of the rhythmic textures except pop-rate, drum-tempo, and drum-rev. Since FAD preserves some information about exact spectro-temporal structure, the result is variable sensitivity to parameters (Figure 4, FAD row).

The statistics designed by [1] are known to synthesize good quality sounds for natural textures such as bees and wind which is clearly reflected in our results. However, these statistics are also known for low realism scores for synthesized pitched sounds such as windchimes, and rhythmic sounds such as drum break. Compared to the Gram matrix based metrics, CPM performs well except for windchimes-strength, pop-rate, drum-rev, and water-fill.

When an event rate varies, the features captured by the Gram matrix may reflect some individual events, especially for shorter kernel sizes. In such cases, AGM captures the summary of those statistics instead of individual events, thereby showing a higher parameter sensitivity than metrics based on the entire matrix. This trend can be seen

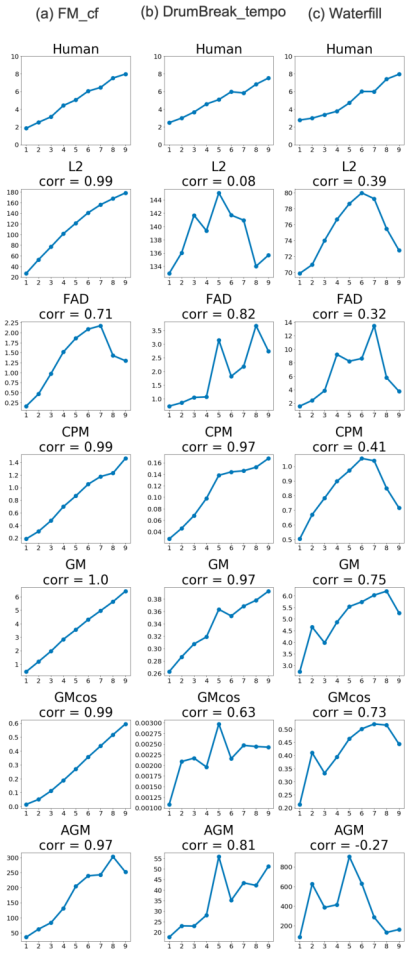


Figure 4. Trends of human responses and objective metrics to parameter variations for three texture-param combinations, (a) FM with carrier frequency variable parameter, (b) Drum-break with tempo a variable parameter, and (c) Waterfill with fill-level parameter, along with the Pearson's correlation between the objective metric values and the subjective responses.

in applause-rate, pop-rate, tapping-rate, chirp-rate, as well as bees-busy (increase in busy body parameter sounds like increase in the rate of buzzing vibrations), where AGM outperforms GM.

The cases where GM outperforms AGM are FM-mf, bees-cf, windchimes-strength, waterfill, and wind-strength, where the parameter variation occurs in the frequency domain, i.e. the center frequency in bees-cf, wind and windchimes-strength, modulation frequency in FM-mf, and resonant frequencies in water-fill. AGM summarization loses information about the statistics in the frequency domain, and this suggests a future study where methods such as eigenvalue decomposition might be better for extracting key information from the Gram matrix.

5.3.2 Some curious cases

Water-fill shows an interesting trend in most of the metrics, where there's a gradual increase, and then a decrease in the metric values (Figure 4(c)), possibly because the resonances of the container wrt changing water and air column heights cause the audio textures of an almost full container to have some statistical similarities with that of an almost empty container. That is, the high-level fill-level parameter has a non-linear affect on the resonant frequencies of the system. For a sound such as water-fill where multiple perceptual dimensions are varying, GM and GMcos show higher correlation with humans than others.

Nsynth-pitch is a curious example where none of the metrics perform well except CPM. The sustained portion of musical tones are atypical as textures because of the lack of temporal variation. Each spectrogram frame is almost exactly the same. Also, as the pitch of the brass instrument increases, the harmonics also change. The Gram matrix summarizes the temporal statistics through the 1D audio representations, but not the frequency statistics, thereby showing a noisy performance.

6. DISCUSSION

We explore a variety of metrics for use with audio textures for their sensitivity to controlled parametric variation. However, in a realistic situation, there may effectively be multiple simultaneous dynamic parameters. One such example in our dataset was the waterfill texture-type, where the fill parameter maps to both rising and falling resonant frequencies. We saw that most of the metrics were responding to a combination of fill parameter and resonances. Further investigation is required to understand how metrics behave in such complex realistic situations.

The systematic parameter variation we used creates textures perceptually close to each other. Further study is required to understand the behavior of these metrics for cross texture-type comparisons, for example, bees compared to water. Such a study would help in mapping audio textures to a perceptual space, the way musical instrument space has been mapped in previous work [19, 20].

Our perceptual study involved placing textures within a 1D space between two reference textures, but there is an inherent lack of an absolute perceptual frame of reference for the meaning of control parameter variation. A systematic study is needed to understand perceptual "just noticeable differences" in parameter variations for complex sounds. Different parametric variations could then be compared on a unified scale.

7. CONCLUSIONS

We present a comprehensive study on the parameter change sensing property of various existing audio evaluation metrics as well as three potential audio statistical and deep-feature based metrics [4]. CPM and FAD emerge as the best metrics, while GM based metrics show promising results. This shows the potential of these deep-features for the purpose of evaluation of audio textures. This study is a fruitful first step towards understanding audio textures, metric design for audio textures, building better synthesis models of this rich and complex class of sounds, and generally toward mapping the space of audio textures.

[4] Code base: `https://github.com/chitralekha18/ParamSensitiveMetrics`

8. REFERENCES

[1] J. H. McDermott and E. P. Simoncelli, "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis," *Neuron*, vol. 71, no. 5, pp. 926–940, 2011.

[2] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1068–1077.

[3] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms." in *INTERSPEECH*, 2019, pp. 2350–2354.

[4] R. McWalter and J. H. McDermott, "Adaptive and selective time averaging of auditory scenes," *Current Biology*, vol. 28, no. 9, pp. 1405–1418, 2018.

[5] J. M. Antognini, M. Hoffman, and R. J. Weiss, "Audio texture synthesis with random neural networks: Improving diversity and quality," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3587–3591.

[6] C. S. Sastry and S. Oore, "Detecting out-of-distribution examples with in-distribution examples and gram matrices," *NeurIPS Workshop on Safety and Robustness in Decision Making*, 2019.

[7] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 586–595.

[8] D. Ulyanov and V. Lebedev, "Audio texture synthesis and style transfer," *[Blog post]. Available from: https://dmitryulyanov. github. io/audio-texture-synthesis-and-style-transfer/[accessed 16 Jan 2022]*, 2016.

[9] L. Wyse and P. T. Ravikumar, "Syntex: parametric audio texture datasets for conditional training of instrumental interfaces." *International Conference on New Interfaces for Musical Expression*, 4 2022, https://nime.pubpub.org/pub/0nl57935. [Online]. Available: https://nime.pubpub.org/pub/0nl57935

[10] J. Bruna and S. Mallat, "Audio texture synthesis with scattering moments," *arXiv preprint arXiv:1311.0407*, 2013.

[11] J. Andén, V. Lostanlen, and S. Mallat, "Joint time–frequency scattering," *IEEE Transactions on Signal Processing*, vol. 67, no. 14, pp. 3704–3718, 2019.

[12] L. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," *Advances in neural information processing systems*, vol. 28, pp. 262–270, 2015.

[13] H. Caracalla and A. Roebel, "Sound texture synthesis using ri spectrograms," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 416–420.

[14] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," *Advances in neural information processing systems*, vol. 29, pp. 2234–2242, 2016.

[15] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," *arXiv preprint arXiv:1902.08710*, 2019.

[16] D. Dowson and B. Landau, "The fréchet distance between multivariate normal distributions," *Journal of multivariate analysis*, vol. 12, no. 3, pp. 450–455, 1982.

[17] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "Cnn architectures for large-scale audio classification," in *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2017, pp. 131–135.

[18] A. J. Neto, A. G. Pacheco, and D. C. Luvizon, "Improving deep learning sound events classifiers using gram matrix feature-wise correlations," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3780–3784.

[19] J. M. Grey, "Multidimensional perceptual scaling of musical timbres," *the Journal of the Acoustical Society of America*, vol. 61, no. 5, pp. 1270–1277, 1977.

[20] P. Esling, A. Chemla-Romeu-Santos, and A. Bitton, "Bridging audio analysis, perception and synthesis with perceptually-regularized variational timbre spaces." in *ISMIR*, 2018, pp. 175–181.

STABILITY OF SYMBOLIC FEATURE GROUP IMPORTANCE IN THE CONTEXT OF MULTI-MODAL MUSIC CLASSIFICATION

Igor Vatolkin
TU Dortmund University
Department of Computer Science
igor.vatolkin@udo.edu

Cory McKay
Marianopolis College
Department of Liberal and Creative Arts
cory.mckay@mail.mcgill.ca

ABSTRACT

Multi-modal music classification creates supervised models trained on features from different sources (modalities): the audio signal, the score, lyrics, album covers, expert tags, etc. A concept of "multi-group feature importance" not only helps to measure the individual relevance of features of a feature type under investigation (such as the instruments present in a piece), but also serves to quantify the potential for further improving classification by adding features from other feature types or extracted from different kinds of sources, based on a multi-objective analysis of feature sets after evolutionary feature selection. In this study, we investigate the stability of feature group importance when different classification methods and different measures of classification quality are applied. Since musical scores are particularly helpful in deriving semantically meaningful, robust genre characteristics, we focus on the feature groups analyzed by the jSymbolic feature extraction software, which describe properties associated with instrumentation, basic pitch statistics, melody, chords, tempo, and other rhythmic aspects. These symbolic features are analyzed in the context of musical information drawn from five other modalities, and experiments are conducted involving two datasets, one small and one large. The results show that, although some feature groups can remain similarly important compared to others, differences can also be evident in various applications, and can depend on the particular classifier and evaluation measure being used. Insights drawn from this type of analysis can potentially be helpful in effectively matching specific features or feature groups to particular classifiers and evaluation measures in future feature-based MIR research.

1. INTRODUCTION AND RELATED WORK

There are music research scenarios where manually designed features can have value, and where gaining understanding of which features are particularly effective for various classification tasks can be of central importance. Researchers seeking to gain domain knowledge can be interested in more than just optimizing performance. Those working on composer attribution or genre classification, for example, might be interested in learning about the specific qualities that delineate musical style, so comparing the performance of different musically meaningful features can provide important insight. There can also be situations where only very limited training data are available, such as when there is only so much extant music by a given composer, and where data augmentation techniques can only improve matters so much. In such situations, deep learning that self-learns features can be less useful, and handmade features become valuable, as do approaches for selecting more promising features when many are available.

Multi-modal features drawn from a variety of data sources can be of particular interest. Although there can be redundancy across types of musical data (e.g., both audio and symbolic data specify pitch), such information is not always equally accessible (e.g., pitch is harder to extract from dense polyphonic audio), and there are other times when different source types contain complementary information (e.g., album art can provide cultural information that is inaccessible from audio or scores). Multi-modal research often plays an essential role in musicology, and involving sources as diverse as concert programs, manuscript illuminations, critical accounts, contemporary visual portrayals, and scores themselves. MIR research can and has similarly taken advantage of multi-modal information, both for improving performance in an engineering sense and in learning more about the connections between different types of musical data in a scientific or musicological sense. There is a rich MIR literature involving two modalities, but relatively few studies have considered more [1–6], and relatively few multi-modal public datasets are available (standouts are described in [5, 7–10]). There are also several papers that provide useful summaries of multi-modal MIR research [11–14].

Of course, with more types of data come more features, and the curse of dimensionality becomes a concern. Although handmade features can help, since they tend to represent information more concisely than raw data, too many can still present problems when training data are limited. Although there is substantial literature on dimensionality reduction (e.g., [15]), and important related MIR studies [4, 16–23], some approaches sacrifice feature independence (e.g., PCA), which compromises interpretability,

© I. Vatolkin and C. McKay. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** I. Vatolkin and C. McKay, "Stability of Symbolic Feature Group Importance in the Context of Multi-Modal Music Classification", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

and feature selection approaches can be sensitive to particular classifier and evaluation methodologies.

So, it can be of value to get a sense of how consistently features perform across different classification and evaluation methodologies, in order to gain a deeper musical understanding of which features and groups of features are most important in delineating classes of interest. This paper is concerned with exactly this: attempting to measure the stability in importance of various features for a given music classification task, which can ultimately help to avoid overfitting the choice of features (or modalities), and can provide insights into underlying musical meaning.

We focus here on symbolic features, as they tend to be particularly interpretable, especially when collaborating with musicologists or theorists. This research maintains a fundamentally multi-modal character, as it also uses information from five other source types to ground feature stability measurements in a broader context: we used the same audio, album cover image, playlist co-occurrence, semantic tag, and lyric text feature data as [10]. These data are particularly useful because they include both a small but clean dataset and a large but noisy dataset, permitting stability to be measured in both scenarios. All symbolic features were extracted with jSymbolic [24], which provides access to many features usefully grouped into feature types whose relative stability can be compared.

We selected genre classification as the test domain for this research, but there is nothing about the techniques we propose that is specific to genre. Furthermore, the same techniques could just as easily be refocused on modalities other than symbolic music. It is also important to note that there are fundamental concerns related to the evaluation of musical classification [25–28]; although there is insufficient space to detail these here, this is essential reading for MIR researchers working in classification.

2. MULTI-GROUP FEATURE IMPORTANCE

We have proposed the concept of *multi-group feature importance* in [10], as an extension of earlier work [29–31]. A feature selection scenario based on two objectives was constructed, with the goal of simultaneously minimizing balanced relative error m_{BRE} (defined as a mean of relative errors for positive and negative instances) and maximizing the share of features g_i belonging to a given group i under investigation, such as rhythmic descriptors. This approach makes it possible to answer not only the questions "what is the lowest error rate achieved by rhythmic features in predicting a musical category?" and "which are the best rhythmic features for the current prediction task?", but also "how can performance be improved when features of other groups, domains, or modalities are also introduced?"

While the formal details of "importance" and the mathematical backing of multi-objective optimization are left to [10], we will briefly illustrate the core ideas here with the help of Figure 1. The connected circles in Subfigure (a) show a *non-dominated front* after feature selection, which simultaneously maximizes the share of rhythmic features g_{RHYTHM} and minimizes m_{BRE}, when features from all six modalities are considered (i.e. not just other symbolic features). Each circle corresponds to a feature set that is not *dominated* by any other feature set. This means that no other feature sets have been identified after feature selection that are "better" than the one under consideration, in combined terms of the two measures being optimized (i.e., using *a greater share* of rhythmic descriptors and achieving *smaller* classification error at the same time). For instance, the feature set in the upper right corner contains only rhythmic descriptors ($g_{RHYTHM} = 1.0$), but $m_{BRE} = 0.4236$ is rather high. No other feature sets that contain only rhythmic descriptors have lower errors, meaning that they are dominated by the upper right corner set, and are not shown in the plot. The feature set in the bottom left corner has the smallest $m_{BRE} = 0.0139$, but has only 37.86% rhythmic descriptors. Other circled feature sets between these corner solutions consist of various trade-offs between the two measures, and are also not dominated by any other found feature sets. The non-dominated fronts in the figures are constructed after ten feature selection repetitions and an independent evaluation on the reserved test set created with music tracks used neither for training the models nor for evaluating the selected feature sets.

Subfigure (a) indicates that rhythmic descriptors do not perform well for Traditional Blues music. This is not only because the error is high in general (this can occur, for example, if a category is too hard to predict or is badly defined), but also because the error is substantially reduced if other features (e.g., audio features) are allowed to contribute to the feature sets used to train the classification models. So, the overall *multi-group feature importance* of rhythmic descriptors is low, which is indicated graphically by the large area shaded grey on the graph. As the number of all possible non-empty feature sets is typically very high ($2^N - 1$ for N features), an evolutionary algorithm is proposed in [10] to explore many different combinations of features in a non-deterministic way, guided by a process inspired by natural evolution, where feature sets can be randomly changed by *mutation* that switches feature dimensions on or off. Only fitter feature sets survive after an exit condition is fulfilled, such as a cap on the overall number of mutations.

In [10], we have also investigated the importances of different modalities and proposed the complementary concept of *multi-group feature redundancy*. However, despite a large number of experiments, the results should be treated with caution because of two limitations of the study: only random forest classifiers were used and balanced relative error was the only measure of classification quality. It was therefore unclear whether estimated importances would remain similar if other classification methods were applied or other evaluation measures used. The present study addresses both of these gaps, by investigating how stable multi-group feature performance is when classification method or evaluation measure are varied.

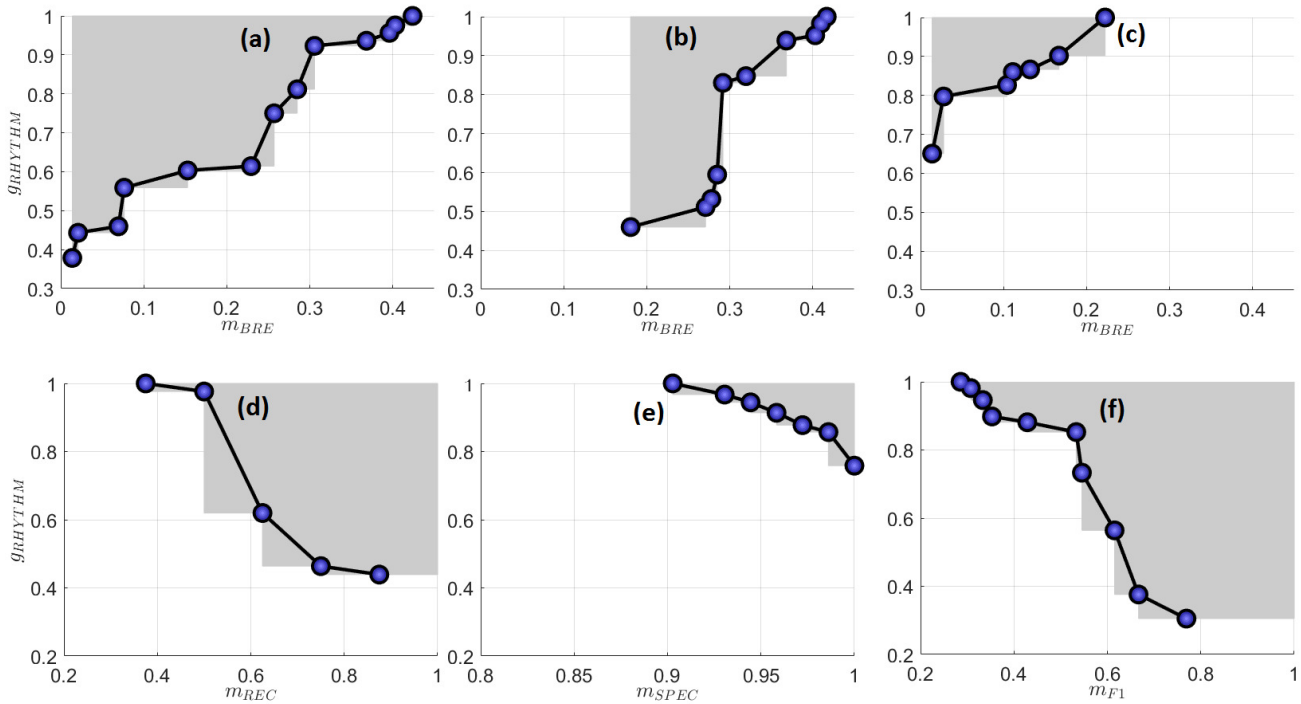


Figure 1. Non-dominated fronts after multi-objective feature selection for the identification of Traditional Blues in the SLAC dataset [2], with rhythmic descriptors the feature group being focused on. Top row: the share of rhythmic descriptors g_{RHYTHM} is maximized and the balanced relative error m_{BRE} is minimized using random forest (a), k-nearest neighbors (b), or support vector machine (c) classifiers. Bottom row: random forest classifiers are used to maximize both g_{RHYTHM} and recall m_{REC} (d), specificity m_{SPEC} (e), or F1-measure m_{F1} (f).

3. DESIGN OF EXPERIMENTS

We conducted our experiments on two pre-existing datasets, namely LMD-aligned [32] and SLAC [2], involving a total of 20 genres and sub-genres. While LMD-aligned is a larger dataset, it has problems with noisiness and structure (e.g., it is unbalanced, with an overly large representation of Pop/Rock songs). SLAC is smaller but carefully designed with, for example, an equal number of pieces belonging to each genre, each of which can be broken into two equally represented sub-genres. We made use of multi-modal features extracted from 1,575 Lakh and 250 SLAC tracks, which have been made publicly available. [1]

Table 1 shows sample jSymbolic features, divided into feature groups; the full list of symbolic (and other) features is available online. [2] We excluded the dynamics features used in [10], as dynamics are often inconsistently encoded.

In the first part of this study we measure the importance of eight groups of symbolic features (relative to features from all six modalities considered) not only with the random forests (RF) classifiers used in [10], but also with k-nearest neighbor (kNN) and support vector machine (SVM) classifiers. As can be observed in Figure 1, subfigures (a)-(c), notable differences in importance can be evident when the classification method is changed. Of course, a limitation of this study is that all classifiers are applied with default hyperparameters in the AMUSE framework [33]: 100 trees for RF, $k = 1$ for kNN and a linear kernel for SVM. In practice, varying hyperparameters may well also introduce meaningful variance in measured feature importances.

We chose to omit deep neural networks from our experiments, despite their popularity in MIR, for two reasons. First, they typically learn their own features, which can make it difficult to analyze the relative relevance of interpretable semantic descriptors for a given musical category. Second, the typically very large number of parameters they involve can lead to overfitting in situations with limited available data, such as when a musical category is defined by a small number of "positive" and "negative" tracks, as in real-world situations where a listener may wish to provide only a few labeled examples to train a supervised classification model.

In the second part of this study we vary the evaluation measures used in the two-objective feature selection. Although m_{BRE} is generally a good choice for binary classification tasks, as it can help to measure performance with unbalanced test sets [34, p.344], it is not often used in MIR classification studies. We have therefore extended the setup from [10] with three additional evaluation criteria. Recall and specificity measure classification errors associated with, respectively, instances annotated in the ground truth as belonging or not belonging to a category [34, p.342]. F1-measure is a weighted combination of precision and recall that, like m_{BRE}, can be useful for less

[1] https://zenodo.org/record/5651429
[2] https://doi.org/10.5334/tismir.67.s1

Group	Feature Examples
Pitch	First pitch, last pitch, major or minor, pitch class histogram, pitch variability, range
Melodic	Amount of arpeggiation, direction of melodic motion, melodic intervals, repeated notes
Chords	Chord type histogram, dominant seventh chords, variability of number of simultaneous pitches
Rhythm	Initial time signature, metrical diversity, note density per quarter note, prevalence of dotted notes
Tempo	Initial tempo, mean tempo, minimum and maximum note duration, note density and its variation
Instrument presence	Note prevalences of pitched and unpitched instruments, pitched instruments present
Instrument prevalence	Prevalences of individual instruments/instrument groups: acoustic guitar, string ensemble, etc.
Texture	Average number of independent voices, parallel fifths and octaves, voice overlap

Table 1. Sample jSymbolic [24] features grouped into eight semantically meaningful groups.

balanced datasets [34, p.344]. The potential for differences in measured importances based on recall, specificity, and F1 is demonstrated in subfigures (d), (e), and (f).

We conducted 24,000 feature selection experiments involving 20 genres and sub-genres $\times$ 3 cross-validation folds $\times$ 8 feature groups $\times$ 5 new combinations of classifiers and measures $\times$ 10 statistical repetitions, with runtimes between about 2 and 50 hours per experiment.

4. DISCUSSION OF RESULTS

Tables 2 and 3 show multi-group feature importances for all 20 classes, with each value indicating results aggregated over three folds. In each row, the importance values across the eight feature groups are shown with color: the group with the highest importance (compared to the other feature groups in the same row) is marked in deep red, and the group with the lowest importance in deep blue. The four "more important" feature groups are shown in shades of red, and four "less important" groups in shades of blue.

The instrument presence feature group seems to be the most important group in most of the experiments, with the instrument prevalence group being the most important in a few others. Features measuring pitch statistics are the most important in five experiments (e.g., RnB prediction using kNN and m_{BRE}). Interestingly, instrument prevalence is the least important group for almost all LMD-aligned genres (even when instrument presence is the most important group). Another intriguing result is that melodic features seem to be particularly unimportant for the Classical SLAC genre and its ClassBaroq and ClassRomant sub-genres.

As anticipated in Figure 1, these results reveal differences between importances when the classifier is varied. Such difference may be particularly meaningful when a cell's color changes from red to blue or vice versa: for example, chords are among the four more important groups for predicting Country using RF and SVM, but are among the less important groups when kNN is used. On the other hand, for this same genre tempo-based features become more important using kNN, not only relatively (the cell goes from blue to red), but also with respect to mean importance values (0.861 instead of 0.656 and 0.651, respectively). This supports our preliminary suspicion that the choice of classifier can have a strong impact, even for "robust" features groups (e.g., symbolic characteristics arguably describe musical properties in concise and clearly understood ways compared to audio descriptors). However, this is certainly not always the case: for 480 combinations of 20 classes, 8 feature groups, and 3 classifiers, in 62 cases (12.92%) the cell shade remains the same when the classifier is changed for a fixed genre and feature group.

When the evaluation measures are varied for a fixed classifier (note that here results were restricted to RF), the changes seem to be slightly less impactful: for 640 related combinations (20 classes, 8 feature groups, 4 measures), the shade changed in only 53 cases (8.28%).

A randomly chosen decision will assign a feature group to be either "more" or "less" relatively important with an expected probability of 25% for three classifiers (1 case with all more important values, 1 case with all less important values, and 6 remaining cases), and with 12.5% for the four measures. Although this interpretation is not perfect, as the change of a light red shade to a light blue shade will indicate a switch between "more" and "less" important, our main inference from the complete study is that it is not enough to claim that some feature group is "generally" more or less important for a particular musical category. Our results suggest that it is indeed necessary to accompany feature group evaluations with specification of the classifier and evaluation measure used. More general claims about the suitability of particular feature groups should be substantiated with broader experiments involving multiple classifiers and evaluation measures (and, perhaps, classifier hyper-parameters).

5. CONCLUSIONS

This paper studied the stability of multi-group feature importance when classifiers and evaluation measures are varied. Eight symbolic feature groups were focused on within a multi-modal classification context involving features extracted from six modalities (symbolic and five others). Various combinations of features and classification approaches were used to predict genres and sub-genres for two datasets with publicly available pre-extracted feature values. The results show that, although in most cases the relative importance of individual feature groups is not affected by the choice of classification method or evaluation measure, either or both of these do nonetheless have an influence in a non-negligible number of cases. Multiple parameters of the feature importance estimation chain can impact determination of which musical properties are more or less relevant for a particular musical category.

In the future, we plan to continue our experiments by

		Pitch	Melodic	Chords	Rhythm	Tempo	Instr. Pres.	Instr. Prev.	Texture
Country	RF	0.796±0.05	0.675±0.07	0.798±0.02	0.820±0.04	0.656±0.03	0.945±0.01	0.614±0.02	0.700±0.03
	kNN	0.924±0.08	0.704±0.07	0.832±0.03	0.865±0.04	0.861±0.06	0.952±0.03	0.645±0.02	0.779±0.03
	SVM	0.756±0.03	0.688±0.04	0.785±0.04	0.730±0.03	0.651±0.05	0.861±0.03	0.633±0.01	0.665±0.02
	RF-F1	0.610±0.09	0.439±0.03	0.576±0.10	0.654±0.05	0.413±0.11	0.889±0.01	0.285±0.12	0.381±0.08
	RF-Rec	0.849±0.01	0.722±0.06	0.823±0.03	0.882±0.02	0.755±0.08	0.965±0.01	0.668±0.06	0.733±0.08
	RF-Spec	0.760±0.03	0.678±0.06	0.768±0.04	0.785±0.04	0.618±0.06	0.927±0.02	0.604±0.09	0.674±0.03
Electronic	RF	0.825±0.04	0.774±0.02	0.815±0.01	0.871±0.02	0.754±0.07	0.951±0.02	0.697±0.03	0.746±0.02
	kNN	0.867±0.07	0.748±0.11	0.861±0.06	0.914±0.06	0.954±0.01	0.964±0.02	0.700±0.04	0.708±0.05
	SVM	0.824±0.04	0.773±0.01	0.770±0.05	0.836±0.05	0.768±0.00	0.916±0.01	0.683±0.03	0.758±0.03
	RF-F1	0.695±0.05	0.583±0.05	0.681±0.02	0.771±0.02	0.542±0.08	0.934±0.01	0.450±0.01	0.553±0.02
	RF-Rec	0.877±0.03	0.751±0.08	0.830±0.05	0.911±0.03	0.738±0.12	0.944±0.01	0.718±0.04	0.759±0.04
	RF-Spec	0.757±0.09	0.738±0.05	0.748±0.02	0.835±0.02	0.742±0.04	0.956±0.01	0.660±0.03	0.733±0.01
Pop	RF	0.752±0.05	0.694±0.07	0.735±0.04	0.806±0.04	0.701±0.04	0.927±0.02	0.626±0.06	0.713±0.04
	kNN	0.946±0.04	0.898±0.01	0.908±0.01	0.982±0.01	0.961±0.03	0.993±0.00	0.837±0.06	0.906±0.08
	SVM	0.799±0.08	0.718±0.08	0.777±0.02	0.780±0.04	0.781±0.01	0.875±0.02	0.738±0.02	0.795±0.04
	RF-F1	0.788±0.02	0.668±0.06	0.782±0.02	0.786±0.03	0.732±0.05	0.916±0.04	0.649±0.05	0.748±0.01
	RF-Rec	0.803±0.10	0.729±0.04	0.769±0.11	0.870±0.07	0.722±0.13	0.959±0.02	0.643±0.03	0.745±0.09
	RF-Spec	0.575±0.08	0.510±0.10	0.593±0.11	0.612±0.06	0.497±0.04	0.899±0.03	0.475±0.09	0.569±0.08
RnB	RF	0.808±0.04	0.689±0.10	0.807±0.05	0.844±0.03	0.716±0.02	0.958±0.02	0.621±0.09	0.697±0.07
	kNN	0.921±0.06	0.601±0.08	0.778±0.00	0.892±0.06	0.818±0.20	0.920±0.06	0.402±0.02	0.743±0.07
	SVM	0.764±0.07	0.699±0.03	0.780±0.04	0.788±0.04	0.735±0.01	0.889±0.03	0.663±0.10	0.691±0.07
	RF-F1	0.711±0.06	0.418±0.16	0.603±0.03	0.661±0.03	0.448±0.17	0.892±0.00	0.306±0.07	0.417±0.03
	RF-Rec	0.818±0.01	0.778±0.03	0.834±0.03	0.861±0.04	0.738±0.06	0.972±0.00	0.665±0.04	0.679±0.08
	RF-Spec	0.829±0.06	0.683±0.02	0.738±0.02	0.832±0.02	0.704±0.04	0.930±0.01	0.601±0.07	0.637±0.02
Rock	RF	0.726±0.06	0.602±0.03	0.713±0.07	0.748±0.03	0.655±0.03	0.917±0.02	0.598±0.02	0.657±0.05
	kNN	0.949±0.05	0.856±0.03	0.919±0.02	0.961±0.04	0.983±0.01	0.984±0.01	0.813±0.05	0.948±0.02
	SVM	0.768±0.03	0.756±0.01	0.766±0.02	0.780±0.03	0.710±0.04	0.886±0.04	0.625±0.03	0.686±0.04
	RF-F1	0.767±0.03	0.641±0.08	0.742±0.01	0.773±0.05	0.679±0.08	0.928±0.02	0.608±0.03	0.663±0.04
	RF-Rec	0.808±0.04	0.693±0.10	0.766±0.05	0.887±0.01	0.784±0.06	0.968±0.01	0.641±0.07	0.731±0.01
	RF-Spec	0.612±0.08	0.446±0.04	0.586±0.08	0.603±0.05	0.482±0.07	0.859±0.06	0.509±0.12	0.508±0.05
Blues	RF	0.923±0.03	0.742±0.07	0.858±0.06	0.902±0.03	0.759±0.05	0.975±0.00	0.769±0.10	0.723±0.13
	kNN	0.743±0.06	0.650±0.20	0.659±0.11	0.841±0.04	0.725±0.14	0.945±0.00	0.576±0.04	0.624±0.07
	SVM	0.792±0.02	0.557±0.14	0.665±0.20	0.795±0.08	0.584±0.04	0.916±0.01	0.621±0.13	0.627±0.03
	RF-F1	0.806±0.09	0.551±0.20	0.656±0.11	0.801±0.07	0.534±0.04	0.953±0.02	0.645±0.03	0.525±0.12
	RF-Rec	0.984±0.01	0.865±0.07	0.910±0.10	0.982±0.02	0.891±0.05	0.994±0.00	0.904±0.09	0.865±0.12
	RF-Spec	0.943±0.03	0.826±0.10	0.954±0.02	0.899±0.01	0.894±0.01	0.984±0.00	0.898±0.05	0.857±0.06
Classical	RF	0.934±0.02	0.782±0.04	0.962±0.01	0.949±0.01	0.872±0.02	0.996±0.00	0.945±0.05	0.885±0.05
	kNN	0.858±0.04	0.690±0.08	0.844±0.04	0.829±0.04	0.811±0.01	0.990±0.01	0.945±0.05	0.791±0.08
	SVM	0.872±0.01	0.706±0.06	0.915±0.01	0.836±0.04	0.837±0.04	0.979±0.02	0.963±0.03	0.812±0.03
	RF-F1	0.848±0.04	0.679±0.04	0.910±0.04	0.878±0.04	0.722±0.04	0.989±0.01	0.889±0.10	0.738±0.08
	RF-Rec	0.987±0.01	0.925±0.07	0.989±0.02	0.982±0.02	0.952±0.03	1.000±0.00	1.000±0.00	0.983±0.03
	RF-Spec	0.953±0.02	0.836±0.07	0.981±0.02	0.960±0.00	0.892±0.04	0.992±0.01	0.925±0.07	0.905±0.06
Rock	RF	0.933±0.00	0.887±0.03	0.911±0.07	0.917±0.03	0.833±0.05	0.995±0.00	0.915±0.04	0.905±0.02
	kNN	0.791±0.15	0.716±0.09	0.753±0.08	0.802±0.13	0.596±0.32	0.995±0.00	0.804±0.01	0.702±0.11
	SVM	0.808±0.02	0.827±0.09	0.816±0.09	0.797±0.04	0.758±0.03	0.982±0.00	0.866±0.06	0.749±0.06
	RF-F1	0.799±0.03	0.705±0.06	0.843±0.05	0.804±0.02	0.628±0.05	0.984±0.01	0.787±0.04	0.743±0.03
	RF-Rec	0.960±0.03	0.971±0.03	0.982±0.03	0.969±0.03	0.910±0.07	1.000±0.00	1.000±0.00	0.974±0.02
	RF-Spec	0.956±0.03	0.878±0.01	0.955±0.01	0.947±0.01	0.870±0.02	0.994±0.00	0.928±0.02	0.941±0.06
Jazz	RF	0.967±0.01	0.924±0.02	0.965±0.03	0.966±0.00	0.900±0.02	0.996±0.00	0.908±0.02	0.888±0.06
	kNN	0.886±0.03	0.566±0.05	0.764±0.10	0.772±0.11	0.754±0.11	0.985±0.01	0.606±0.11	0.618±0.23
	SVM	0.898±0.03	0.832±0.04	0.888±0.05	0.777±0.05	0.821±0.04	0.985±0.00	0.820±0.01	0.813±0.04
	RF-F1	0.913±0.01	0.815±0.05	0.929±0.05	0.898±0.04	0.837±0.10	0.990±0.00	0.805±0.02	0.748±0.10
	RF-Rec	0.972±0.02	0.966±0.04	0.991±0.01	0.988±0.02	0.933±0.04	0.998±0.00	0.981±0.03	0.927±0.01
	RF-Spec	0.970±0.02	0.952±0.01	0.983±0.02	0.967±0.01	0.971±0.01	0.999±0.00	0.940±0.01	0.941±0.01
Rap	RF	0.967±0.01	0.920±0.02	0.930±0.05	0.928±0.02	0.861±0.02	0.996±0.00	0.849±0.03	0.912±0.04
	kNN	0.955±0.01	0.731±0.10	0.839±0.04	0.828±0.04	0.797±0.07	0.979±0.00	0.711±0.09	0.555±0.06
	SVM	0.880±0.04	0.705±0.07	0.887±0.06	0.793±0.01	0.788±0.04	0.955±0.01	0.798±0.07	0.823±0.07
	RF-F1	0.918±0.03	0.796±0.01	0.818±0.10	0.847±0.07	0.748±0.06	0.992±0.01	0.762±0.03	0.737±0.21
	RF-Rec	0.990±0.02	0.936±0.01	0.988±0.02	0.959±0.06	0.936±0.03	0.999±0.00	0.960±0.04	0.969±0.04
	RF-Spec	0.975±0.02	0.945±0.01	0.939±0.06	0.950±0.03	0.898±0.03	0.999±0.00	0.938±0.02	0.910±0.05

Table 2. Multi-group symbolic feature importances for five LMD-aligned genres (top half of the table) and five SLAC parent genres (bottom half of the table), aggregated over 3 folds. F1: F1-measure; Rec: recall; Spec: specificity. The first three rows of each genre block were evaluated with balanced relative error m_{BRE}.

measuring the impact of other feature types and modalities. We will also further examine the effects of varying other parameters in the experimental setup, such as classifier hyper-parameters, and also systematically consider as-

		Pitch	Melodic	Chords	Rhythm	Tempo	Instr. Pres.	Instr. Prev.	Texture
BluesModern	RF	0.922±0.05	0.711±0.07	0.775±0.04	0.701±0.18	0.732±0.19	0.965±0.04	0.826±0.05	0.837±0.14
	kNN	0.762±0.07	0.772±0.13	0.631±0.08	0.698±0.03	0.773±0.08	0.966±0.04	0.536±0.07	0.774±0.16
	SVM	0.865±0.03	0.602±0.02	0.660±0.16	0.758±0.12	0.641±0.13	0.946±0.03	0.684±0.17	0.614±0.07
	RF-F1	0.916±0.07	0.488±0.39	0.603±0.40	0.704±0.33	0.511±0.29	0.929±0.09	0.510±0.27	0.711±0.22
	RF-Rec	0.769±0.20	0.629±0.17	0.640±0.07	0.498±0.30	0.895±0.10	0.991±0.02	0.885±0.20	0.794±0.07
	RF-Spec	0.984±0.01	0.966±0.02	0.970±0.03	0.982±0.02	0.963±0.01	0.997±0.00	0.928±0.02	0.962±0.04
BluesTradit	RF	0.836±0.10	0.664±0.10	0.909±0.02	0.949±0.03	0.778±0.18	0.971±0.01	0.844±0.09	0.673±0.14
	kNN	0.798±0.12	0.619±0.14	0.755±0.09	0.851±0.07	0.768±0.10	0.935±0.04	0.805±0.18	0.537±0.10
	SVM	0.844±0.05	0.672±0.14	0.618±0.15	0.773±0.14	0.748±0.10	0.964±0.05	0.598±0.13	0.608±0.11
	RF-F1	0.852±0.04	0.595±0.23	0.645±0.13	0.852±0.04	0.629±0.24	0.960±0.04	0.694±0.19	0.505±0.16
	RF-Rec	0.872±0.10	0.747±0.17	0.943±0.10	0.905±0.14	0.944±0.10	0.999±0.00	0.624±0.45	0.782±0.38
	RF-Spec	0.993±0.01	0.911±0.05	0.916±0.05	0.971±0.03	0.918±0.10	0.991±0.01	0.922±0.09	0.855±0.14
ClassBaroq	RF	0.914±0.07	0.747±0.26	0.921±0.06	0.804±0.11	0.815±0.14	0.997±0.00	0.816±0.03	0.860±0.14
	kNN	0.821±0.06	0.486±0.40	0.754±0.14	0.654±0.03	0.572±0.18	0.985±0.01	0.964±0.04	0.421±0.12
	SVM	0.901±0.02	0.595±0.04	0.856±0.05	0.774±0.06	0.625±0.04	0.985±0.01	0.909±0.08	0.740±0.10
	RF-F1	0.851±0.21	0.520±0.14	0.701±0.30	0.688±0.21	0.593±0.08	0.987±0.02	0.728±0.18	0.539±0.17
	RF-Rec	0.995±0.01	0.861±0.13	0.971±0.05	0.875±0.22	0.778±0.29	1.000±0.00	0.883±0.11	0.976±0.02
	RF-Spec	0.970±0.02	0.922±0.08	0.986±0.01	0.988±0.01	0.954±0.04	1.000±0.00	1.000±0.00	0.934±0.07
ClassRomant	RF	0.831±0.12	0.710±0.14	0.940±0.04	0.893±0.07	0.781±0.12	1.000±0.00	0.895±0.14	0.670±0.09
	kNN	0.888±0.09	0.670±0.25	0.789±0.12	0.947±0.05	0.817±0.14	0.983±0.01	0.804±0.13	0.722±0.23
	SVM	0.817±0.07	0.723±0.18	0.952±0.04	0.860±0.05	0.738±0.04	0.983±0.02	0.971±0.02	0.746±0.11
	RF-F1	0.744±0.04	0.510±0.23	0.864±0.09	0.855±0.03	0.579±0.13	1.000±0.00	0.731±0.10	0.621±0.09
	RF-Rec	0.792±0.14	0.724±0.24	0.974±0.02	0.848±0.16	0.480±0.25	0.999±0.00	1.000±0.00	0.806±0.09
	RF-Spec	0.979±0.02	0.942±0.05	0.999±0.00	0.993±0.01	0.980±0.01	0.999±0.00	0.986±0.01	0.971±0.02
JazzBop	RF	0.957±0.01	0.767±0.15	0.855±0.04	0.824±0.08	0.863±0.03	0.966±0.03	0.818±0.10	0.865±0.03
	kNN	0.761±0.11	0.604±0.07	0.838±0.04	0.751±0.02	0.711±0.08	0.994±0.00	0.713±0.16	0.572±0.06
	SVM	0.844±0.01	0.720±0.06	0.840±0.02	0.784±0.06	0.816±0.06	0.973±0.03	0.713±0.12	0.759±0.05
	RF-F1	0.788±0.03	0.609±0.35	0.722±0.16	0.811±0.03	0.681±0.03	0.962±0.05	0.571±0.15	0.568±0.09
	RF-Rec	0.949±0.04	0.653±0.33	0.774±0.06	0.904±0.11	0.965±0.03	0.954±0.08	1.000±0.00	0.839±0.18
	RF-Spec	0.994±0.01	0.985±0.02	0.995±0.01	0.994±0.01	0.991±0.01	1.000±0.00	0.960±0.05	0.930±0.05
JazzSwing	RF	0.957±0.01	0.863±0.07	0.911±0.06	0.946±0.05	0.932±0.04	0.947±0.04	0.876±0.05	0.861±0.08
	kNN	0.939±0.07	0.840±0.10	0.887±0.08	0.825±0.12	0.707±0.14	0.983±0.01	0.738±0.05	0.749±0.22
	SVM	0.870±0.06	0.823±0.05	0.903±0.07	0.840±0.07	0.807±0.06	0.946±0.03	0.822±0.07	0.800±0.10
	RF-F1	0.868±0.07	0.658±0.05	0.788±0.10	0.888±0.03	0.604±0.16	0.943±0.03	0.759±0.16	0.650±0.08
	RF-Rec	0.988±0.02	0.942±0.06	0.840±0.16	0.943±0.08	0.901±0.12	0.962±0.03	0.867±0.13	0.828±0.26
	RF-Spec	0.994±0.01	0.966±0.01	0.978±0.02	0.996±0.01	0.964±0.02	0.999±0.00	0.978±0.00	0.979±0.02
RapHardcore	RF	0.845±0.00	0.790±0.07	0.898±0.06	0.880±0.03	0.688±0.18	0.965±0.01	0.693±0.11	0.883±0.11
	kNN	0.816±0.09	0.685±0.14	0.801±0.13	0.725±0.07	0.661±0.06	0.969±0.03	0.677±0.24	0.686±0.02
	SVM	0.768±0.10	0.779±0.05	0.820±0.13	0.718±0.18	0.712±0.12	0.913±0.07	0.719±0.10	0.813±0.06
	RF-F1	0.842±0.14	0.727±0.05	0.649±0.23	0.949±0.05	0.838±0.09	0.978±0.01	0.576±0.39	0.842±0.16
	RF-Rec	0.660±0.27	0.700±0.24	0.981±0.02	0.985±0.03	0.888±0.07	0.970±0.01	0.829±0.15	0.954±0.05
	RF-Spec	0.993±0.01	0.959±0.01	0.985±0.01	0.987±0.02	0.964±0.03	1.000±0.00	0.972±0.02	0.989±0.01
RapPop	RF	0.851±0.08	0.637±0.03	0.926±0.04	0.883±0.06	0.787±0.19	0.953±0.01	0.644±0.16	0.658±0.17
	kNN	0.907±0.09	0.629±0.11	0.790±0.12	0.736±0.04	0.732±0.12	0.987±0.01	0.643±0.17	0.668±0.08
	SVM	0.854±0.07	0.776±0.08	0.863±0.09	0.682±0.10	0.625±0.32	0.895±0.06	0.511±0.17	0.756±0.09
	RF-F1	0.835±0.12	0.626±0.38	0.862±0.11	0.755±0.22	0.637±0.32	0.971±0.03	0.710±0.25	0.582±0.21
	RF-Rec	0.972±0.05	0.761±0.18	0.910±0.16	0.529±0.09	0.603±0.41	0.957±0.07	0.734±0.06	0.681±0.48
	RF-Spec	0.982±0.01	0.965±0.04	0.961±0.02	0.961±0.04	0.960±0.02	1.000±0.00	0.935±0.04	0.944±0.03
RockAltern	RF	0.787±0.06	0.768±0.10	0.815±0.07	0.891±0.02	0.824±0.08	0.971±0.01	0.728±0.09	0.681±0.05
	kNN	0.658±0.14	0.678±0.07	0.639±0.10	0.830±0.14	0.751±0.08	0.952±0.05	0.423±0.24	0.593±0.19
	SVM	0.670±0.12	0.679±0.13	0.797±0.08	0.804±0.14	0.751±0.11	0.903±0.07	0.695±0.10	0.592±0.04
	RF-F1	0.535±0.20	0.458±0.05	0.764±0.14	0.740±0.05	0.597±0.13	0.911±0.07	0.442±0.15	0.498±0.24
	RF-Rec	0.578±0.19	0.794±0.19	0.855±0.04	0.884±0.03	0.847±0.22	0.942±0.04	0.698±0.08	0.765±0.10
	RF-Spec	0.977±0.03	0.918±0.06	0.987±0.02	0.980±0.03	0.907±0.11	1.000±0.00	0.963±0.04	0.934±0.01
RockMetal	RF	0.914±0.03	0.840±0.06	0.873±0.09	0.847±0.11	0.840±0.04	0.988±0.00	0.950±0.03	0.840±0.09
	kNN	0.986±0.02	0.797±0.12	0.797±0.18	0.835±0.04	0.720±0.30	0.985±0.01	0.795±0.17	0.601±0.17
	SVM	0.758±0.25	0.788±0.05	0.915±0.04	0.826±0.03	0.750±0.08	0.981±0.02	0.946±0.04	0.810±0.03
	RF-F1	0.943±0.04	0.605±0.14	0.742±0.15	0.739±0.18	0.630±0.15	0.985±0.01	0.802±0.04	0.775±0.04
	RF-Rec	0.965±0.03	0.703±0.23	0.795±0.26	0.757±0.16	0.823±0.05	0.991±0.01	1.000±0.00	0.807±0.16
	RF-Spec	0.995±0.01	0.972±0.02	0.985±0.00	0.992±0.01	0.965±0.03	1.000±0.00	0.992±0.01	0.990±0.01

Table 3. Multi-group symbolic feature importances for ten SLAC sub-genres, aggregated over 3 folds. F1: F1-measure; Rec: recall; Spec: specificity. The first three rows of each sub-genre block were evaluated with balanced relative error m_{BRE}.

pects like extraction times, statistical properties, and suitability for data augmentation. We will also run further trials in order to be able to apply more developed statistical significance testing.

6. ACKNOWLEDGMENTS

The experiments were carried out on the Linux HPC cluster at TU Dortmund (LiDO3), which was partially funded by the Large-Scale Equipment Initiative by the German Research Foundation (DFG) grant 271512359. The second author's work was funded by the Fonds de recherche du Québec – Société et culture, under grants 2021-CHZ-282456 and 2022-CHZ-309882.

7. REFERENCES

[1] C. McKay and I. Fujinaga, "Combining features extracted from audio, symbolic and cultural sources," in *Proc. of the 9th International Conference on Music Information Retrieval, ISMIR*, 2008, pp. 597–602.

[2] C. McKay, J. A. Burgoyne, J. Hockman, J. B. L. Smith, G. Vigliensoni, and I. Fujinaga, "Evaluating the genre classification performance of lyrical features relative to audio, symbolic and cultural features," in *Proc. of the 11th International Society for Music Information Retrieval Conference, ISMIR*, 2010, pp. 213–218.

[3] B. McFee and G. R. G. Lanckriet, "Hypergraph models of playlist dialects," in *Proc. of the 13th International Society for Music Information Retrieval Conference, ISMIR*, 2012, pp. 343–348.

[4] R. Panda, R. Malheiro, B. Rocha, A. Oliveira, and R. P. Paiva, "Multi-modal music emotion recognition: A new dataset, methodology and comparative analysis," in *Proc. of the 10th International Symposium on Computer Music Multidisciplinary Research, CMMR*. Springer, 2013.

[5] S. Oramas, O. Nieto, F. Barbieri, and X. Serra, "Multi-label music genre classification from audio, text and images using deep features," in *Proc. of the 18th International Society for Music Information Retrieval Conference, ISMIR*, 2017, pp. 23–30.

[6] S. Oramas, F. Barbieri, O. Nieto, and X. Serra, "Multimodal deep learning for music genre classification," *Transactions of the International Society for Music Information Retrieval*, vol. 1, no. 1, pp. 4–21, 2018.

[7] N. Orio, D. Rizo, R. Miotto, M. Schedl, N. Montecchio, and O. Lartillot, "Musiclef: a benchmark activity in multimodal music information retrieval," in *Proc. of the 12th International Society for Music Information Retrieval Conference, ISMIR*, 2011, pp. 603–608.

[8] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "DALI: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm." in *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR*, 2018, pp. 431–437.

[9] D. Bogdanov, A. Porter, H. Schreiber, J. Urbano, and S. Oramas, "The acousticbrainz genre dataset: Multi-source, multi-level, multi-label, and large-scale," in *Proc. of the 20th International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 360–367.

[10] I. Vatolkin and C. McKay, "Multi-objective investigation of six feature source types for multi-modal music classification," *Transactions of the International Society for Music Information Retrieval*, vol. 5, no. 1, pp. 1–19, 2022.

[11] R. Mayer and A. Rauber, "Multimodal aspects of music retrieval: Audio, song lyrics - and beyond?" in *Advances in Music Information Retrieval*, Z. W. Ras and A. Wieczorkowska, Eds. Springer, 2010, pp. 333–363.

[12] P. Knees and M. Schedl, "A survey of music similarity and recommendation from music context data," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 10, no. 1, pp. 2:1–2:21, 2013.

[13] D. Jannach, I. Vatolkin, and G. Bonnin, "Music data: beyond the signal level," in *Music Data Analysis: Foundations and Applications*, C. Weihs, D. Jannach, I. Vatolkin, and G. Rudolph, Eds. CRC Press, 2017, pp. 197–215.

[14] F. Simonetta, S. Ntalampiras, and F. Avanzini, "Multimodal music information processing and retrieval: Survey and future challenges," in *Proc. of the International Workshop on Multilayer Music Representation and Processing, MMRP*, 2019, pp. 10–18.

[15] I. Guyon, M. Nikravesh, S. Gunn, and L. A. Zadeh, Eds., *Feature Extraction. Foundations and Applications*, ser. Studies in Fuzziness and Soft Computing. Berlin Heidelberg: Springer, 2006, vol. 207.

[16] I. Fujinaga, "Machine recognition of timbre using steady-state tone of acoustic musical instruments," in *Proc. of the International Computer Music Conference, ICMC*, 1998, pp. 207–210.

[17] R. Fiebrink and I. Fujinaga, "Feature selection pitfalls and music classification," in *Proc. of the 7th International Conference on Music Information Retrieval, ISMIR*, 2006, pp. 340–341.

[18] S. Doraisamy, S. Golzari, N. M. Norowi, M. N. Sulaiman, and N. I. Udzir, "A study on feature selection and classification techniques for automatic genre classification of traditional malay music," in *Proc. of the 9th International Conference on Music Information Retrieval, ISMIR*, J. P. Bello, E. Chew, and D. Turnbull, Eds., 2008, pp. 331–336.

[19] C. N. Silla Jr., A. L. Koerich, and C. A. A. Kaestner, "A feature selection approach for automatic music genre classification," *International Journal of Semantic Computing*, vol. 3, no. 2, pp. 183–208, 2009.

[20] R. Mayer, A. Rauber, P. J. P. de León, C. Pérez-Sancho, and J. M. Iñesta, "Feature selection in a cartesian ensemble of feature subspace classifiers for music categorisation," in *Proc. of the 3rd International Workshop on Machine Learning and Music, MML*. ACM, 2010, pp. 53–56.

[21] P. Saari, T. Eerola, and O. Lartillot, "Generalizability and simplicity as criteria in feature selection: Application to mood classification in music," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 6, pp. 1802–1812, 2011.

[22] S.-C. Lim, J.-S. Lee, S.-J. Jang, S.-P. Lee, and M. Y. Kim, "Music-genre classification system based on spectro-temporal features and feature selection," *IEEE Transactions on Consumer Electronics*, vol. 58, no. 4, pp. 1262–1268, 2012.

[23] Y. Huang, S. Lin, H. Wu, and Y. Li, "Music genre classification based on local feature selection using a self-adaptive harmony search algorithm," *Data Knowledge Engineering*, vol. 92, pp. 60–76, 2014.

[24] C. McKay, J. Cumming, and I. Fujinaga, "jSymbolic 2.2: Extracting features from symbolic music for use in musicological and MIR research," in *Proc. of the 19th International Society for Music Information Retrieval Conference, ISMIR*, 2018, pp. 348–354.

[25] B. L. Sturm, "A survey of evaluation in music genre recognition," in *Adaptive Multimedia Retrieval: Semantics, Context, and Adaptation, 10th International Workshop, AMR*, 2012, pp. 29–66.

[26] ——, "Two systems for automatic music genre recognition: What are they really recognizing?" in *Proc. of the 2nd International ACM Workshop on Music Information Retrieval with User-Centered and Mulitmodal Strategies, MIRUM*, 2012, pp. 69–74.

[27] ——, "Classification accuracy is not enough," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 371–406, 2013.

[28] ——, "Evaluating music emotion recognition: Lessons from music genre recognition?" in *IEEE International Conf. on Multimedia and Expo Workshops, ICMEW*, 2013, pp. 1–6.

[29] I. Vatolkin, M. Preuß, and G. Rudolph, "Multi-objective feature selection in music genre and style recognition tasks," in *Proc. of the 13th Annual Genetic and Evolutionary Computation Conference, GECCO*, N. Krasnogor and P. L. Lanzi, Eds. ACM Press, 2011, pp. 411–418.

[30] I. Vatolkin, "Exploration of two-objective scenarios on supervised evolutionary feature selection: A survey and a case study (application to music categorisation)," in *Proc. of the 8th International Conference on Evolutionary Multi-Criterion Optimization, EMO*. Springer, 2015, pp. 529–543.

[31] I. Vatolkin, G. Rudolph, and C. Weihs, "Evaluation of album effect for feature selection in music genre recognition," in *Proc. of the 16th International Society for Music Information Retrieval Conference, ISMIR*, 2015, pp. 169–175.

[32] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, Graduate School of Arts and Sciences, Columbia University, 2016.

[33] I. Vatolkin, P. Ginsel, and G. Rudolph, "Advancements in the music information retrieval framework AMUSE over the last decade," in *Proc. of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2021, pp. 2383–2389.

[34] C. Weihs, D. Jannach, I. Vatolkin, and G. Rudolph, Eds., *Music Data Analysis: Foundations and Applications*. CRC Press, 2016.

MULTI-PITCH ESTIMATION MEETS MICROPHONE MISMATCH: APPLICABILITY OF DOMAIN ADAPTATION

Franca Bittner **Marcel Gonzalez** **Maike Richter**
Hanna Lukashevich **Jakob Abeßer**
Semantic Music Technologies Group, Fraunhofer IDMT, Ilmenau, Germany
jakob.abesser@idmt.fraunhofer.de

ABSTRACT

The performance of machine learning (ML) models is known to be affected by discrepancies between training (source) and real-world (target) data distributions. This problem is referred to as domain shift and is commonly approached using domain adaptation (DA) methods. As one relevant scenario, automatic piano transcription algorithms in music learning applications potentially suffer from domain shift since pianos are recorded in different acoustic conditions using various devices. Yet, most currently available datasets for piano transcription only cover ideal recording situations with high-quality microphones. Consequently, a transcription model trained on these datasets will face a mismatch between source and target data in real-world scenarios. To address this issue, we employ a recently proposed dataset which includes annotated piano recordings covering typical real-life recording settings for a piano learning application on mobile devices. We first quantify the influence of the domain shift on the performance of a deep learning-based piano multi-pitch estimation (MPE) algorithm. Then, we employ and evaluate four unsupervised DA methods to reduce domain shift. Our results show that the studied MPE model is surprisingly robust to domain shift in microphone mismatch scenarios and the DA methods do not notably improve the transcription performance.

1. INTRODUCTION

Recent advances in Automatic Music Transcription (AMT) enable its practical application in musical education applications where students can record themselves while playing a musical instrument and retrieve a performance feedback in near real-time. The underlying algorithms are driven by deep learning and commonly trained on audio data (source domain), which was gathered in specific and ideal recording setups such as music studios with high-quality microphones [1–3]. In real-life scenarios however, the recording setups may vary from user to user due to different recording devices, room acoustics, and music instrument timbres (target domain). Due to the resulting distribution discrepancy between both domains (domain shift), AMT algorithms might exhibit performance degradation. To overcome this issue, one approach would be to fine-tune pre-trained transcription models using labeled data recorded in real-world settings [4]. This comes with two main drawbacks. First, this procedure would have to be repeated for each user. Second, it requires a lot of effort to obtain perfectly aligned score annotations by manually transcribing audio recordings. For this reason, domain adaptation (DA) methods are used to bridge the gap between different data domains and ensure a good model performance even on previously unseen data. These methods can align the target data distribution to the source data distribution (or vice-versa) [5].

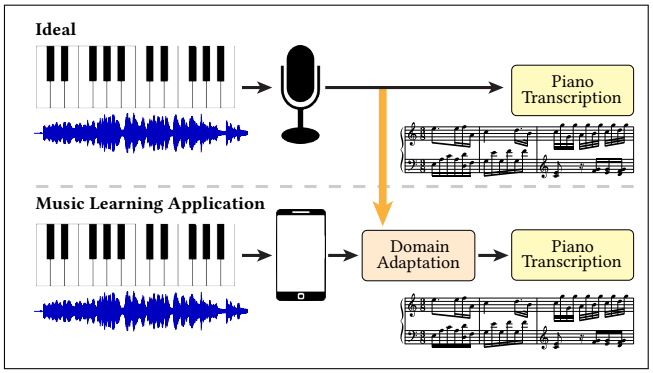


Figure 1. Illustration of the piano transcription process including domain adaptation. Piano music is captured with different recording devices (high-quality microphone, mobile devices), adapted to the source domain, and transcribed by an MPE model.

The contributions of the paper are as follows: We study the task of piano multi-pitch estimation (MPE), i. e., the estimation of simultaneously sounding note pitches, from audio recordings captured with different mobile devices. We first analyze and quantify the mismatch of recording devices by comparing their frequency responses. Then, we study the effectiveness of four different DA methods as a pre-processing step to improve MPE algorithms. We do this by investigating to what extent the microphone mismatch impacts the performance of a deep learning-based

© F. Bittner, M. Gonzalez, M. Richter, H. Lukashevich, and J. Abeßer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** F. Bittner, M. Gonzalez, M. Richter, H. Lukashevich, and J. Abeßer, "Multi-pitch Estimation meets Microphone Mismatch: Applicability of Domain Adaptation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

MPE model with and without DA. We use the representation shift metric to assess whether the domain shift was reduced by DA.

2. DOMAIN ADAPTATION

2.1 Application in Audio Domain

DA was successfully applied for various tasks in different audio domains. Several acoustic scene classification (ASC) algorithms were proposed during the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge [6,7] to compensate domain shift caused by mismatched recording devices. Furthermore, Yang et al. [8] proposed a two-stage domain adaptation approach to improve the robustness of sound event detection (SED) models by aligning synthetic and real audio data distributions in feature space during training. DA was also applied in Music Information Retrieval (MIR) tasks, such as expressivity analysis in piano recordings [9], representation learning for music processing [10], and instrument activity detection [11], and in automatic speech recognition, DA methods are employed to avoid overfitting of a model which was trained on limited data by transferring knowledge from a source model [12].

2.2 Selected Methods

2.2.1 Zero-mean Unit Variance (ZMUV) Normalization

A common pre-processing step for machine learning (ML) models is the standardization of input features to zero mean and a standard deviation of one. We implemented four variants of the zero-mean unit variance (ZMUV) standardization process from [4] as unsupervised DA methods, which do not require data annotations. The statistics used to standardize the target domain data (mean and standard deviation coefficients) are computed either from the source dataset (*global* variants) or from the target dataset (*adaptive* variants). As a consequence, global variants require access to the source domain data whereas adaptive variants only need fractions of the target domain data.The statistics can be computed either over all available files of the selected domain or individually per file. Furthermore, it is possible to have a finer resolution when computing frequency-wise, i. e., by averaging over all time frames and obtaining coefficients per frequency bin, or patch-wise statistics, i. e., by averaging all frequencies within a patch of 16 time frames. These methods are summarized in Table 1. Finally, in addition to standardizing the target domain features, the source domain data will also be standardized before training the model.

2.2.2 Band-wise Statistics Matching (BWSM)

Band-Wise Statistics Matching (BWSM) is an unsupervised DA method, which involves a band-wise alignment of the first and second statistical moments of the target domain data to the ones of the source domain data [13]. Similarly to standardization, the method is applied on the data level and avoids a re-training of a ML model. First, the sample mean and standard deviation values are computed over source and target domain data per frequency bands. Then, a band-wise standardization is applied to the target domain in a similar way as in the adaptive ZMUV normalization per frequency as discussed in the previous section. At the final stage, the adapted features in the target domain $\mathbf{X}_\mathrm{T}$ are aligned to the source domain $\mathbf{X}_\mathrm{S}$, sharing the same means and standard deviations. In contrast to other assessed DA methods, source domain data remains unchanged and the originally trained ML models can be re-used.

Type	Data Scope	Resolution
Global	Domain (all)	Frequency
Adaptive	Domain (all)	Frequency
Adaptive	File	Frequency
Adaptive	File	Patch

Table 1. Overview of the implemented standardization methods w.r.t. their type (global or adaptive), data scope (whole domain or per file), and resolution (subdivision by frequency bins or patches).

2.2.3 Correlation Alignment (CORAL)

Sun et al. [14] proposed CORrelation ALignment (CORAL) as an unsupervised DA method to align the statistical moments of source and target data. After whitening the source domain data (i. e., removing correlation among features), the covariance matrix of the target domain data distribution $\mathbf{C}_\mathrm{T}$ is transferred to the source distribution (re-coloring). Then, a new model needs to be trained on the adapted source domain data.

Given that the distribution of target domain data varies with each mobile device, room, and piano type in our given scenario, it is not feasible to train a new MPE model each time one of these parameters changes. In contrast to the original publication, we implemented CORAL such that the target domain data is adapted using the statistics obtained from the source domain data. We follow [14] and use a regularization parameter $\lambda = 1$ for traditional whitening without a singular value decomposition (SVD). The target domain data $\mathbf{D}_\mathrm{T}$ is whitened as in [14] and then re-colored with the source domain covariance matrix $\mathbf{C}_\mathrm{S}$ as $\mathbf{D}_\mathrm{T} \leftarrow \mathbf{D}_\mathrm{T} * \mathbf{C}_\mathrm{T}^{-\frac{1}{2}} * \mathbf{C}_\mathrm{S}^{\frac{1}{2}}$. This way, the source domain data remains unchanged and the same classifier can be used for all target domains. However, Sun et al. [14] observed a lower performance with this approach and supposed that a model trained on adapted source domain data may benefit from the knowledge inherited by the target data distribution. This is not possible if only the target domain data is modified by DA, as the model is trained on the original source domain data independent of the target domain data.

2.2.4 Feature Projection-Based Unsupervised Domain Adaptation (FPDA)

Mezza et al. [15] proposed Feature Projection-based Unsupervised Domain Adaptation (FPDA) to address the mismatch between the distributions of training and test data acquired under different recording conditions. FPDA is based on the projection of both source and target domain features onto a common subspace using a subset of the eigenvectors of the sample covariance matrix of the source domain data. After standardizing the spectrograms extracted from the source domain in a band-wise fashion, the sample covariance matrix $\mathbf{V}_{\mathrm{s}}$ of the source domain data matrix and the respective eigenvectors and eigenvalues are calculated. The eigenvectors corresponding to the L largest eigenvalues are retained as matrix $\mathbf{V}_{\mathrm{s}}^{(L)}$ and using this matrix source domain and target domain data are projected onto a common subspace as $\mathbf{X}_{\mathrm{S}} = \bar{\mathbf{S}}_{\mathrm{2D}}\mathbf{V}_{\mathrm{s}}^{(L)}$ and $\mathbf{X}_{\mathrm{T}} = \bar{\mathbf{T}}_{\mathrm{2D}}\mathbf{V}_{\mathrm{s}}^{(L)}$.

There are several differences between the application of FPDA in Acoustic Scene Classification (ASC) [15] and our application scenario for real-time piano transcription. Firstly, in [15], source and target domain data contain the same audio signal recorded simultaneously on different recording devices with a constant length of ten seconds, whereas the files contained in our source domain data have entirely different musical content as compared to the target domain data. Moreover, the number of samples varies greatly throughout both the source and target domain dataset in our case. For this reason and for FPDA to be applicable w.r.t. a real-time piano transcription scenario, we chose $L = 16$, so that the DA can be applied in a block-wise fashion, 16 frames at a time. To use the domain-adapted features as input for the piano transcription model, we project the features back to the original feature space as $\hat{\mathbf{S}}_{\mathrm{2D}} = \mathbf{X_S}\mathbf{V}_{\mathrm{s}}^{(L)^T}$ as $\hat{\mathbf{T}}_{\mathrm{2D}} = \mathbf{X_T}\mathbf{V}_{s}^{(L)^T}$. Then, the MPE model is trained on the resulting modified source domain data.

3. MULTI-PITCH ESTIMATION

3.1 Recent Approaches

While traditional MPE methods mostly relied on spectral decomposition techniques such as non-negative matrix factorization (NMF) [16], modern methods are based on different neural network (NN) architectures [17]. Hawthorne et al. [18] propose a general purposed piano transcription model that combines two branches of convolutional recurrent neural networks (CRNNs), which jointly predict the pitches and onset times of played piano notes. Kong et al. [19] proposed a high-resolution AMT system consisting of several CRNNs acoustic models dedicated to different tasks such as velocity, onset and offset regression. More recently, generic encoder-decoder architectures such as transformer networks have been used to remove the need of custom NN design, as done by Hawthorne et al. [20].

3.2 Model Architecture

Following recent advances in the field of computer vision, the U-net architecture is employed for different MIR tasks. In particular, it was used for different AMT tasks such as bass transcription [21], melody transcription [22, 23], as well as polyphonic piano transcription [24]. The structure of a U-net resembles an autoencoder, but is extended by skip connections between encoder and decoder blocks. Input features are first processed by an encoder that sequentially reduces the time-frequency resolution. The intermediate features are then handed over to the decoder part at the so-called bottleneck, where the compression is strongest. In the decoder, the time-frequency resolution is gradually restored by upsampling and interpolation. The main goal is to train the U-net such that it learns a mapping function from a spectrogram-like audio representation to a piano roll representation. In this study, we consider MPE as binary classification task, where each pitch can be either active or inactive at a given time frame. The encoder of our U-net model comprises three layers with alternating blocks of convolutional filter blocks and max pooling. In the decoder, bilinear interpolation operations are used for upsampling. Intermediate activations at similar levels in the encoder and decoder are connected by skip connections. The model output lies in the same feature space as the input features, but with a reduced frequency resolution of 72 bins instead of 216 bins.

The model was trained for 300 epochs with the Adam optimizer at an initial learning rate of $5 \cdot 10^{-4}$, early stopping patience of 20, and the binary crossentropy loss function.

3.3 Audio Representation

The MPE model expects input features as an harmonic Constant-Q transform (HCQT) [25] $H_{h,f,t}$ indexed by harmonic ratio h, frequency f, and time t. For any harmonic $h > 0$, we compute a standard Constant-Q transform (CQT), where the minimum frequency is scaled by the harmonic ratio h. For this MPE task, the HCQTs are computed for the harmonic ratios $h \in \{0.5, 1, 2, 3, 4, 5\}$, that is, one sub-harmonic below the fundamental frequency ($h = 0.5$), the fundamental frequency ($h = 1$) and four harmonics above ($h = [2, 3, 4, 5]$). As the fundamental frequency and the first harmonic often contain similar harmonic patterns [25], one sub-harmonic below the fundamental is included to distinguish between the two.

Features are extracted with the *librosa* Python library [26] with a hopsize of 512 samples, a sampling rate $f_{\mathrm{s}} = 22.05\,\mathrm{kHz}$, a frequency resolution of 36 bins per octave (bpo), and a minimum frequency for the CQT of $f_{\mathrm{min}} = 32.7\,\mathrm{Hz}$. This results in input features of dimension $\mathbf{X} \in \mathbb{R}^{M \times K \times C}$, with the number of time frames $M = 16$ for patch-wise processing, number of frequency bins $K = 216$, and amount of channels $C = 6$.

4. DATASET

4.1 Source Dataset

The U-net MPE model was trained and tested on subsets of the MIDI Aligned Piano Sounds (MAPS) dataset [1]. We used the dataset split labelled "Configuration 2" by Sigtia et al. [27], which divides MAPS into 210 synthesized audio files for training and 60 real-world audio recordings for test data. The audio and aligned MIDI data used for training were generated using synthesis software based on high-quality sample libraries, whereas the test set contains piano recordings from a MIDI-controlled Disklavier which was recorded on two omnidirectional Schoeps microphones in a studio setting. We refer to the training data as `MAPS-train` and test data as `MAPS-test`.

4.2 Target Dataset

To assess the selected DA methods, we require a dataset of multiple transcribed piano recordings which cover different mobile devices as recording devices, recording locations with different acoustical properties, and various acoustic piano models (upright and grand piano). Because no openly available dataset fulfilled all our requirements, we created a new dataset called `IDMT-PIANO-MM` [28].

`IDMT-PIANO-MM` contains a total of 432 audio files (about 4 h 7 min) and 72 MIDI files corresponding to 72 unique piano performances recorded simultaneously on five recording devices (3 smartphones, 2 tablets) and a high-quality stereo microphone. The 72 MIDI files were generated and aligned manually to match the actual piano performance. The recording environments range from small rooms to large lecture halls. Three grand pianos, four upright pianos, and one electronic stage piano of various age, brand and price segments were recorded. In each room we recorded a human piano performance of a self-composed swing pattern, chord progression, a chromatic scale, as well as sections (between 24 s and 54 s) out of five classical music pieces and one ragtime piece.

5. ANALYZING DOMAIN SHIFT

5.1 Investigation of Microphone Mismatch

To validate the microphone mismatch, we compute the spectrum correction coefficients as described in [29] to compare the acoustic characteristics of different recording devices. We choose the high-quality stereo microphone as reference device r. The frequency response of each mobile device d is compared per frequency bin with the reference to obtain a vector of multiplicative correction coefficients, which allows to make the frequency response of the device d equal to the one of the reference device r. Figure 2 illustrates the band-wise coefficients obtained for each mobile device. The correction coefficients show that there are considerable differences among the recordings associated to the different devices and their microphones. This confirms the microphone mismatch condition and suggests there is a domain shift between the training data from MAPS and the test data recorded with different mobile devices.

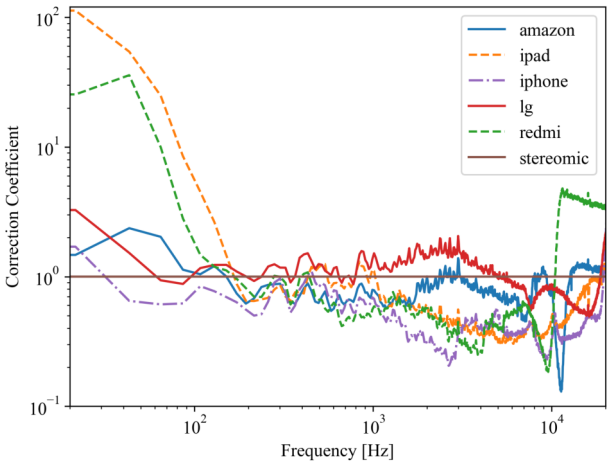


Figure 2. Frequency-dependence of the spectrum correction coefficients for all devices in respect to the `stereomic` microphone.

5.2 Quantifying Domain Shift

Due to the observed differences in the acoustic characteristics of the recording devices, we expect a measurable domain shift, which we quantify using the representation shift metric proposed by Stacke et al. [30]. The main idea is to compute an abstract representation of source and target domain features from a latent representation within a deep neural network and compare the data distributions between both domains in this latent feature space. In our experiments, we computed the activations after the last convolutional layer of the encoder in the U-net MPE model to measure domain shift, as we expect the highest degree of abstraction here.

In [30], the layer activations at layer l and filter f are averaged across the two spatial dimensions of a feature map. Since we deal with audio instead of images, these dimensions correspond to time frames and frequency bins. As the convolutional neural network (CNN) structure used in [30] is not directly comparable with the U-net architecture, we assumed that the U-net only has one filter operation per layer for simplicity. The mean value of a layer activation $\phi_l(x)$ at layer l is denoted as

$$c_l(x) = \frac{1}{N}\frac{1}{K}\sum_{i=1}^{N}\sum_{j=1}^{K}\phi_l(x)_{i,j} \qquad (1)$$

with the number of time frames N, number of frequency bands K, and i and j as time and frequency index of the feature map, respectively. The mean values of layer activations are aggregated over all input features of a domain. The probability density function (PDF) of all mean values is approximated by computing a probability density histogram with 500 bins for each HCQT channel. The resulting PDFs are averaged over all HCQT channels. Finally, we compute the representation shift r as the mean distance between the distributions of the source and target domain. We employ the Wasserstein distance, which is a symmetric distance measure that can be seen as the least amount of

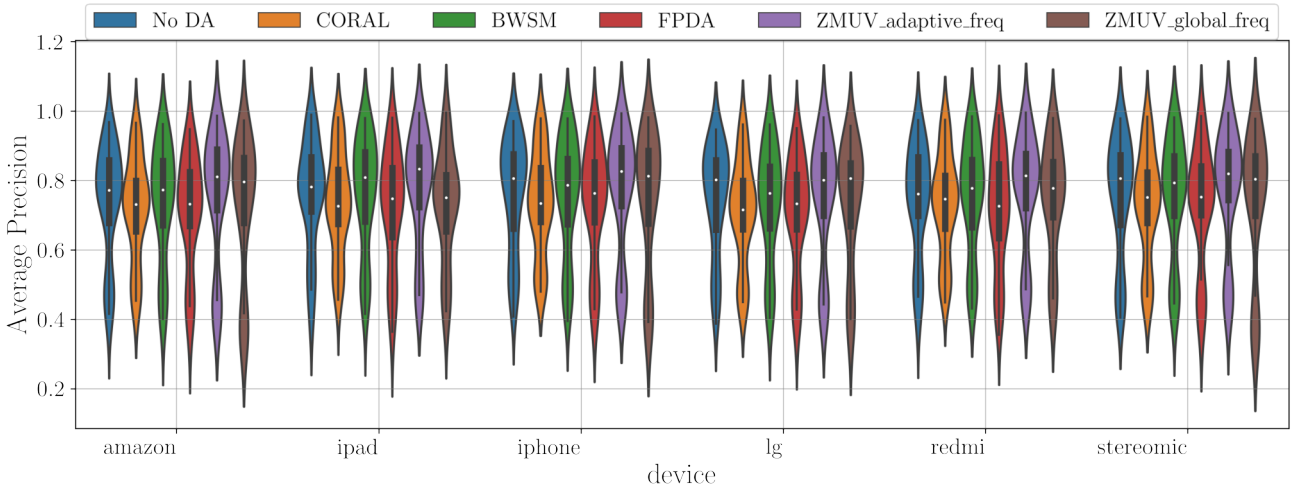


Figure 3. Performance of the MPE model on `IDMT-PIANO-MM`. The average precision score is presented for different DA methods grouped by recording devices.

Domain Pairs	r
`MAPS-train` vs. `IDMT-PIANO-MM`	0.035
`MAPS-train` vs. `IDMT-PIANO-MM-BWSM`	0.029
`MAPS-train` vs. `MAPS-test`	0.029

Table 2. Representation shift r based on the Wasserstein distance between source training data and source test data (`MAPS-train` vs. `MAPS-test`), source training data and target data (`MAPS-train` vs. `IDMT-PIANO-MM`), source training data and with BWSM domain-adapted target data (`MAPS-train` vs. `IDMT-PIANO-MM-BWSM`).

effort required to align two distributions w.r.t. the amount of data and the distance that has to be moved [30]. The representation shift between the used datasets is given in Table 2. The DA method BWSM was chosen as, in contrast to other DA methods, source domain data is not modified, which enables to directly compare the implementation with and without DA with identical MPE models.

6. IMPACT OF DOMAIN SHIFT ON PERFORMANCE

The performance of the MPE model is tested by comparing the estimated pitches with MIDI data as ground truth per frame. We chose mean Average Precision (mAP), i. e., the area under the precision-recall curve [31], as an evaluation metric, which is denoted as Average Precision (AP) in the following. Unlike other popular ML evaluation metrics like F-score or accuracy, AP does not depend on a particular binarization threshold for the pitch activity predictions.

6.1 Effect of Domain Adaptation

Figure 3 shows the AP scores of the U-net MPE model on the benchmark dataset separated by recording device. It shows that the microphone mismatch has no significant effect on the performance of the MPE model. Presumably as a direct consequence, we observe only little variation in transcription performance caused by DA. `ZMUV adaptive` performs consistently best, `ZMUV global` and `BWSM` report a similar performance as using no DA, and with `FPDA` and `CORAL` the AP scores are lowest amongst all DA methods, including not applying any DA.

6.2 Impact of Musical Content and Acoustic Context

We also assessed the impact of other parameters on the MPE performance, such as room acoustics and musical pieces. Our results showed no significant influence of the room and instrument characteristics towards the MPE performance. In contrast, the musical content shows the greatest impact on transcription performance. Figure 4 displays the influence of music content on the AP score when no DA or different DA methods are applied. Due to missing annotations of offsets, we assume that the MPE performance is lower for `beethoven1` and `beethoven2` as these pieces involve the use of the sustain pedal. While some of the DA methods vary in effectiveness for different music content, like FPDA and CORAL, overall the results of DA seem to be quite consistent and do not improve performance significantly regardless of the musical content.

7. DISCUSSION

Although the examined DA methods reduced the domain shift between `MAPS-train` and our test dataset `IDMT-PIANO-MM` (see Table 2), the MPE performance could not be improved significantly by DA. We see several possible explanations for this.

First, it should be noted that the applied method to quantify domain shift assumes that each convolutional layer of the MPE model contains only one filter, which is not given with the U-net MPE model. Further research must investigate how this discrepancy leads to a biased measurement. Yet, we expect that the computed domain shift values can

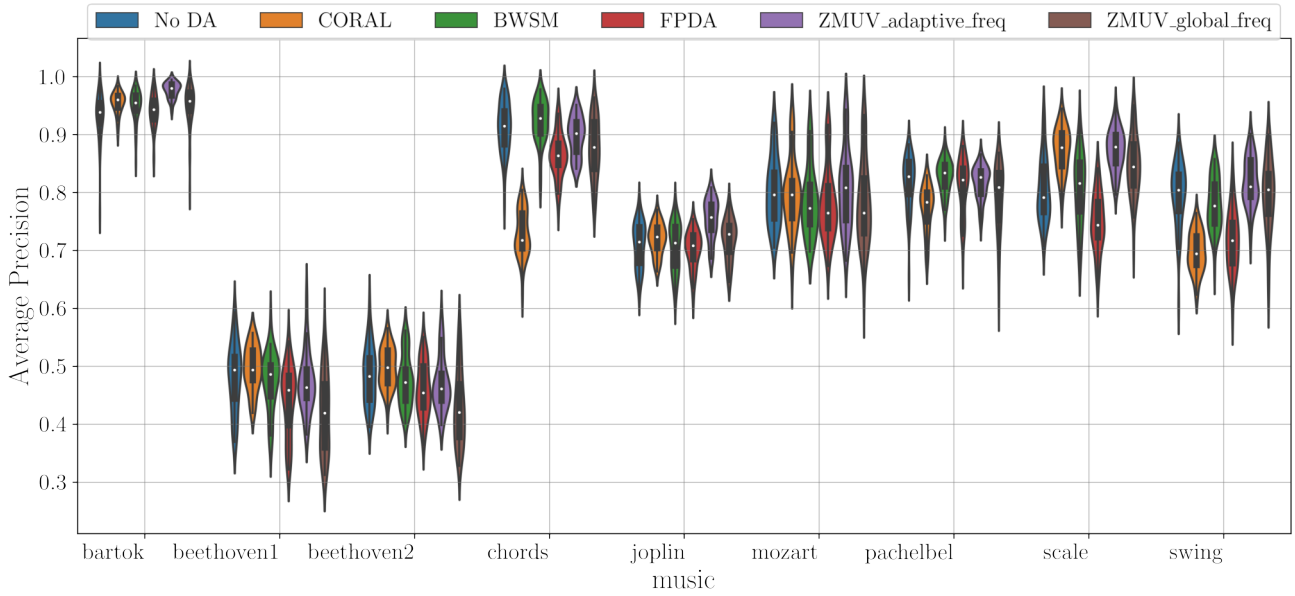


Figure 4. Performance of the MPE model on `IDMT-PIANO-MM`. The average precision score is presented for different DA methods grouped by music pieces.

still provide a general tendency.

Second, the employed MPE model might already generalize well enough to the given task. In fact, the U-net model itself can be considered as a reconstruction-based deep DA implementation as defined in [5]. The encoding procedure ensures that domain-specific features are disregarded, while the decoding part enables to attribute a feature within a domain. Therefore, it would be interesting to examine the presented DA methods for MPE models with other network architectures.

Third, the benchmark dataset `IDMT-PIANO-MM` does not provide precise offset annotations, which can introduce label noise to the frame-wise MPE evaluation.

Finally, there are further restrictions for particular DA methods. Our implementation of FPDA involves cutting the features after 16 time frames (about 0.372 s), whereas the original approach [15] keeps 10 s of data for the transformation matrix. As we reduced the size of the transformation matrix extremely, our implementation could be inaccurate. Moreover, unlike in the original paper [15] it was not possible to use recordings with identical content (denoted as "parallel data" in [13]). This requires audio data to be simultaneously captured using source and target domain devices, which is not applicable in this study. Besides, CORAL is implemented as time-independent DA since frequency patterns are only compared within one time frame. The domain-specific knowledge of time is disregarded due to buffer memory limits. As there is no reference implementation for CORAL in the audio domain to our knowledge, we cannot compare this approach to other findings. The effectiveness of adaptive ZMUV normalization is restricted by the sole dependency on recorded target data. Unlike in global ZMUV normalization, no source domain data is used to modify target data. Hence, if the amount of cached target data is too small, the DA may fail.

8. CONCLUSION

In this paper, we studied the influence of domain shift to piano MPE under microphone mismatch conditions. As our first contribution, we created and published a novel benchmark dataset of piano recordings captured with various mobile devices. In our initial experiments, we verified differences in the used recording devices using the spectrum correction coefficient method. As a second step, we quantified the domain shift between the source domain (`MAPS-train`) and different target domains of a novel benchmark dataset (`IDMT-PIANO-MM`) using the representation shift based on the Wasserstein distance between distributions of intermediate activation map values of the MPE model. We investigated in particular the performance of an MPE model based on the U-net architecture. As expected, the domain shift was greater between source and target data than between two subsets of the MAPS dataset (0.035 vs. 0.029 respectively), possibly due to different types of recording settings.

As a third step, we investigated the influence of domain shift on the MPE performance and found that the U-net model is surprisingly robust to domain shift conditions. We assume the main reason for this is that spectral peaks contain the main information for the MPE. Domain shift, however, mainly causes deviations in the overall spectral envelope of the signal, which are mostly irrelevant for the given task. Finally, we evaluated four different unsupervised DA methods in order to reduce the domain shift. The DA method BWSM was able to reduce the measured domain shift from the initial 0.035 to 0.029.

In future research, we aim to evaluate additional MPE models based on other neural network architectures, such as CRNNs or transformer models, to assess the influence of the network architecture on the robustness against domain shift.

9. ACKNOWLEDGEMENTS

This work has been supported by the German Research Foundation (AB 675/2-2). We would like to thank the reviewers for their valuable suggestions and comments.

10. REFERENCES

[1] V. Emiya, N. Bertin, B. David, and R. Badeau, "MAPS - A piano database for multipitch estimation and automatic transcription of music," Telecom ParisTech, Paris, France, Research Report, 2010.

[2] M. Müller, V. Konz, W. Bogler, and V. Arifi-Müller, "Saarland music data (SMD)," in *Late-Breaking and Demo Session of the 12th International Conference on Music Information Retrieval (ISMIR)*, Miami, United States, 2011.

[3] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset," in *Proceedings of the International Conference on Learning Representations (ICLR)*, New Orleans, United States, 2019.

[4] D. S. Johnson and S. Grollmisch, "Techniques Improving the Robustness of Deep Learning Models for Industrial Sound Analysis," in *Proceedings of the 28th European Signal Processing Conference*. Amsterdam, The Netherlands: IEEE, 2021, pp. 81–85.

[5] M. Wang and W. Deng, "Deep Visual Domain Adaptation: A Survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[6] A. Mesaros, T. Heittola, and T. Virtanen, "Acoustic Scene Classification in DCASE 2019 Challenge: Closed and Open Set Classification and Data Mismatch Setups," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*. New York City, United States: New York University, 2019, pp. 164–168.

[7] Y. Kawaguchi, K. Imoto, Y. Koizumi, N. Harada, D. Niizumi, K. Dohi, R. Tanabe, H. Purohit, and T. Endo, "Description and Discussion on DCASE 2021 Challenge Task 2: Unsupervised Anomalous Sound Detection for Machine Condition Monitoring under Domain Shifted Conditions," *arXiv e-prints: 2106.04492*, 2021.

[8] L. Yang, J. Hao, Z. Hou, and W. Peng, "Two-stage Domain Adaptation for Sound Event Detection," in *Detection and Classification of Acoustic Scenes and Events (DCASE)*, Tokyo, Japan, 2020, pp. 230–234.

[9] S. Chowdhury and G. Widmer, "Towards Explaining Expressive Qualities in Piano Recordings: Transfer of Explanatory Features Via Acoustic Domain Adaptation," in *Proceedings of the ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 561–565.

[10] Y.-J. Luo and L. Su, "Learning Domain-Adaptive Latent Representations of Music Signals Using Variational Autoencoders," in *19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 653–660.

[11] C. Nyströmer, "Musical Instrument Activity Detection using Self-Supervised Learning and Domain Adaptation," Master's thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2020.

[12] T. Asami, R. Masumura, Y. Yamaguchi, H. Masataki, and Y. Aono, "Domain adaptation of DNN acoustic models using knowledge distillation," in *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 5185–5189.

[13] A. I. Mezza, E. A. P. Habets, M. Müller, and A. Sarti, "Unsupervised Domain Adaptation for Acoustic Scene Classification Using Band-Wise Statistics Matching," in *Procedings of the 2020 28th European Signal Processing Conference (EUSIPCO)*, Amsterdam, The Netherlands, 2021, pp. 11–15.

[14] B. Sun, J. Feng, and K. Saenko, "Return of Frustratingly Easy Domain Adaptation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, pp. 2058–2065, 2016.

[15] A. I. Mezza, E. A. P. Habets, M. Müller, and A. Sarti, "Feature Projection-Based Unsupervised Domain Adaptation for Acoustic Scene Classification," in *Proceedings of the 2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, Espoo, Finland, 2020, pp. 1–6.

[16] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *Proceedings of the 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) (IEEE Cat. No.03TH8684)*, New Paltz, NY, USA, 2003, pp. 177–180.

[17] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic Music Transcription: An Overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[18] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and Frames: Dual-Objective Piano Transcription," in *19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 50–57.

[19] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-Resolution Piano Transcription With Pedals by Regressing Onset and Offset Times," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3707–3717, 2021.

[20] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, "Sequence-to-Sequence Piano Transcription with Transformers," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, Online, 2021, pp. 246–253.

[21] J. Abeßer and M. Müller, "Jazz Bass Transcription Using a U-Net Architecture," *Electronics*, vol. 10, no. 6, 670, 2021.

[22] T.-H. Hsieh, L. Su, and Y.-H. Yang, "A Streamlined Encoder/decoder Architecture for Melody Extraction," in *Proceedings of the ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 2019, pp. 156–160.

[23] G. Doras, P. Esling, and G. Peeters, "On the use of U-Net for dominant melody estimation in polyphonic music," in *Proceedings of the International Workshop on Multilayer Music Representation and Processing (MMRP) 2019*, Milano, Italy, 2019, pp. 66–70.

[24] Y.-T. Wu, B. Chen, and L. Su, "Polyphonic Music Transcription with Semantic Segmentation," in *Proceedings of the ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brighton, UK, 2019, pp. 166–170.

[25] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep Salience Representations for F0 Estimation in Polyphonic Music," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, Suzhou, China, 2017, pp. 63–70.

[26] B. McFee, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, D. Ellis, J. Mason, E. Battenberg, S. Seyfarth, R. Yamamoto, viktorandreevichmorozov, K. Choi, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, D. Hereñú, F.-R. Stöter, P. Friesch, A. Weiss, M. Vollrath, T. Kim, and Thassilo, "Librosa/librosa: 0.8.1rc2," DOI: 10.5281/zenodo.4792298, 2021.

[27] S. Sigtia, E. Benetos, and S. Dixon, "An End-to-End Neural Network for Polyphonic Piano Music Transcription," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 5, pp. 927–939, 2016.

[28] J. Abeßer, F. Bittner, M. Richter, M. Gonzalez, and H. Lukashevich, "A Benchmark Dataset to Study Microphone Mismatch Conditions for Piano Multipitch Estimation on Mobile Devices," in *Proceedings of the Digital Music Research Network One-day Workshop*. London, United Kindom: Centre for Digital Music (C4DM), 2021, p. 22.

[29] M. Kośmider, "Spectrum Correction: Acoustic Scene Classification with Mismatched Recording Devices," in *Proceedings of INTERSPEECH 2020*, Shanghai, China, 2020, pp. 4641–4645.

[30] K. Stacke, G. Eilertsen, J. Unger, and C. Lundström, "Measuring Domain Shift for Deep Learning in Histopathology," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 2, pp. 325–336, 2021.

[31] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

MELODY TRANSCRIPTION VIA GENERATIVE PRE-TRAINING

Chris Donahue
Stanford University

John Thickstun
Stanford University

Percy Liang
Stanford University

ABSTRACT

Despite the central role that melody plays in music perception, it remains an open challenge in MIR to reliably detect the notes of the melody present in an arbitrary music recording. A key challenge in *melody transcription* is building methods which can handle broad audio containing any number of instrument ensembles and musical styles—existing strategies work well for *some* melody instruments or styles but not all. To confront this challenge, we leverage representations from Jukebox [1], a generative model of broad music audio, thereby improving performance on melody transcription by 20% relative to conventional spectrogram features. Another obstacle in melody transcription is a lack of training data—we derive a new dataset containing 50 hours of melody transcriptions from crowdsourced annotations of broad music. The combination of generative pre-training and a new dataset for this task results in 77% stronger performance on melody transcription relative to the strongest available baseline. [1] By pairing our new melody transcription approach with solutions for beat detection, key estimation, and chord recognition, we build Sheet Sage, a system capable of transcribing human-readable lead sheets directly from music audio.

1. INTRODUCTION

In the Western music canon, *melody* is a defining characteristic of musical composition, and can even constitute the very identity of a piece of music within the collective consciousness. Because of the significance of melody to our music perception, the ability to automatically transcribe the melody notes present in an arbitrary recording could enable numerous applications in interaction [2], education [3], informatics [4], retrieval [5], source separation [6], and generation [7]. Despite the potential benefits, reliable melody transcription remains an open challenge.

A closely-related problem that has received considerable attention from the MIR community is *melody extraction* [8–11], where the goal is to estimate the time-varying, continuous F0 trajectory of the melody in an audio mixture. In contrast, the goal of melody transcription is to output the *notes* of the melody, where a note is defined by an onset time, a pitch, and an offset time. While F0 trajectories are useful for several downstream tasks (e.g., query by humming) and more inclusive of music which does not use equal-tempered pitches, unlike notes, trajectories cannot be readily converted into formats like MIDI or scores which are more convenient for musicians.

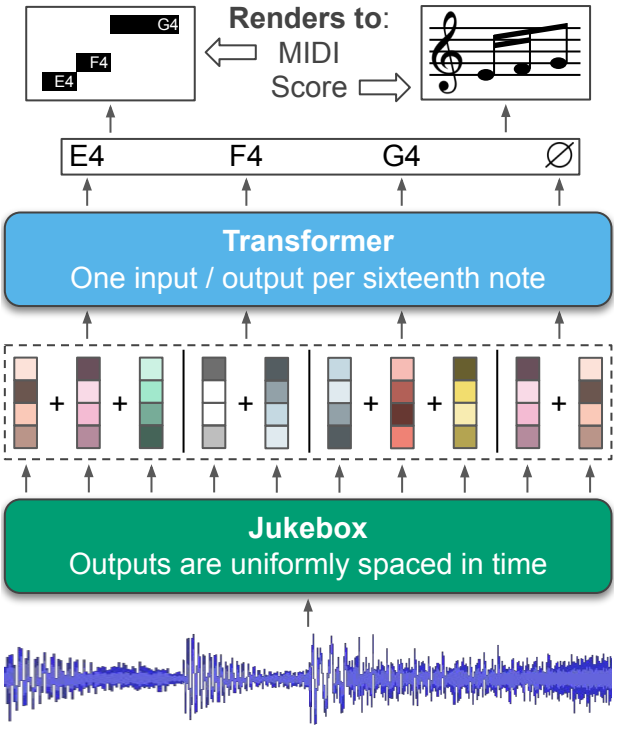


Figure 1. Our melody transcription approach involves (1) extracting audio representations from Jukebox [1], a generative model of music, (2) averaging these representations across time to their nearest sixteenth note (dashed outline—uses `madmom` [12, 13] for beat detection), and (3) training a Transformer [14] to detect note onsets (or absence thereof) per sixteenth note. Outputs can be rendered to MIDI (by mapping beats back to time) or a score.

The relative lack of progress on melody transcription is perhaps counterintuitive when compared to the considerable progress on seemingly more difficult tasks like piano transcription [15, 16]. This circumstance stems from two primary factors. First, unlike in piano transcription, melody transcription involves operating on *broad* audio mixtures from arbitrary instrument ensembles and musical styles. Second, there is a deficit of training data for melody transcription, which particularly impedes the deep learning approaches central to recent improvements on other transcription tasks. Moreover, collecting data for melody tran-

[1] Examples: `https://chrisdonahue.com/sheetsage`
Code: `https://github.com/chrisdonahue/sheetsage`

© C. Donahue, J. Thickstun, and P. Liang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: C. Donahue, J. Thickstun, and P. Liang, "Melody transcription via generative pre-training", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.*, Bengaluru, India, 2022.

scription is difficult compared to collecting data for tasks like piano transcription, where a Disklavier can be used to create aligned training data in real time.

To overcome the challenge of transcribing broad audio, in this work we leverage representations from Jukebox [1], a large-scale generative model of music audio pre-trained on 1M songs. In [17], Castellon et al. demonstrate that representations from Jukebox are useful for improving performance on a wide variety of MIR tasks. Here we show that, when used as input features to a Transformer model [14], representations from Jukebox yield 27% stronger performance on melody transcription (as measured by note-wise F1) relative to handcrafted spectrogram features conventionally used for transcription. To our knowledge, this is the first evidence that representations learned via generative modeling are useful for time-varying MIR tasks like transcription, as opposed to the song-level tasks (e.g. tagging, genre detection) examined in [17].

To address the data deficit for melody transcription, we release a new dataset containing 50 hours of melody annotations for broad audio which we derive from HookTheory. [2] The user-specified alignments between the audio and melody annotations in HookTheory are crude—we refine these alignments using beat detection. To overcome remaining alignment jitter, we resample features to be uniformly spaced in beats (rather than time) and pass these *beat-wise resampled* features as input to melody transcription models. This procedure has a secondary benefit of enabling simple conversion from raw model outputs to human-readable scores (Figure 1).

By training Transformer models on this new dataset using representations from Jukebox as input, we are able to improve overall performance on melody transcription by 70% relative to the strongest available baseline. A summary of our primary **contributions** follows:

- We show that representations from generative models can improve melody transcription (Section 6).
- We collect, align, and release a new dataset with 50 hours of melody and chord annotations (Section 4).
- We propose a method for training transcription models on data with imprecise alignment (Section 5.3).
- As a bonus application of our melody transcription approach, we build a system which can transcribe music audio into lead sheets (Section 7).

2. RELATED WORK

Melody transcription is closely related to but distinct from the task of melody extraction, originally referred to as predominant fundamental frequency (F0) estimation [8, 9]. Melody extraction has received significant interest from the MIR community over the last two decades (see [10, 11] for comprehensive reviews), and is the subject of an annual MIREX competition [18]. Melody extraction may be a component of a melody transcription pipeline in combination with a strategy to segment F0 into notes [19–21]—we directly compare to such a pipeline in Section 6.2.

Compared to melody extraction, melody transcription has received considerably less attention. Earlier efforts use sophisticated DSP-based pipelines [22–25]—unfortunately none of these methods provide code, though [24] provides example transcriptions which we use to facilitate direct comparison. A more recent effort uses ground truth chord labels as extra information to improve melody transcription [26]—in contrast, our method does not require extra information. Another line of work seeks to transcribe solo vocal performances into notes [27–30]. As singing voice often carries the melody in popular music, we directly compare to a baseline which firsts isolates the vocals (using Spleeter [31]) and then transcribes them.

Polyphonic music transcription is another related task which involves transcribing *all* of the notes present in a recording (not just the melody). This task has its own MIREX contest (Multiple Fundamental F0 Estimation) alongside a growing collection of supervised training data resources [7, 32–34]. The similarity of the polyphonic and melody transcription problems motivates us to experiment with representations learned by a polyphonic system—specifically, MT3 [35]—for melody transcription.

3. TASK DEFINITION

In this work, melody transcription refers to the task of converting a music recording into a *monophonic* (non-overlapping) sequence of *notes* which constitute its dominant melody. [3] More precisely, given a music waveform $\boldsymbol{a}$ of length T seconds, our task is to uncover the sequence of N notes $\boldsymbol{y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N]$ that represent the melody of $\boldsymbol{a}$. For many MIR tasks, including transcription, it can be convenient to work with *features* of audio $\boldsymbol{X} = \texttt{Featurize}(\boldsymbol{a})$, rather than waveforms. Hence, a melody transcription algorithm is a procedure that maps featurized audio to notes, i.e. $\boldsymbol{y} = \texttt{Transcribe}(\boldsymbol{X})$.

Canonically, a musical note consists of an onset time, a musical pitch, and an offset time. However, in this work we disregard offsets and define a note to be a pair $\boldsymbol{y}_i = (t_i, n_i)$ consisting of an onset time $t_i \in [0, T)$ and discrete musical pitch $n_i \in \mathbb{V} = \{\text{A0}, \ldots, \text{C8}\}$. We ignore offsets for two reasons. First, accurate offsets have been found to be considerably less important for human perception of transcription quality compared to accurate onsets [36]. Second, in our dataset, a heuristically-determined offset is identical to the user-annotated offset for 89% of notes. [4]

Formally, a musical audio recording of length T seconds sampled at rate f_s is a vector $\boldsymbol{a} \in \mathbb{R}^{Tf_s}$. A featurization of audio $\boldsymbol{X} \in \mathbb{R}^{Tf_k \times d}$ is a matrix of d-dimensional features of audio, sampled uniformly at some rate $f_k \ll f_s$ (for example, $\boldsymbol{X}$ could be a spectrogram). Intuitively, the function $\texttt{Featurize} : \mathbb{R}^{Tf_s} \to \mathbb{R}^{Tf_k \times d}$ defined by

[2] https://www.hooktheory.com/theorytab

[3] Melody is difficult to precisely define—here we adopt an implicit definition based on a dataset of crowdsourced melody annotations.

[4] The specific heuristic that we use sets the offset of one note equal to the onset of the next, i.e., it assumes the melody is legato.

$\boldsymbol{a} \mapsto \boldsymbol{X}$ maps raw audio to a feature representation more conducive to learning. A melody of length N is a sequence of notes $\boldsymbol{y} = [\boldsymbol{y}_1, \ldots, \boldsymbol{y}_N] \in \mathbb{Y}^N$ consisting of onset-pitch pairs $\boldsymbol{y}_i = (t_i, n_i) \in \mathbb{Y} = \mathbb{R}^+ \times \mathbb{V}$ where $t_i < t_j$ if $i < j$. Given a featurization $\boldsymbol{X}$, the melody transcription task is to construct a transcription algorithm $\texttt{Transcribe} : \mathbb{R}^{Tf_k \times d} \to \mathbb{Y}^N$ such that $\boldsymbol{X} \mapsto \boldsymbol{y}$.

3.1 Evaluation

To evaluate a melody transcription method `Transcribe`, we adopt a standard metric commonly used for evaluation in polyphonic music transcription tasks, namely, "onset-only note-wise F-measure" [36]. This metric scores an estimated transcript $\texttt{Transcribe}(\boldsymbol{X})$ by first matching its note onsets to those in the reference $\boldsymbol{y}$ with 50ms of tolerance (default in [37]), and then computes a standard `F1` score where an estimated note is treated as correct if it is the same pitch as its matched reference note. This "note-wise" metric represents a departure from the "frame-based" metrics typically used to evaluate melody extraction algorithms—Ycart et al. demonstrate in [36] that this particular note-wise metric correlates more strongly with human perception of transcription quality than any other common metric, including frame-based ones.

We make a slight modification to this note-wise metric specific to the melody transcription setting: an estimate $\texttt{Transcribe}(\boldsymbol{X})$ may receive full credit if it is off by a fixed octave shift but otherwise identical to the reference. In downstream settings, melody transcriptions are likely to be used in an octave-invariant fashion, e.g., they may be shifted to read more comfortably in treble clef, or performed by singers with different vocal ranges. Hence, we modify the evaluation criteria by simply taking the highest score over octave shifted versions of the estimate:

$$\max_{\sigma \in \mathbb{Z}} \texttt{F1}(\texttt{OctaveShift}(\texttt{Transcribe}(\boldsymbol{X}), \sigma), \mathbf{y}).$$

Henceforth, we refer to this octave-invariant metric as `F1`.

4. DATASET OVERVIEW

A major obstacle to progress on melody transcription is the lack of a large volume of data for training. To the best of our knowledge, there are only two datasets available with annotations suitable for melody transcription: the RWC Music Database [38–40] (RWC-MDB), and a dataset labeled by Laaksonen [26]. The former is larger but the annotations are inconsistent—Ryynänen and Klapuri note that only 8.7 hours (130 songs) are usable for melody transcription [24], while the latter only contains 1.5 hours.

We derive a suitably large dataset for melody transcription using crowdsourced annotations from HookTheory.[5] HookTheory is a platform where users can easily create and share musical analyses of particular recordings hosted on YouTube, with Wikipedia-style editing. The dataset contains annotations for 22k segments of 13k unique recordings totaling 50 hours of labeled audio. The audio content covers a wide range of genres—there is a skew towards pop and rock but many other genres are represented including EDM, jazz, and even classical. We create an artist-stratified 8:1:1 split of the dataset for training, validation, and testing. The dataset also includes chord annotations which may facilitate chord recognition research.

While HookTheory data has been used previously for MIR tasks like harmonization [41, 42], chord recognition [43], and representation learning [44], making use of this platform for MIR is currently cumbersome. One obstacle is that the annotations are created via a "functional" interface, i.e., one which uses scale degrees and roman numerals relative to a key signature instead of absolute notes and chord names. In contrast, most MIR research favors absolute labels. Hence, we convert annotations from this functional format to a simple (JSON-based) absolute format. One caveat is that the HookTheory annotation interface uses a relative octave system, so there is no way to reliably map annotations to a ground truth octave. Thus, melodies in our dataset also contain only relative octave information, consistent with the octave-invariant evaluation proposed in Section 3.1.

5. METHODS

Similar to state-of-the-art methodology used for polyphonic transcription [45], our approach to melody transcription involves training Transformer models [14] to predict notes from audio features. However, to address the unique challenges of melody transcription, our approach differs in two distinct ways. First, because melody transcription involves operating on broad audio, we leverage representations from pre-trained models as drop-in replacements for the handcrafted spectrogram features used as inputs to other transcription systems. Secondly, because alignments in our dataset are approximate, we propose a new strategy for training transcription models under such conditions.

5.1 Pre-trained representations

We explore representations from two different pre-trained models for use as input features to transcription models. In [17], Castellon et al. demonstrate that representations from Jukebox [1]—a generative model of music audio pre-trained on 1M songs—constitute effective features for many MIR tasks, though notably they do not experiment on transcription. We adopt their approach to extract features from Jukebox ($f_k \approx 345$ Hz, $d = 4800$), though we use a deeper layer (53) than their default (36) which improved transcription performance in our initial experiments.

We also explore features from MT3 [35], an encoder-decoder transcription model pre-trained on a multitude of different transcription tasks (though not melody transcription). For this model, we use the encoder's outputs as features ($f_k = 125$ Hz, $d = 512$). The two models have different trade-offs with respect to our setting: Jukebox was pre-trained on audio similar to that found in our dataset but in a generative fashion, whereas MT3 is pre-trained on transcription but for different audio domains.

[5] HookTheory annotations are published under a CC BY-NC-SA 3.0 license, which our dataset inherits.

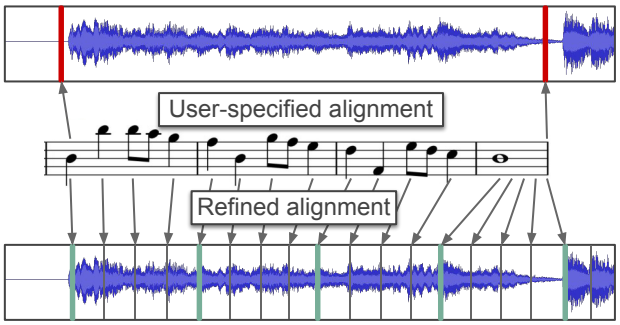


Figure 2. We refine the crude user-specified alignments from HookTheory by using beat and downbeat tracking. The first segment beat is mapped to the detected downbeat nearest to the user-specified starting timestamp, and remaining beats are mapped to subsequent detected beats.

5.2 Refined Alignments

The alignments between audio and HookTheory annotations are crude—users provide only an approximate starting and ending timestamp of their annotated segment within the audio. Because transcription methodology generally depends on precise alignments, we make an effort to refine the user-specified ones. To this end, we make use of the beat and downbeat detection algorithm from `madmom` [12, 13]. Specifically, our approach aligns the first beat of the segment to the detected downbeat which is nearest to the user-specified starting timestamp. Then, we align the remaining beats to the subsequent detected beats (see Figure 2 for an example). This provides a beat-level alignment for the entire segment, which we linearly interpolate to fractional subdivisions of the beat. Formally, we construct an alignment function $\texttt{Align} : [0, B) \to [0, T)$ that assigns each of B beats in the metrical structure to a time $t \in [0, T)$ in the audio. In an informal listening test, this produced an improved alignment for 95 of 100 segments, where the primary failure mode in the remaining 5 segments occurred when `madmom` detected the wrong beat as the downbeat. We use these refined alignments for training and evaluation and release them alongside the dataset.

5.3 Beat-wise resampling

Here we outline our approach for training transcription models in the presence of imprecise alignments. Existing transcription methods were largely designed for domains where perfect alignments are readily available, e.g., piano transcription data captured by a Disklavier. Despite our best efforts, the refined HookTheory alignments are still imprecise when compared to alignments in the datasets used to develop existing methods. Consequently, in initial experiments, we found that naively adopting existing methods (specifically, [16, 45]) resulted in poor performance on our dataset and task. Additionally, initial experiments on training models with an alignment-free approach [46] also resulted in poor performance.

Accordingly, to sidestep small alignment deviations, we perform a beat-wise resampling of audio features $\boldsymbol{X} \in \mathbb{R}^{Tf_k \times d}$ to yield features that are uniformly spaced in subdivisions of the beat (using `Align`—see Section 5.2) rather than in time. For an audio recording with B beats, we sample features $\tilde{\boldsymbol{X}} \in \mathbb{R}^{4B \times d}$ at sixteenth-note intervals. The value $\tilde{\boldsymbol{X}}_i$ is constructed by averaging all feature vectors in $\boldsymbol{X}$ that are nearest to the i'th sixteenth note into a single vector which acts as a proxy feature. For example, if a recording has a tempo of 120 BPM, a sixteenth note represents 125 ms of time, which would entail averaging across 43 feature vectors from Jukebox ($f_k \approx 345$ Hz). The intuition is that, while our alignments may not be precise enough to identify which of those 43 frames contains an onset, we can be reasonably confident that it occurs *somewhere* within them, and thus the relevant frame will be incorporated into the proxy. A similar approach was previously explored for song structure analysis in [47].

5.4 Modeling

Together with the beat-wise resampling $\tilde{\boldsymbol{X}} \in \mathbb{R}^{4B \times d}$, we convert the sparse task labels $\boldsymbol{y} \in (\mathbb{R}^+ \times \mathbb{V})^N$ into a dense sequence $\tilde{\boldsymbol{y}} \in \{\{\varnothing\} \cup \mathbb{V}\}^{4B}$, which indicates whether or not an onset occurs at each sixteenth note.[6] Formally,

$$\tilde{\boldsymbol{y}}_i = \begin{cases} n_j & \text{if } \texttt{Align}(\frac{i}{4}) = t_j \text{ for some note } \boldsymbol{y}_j, \\ \varnothing & \text{otherwise.} \end{cases}$$

We formulate melody transcription as an aligned sequence-to-sequence modeling problem and attempt to predict the sequence $\tilde{\boldsymbol{y}}$ given $\tilde{\boldsymbol{X}}$. Specifically, we train models of the form $f_\theta : \mathbb{R}^{4B \times d} \to \mathbb{R}^{4B \times (|\mathbb{V}|+1)}$, which parameterize probability distributions $p_\theta(\tilde{\boldsymbol{y}}_i | \tilde{\boldsymbol{X}}) = \texttt{SoftMax}(f_\theta(\tilde{\boldsymbol{X}})_i)$ over elements of the sequence $\tilde{\boldsymbol{y}}$. One unique aspect of our dataset is that absolute octave information is absent (see Section 4). Hence, we construct an octave-tolerant cross-entropy loss by identifying the octave shift amount that minimizes the standard cross-entropy loss (denoted `CE`) when applied to the labels:

$$\min_{\sigma \in \mathbb{Z}} \sum_{i=0}^{4B-1} \texttt{CE}(p_\theta(\tilde{\boldsymbol{y}}_i | \tilde{\boldsymbol{X}}), \texttt{OctaveShift}(\tilde{\boldsymbol{y}}_i, \sigma)).$$

We require a thresholding scheme to convert the dense sequence of soft probability estimates $p_\theta(\tilde{\boldsymbol{y}}_i | \tilde{\boldsymbol{X}})$ into a sparse sequence of notes required by our task (see Section 3). Given a threshold $\tau \in \mathbb{R}$ (in practice, tuned on validation data), we define a sorted *onset list*

$$\mathcal{I} = \texttt{Sort}(\{i \in \{0, \ldots, 4B-1\} : p_\theta(\tilde{\boldsymbol{y}}_i = \varnothing | \tilde{\boldsymbol{X}}) < \tau\}).$$

This should be interpreted as a list of N metrical positions where an onset likely occurs. The timings of these onsets are given by the alignment, and we will predict the note-value with the highest probability. The sparse melody transcription is thus defined for $j = 1, \ldots, N$ by

$$\texttt{Transcribe}(\tilde{\boldsymbol{X}})_j = (t_j, n_j), \text{ where}$$
$$t_j = \texttt{Align}\left(\frac{\mathcal{I}_j}{4}\right),$$
$$n_j = \arg\max_{v \in \mathbb{V}} p_\theta(\tilde{\boldsymbol{y}}_{\mathcal{I}_j} = v | \tilde{\boldsymbol{X}}).$$

[6] This requires quantizing labels to the nearest sixteenth note. In practice, less than 1% of notes in our dataset are affected by this quantization.

Features	d	F1
Mel	229	0.514
MT3	512	0.550
Jukebox	4800	**0.615**
Mel, MT3	741	0.548
Mel, Jukebox	5029	0.617
MT3, Jukebox	5312	0.622
Mel, MT3, Jukebox	5541	**0.623**

Table 1. HookTheory test set performance for Transformers trained with different features (top) and combinations (bottom). Features are complementary—combining all three yields highest performance—but marginally so compared to Jukebox alone.

Approach	F1 (All)	F1 (Vocal)
MT3 Zero-shot [35]	0.133	0.085
Melodia [48] + Segmentation	0.201	0.268
Spleeter [31] + Tony [28]	0.341	**0.462**
DSP + HMM [24]	**0.420**	0.381
Mel + Transformer	0.631	0.621
MT3 + Transformer	0.701	0.659
Jukebox + Transformer	**0.744**	**0.786**

Table 2. Performance of different approaches on a subset of RWC-MDB [38–40]. The bottom three approaches were trained on the HookTheory dataset. For fair comparison to vocal transcription baselines, we also separately report performance on the vocal portions of this dataset.

6. EXPERIMENTS

Here we describe our experimental protocol for training melody transcription models on the HookTheory dataset. The purpose of these experiments is two-fold. First, we compare representations from different pre-trained models to handcrafted spectrogram features to determine if pre-training is helpful for the task of melody transcription (Section 6.1). Second, we compare our trained models holistically to other melody transcription baselines (Section 6.2).

All transcription models are encoder-only Transformers with the default hyperparameters from [14], except that we reduce the number of layers from 6 to 4 to allow models to be trained on GPUs with 12GB of memory. During training, we select random slices from the annotated segments of up to 96 beats or 24 seconds in length (whichever is shorter). We train using our proposed loss function from Section 5.4 and perform early stopping based on max F1 score across thresholds τ on the validation set, using the best validation τ for testing. All models converge within 15k steps or about a day on a single K40 GPU.

6.1 Comparing input features

We compare representations from Jukebox [1] and MT3 [35] (see Section 5.1) to handcrafted spectrogram features, which are commonly used by existing transcription methods. Specifically, we compare to log-amplitude Mel spectrograms using the formulation from [16] ($f_k \approx 31$, $d = 229$). Because features may contain complementary information, we also experiment with all combinations of these three features. Note that our beat-wise resampling strategy allows for trivial combination of these features (by concatenation) despite their differing rates. In Table 1, we report F1 (as described in Section 3.1) on the HookTheory test set for all input features.

Overall, using representations from Jukebox as input features results in stronger melody transcription performance than using either representations from MT3 or conventional handcrafted features. Representations from both MT3 and Jukebox outperform conventional handcrafted features, implying that both pre-training strategies are helpful for melody transcription. Note that these two pre-training approaches are compared holistically—these models differ on several axes (number of parameters, pre-training data semantics, pre-training task), and thus it is impossible to disentangle the individual contributions of these different factors without retraining the models.

Qualitatively speaking, there is a noticeable difference in performance across the three different input features which correlates with quantitative performance (see footnote 1 for sound examples). Using representations from Jukebox tends to result in fewer wrong notes than the other features, and substantially reduces the number of egregiously wrong notes (e.g., notes outside of the key signature). Representations from Jukebox also appear to aid in the detection of more nuanced rhythmic patterns. Moreover, using handcrafted features will often result in several repeated onsets during a longer sustained melody note—in contrast, using representations from Jukebox appears to mitigate this failure mode.

Different features also appear to complement one another to a degree. The strongest performance overall is obtained by combining all three features, though the improvement over Jukebox alone is marginal. The practical downsides of combining all features outweigh the marginal benefits—running both pre-trained models effectively doubles the overall runtime, and the models have incompatible software dependencies. Hence, in the remainder of this paper we focus on models trained on individual features.

6.2 Comparison to melody transcription baselines

We compare overall performance of our proposed melody transcription approach to several baselines. We evaluate all methods on a small subset of 10 songs from RWC-MDB [38–40], another dataset which includes melody transcription labels. We chose this specific subset in an effort to compare to early DSP-based work on melody transcription—none of the early approaches [22–25] shared code, however [24] shared melody transcriptions for this 10-song subset.

In addition to [24], we also compare to a baseline which applies a note segmentation heuristic [19] to a melody extraction algorithm [48]. We additionally compare to MT3 in a zero-shot fashion—this model was not trained on melody transcription but was trained on some tasks which incorporate vocal transcription. Finally, because the vo-

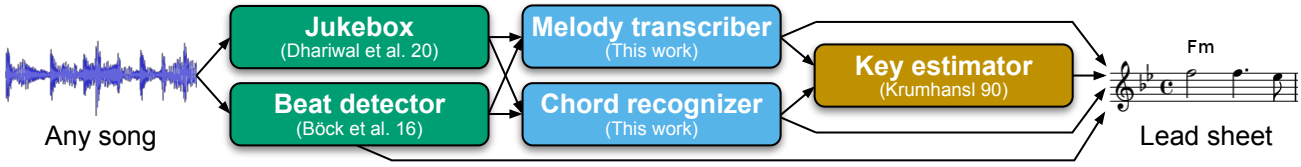


Figure 3. Inference procedure for Sheet Sage, our proposed system which transcribes any Western music audio into lead sheets (scores which depict melody as notes and harmony as chord names). The green, blue, and yellow boxes respectively take audio, features, and symbolic music data as input. Green boxes are modules that we built as part of this work—both are Transformers [14] trained on their respective tasks using audio features from Jukebox [1] and data from HookTheory [49].

cals often carry the melody in popular music, we compare to a baseline of running the Tony [28] monophonic transcription software on source-separated vocals isolated with Spleeter [31]. Because this approach will only work for vocals, we also separately report performance on a subset of our evaluation set where the vocals represent the melody. Scores for all methods and baselines appear in Table 2.

Overall, our approach to training Transformers with features from Jukebox significantly outperforms the strongest baseline in both the vocals-only and unrestricted settings ($p < 0.01$ using a two-sided t-test for paired samples). Qualitatively speaking, the stronger baselines produce transcriptions where a reasonable proportion of the notes are the correct pitches, but they have poor rhythmic consistency with respect to the ground truth. In contrast, our best model produces the correct pitches more often and with a higher degree of rhythmic consistency.

7. SHEET SAGE

As a bonus demo, here we describe Sheet Sage, a system we built to automatically convert music audio into lead sheets (see footnote on first page for examples), powered by our Jukebox-based melody transcription model. In Western music, a piece can often be characterized by its melody and harmony. When engraved as a lead sheet—a musical score containing the melody as notes on a staff and the harmony as chord names—melody and harmony can be readily interpreted by musicians, enabling recognizable performances of existing pieces. Hence, for some music, a lead sheet represents the essence of its underlying composition. Existing services like Chordify [50] can already detect a subset of the information needed to produce lead sheets (specifically, chords, beats, and keys) for broad music audio. However, despite past research efforts [24, 25], no user-facing service yet exists which can convert broad music audio into lead sheets, presumably due to the poor performance of existing melody transcription systems.

To build Sheet Sage, we also train a Jukebox-based chord recognition model on the HookTheory data, using the same methodology that we propose for melody transcription (we simply replace the target vocabulary of onset pitches with one containing chord labels). Passing audio through our Jukebox-based melody transcription and chord recognition models results in a score like format containing raw note names and chord labels per sixteenth note. Engraving this information as a lead sheet requires additional information: the key signature and the time signature. We estimate the former using the Krumhansl-Schmuckler algorithm [51, 52], which takes the symbolic melody and chord information as input. For the latter, we use `madmom` [12, 13]. Finally, we engrave a lead sheet using Lilypond [53]. See Figure 3 for a full schematic.

Subjectively speaking, Sheet Sage often produces high-quality lead sheets, especially for the chorus and verse segments of pop music which have more prominent melodies. Performance is fairly robust across styles and instruments, even those which are less represented in the training data—one user reported particularly strong success on Bollywood music. However, the system occasionally struggles, especially with quieter vocals, layered harmonies, unusual time signatures, or poor intonation. Sheet Sage is also limited to fixed time and key signatures due to limitations of its downbeat detection and key estimation modules.

8. CONCLUSION

We present a new method and dataset which together improve melody transcription on broad music audio. Our method benefits from the rich representations learned by generative models pre-trained on broad audio. This suggests that further improvement in melody transcription may be possible without additional data, i.e., by scaling up or otherwise improving the pre-training procedure. By open sourcing our models and dataset, we hope to spark renewed interest for melody transcription in the MIR community, which may in turn reduce the gap between human perception and machine recognition of a fundamental aspect of music.

9. ETHICAL CONSIDERATIONS

Our definition of melody transcription incorporates equal temperament, a Western-centric tuning system. This could lead to disparate treatment of non equal-tempered music, e.g., if a streaming service were to use melody transcriptions for recommendation. We therefore advocate for the deployment of transcription only in contexts where users are self-selecting music to listen or play along to. Transcription may also be used to create training data for generation—as with any work on generation, there are risks of plagiarism and labor displacement. We recommend that any work on generation involve careful auditing and mitigation of plagiarism. Due to the incomplete nature of a melody, we argue that melody generation tools are more likely to be *incorporated* into co-creation workflows (see [54]) rather than used to displace musicians.

10. ACKNOWLEDGEMENTS

Thanks to Annie Hui-Hsin Hsieh, John Hewitt, Maggie Henderson, Megha Srivastava, Nelson Liu, Pang Wei Koh, Rodrigo Castellon, Sam Ainsworth, and Zachary C. Lipton for helpful discussions, support, and advice. We thank all reviewers for their helpful comments.

11. REFERENCES

[1] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv:2005.00341*, 2020.

[2] M. Ryynänen, T. Virtanen, J. Paulus, and A. Klapuri, "Accompaniment separation and karaoke application based on automatic melody transcription," in *IEEE International Conference on Multimedia and Expo*, 2008.

[3] K. Droe, "Music preference and music education: A review of literature," *Applications of Research in Music Education*, 2006.

[4] D. Bainbridge, C. G. Nevill-Manning, I. H. Witten, L. A. Smith, and R. J. McNab, "Towards a digital library of popular music," in *ACM Conference on Digital Libraries*, 1999.

[5] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query by humming: Musical information retrieval in an audio database," in *ACM International Conference on Multimedia*, 1995.

[6] S. Ewert, B. Pardo, M. Muller, and M. D. Plumbley, "Score-informed source separation for musical audio recordings: An overview," *IEEE Signal Processing Magazine*, 2014.

[7] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *ICLR*, 2019.

[8] M. Goto and S. Hayamizu, "A real-time music scene description system: Detecting melody and bass lines in audio signals," in *International Joint Conference on Artificial Intelligence Workshop on Computational Auditory Scene Analysis*, 1999.

[9] M. Goto, "A real-time music-scene-description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals," *Speech Communication*, 2004.

[10] J. Salamon, E. Gómez, D. P. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *IEEE Signal Processing Magazine*, 2014.

[11] K. S. Rao, P. P. Das *et al.*, "Melody extraction from polyphonic music by deep learning approaches: A review," *arXiv:2202.01078*, 2022.

[12] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *ISMIR*, 2016.

[13] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "`madmom`: A new python audio and music signal processing library," in *International Conference on Multimedia*, 2016.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, 2017.

[15] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE Transactions on Audio, Speech, and Language Processing*, 2016.

[16] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *ISMIR*, 2018.

[17] R. Castellon, C. Donahue, and P. Liang, "Codified audio language modeling learns useful representations for music information retrieval," in *ISMIR*, 2021.

[18] J. S. Downie, X. Hu, J. H. Lee, K. Choi, S. J. Cunningham, and Y. Hao, "Ten years of MIREX: reflections, challenges and opportunities," in *ISMIR*, 2014.

[19] J. Salamon, "`audio_to_midi_melodia`," https://github.com/justinsalamon/audio_to_midi_melodia, 2015.

[20] R. Nishikimi, E. Nakamura, K. Itoyama, and K. Yoshii, "Musical note estimation for F0 trajectories of singing voices based on a Bayesian semi-beat-synchronous HMM," in *ISMIR*, 2016.

[21] R. Nishikimi, E. Nakamura, M. Goto, K. Itoyama, and K. Yoshii, "Scale-and rhythm-aware musical note estimation for vocal F0 trajectories based on a semi-tatum-synchronous hierarchical hidden semi-markov model," in *ISMIR*, 2017.

[22] R. P. Paiva, T. Mendes, and A. Cardoso, "An auditory model based approach for melody detection in polyphonic musical recordings," in *International Symposium on Computer Music Modeling and Retrieval*, 2004.

[23] R. P. Paiva, T. Mendes, and A. Cardoso, "On the detection of melody notes in polyphonic audio," in *ISMIR*, 2005.

[24] M. P. Ryynänen and A. P. Klapuri, "Automatic transcription of melody, bass line, and chords in polyphonic music," *Computer Music Journal*, 2008.

[25] J. Weil, T. Sikora, J.-L. Durrieu, and G. Richard, "Automatic generation of lead sheets from polyphonic music signals," in *ISMIR*, 2009.

[26] A. Laaksonen, "Automatic melody transcription based on chord transcription," in *ISMIR*, 2014.

[27] E. Molina, L. J. Tardón, A. M. Barbancho, and I. Barbancho, "SiPTH: Singing transcription based on hysteresis defined on the pitch-time curve," *IEEE Transactions on Audio, Speech, and Language Processing*, 2014.

[28] M. Mauch, C. Cannam, R. Bittner, G. Fazekas, J. Salamon, J. Dai, J. Bello, and S. Dixon, "Computer-aided melody note transcription using the Tony software: Accuracy and efficiency," in *International Conference on Technologies for Music Notation and Representation*, 2015.

[29] R. Nishikimi, E. Nakamura, M. Goto, K. Itoyama, and K. Yoshii, "Bayesian singing transcription based on a hierarchical generative model of keys, musical notes, and F0 trajectories," *IEEE Transactions on Audio, Speech, and Language Processing*, 2020.

[30] R. Nishikimi, E. Nakamura, M. Goto, and K. Yoshii, "Audio-to-score singing transcription based on a CRNN-HSMM hybrid model," *APSIPA Transactions on Signal and Information Processing*, 2021.

[31] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, 2020.

[32] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: challenges and future directions," *Journal of Intelligent Information Systems*, 2013.

[33] J. Thickstun, Z. Harchaoui, and S. Kakade, "Learning features of music from scratch," in *ICLR*, 2017.

[34] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2019.

[35] J. Gardner, I. Simon, E. Manilow, C. Hawthorne, and J. Engel, "MT3: Multi-task multitrack music transcription," in *ICLR*, 2022.

[36] A. Ycart, L. Liu, E. Benetos, and M. Pearce, "Investigating the perceptual validity of evaluation metrics for automatic piano music transcription," *TISMIR*, 2020.

[37] C. Raffel, B. McFee, E. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. Ellis, "`mir_eval`: A transparent implementation of common MIR metrics," in *ISMIR*, 2014.

[38] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Popular, classical and jazz music databases," in *ISMIR*, 2002.

[39] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC Music Database: Music genre database and musical instrument sound database," in *ISMIR*, 2003.

[40] M. Goto, "Development of the RWC Music Database," in *International Congress on Acoustics*, 2004.

[41] Y.-W. Chen, H.-S. Lee, Y.-H. Chen, and H.-M. Wang, "SurpriseNet: Melody harmonization conditioning on user-controlled surprise contours," in *ISMIR*, 2021.

[42] Y.-C. Yeh, W.-Y. Hsiao, S. Fukayama, T. Kitahara, B. Genchel, H.-M. Liu, H.-W. Dong, Y. Chen, T. Leong, and Y.-H. Yang, "Automatic melody harmonization with triad chords: A comparative study," *Journal of New Music Research*, 2021.

[43] J. Jiang, G. G. Xia, and D. B. Carlton, "MIREX 2019 submission: Crowd annotation for audio key estimation," *MIREX*, 2019.

[44] J. Jiang, G. G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa, "TransformerVAE: A hierarchical model for structure-aware and interpretable music representation learning," in *ICASSP*, 2020.

[45] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, "Sequence-to-sequence piano transcription with transformers," in *ISMIR*, 2021.

[46] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[47] B. McFee and D. Ellis, "Analyzing song structure with spectral clustering," in *ISMIR*, 2014.

[48] J. Salamon and E. Gómez, "Melody extraction from polyphonic music signals using pitch contour characteristics," *IEEE Transactions on Audio, Speech, and Language Processing*, 2012.

[49] C. Anderson, D. Carlton, R. Miyakawa, and D. Schwachhofer, "Hooktheory TheoryTab DB," https://www.hooktheory.com/theorytab.

[50] B. de Haas, J. P. Magalhaes, D. Ten Heggeler, G. Bekenkamp, and T. Ruizendaal, "Chordify: Chord transcription for the masses," in *ISMIR Late-Breaking Demos*, 2014.

[51] C. L. Krumhansl, *Cognitive foundations of musical pitch*. Oxford University Press, 1990.

[52] D. Temperley, "What's key for key? The Krumhansl-Schmuckler key-finding algorithm reconsidered," *Music Perception*, 1999.

[53] H.-W. Nienhuys and J. Nieuwenhuizen, "LilyPond, a system for automated music engraving," in *Colloquium on Musical Informatics*, 2003.

[54] C.-Z. A. Huang, H. V. Koops, E. Newton-Rex, M. Dinculescu, and C. J. Cai, "AI Song Contest: Human-AI co-creation in songwriting," in *ISMIR*, 2020.

SOURCE SEPARATION OF PIANO CONCERTOS WITH TEST-TIME ADAPTATION

Yigitcan Özer **Meinard Müller**
International Audio Laboratories Erlangen, Germany
{yigitcan.oezer, meinard.mueller}@audiolabs-erlangen.de

ABSTRACT

Music source separation (MSS) aims at decomposing a music recording into its constituent sources, such as a lead instrument and the accompaniment. Despite the difficulties in MSS due to the high correlation of musical sources in time and frequency, deep neural networks (DNNs) have led to substantial improvements to accomplish this task. For training supervised machine learning models such as DNNs, isolated sources are required. In the case of popular music, one can exploit open-source datasets which involve multitrack recordings of vocals, bass, and drums. For western classical music, however, isolated sources are generally not available. In this article, we consider the case of piano concertos, which is a genre composed for a pianist typically accompanied by an orchestra. The lack of multitrack recordings makes training supervised machine learning models for the separation of piano and orchestra challenging. To overcome this problem, we generate artificial training material by randomly mixing sections of the solo piano repertoire (e.g., piano sonatas) and orchestral pieces without piano (e.g., symphonies) to train state-of-the-art DNN models for MSS. As our main contribution, we propose a test-time adaptation (TTA) procedure, which exploits random mixtures of the piano-only and orchestra-only parts in the test data to further improve the separation quality.

1. INTRODUCTION

Music source separation (MSS) is a central task in music information retrieval, which seeks to recover individual musical sources in audio recordings. In general, a musical source can refer to singing, an instrument, or an entire group of instruments providing an accompaniment. Since musical signals often exhibit non-stationary spectro–temporal properties and may be highly correlated in time and frequency, MSS proves to be a challenging task in music processing [1]. In the last years, deep neural networks (DNNs) have led to substantial improvements in separating musical sources [2–11]. Despite their effectiveness, one disadvantage of data-driven deep models is their need for a large training dataset, which in the case of MSS consists of multitrack recordings with (isolated) individual sources or stems. Most of the open-source datasets with isolated stems are limited to popular music, e.g., MUSDB18 [12]. However, for western classical music, professionally produced multitrack recordings are quite rare [13, 14].

In this article, we consider the separation of piano concertos, which are a genre of central importance in western classical music. From the Baroque era onward, numerous composers have written piano concertos, which are compositions highlighting the virtuosity of pianists. As an example, Figure 1 shows an excerpt from the first movement of Piano Concerto in D minor (KV 466) by Wolfgang Amadeus Mozart. In addition to the large number of compositions, there also are many prominent historical recordings of piano concertos in classical music archives. In this context, separation of piano and orchestra can enable applications such as editing and upmixing historical and modern piano concerto recordings.

As the piano is the lead instrument and the orchestra takes over the accompaniment, separation of piano concertos can be regarded as a lead instrument and accompaniment separation task [15–17]. The piano has distinct timbral characteristics, e.g., clear onsets, which intuitively may help a separation model in distinguishing it from orchestral instruments such as strings, woodwinds, and brass. Nevertheless, the high spectro–temporal correlations between the piano and orchestral parts in concertos constitute a challenging problem.

In this paper, we adapt the spectral-based Spleeter model [5] to address the separation of piano and orchestra in piano concertos. Spleeter has achieved impressive results for the separation of four stems (vocals, drums, base, and others) in the SiSEC challenge [18]. Building upon its standard architecture, which is an encoder–decoder convolutional neural network (CNN), we train our baseline model using a proprietary dataset.

When training deep MSS models, generating random mixes of solo instrument recordings may improve the separation quality [4, 19]. Random mixing for data generation and augmentation has opened up new paths for separating instrument mixtures, for which multitrack recordings are not available. For example, Chiu et al. [20] created their own synthetic dataset comprising classical violin and pop piano solo recordings, which serve as training material of an MSS model for the separation of piano and violin duos. Inspired by the recent advances in deep learning and data

© Yigitcan Özer, Meinard Müller. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Yigitcan Özer, Meinard Müller, "Source Separation of Piano Concertos with Test-Time Adaptation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

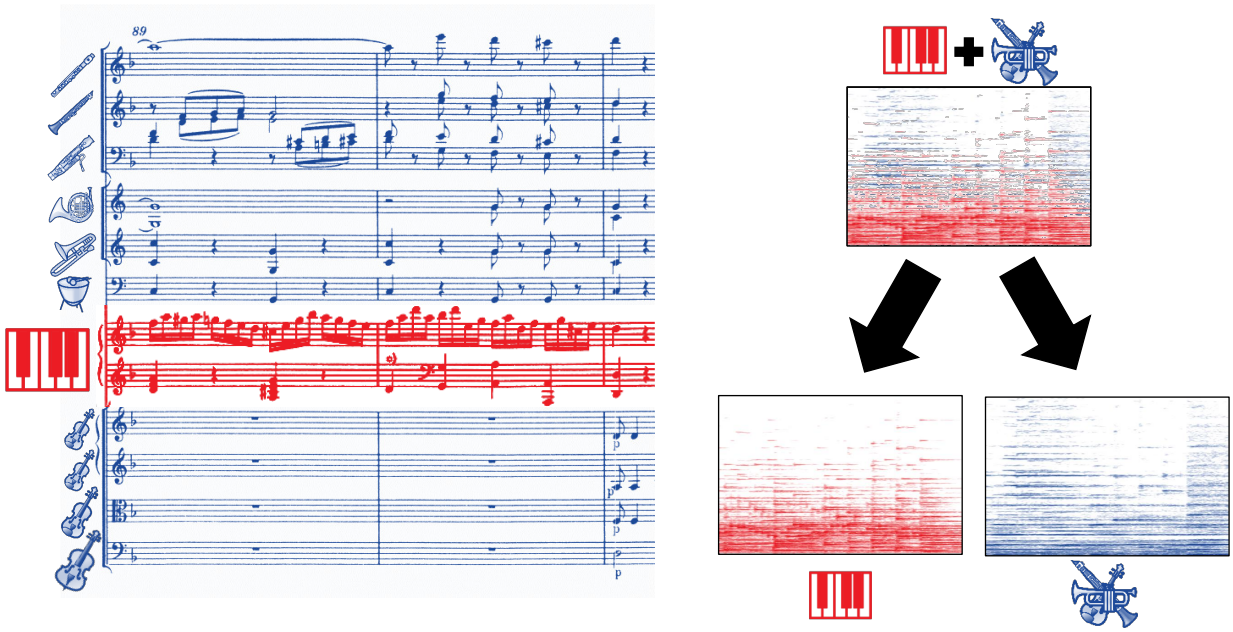


Figure 1: An excerpt from the first movement of the Piano Concerto in D minor (KV466) by Wolfgang Amadeus Mozart. Our goal is to estimate the magnitude spectrograms of the piano part (red) and orchestral part (blue).

augmentation, we generate in our setting an artificial training dataset through randomly mixing samples from the solo piano repertoire (e.g., piano sonatas, mazurkas, etc.) and orchestral pieces without piano (e.g., symphonies) to simulate piano concertos.

Whereas one can improve the performance of data-based models using artificially generated data, a supervised machine learning model necessitates a representative training set to ensure its robustness during the testing phase. In the case of MSS, many recordings have specific acoustic properties (e.g., historical recordings) that are not reflected in training datasets, thus leading to a poor separation performance. To overcome this issue, one can exploit the occurence of repeating patterns in the same recording [21], use bootstrapping to improve separation results [22, 23], or adapt a pre-trained model to one specific target mixture [24, 25]. In this work, our approach is based on the latter strategy. To this end, we first train on our artificial dataset and then finetune the model at the testing stage. As our main contribution, we propose a test-time adaptation (TTA) method similar to [26], where we exploit that a piano concerto typically has relatively long piano-only and orchestra-only passages. Generating random mixes of these sections, we adapt the separation model at the test time individually for each piano concerto in our test dataset. Our systematic experiments highlight the benefits of TTA trained with the spectrogram-domain MSS model Spleeter [5]. To evaluate the performance of our models, we use the widely-used Signal to Distortion Ratio (SDR) [27], and the *2f-model* [28], which is an objective quality measure. Furthermore, we conduct Multiple Stimulus with Hidden Reference and Anchors (MUSHRA) listening tests [29] to assess the perceptual separation quality.

The remainder of the paper is organized as follows. In Section 2, we describe our MSS approach, explore the recent state-of-the-art spectrogram-domain DNN model Spleeter to address the MSS task and describe our experimental setting. In Section 3, we introduce the TTA procedure to improve the separation quality of piano concertos and present our dataset. In Section 4, we report on the quantitative empirical results, including a subjective evaluation. Finally, we conclude in Section 5 with prospects on future work.

2. MUSIC SOURCE SEPARATION APPROACH

Recent DNN approaches for MSS can be divided into two categories: waveform-domain architectures [6, 7] and spectrogram-domain approaches [2–5]. In this paper, our focus is the latter type of models, which learn to approximate the magnitude spectrogram of a target source. To reconstruct the separated audio signals, spectrogram-based models typically use binary masking, soft masking or multichannel Wiener filtering [30].

In particular, we adapt the Spleeter model [5] for separating piano concertos. Its default setting is based on a *U-Net* [31], which has recently been a widely-used architecture in the MIR community to address the MSS task [2, 6, 25, 32, 33]. Following this trend of research, we adapt a U-Net model to predict the magnitude spectrograms of the constituent piano and orchestral parts in a piano concerto. In the following, we revisit the U-Net architecture in Section 2.1 and present our experimental setting in Section 2.2.

2.1 U-Net Model

The U-Net model is composed of a convolutional encoder-decoder architecture with skip connections, which account for the resurrection of fine-grained details in the reconstructed representation. Following the default setting of the U-Net model in [2], we use a 12-layer network (6 layers for the encoder and 6 for the decoder). Each encoder layer uses a strided 2D convolution with a kernel size of 5×5 and a stride size of 2, preceded by a leaky rectified linear unit (ReLU) activation function, and batch normalization. The decoder is composed of strided deconvolution layers with a kernel size of 5×5 and a stride size of 2, as in the encoder. The decoder uses ReLU as the activation function, different from the encoder. To avoid overfitting, we use here dropout with a probability of 0.5 in the first three layers of the decoder. The final layer of the network is a sigmoid activation function, yielding a soft mask for each target source, which contains values between 0 and 1. As the loss function, we use ℓ^1-norm between masked input mixture and target spectrograms. For further details about network architecture, we refer to [2, 5].

2.2 Experimental Setting

In our experiments, we use mono recordings, which are sampled at 22.05 kHz. We generate the magnitude spectrograms using a Hann window size of 2048 and hop size of 512. In a first step, we train our models using an artificial dataset which contains 20-second random chunks from the mixtures of solo piano recordings (e.g., piano sonatas) and orchestral pieces without piano (e.g., symphonies) by 16 different composers from different periods. The total duration of our randomly generated proprietary dataset is circa 45 hours. We regard this model as our pre-trained model, which we denote as `PT`.

We train all our models on a single NVIDIA GeForce 1080 Ti GPU, using a batch size of 8, and a learning rate of $1\mathrm{e}{-4}$ with ADAM optimizer. To improve the separation quality of real piano concerto recordings, we finetune the model with TTA, which we describe in the next section.

3. TEST-TIME ADAPTATION

Supervised deep learning models addressing the MSS task typically require a large dataset that consists of isolated recordings. As a data augmentation method, one can use random mixes to provide training material for an MSS model in the case isolated stems are missing [19, 20]. While this approach cannot simulate the harmonic and rhythmic relationships between various instruments in a real recording, it helps the model to distinguish timbral characteristics of the concurrent musical sources. However, the acoustic properties of recordings (including reverberation, and background noise) play an essential role when upmixing and separating different musical tracks. For instance, in the case of poor recording conditions, (e.g., historical recordings) the properties of the test data may not be reflected well in the training set, thus resulting in a poor separation quality. Finetuning a pre-trained MSS model in the testing phase using a few samples drawn from the test data (also called *test-time adaptation (TTA)* [26]) can improve the separation quality by capturing the specific acoustic features found in a music recording.

From this perspective, separation of piano concertos is a particularly suitable scenario for applying TTA thanks to their compositional form. Depending on the period in which the work was composed, these compositions often comprise long piano-only (e.g., in the cadenza) and orchestra-only parts (e.g., in the exposition, also called *opening riternello*). Using these sections, one can create artificial mixes which come from the audio material of the given test item. As a result, the mixes share the same recording conditions as the test data.

To investigate this approach, we consider seven piano concerto recordings (see Figure 2a). The selected movements of these piano concertos have a long cadenza, which contains only the piano (see Figure 2b). Note that, with the exception of `BrahOp015`, these musical pieces also comprise a long exposition in which only the orchestra plays. For our experiments, we annotate the piano-only and orchestra-only sections, which are publicly available. [1] Exploiting the structural characteristics of piano concertos, we create random mixes of piano-only and orchestra-only sections, which serve as further training data for model adaptation for each piano concerto individually. In the next section, we investigate the improvement of qualitative and subjective separation quality via TTA.

4. EVALUATION

In this section, we report on the separation results acquired by our pre-trained model `PT` and the finetuned models `TTA`. In Section 4.1 we describe our test dataset. We discuss the quantitative empirical results in Section 4.2 and present the subjective evaluation in Section 4.3.

4.1 Test Dataset

For the evaluation of our models, we generate 30-second random mixes of piano-only and orchestra-only parts sampled from the annotated piano concertos (see Figure 2b). These are different from the artificial training set, which we use for the pre-trained model, as they share harmonic and acoustic properties originating from the same recording. Note that we ensure that the samples used for training do not overlap with the test mixtures which we use for the evaluation purposes.

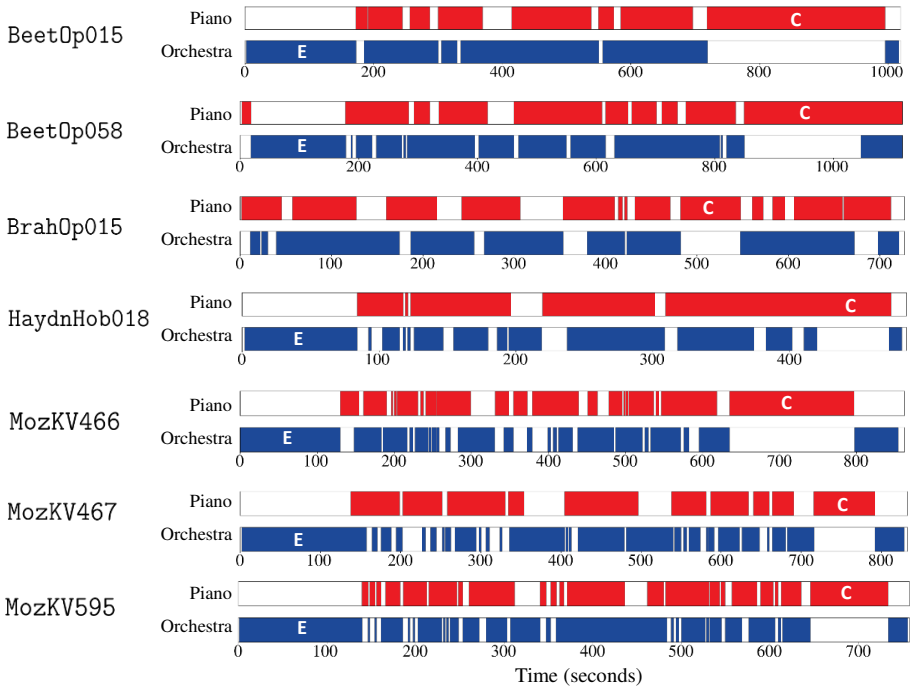
4.2 Quantitative Evaluation

To get a first impression of the performance of the models `PT` and `TTA`, we use the SDR [27] as our quantitative evaluation metric for the separation. Table 1 provides a comparison of the resulting SDR values between a baseline for the SDR values (denoted as `BL`), which we compute using the test mixture as the target signal and ground-truth sources as the reference, pre-trained `PT`, and finetuned `TTA`

[1] https://www.audiolabs-erlangen.de/resources/MIR/2022-PianoSep/

Composer	Full Name	Performer	Work ID	Year	M.	Dur. (T)	Dur. (E)	Dur. (C)
Beethoven	Piano Concerto No.1 in C major, Op.15	Schnabel	BeetOp015	1932	1	1020	170	277
Beethoven	Piano Concerto No.4 in G major, Op.58	Gulda	BeetOp058	1960	1	1116	159	197
Brahms	Piano Concerto No.1 in D minor, Op.15	Arrau	BrahOp015	1958	3	728	N/A	65
Haydn	Piano Concerto No.11 in D major, Hob. XVIII:11	Gulda	HaydnHob018	1962	1	486	83	52
Mozart	Piano Concerto No.20 in D minor, KV. 466	Renzi	MozKV466	N/A	1	862	129	161
Mozart	Piano Concerto No.21 in C major, KV. 467	N/A	MozKV467	1962	1	833	136	76
Mozart	Piano Concerto No.27 in B–flat major, KV. 595	Casadesus	MozKV595	1963	1	778	128	88
					Σ	5823	805	916

(a) The table shows the composer, full name of the work, performer, identifier, recording year, movement (M.), duration (Dur.) in seconds of total recording (T), exposition (E), and cadenza (C).



(b) Annotation of the piano concertos in our dataset into the piano (red), orchestral (blue) parts. To finetune the pre-trained model with the test-time adaptation (TTA) approach, we generate random mixtures of the piano-only (e.g., in the *Cadenza*, denoted as **C**) and orchestra-only (e.g. in the *Exposition*, denoted as **E**) sections.

Figure 2: Overview of the piano concertos in our test dataset.

after 100 iterations. One can observe that PT leads to a substantial improvement in SDR values compared to BL over the whole dataset, both for the piano and orchestra. It is interesting to observe that PT improves the SDR value of BeetOp015 for the separation of piano from 4.48 to 4.60, which is a relatively low improvement compared to other piano concertos in the dataset. Note that BeetOp015 is a historical recording (see Figure 2a for the recording year of the piano concertos), whose inadequate recording conditions may not be well represented in the random mixes used for the training of the PT, thus leading to a relatively poor separation performance.

Now, we focus on the comparison between PT and TTA. In general, the SDR-based results demonstrate that TTA enhances the separation of PT across all the piano concertos, for both the piano and the orchestra. For example, in the case of BeetOp015, PT yields an SDR value of 4.60 for the separated piano. After finetuning with TTA for 100 iterations, this improves to 8.95. For the separated orchestra of BeetOp015, TTA also improves the SDR from 0.09 to 3.67. In the case of better quantitative separation results by PT, e.g., MozKV595, we observe that the improvement via TTA is relatively lower. Here, the SDR values improve from 12.74 to 13.00 for the piano and from 5.25 to 5.58 for the orchestra. Furthermore, our analysis reveals that the SDR value for the separated orchestra is generally lower than piano for both PT and TTA over the whole test dataset, except for MozKV467. An informal inspection states that the TTA leads to a significant improvement in the separation performance for historical recordings, which are not well-reflected in the training dataset of the pre-trained model PT.

In our next experiment, we investigate the performance of the finetuned models TTA per iteration. Figure 3 illustrates the evolution of the SDR values for each piano concerto in our test dataset. The overall convergence behavior exhibits a general trend of improvement of SDR values through TTA over PT for the separation of the piano and the orchestra. In particular, the SDR values for the separation of BeetOp015 depict a rapid improvement within

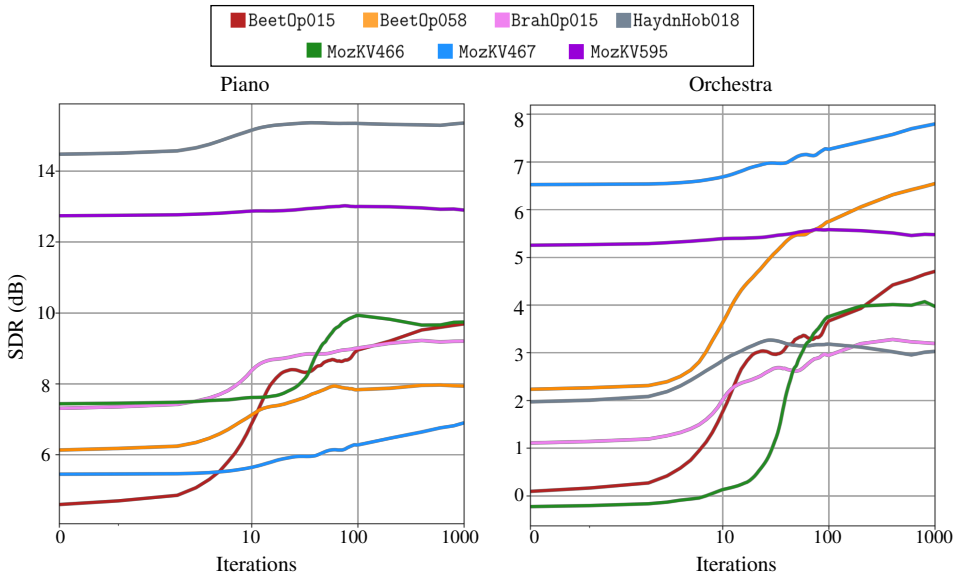


Figure 3: Evolution of SDR values based on our test dataset, applying TTA on the pre-trained model PT.

Work ID	Piano			Orchestra		
	BL	PT	TTA	BL	PT	TTA
BeetOp015	4.48	4.60	8.95	−4.36	0.09	3.67
BeetOp058	1.62	6.13	7.83	−1.58	2.23	5.75
BrahOp015	4.75	7.31	9.02	−4.60	0.09	3.67
HaydnHob018	10.99	14.47	15.34	−10.93	1.97	3.18
MozKV466	5.01	7.44	9.93	−5.06	−0.23	3.76
MozKV467	−0.72	5.45	6.28	0.73	6.52	7.26
MozKV595	6.64	12.74	13.00	−6.89	5.25	5.58
ϕ	4.67	8.31	10.05	−4.67	2.27	4.70

Table 1: Comparison of the SDR (dB) values between the baseline BL, and the separated sources by the pre-trained model PT and the finetuned model TTA after 100 iterations. The average SDR values are denoted with ϕ.

Work ID	Piano			Orchestra		
	BL	PT	TTA	BL	PT	TTA
BeetOp015	21.60	21.50	28.39	15.51	25.85	29.52
BeetOp058	22.08	27.13	36.19	27.02	38.65	38.68
BrahOp015	24.01	30.79	36.43	22.63	35.36	33.20
HaydnHob018	19.27	34.57	38.31	27.10	41.19	40.35
MozKV466	19.25	32.30	39.49	26.07	35.62	40.18
MozKV467	15.61	28.80	31.52	28.43	40.26	41.21
MozKV595	14.88	27.82	31.52	18.08	30.36	31.49
ϕ	19.53	28.99	34.55	23.55	35.33	36.38

Table 2: Comparison of the 2f-model values between the baseline BL, and the separated sources using the pre-trained model PT and finetuned model TTA after 100 iterations. The average 2f-model values are denoted with ϕ.

the first 10 iterations. For the other piano concertos, the improvement of SDR values generally accelerates after the 10th iteration. After the 100th iteration, the separation performance remains steady for most of the piano concertos. Furthermore, after the 100th iteration, the SDR values constantly increase in the case of BeetOp015 and MozKV467 for both piano and orchestra.

Although SDR is widely used as a quantitative evaluation metric for MSS, it is well known that it may not be suitable for determining the perceptual sound quality of separated musical sources [34]. The work by Torcoli et al. [35] provides a comparison of objective quality measures in the source separation domain. Their analysis indicates that a quantitative evaluation using the metric called *2f-model* exhibits the best correlation with ground-truth data based on the subjective ratings from MUSHRA listening tests. For a detailed account of the 2f-model, we refer to [28]. Note that the 2f-model values range from 0 to 100 following MUSHRA rating scores (see Section 4.3). Table 2 provides the resulting 2f-model values for the separated sources by PT and TTA using 100 iterations. In general, one can observe here a similar trend as for the SDR. PT mostly reveals better 2f-model scores than the baseline BL, except for the piano separation of BeetOp015, which presumably suffers from its poor recording conditions that are not well-represented in the artificial training set.

As for the SDR-based results, 2f-model values increase via TTA after 100 iterations for both the piano and the orchestra. For example, in the case of BeetOp015, PT yields a 2f-model value of 21.50 for the separated piano, improving to 28.39 after applying TTA. Interestingly, the separation of the orchestral part yields better results than the piano according to 2f-model values. This is opposed to the evaluation based on the SDR, where the separation results are significantly better in the case of piano separation (see Table 1).

4.3 Subjective Evaluation

In this section, we describe the experimental setting for our subjective evaluation to assess the perceived separation quality. We carried out two listening tests using the MUSHRA methodology following the ITU-R BS.1534-3 recommendation [36]. It is a double-blind multi-stimulus test method with a hidden reference and an additional lower anchor signal. The rating scores range from 0 to

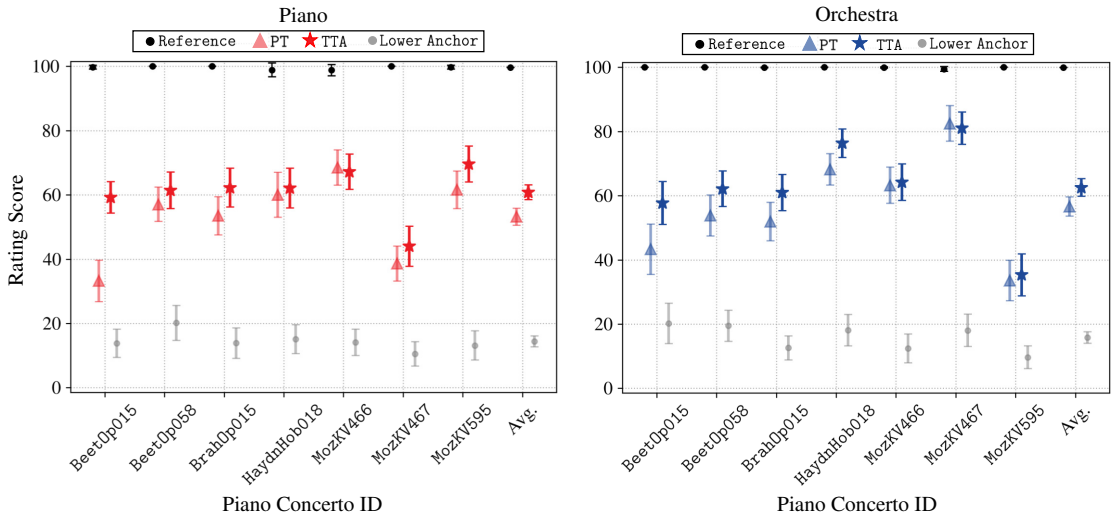


Figure 4: Results of our subjective evaluation based on MUSHRA listening tests for the piano (left) and orchestra (right). The colored markers indicate the average rating scores enclosed by 95% confidence intervals (indicated by the vertical lines).

100, involving five categories: Bad (0-20), Poor (20-40), Fair (40-60), Good (60-80), and Excellent (80-100).

In total, 34 participants were involved in our listening tests (31 experienced listeners and 3 inexperienced listeners). The MUSHRA methodology suggests a post-screening of the participants stating that participants should be excluded from the listening test if they give the hidden reference a score lower than 90 for more than 15% of the test items. Concerning our tests, one of the participants was excluded after post-screening.

Each of the two listening tests contains seven test items. For each test item, we generated four different signals with a duration of 12 seconds (maximum allowed signal duration for MUSHRA listening tests), which are excerpts from the test mixtures that we use for our quantitative evaluation. In our first listening test, we asked the participants to rate the overall audio quality for each of the four signals (also called *conditions*) with respect to a reference signal, which is a clean piano-only section. Similar to [37], we created the lower anchor signals by low-pass filtering the test mixtures to a 3.5 kHz cut-off frequency and by adding musical noise, i.e., randomly setting 20% of the remaining time/frequency coefficients to zero. The other two signals involve separated piano parts by `PT` and `TTA`. Similarly, our second listening test evaluates the overall quality of the separated orchestral parts following the same procedure as the first listening test.

Figure 4 summarizes the results from our listening tests. First, one can observe that the participants rated the reference signal with an average MUSHRA rating score of 99 and the lower anchor is significantly below the conditions `PT` and `TTA`. Remarkably, the general trend of the performances by `PT` and `TTA` support our quantitative analyses, inferring that the TTA procedure generally improves the separation of both the piano and orchestra. When observing the rating score of the piano concertos individually, one can observe that the rating of the historical recording `BeetOp015` is significantly lower than other items for `PT`. Intuitively, this is due to its poor recording conditions. After applying TTA, the average rating score of `BeetOp015` improves from 33 to 59 for the piano and from 43 to 58 for the orchestra. Furthermore, the orchestra separation led to a lower MUSHRA score in the case of `MozKV595`, both for `PT` and `TTA`. One reason may be the audible clipping artifacts in the reference signal and hidden separated orchestra, which a subset of the participants noted during the listening test.

As a final remark, the subjective results demonstrate that the average separation quality of the orchestra is better than for the piano, which is in favor of the results based on the 2f-model (see Table 2). This again illustrates that quantitative and subjective evaluations need to be carefully interpreted.

5. CONCLUSION

In this paper, we investigated the separation of piano and orchestra in piano concertos. We trained our model using a U-Net architecture based on the Spleeter implementation with random mixes of solo piano and orchestral recordings, which we regarded as our baseline pre-trained model. As the main contribution, we proposed a TTA procedure to enhance the separation quality using the random mixes created from the samples found in the test data. We showed that TTA substantially improved the quantitative and subjective evaluation results, both for the piano and orchestra. For the future, we aim to explore musically plausible data augmentation methods that simulate more realistic mixtures. To further improve the separation quality, avenues of research may be to integrate a transcription model as proposed by [38] or to incorporate phase information into the network by using, e.g., a complex U-Net [39]. Moreover, we intend to further investigate objective evaluation measures for the source separation of piano concertos.

Acknowledgements: This work was supported by the German Research Foundation (DFG MU 2686/10-2). The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

6. REFERENCES

[1] E. Cano, D. FitzGerald, A. Liutkus, M. D. Plumbley, and F. Stöter, "Musical source separation: An introduction," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 31–40, 2019.

[2] A. Jansson, E. J. Humphrey, N. Montecchio, R. M. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep U-net convolutional networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Suzhou, China, 2017, pp. 745–751.

[3] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band densenets for audio source separation," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2017, pp. 21–25.

[4] F. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-Unmix – A reference implementation for music source separation," *Journal of Open Source Software*, vol. 4, no. 41, 2019. [Online]. Available: https://doi.org/10.21105/joss.01667

[5] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020, deezer Research. [Online]. Available: https://doi.org/10.21105/joss.02154

[6] D. Stoller, S. Ewert, and S. Dixon, "Wave-U-net: A multi-scale neural network for end-to-end audio source separation," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018, pp. 334–340.

[7] A. Défossez, N. Usunier, L. Bottou, and F. R. Bach, "Music source separation in the waveform domain," 2019. [Online]. Available: http://arxiv.org/abs/1911.13254

[8] A. Défossez, "Hybrid spectrogram and waveform source separation," in *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, Online, 2021.

[9] H. Liu, Q. Kong, and J. Liu, "CWS-PResUNet: Music source separation with channel-wise subband phase-aware ResUNet," in *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, Online, 2021.

[10] M. Kim, W. Choi, J. Chung, D. Lee, and S. Jung, "KUIELab-MDX-Net: A two-stream neural network for music demixing," in *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, Online, 2021.

[11] R. Sawata, S. Uhlich, S. Takahashi, and Y. Mitsufuji, "All for one and one for all: Improving music separation by bridging networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Toronto, ON, Canada, 2021, pp. 51–55.

[12] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[13] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "MedleyDB: A multitrack dataset for annotation-intensive MIR research," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Taipei, Taiwan, 2014, pp. 155–160.

[14] M. Schedl, D. Hauger, M. Tkalčič, M. Melenhorst, and C. C. S. Liem, "A dataset of multimedia material about classical music: PHENICX-SMM," in *Proceedings of International Workshop on Content-Based Multimedia Indexing (CBMI)*, 2016, pp. 1–4.

[15] C. Joder and B. W. Schuller, "Score-informed leading voice separation from monaural audio," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, 2012, pp. 277–282.

[16] E. Cano, G. Schuller, and C. Dittmar, "Pitch-informed solo and accompaniment separation towards its use in music education applications," *EURASIP Journal on Advances in Signal Processing*, vol. 2014, no. 23, 2014. [Online]. Available: https://doi.org/10.1186/1687-6180-2014-23

[17] Z. Rafii, A. Liutkus, F. Stöter, S. I. Mimilakis, D. FitzGerald, and B. Pardo, "An overview of lead and accompaniment separation in music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1307–1335, 2018.

[18] F. Stöter, A. Liutkus, and N. Ito, "The 2018 Signal Separation Evaluation Campaign," in *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA)*, ser. Lecture Notes in Computer Science, vol. 10891. Springer, 2018, pp. 293–305.

[19] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, New Orleans, Louisiana, USA, March 2017, pp. 261–265.

[20] C.-Y. Chiu, W.-Y. Hsiao, Y.-C. Yeh, Y.-H. Yang, and A. W.-Y. Su, "Mixing-specific data augmentation techniques for improved blind violin/piano source separation," in *Proceedings of the Workshop on Multimedia Signal Processing (MMSP)*, 2020, pp. 1–6.

[21] Z. Rafii and B. Pardo, "Repeating pattern extraction technique (REPET): A simple method for music/voice separation." *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 1, pp. 71–82, 2013.

[22] B. Lehner, R. Sonnleitner, and G. Widmer, "Towards light-weight, real-time-capable singing voice detection," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Curitiba, Brazil, 2013, pp. 53–58.

[23] C. Dittmar, B. Lehner, T. Prätzlich, M. Müller, and G. Widmer, "Cross-version singing voice detection in classical opera recordings," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, October 2015, pp. 618–624.

[24] G. Cantisani, A. Ozerov, S. Essid, and G. Richard, "User-guided one-shot deep model adaptation for music source separation," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, New York, USA, 2021, pp. 111–115.

[25] Y. Wang, J. P. Bello, D. Stoller, and R. Bittner, "Few-shot musical source separation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Singapore, Singapore, 2022, pp. 121–125.

[26] Y. Sun, X. Wang, L. Zhang, J. Miller, M. Hardt, and A. A. Efros, "Test-time training with self-supervision for generalization under distribution shifts," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[27] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[28] T. Kastner and J. Herre, "An efficient model for estimating subjective quality of separated audio source signals," in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, New Paltz, New York, USA, 2019, pp. 95–99.

[29] B. Series, "Method for the subjective assessment of intermediate quality level of audio systems," *International Telecommunication Union Radiocommunication Assembly*, 2014.

[30] A. Liutkus and R. Badeau, "Generalized Wiener filtering with fractional power spectrograms," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, April 2015, pp. 266–270.

[31] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention MICCAI*, Munich, Germany, 2015, pp. 234–241.

[32] G. Meseguer-Brocal and G. Peeters, "Conditioned-U-net: Introducing a control mechanism in the U-net for multiple source separations," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 159–165.

[33] A. Cohen-Hadria, A. Roebel, and G. Peeters, "Improving singing voice separation using deep U-net and wave-U-net with data augmentation," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, A Coruña, Spain, 2019.

[34] E. Cano, D. FitzGerald, and K. Brandenburg, "Evaluation of quality of sound source separation algorithms: Human perception vs quantitative metrics," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, 2016, pp. 1758–1762.

[35] M. Torcoli, T. Kastner, and J. Herre, "Objective measures of perceptual audio quality reviewed: An evaluation of their application domain dependence," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1530–1541, 2021.

[36] International Telecommunications Union, "ITU-R Rec. BS.1534-3: Method for the subjective assessment of intermediate quality levels of coding systems," 2015.

[37] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann, "Subjective and objective quality assessment of audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2046–2057, 2011.

[38] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020, pp. 771–775.

[39] H.-S. Choi, J.-H. Kim, J. Huh, A. Kim, J.-W. Ha, and K. Lee, "Phase-aware speech enhancement with deep complex U-net," in *Proceedings of the International Conference on Learning Representations (ICLR)*, Vancouver, Canada, 2018.

COUNTERPOINT ERROR-DETECTION TOOLS FOR OPTICAL MUSIC RECOGNITION OF RENAISSANCE POLYPHONIC MUSIC

Martha E. Thomae
McGill University
martha.thomaeelias@mail.mcgill.ca

Julie E. Cumming
McGill University
julie.cumming@mcgill.ca

Ichiro Fujinaga
McGill University
ichiro.fujinaga@mcgill.ca

ABSTRACT

In this paper, we present a music information retrieval (MIR) pipeline to aid musicologists in making editions of mensural music sources. We designed this pipeline by improving existing MIR tools and allowing for their interoperability rather than implementing a new monolithic tool. These MIR tools include technologies such as optical music recognition (OMR), automatic voice alignment for mensural notation, editorial correction software, and computational counterpoint error detection. To ease the editorial correction process necessary to obtain correctly lined-up scores, we evaluate whether the use of counterpoint error-detection tools makes the correction process more efficient. While this idea has been discussed before, this paper presents the first attempt at implementing it. The results confirm that marking illegal dissonances in the score following the rules of Renaissance counterpoint makes the process of editorial correction of scribal errors in Renaissance music more efficient by reducing the time taken and improving the accuracy of such corrections. Moreover, it also allowed us to catch OMR errors that had passed through undetected at a previous step of the pipeline. This paper is part of a larger project to preserve and increase access to a set of Guatemalan polyphonic choirbooks through digital images and symbolic scores.

1. INTRODUCTION

This paper discusses part of a larger project to preserve and increase access to a set of six Guatemalan Cathedral choirbooks (the *GuatC* collection). [1] These choirbooks, written in mensural notation, contain mostly sixteenth-century polyphonic music that was copied in the seventeenth and eighteenth centuries [1]. They document a continuous performance tradition of sacred choral music from the Renaissance until the beginning of the nineteenth century and are valuable sources for studying the transmission of music from Europe to Latin America.

[1] *GuatC*: Guatemala. Guatemala City. Catedral, Archivo Capitular. Other sigla include GCA-Gc.

© M. E. Thomae, J. E. Cumming, and I. Fujinaga. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. E. Thomae, J. E. Cumming, and I. Fujinaga, "Counterpoint Error-Detection Tools for Optical Music Recognition of Renaissance Polyphonic Music", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

Given the limited access to these manuscript sources, it is of utmost importance to digitize and encode this corpus which otherwise might be lost, damaged, or forgotten. We used a set of digitization and music information retrieval (MIR) technologies to obtain digital images and symbolic files encoding scores with editorial corrections for each of the pieces of the first choirbook of the collection, *GuatC 1* (see Figure 1). In the process, we tested the following three-step pipeline: (i) digitization, (ii) optical music recognition, and (iii) automatic voice alignment & editorial correction. The digitization step, conducted with a do-it-yourself scanner, was discussed in a previous publication [2]. In this paper, we focus on the encoding part of the pipeline, the last two steps.

1. **Optical music recognition (OMR) & correction of the results.** We perform OMR on the images to retrieve a symbolic file (a Mensural MEI file) encoding the music of the manuscript. We used the *Music Recognition Encoding and Transcription* (*MuRET*) OMR framework for this step (see Section 2.2).

2. **Automatic voice alignment & editorial corrections.** Since we are dealing with mensural notation, OMR is not enough to encode the full rhythmic information of the pieces and obtain a score with the voices properly lined up (see Section 2.1). Two additional steps are needed: (1) *automatic voice alignment*, which provides the actual duration of each note and returns a preliminary score; and (2) *editorial correction*, which allows for corrections of scribal errors, some of which can affect the alignment of the voices into a score. We used the *Measuring Polyphony* (*MP*) *Editor* for this (see Section 2.3).

This process generates symbolic scores with editorial corrections in a semi-automatic way through OMR and automatic voice alignment. The user manually corrects the output of each step, correcting the results of the OMR—the recognized symbols and their pitches—in MuRET and correcting the results of the voice alignment in the MP Editor. The latter normally implies the correction of scribal errors in the form of editorial corrections and occasional OMR errors that went undetected in the previous step of the pipeline. We evaluate whether the use of a tool that identifies illegal dissonances in Renaissance counterpoint (see Section 2.4) helps in the correction of this last step of the pipeline. The goal of this MIR pipeline is to aid musicologists in making editions of mensural music sources,

with the last step aimed to reduce the challenge of aligning (possibly error-ridden) polyphonic parts by automatically scoring up the voices and by flagging areas of special attention to the human editor—given how time-consuming it can be to find scribal errors. While the idea of using counterpoint rules to detect errors in the original mensural sources has been mentioned before [3], it has not been implemented yet. This paper provides the first attempt at its implementation and evaluation by conducting a small experiment to test our hypothesis.

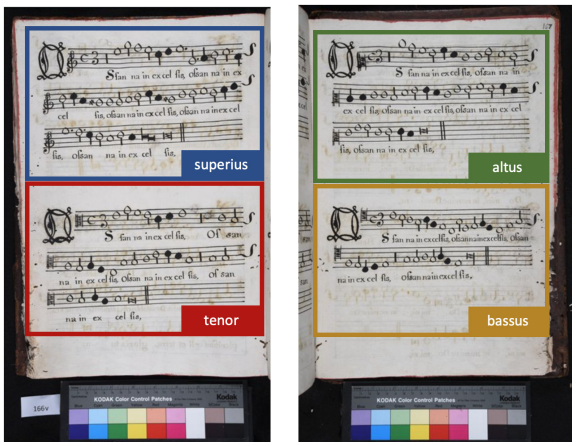


Figure 1: Example of a piece in GuatC 1. The voices are written in choirbook layout, where each voice is in a different area of the book opening.

2. BACKGROUND

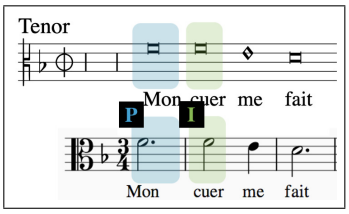
The GuatC collection consists of a set of six manuscript choirbooks written in mensural notation. The first three books have been inventoried [4–6], and the fourth one has been fully transcribed [6]. An overview of the whole collection was presented in [1] and a full inventory is expected [7]. Microfilm images for the first three books were created in the 1980s [8, pp. 3–4]; however, these images are of low quality, with cropped areas and missing folios.

We decided to test the MIR pipeline presented in Section 1 with the first choirbook (GuatC 1, one of the best-preserved manuscripts). The GuatC 1 is a book of masses. It contains twelve masses and fifteen short polyphonic pieces. Eight of the masses are from sixteenth-century composers, one mass is by a seventeenth-century composer, another by an eighteenth-century one, and two by composers whose period of activity remains unknown. On the other hand, most of the short polyphonic pieces are anonymous. Modern transcriptions of ten of the masses and four of the short pieces can be found in the *Choral Public Domain Library* (*CPDL*) wiki, although the provenance of the materials on which the transcriptions were based is shrouded in contradictory accounts. [2]

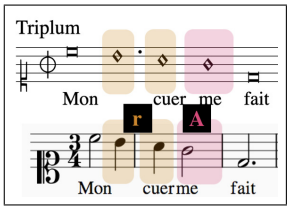
[2] The account found at the *Música Colonial Archive* page (https://www.cpdl.org/wiki/index.php/Música_Colonial_Archive) cannot be corroborated by the Centro de Investigaciones Regionales de Mesoamérica (CIRMA), the institution that holds the original microfilms, as indicated by the director of CIRMA's historical archive (Thelma Porres, personal communication, November 2018).

2.1 Mensural Notation and the Voice-Alignment Issue

Mensural notation was used for polyphonic music in Europe from the thirteenth to the seventeenth centuries. In triple meter, the duration of the notes in mensural notation depends on the context (i.e., the notes preceding or following), as shown in Figure 2. In Figure 2a, the same note shape has two different durational values, a ternary value, which is called *perfect*, and a binary value, which is called *imperfect*. In Figure 2b, another note shape represents two different durations, a regular one and an *altered* one where the note has twice the duration of its regular value. These three durational values of notes, perfect, imperfect, and altered, are common in fourteenth- to sixteenth-century mensural notation.



(a) Same note shape with a perfect (P; triple) and imperfect (I; duple) value.



(b) Same note shape with a regular (one beat) and altered (A; two beats) value.

Figure 2: Example of the different durational values of notes given the context. Both (a) and (b) show an example in mensural notation and its modern transcription below.

The context-dependent duration of mensural notes, together with the separate-parts layout of most mensural music (e.g., the choirbook layout shown in Figure 1), makes it difficult to know what notes are sounding at the same time in the different voices. We implemented an algorithm to compute the duration of notes based on the context and present the piece lined up in a score [9]. This algorithm is referred to as "automatic voice alignment" or "automatic scoring up" of mensural music. After scoring up the voices, it is still necessary to account for errors (e.g., missing or wrong values for notes and rests) to have a correct score. The scoring up of the piece helps to identify OMR and scribal errors that are obscured by the separate-parts arrangement of the music.

2.2 The Music Recognition Encoding and Transcription OMR Framework (MuRET)

There are a few OMR frameworks for early music, including the OMR workflow used by the SIMSSA project through the Rodan workflow manager with the Neon editor [10–13], the OMMR4all framework [14], Aruspix [15],

and MuRET [16, 17]. MuRET, a framework developed by David Rizo at the University of Alicante as part of the Hispamus Project [18], is the only one with support for handwritten mensural notation. [3]

As with other OMR tools, MuRET provides information on the pitch and shape of notes as part of its workflow and allows the user to correct the recognized symbols. It also has support for encoding the imperfect / perfect / altered durational values of mensural notes, though these values are not retrieved automatically—as in the case of pitches and note shapes—and instead have to be entered manually in **mens—a Humdrum format for mensural notation [19]—using MuRET's interface.

At the end of the OMR process, the user can export the encoded music into MEI (Music Encoding Initiative)—a symbolic file format that has support for encoding mensural notation [20]. It is possible to export two kinds of MEI files in MuRET, *score-based* and *parts-based*. The score-based file has the voices stacked and aligned based on the duration of the notes; however, for the notes to be correctly lined up, the user must have provided the durational values for the notes in **mens (i.e., include imperfections and alterations). The parts-based file provides the music encoded as separate parts (just as in the original sources), encoding only pitch and note-shape information for each note. The information about the notes' imperfect, perfect, or altered values is calculated by the next tool in the MIR pipeline, the MP Editor, which will return the score-based Mensural MEI file.

2.3 The Measuring Polyphony Editor (MP Editor)

The MP Editor is an online editor for mensural notation [21], developed under the Measuring Polyphony Project directed by Karen Desmond. [4] Until recently, this was the only mensural notation editor that performed automatic voice alignment, [5] a functionality that I implemented. The MP Editor takes in a series of notes (i.e., pitches and note shape) and automatically lines up the voices into a score using a re-implementation of the scoring-up script in [9]; additionally, after scoring up the voices, the MP Editor allows the user to perform editorial corrections. In a previous paper, I presented the work done to use MuRET's parts-based output as MP Editor's input, which allows for a complete and semi-automatic pipeline of the scoring up of each piece of the GuatC 1 corpus [23].

2.4 The Dissonance Filter (DF)

The goal of this article is to evaluate the efficiency of using counterpoint error markers while making editorial corrections in the last step of the MIR pipeline. On a previous paper, I presented the work done to integrate humlib—Humdrum's data-parsing library—within the MP Editor [23], [6] which allows humlib's Renaissance dissonance filter to label the dissonant notes. [7] The complete list of dissonance labels available in the filter can be consulted on the Verovio Humdrum Viewer (VHV) page. [8] In the MP Editor, the DF marks the dissonances in the score, and makes a distinction between the legal (e.g., passing tones, lower/upper neighbours, and suspensions) and illegal dissonances, rendering the former in blue and the latter in orange. From the set of dissonance labels presented in the VHV documentation page, the ones considered illegal (and rendered in orange font) are *Z*/*z*, *Y*/*y*, *L*/*l*, and *x*. For an example of how the DF labels the dissonances in the MP Editor, see Figures 3–5.

3. METHODOLOGY

3.1 Data Preparation

We randomly selected fifteen pieces from the GuatC 1 manuscript, where each piece was either a mass movement or a short polyphonic piece. The fifteen pieces represent around 20% of the total corpus. To ease the editorial work in the MP Editor, we subdivided the long mass movements into smaller units that we called "self-contained units." Here, we are considering as a "long movement" anything that has more than two openings (i.e., four pages) in the manuscript. We defined the "self-contained unit" as the minimum number of consecutive sections (one or more) that start at a page beginning and end at a page ending. This definition was needed because we had to face the problem of handling sections whose beginning or end do not correspond to the beginning or ending of an image (i.e., a page) since MuRET can only produce an MEI output from a set of one or more selected images.

We obtained a total of 23 self-contained units and divided these into two datasets: a *DF Dataset* where the dissonance filter would be activated during the correction process, and a *NDF Dataset* where no dissonance filter would be used during correction. To guarantee a balanced dataset, we arranged for the DF and NDF datasets to have the same average number of measures, voices, and illegal dissonances. We also made sure that each dataset had close to the same number of CPDL transcriptions to use as a reference, providing another musician's interpretation of where the mistakes are found and how to fix them.

Table 1 shows the pieces in the two datasets. The first column contains the name of the self-contained unit (e.g., Missa9.2_Gloria1), which shows the mass number in the inventory table of the GuatC 1 found in [25] (e.g., Missa9 for the 9th mass in GuatC 1), the movement number and name (e.g., Missa9.2_Gloria1 indicates the second movement in the mass, which is always a *Gloria*), and the unit number within that movement (the first unit of the Gloria).

[3] https://muret.dlsi.ua.es/muret/#/home

[4] https://editor.measuringpolyphony.org

[5] Imperfections and alterations are entered manually in the editor from the Computerized Mensural Music Editing (CMME) project. The new Mensural Rhythm Interpretation Tool (MeRIT) automatically scores up the piece, but its focus is on introspection and pedagogical use and not on producing an edited score [22].

[6] https://github.com/craigsapp/humlib

[7] This filter was developed by Alex Morgan for the Josquin Research Project (https://josquin.stanford.edu). It is based on Peter Schubert's book on modal counterpoint [24].

[8] https://doc.verovio.humdrum.org/filter/dissonant/

3.2 Experiment Setup and Evaluation Procedure

The experiment was conducted by a Bachelor of Music student with a major in voice and a minor in early music. She has knowledge of mensural notation and counterpoint from courses in paleography and Renaissance musicianship, in addition to her participation in early-music vocal ensembles. She corrected each piece following these steps: (1) look at the ends of phrases and sections to see if the voices line up at the cadence in order to determine if there is a note value missing earlier in the piece; and (2) look at the notes preceding that cadence, this would mean looking at all these notes in the NDF Dataset, while focusing on the places with orange labels and the notes preceding them in the DF Dataset. She was also instructed to report the correction time for each piece, annotate any comments about the piece that she considered relevant, and provide the files downloaded from the MP Editor at the end of the correction process. Although most of the corrections at this point in the MIR pipeline should be editorial, we asked the experimenter to still keep an eye out for OMR errors.

We recorded the correction time per piece, the time invested in the correction of the pieces per dataset (DF and NDF), and the accuracy of those corrections. While the average correction time of the DF and NDF datasets is easy to obtain, analyzing the accuracy of the corrections is more involved. To study the accuracy of the corrections, we used three sets of files: (1) the OMR scored-up files obtained by uploading the OMR Parts-based MEI file into the MP Editor and exporting the score with no corrections; (2) the scores corrected by the experimenter; and (3) the CPDL scores. We compared the OMR scores against the experimenter's scores to identify the experimenter's corrections. And we compared the experimenter's scores against the CPDL scores to identify discrepancies between the two transcriptions. [9] While the CPDL scores cannot be considered ground truth, they record another musician's ideas about mistakes in the sources, and help in the process of checking the experimenter's corrections. Whenever we found a discrepancy, we analyzed both versions (the experimenter's and the CPDL's) and chose the best solution, always favoring the one that removed "true" illegal dissonance labels and that guaranteed imitation and motivic consistency. [10] Here, we are distinguishing between "true" and "false" illegal dissonances, where **true** illegal dissonances are notes that are correctly labelled as such (i.e., true positives) and, on the other hand, **false** illegal dissonances are actually legal dissonances that are incorrectly classified by the DF (i.e., false positives). We also reuploaded the experimenter's files into the MP Editor and activated the DF on her files to check if there were any orange labels left to determine whether these were counterpoint errors missed by the experimenter or if they were "false" illegal dissonances. We found a few instances of these false illegal dissonances as reported in Section 4.1.

[9] The OMR and the experimenter's scores are encoded in MEI, while the CPDL scores are in MusicXML. Therefore, to compare them, we wrote transformation scripts for both formats to convert them into text files that encode the notes in each voice as a sequence of tokens and then compared these text files using a *diff* tool.

[10] Imitation refers to repeated melodic motifs in different voices, typical of sacred polyphony.

4. RESULTS AND DISCUSSION

There are two types of results in this experiment: (1) the average correction time of the NDF and DF datasets, which can be seen in Table 1; and (2) the experimenter score files, which are stored in GitHub. [11]

NDF Dataset					
Self-contained Units	CPDL	Number of			Time (min)
		M	V	IllD	
Missa8.4_Sanctus0	yes	29	4	9	8
Missa9.2_Gloria1	yes	44	4	46	13
Missa9.2_Gloria2	yes	37	4	8	17
Missa10.1_Kyrie0	**part**	71	4	4	18
Missa10.3_Credo1	yes	67	4	64	16
Missa10.5_AgnusI0	yes	23	4	2	7
Missa15.3_Credo2	yes	6	4	43	15
Missa15.3_Credo3	yes	48	3	0	15
Missa16.4_Sanctus2	yes	13	4	17	11
Missa17.5_Agnus0	yes	26	4	1	6
Piece21_Surrexit0	**no**	17	5	21	10
Average		**39.2**	**4**	**19.5**	**12.4**
Standard Deviation					**4.2**

DF Dataset					
Self-contained Units	CPDL	Number of			Time (min)
		M	V	IllD	
Missa7.4_Sanctus1	**no**	29	4	0	1
Missa7.4_Sanctus2	**no**	50	4	52	10
Missa9.6_AgnusII0	yes	35	5	14	7
Missa10.3_Credo2	yes	40	4	3	2
Missa10.3_Credo3	yes	64	4	17	30
Missa13.5_Agnus0	yes	30	4	6	6
Missa14.1_Kyrie0	yes	56	4	23	7
Missa15.1_Kyrie0	yes	44	4	0	1
Missa15.3_Credo1	yes	64	4	61	2
Missa16.4_Sanctus1	yes	22	4	2	1
Missa16.4_Sanctus3	yes	27	3	16	9
Piece25_Surrexit0	yes	15	4	13	6
Average		**39.7**	**4**	**17.3**	**6.8**
Standard Deviation					**8.0**

Table 1: Voice-alignment correction time for each piece in the No Dissonance Filter (NDF) Dataset and the Dissonance Filter (DF) Dataset. The pieces shaded in gray did not require correction. The *M* stands for measures, *V* for voices, *IllD* for illegal dissonances, and the *CPDL* column indicates if a modern transcription exists on CPDL.

[11] `https://github.com/martha-thomae/GuatC1/tree/Experiment/MPeditor_files/Score_files`

4.1 Discussion

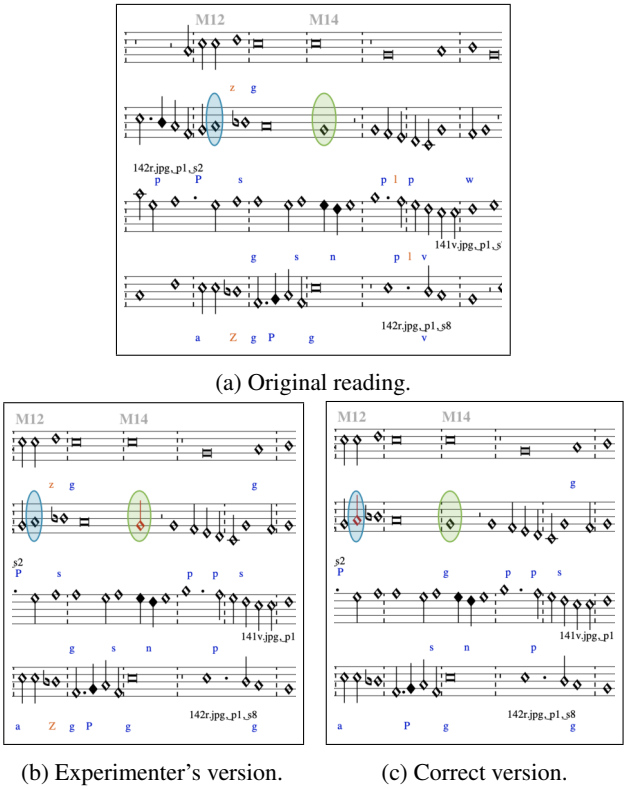
By analyzing the corrections made to the individual pieces and looking at their correction times, the following points become apparent:

1. **The use of the DF reduces the correction time.** As shown in Table 1, the use of the DF reduces the correction time from 12.4 to 6.8 minutes, almost halving it. However, the standard deviation of the DF Dataset is high compared to the one in the NDF Dataset. This is due to an outlier in the DF Dataset. While the correction time for all pieces in the DF Dataset is in the 1–10 minute range, the Missa10.3_Credo3 correction time is 30 minutes. Details about this mass will be provided later in point 4 below. Removing this outlier from the DF Dataset reduces the average correction time to 4.7 minutes (and the standard deviation is reduced from 8.0 to 3.4). Eliminating the outlier reduces the correction time with the DF to almost a third of the correction time for the NDF. Moreover, for pieces that have around the same amount of information—number of measures, voices, and illegal dissonances—and that did not require any corrections (see grey entries in Table 1), the reduction in time when using the DF is considerable. The pieces in grey with 26 and 29 measures took the experimenter 6 minutes to correct without the DF, and 1 minute with the DF. Similarly, the pieces with 48 and 44 measures took the experimenter 15 minutes without the DF and 1 minute with the DF, despite the fact that the latter (the piece in the DF data) has an extra voice.

2. **The use of the DF increases accuracy in the correction.** There were numerous cases where the experimenter made a rhythmic change too late. [12] An example is shown in Figure 3. Figure 3a shows the original manuscript reading of Missa15.3_Credo2. Looking at this scored-up version of the piece, the experimenter noticed that the cadence to G at the end of the example was not correct and knew that she had to cut a minim in the alto voice sometime before the cadence. Her correction, halving a semibreve in measure 14, is shown in Figure 3b. This change still resulted in an illegal dissonance, however, shown in Figure 3b by the orange "z" under the *Bb* in the bass (not shown in the experimenter's NDF dataset). The rhythmic change should have been made a few measures before (in measure 12), as shown in Figure 3c. While the experimenter's correction removes all following illegal dissonances (Figure 3b), it still leaves the ones preceding it. The DF would have shown her where to look—just before the first orange label—to change the note value. This change matches the CPDL correction.

3. **The DF not only aids in identifying scribal errors but also OMR errors that affect the voice alignment.** For pieces where MURET missed a note, or assigned the wrong value of a note or rest in the manuscript, the DF helped to identify the missing or incorrect symbol. [13] Although these pieces were part of the NDF Dataset (where the DF was not used during the experiment), applying the DF on the scored-up OMR file could have saved the experimenter some time as the orange labels help in noticing these errors. By finding the first orange label and checking the manuscript around that spot, the experimenter would have realized that a note/rest in the manuscript was missing (or had the wrong value) in the rendered file.

4. **Style issue – Eighteenth-century works.** The DF was not designed to handle eighteenth-century compositional style. Mass 10 is the only mass in the corpus that we know was composed in the eighteenth century. In many of its self-contained units, one can see the use of seventh chords and ties. [14] In this mass, the DF did not work well, because dissonances that are illegal in Renaissance style are legal in the eighteenth century. The experimenter pointed out a passage in Missa10.3_Credo3, which took her 30 minutes to correct. She invested a lot of time trying to figure out how



(a) Original reading.

(b) Experimenter's version.

(c) Correct version.

Figure 3: Missa15.3_Credo2 beginning at measure 12. The circled semibreve in measure 14 is the one halved by the experimenter, while the circled semibreve in measure 12 is the one that should have been halved. **Clefs from top to bottom:** G, G, suboctave G, suboctave G.

[12] This happened in Missa7.4_Sanctus2, Missa15.3_Credo2, and Missa16.4_Sanctus3.

[13] This happened in Missa9.2_Gloria1, Missa9.2_Gloria2, and Missa10.3_Credo1.

[14] Seventh chords are found in Missas10.1_Kyrie0 (measure 4), Missa10.3_Credo1 (measures 11, 12, and 39), Missa10.3_Credo3 (measure 20), and Missa10.5_AgnusI0 (measure 16). While ties are found in Missa10.3_Credo1 and Missa10.3_Credo3.

to get rid of the orange labels and finally left the passage as it was. This was the correct decision as this passage combines Baroque and Renaissance styles, with seventh chords (see Figure 4), such as the third inversion seventh chord over the red "D," shown by the orange "y."

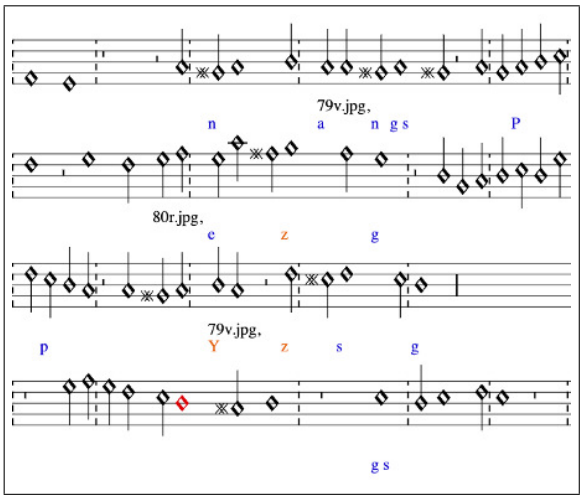


Figure 4: Passage where the experimenter invested a lot of time in Missa10.3_Credo3. It is a combination of Renaissance and Baroque style with fake suspensions and seventh chords. **Clefs from top to bottom:** G, suboctave G, suboctave G, F.

5. **False illegal dissonances.** Rearticulations, minims and semiminims doubling the pitch class of the agent, and extremely short notes like fusas are wrongly labelled as illegal dissonances. This is an issue to correct on the dissonance filter.

Finally, our methodology of re-uploading the file corrected during the experiment, activating the DF, and looking for illegal dissonances left allowed us to catch an error missed by both the experimenter and the CPDL transcriber. Moreover, our methodology of looking for imitative textures allowed us to easily correct this error (see Figure 5).

5. CONCLUSIONS

In this paper, we presented a MIR pipeline designed to obtain corrected symbolic scores for the pieces of a set of Guatemalan choirbooks, and we tested it on the first of these books (GuatC 1). We also presented a way to facilitate correcting the results of the last step of this pipeline, the automatic alignment of the mensural voices recognized by the OMR, conducted in the MP Editor. We used the dissonance filter from the humlib library within the MP Editor to highlight illegal dissonances according to the rules of Renaissance counterpoint. Our findings reveal that using the dissonance filter as a counterpoint error marker for the scoring up of Renaissance music facilitated the spotting of scribal errors and even some OMR errors that went undetected in the previous step of the MIR pipeline. Using the MP Editor and the dissonance filter, the user can see the scored-up OMR file, look for the first illegal dissonance

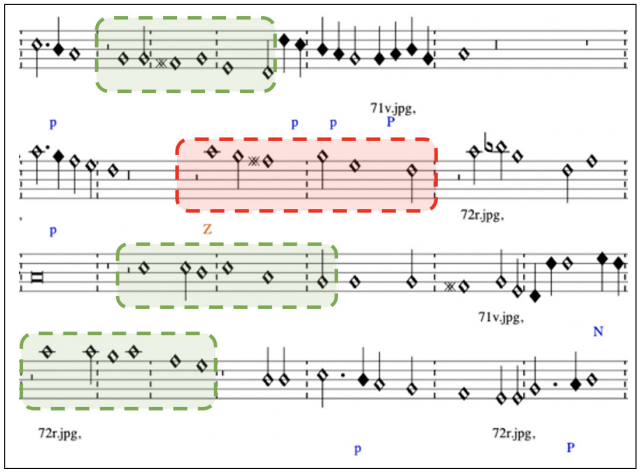


Figure 5: Experimenter and CPDL version of the Christe section of Missa10.1_Kyrie0 with the DF turned on. The alto's A marked by an orange Z label is an undetected error by the experimenter and the CPDL transcriber. Moving this note down to a G results in imitation of the other voices, marked in green boxes. **Clefs from top to bottom:** G, suboctave G, suboctave G, F.

(i.e., the first orange label in the MP Editor), and evaluate if there is an OMR or a scribal error at or before this point. The use of the DF reduces the correction time (to almost a third) and increases the accuracy of the correction process for pieces written in Renaissance style. Further improvements in the time and accuracy of the corrections could be obtained by improving the DF (i.e., resolving the issue of false illegal dissonances). The DF proved to be sensitive to changes in style, as shown in the eighteenth-century mass of the corpus that combines Baroque and Renaissance styles. Future work consists of extending the filter to work on non-Renaissance music.

In conducting this work, the first author has collaborated with the developers of MuRET, MP Editor, and humlib, improving these tools' support for mensural notation and allowing for their interoperability. Interested students and scholars can use the presented MIR pipeline to retrieve editorial music scores for other mensural sources semi-automatically. The OMR part of the proposed pipeline can be substituted by any other OMR tool for mensural notation (e.g., Aruspix) as long as its output conforms with the input expected by the MP Editor [23]. Although this experiment was small, it already showed promising results. The following steps would include replicating these findings with more musicologists and exploring other counterpoint errors (e.g., parallel perfect intervals).

6. ACKNOWLEDGEMENTS

This work draws on research supported by the Fonds de recherche du Québec – Société et culture (FRQSC, Bourse au doctorat en recherche, 13D - Musique, 2019-B2Z-261749) and the Social Sciences and Humanities Research Council of Canada (SSHRC). Special thanks to the developers of the various tools involved in the MIR pipeline,

David Rizo (of MuRET), Juliette Regimbal (of the MP Editor), and Craig Sapp (of humlib), as they assisted in the process of allowing for the interoperability of these tools. Thanks to Alexander Morgan, who developed the dissonance filter for the Josquin Research Project; to Geneviève Gates-Panneton, who was the experimenter and provided valuable feedback for this paper; and to Professor Peter Schubert, who served as an expert consultant for questions regarding modal counterpoint. Finally, thanks to Timothy de Reuse and Andres Lou for their help in editing this paper.

7. REFERENCES

[1] O. Morales Abril, "Música local y música foránea en los libros de polifonía de las Catedrales de Guatemala y México," *Jahrbuch für Renaissancemusik: Troja*, vol. Vokalpolyphonie zwischen Alter und Neuer Welt: Musikalische Austauschprozesse zwischen Europa und Lateinamerika im 16. und 17. Jahrhundert, no. 14, pp. 95–124, 2015.

[2] M. E. Thomae, J. E. Cumming, and I. Fujinaga, "Digitization of choirbooks in guatemala," in *Proceedings of the 9th International Conference on Digital Libraries for Musicology (DLfM)*. Prague, Czech Republic: Association for Computing Machinery, 2022, pp. 19–26. [Online]. Available: https://doi.org/10.1145/3543882.3543885

[3] Y.-H. Huang, X. Chen, S. Beck, D. Burn, and L. Van Gool, "Automatic handwritten mensural notation interpreter: from manuscript to MIDI performance," in *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, Málaga, Spain, 2015, pp. 79–85.

[4] D. Pujol, "Polifonía española desconocida conservada en el Archivo Capitular de la Catedral de Guatemala y de la Iglesia parroquial de Santa Eulalia de Jacaltenango," *Anuario Musical*, vol. 20, pp. 3–10, 1965.

[5] R. Stevenson, *Renaissance and Baroque Musical Sources in the Americas*. General Secretariat, Organization of American States, 1970.

[6] R. J. Snow, *A New-World Collection of Polyphony for Holy Week and the Salve Service: Guatemala City, Cathedral Archive, Music MS 4*, ser. Monuments of Renaissance Music. Chicago: University of Chicago Press, 1996, no. 9.

[7] O. Morales Abril, "Catálogo de los acervos musicales del Archivo Histórico Arquidiocesano de Guatemala," N.D., unpublished.

[8] D. Lehnhoff, *Rafael Antonio Castellanos: vida y obra de un músico guatemalteco*. Universidad Rafael Landívar, Instituto de Musicología, 1994. [Online]. Available: http://www.url.edu.gt/PortalURL/Biblioteca/Contenido.aspx?o=6708&s=49

[9] M. E. Thomae, J. E. Cumming, and I. Fujinaga, "The mensural scoring-up tool," in *Proceedings of the 6th International Workshop on Digital Libraries for Musicology (DLfM)*. The Hague, Netherlands: ACM, pp. 9–19.

[10] I. Fujinaga, "Single interface for music score searching and analysis (SIMSSA) project: Optical music recognition workflow for neume notation," in *Proceedings of the Computers and the Humanities Symposium (JinMonCom)*. Osaka, Japan: Information Processing Society of Japan, 2019, pp. 281–286.

[11] I. Fujinaga and A. Hankinson, "Single interface for music score searching and analysis (SIMSSA)," in *Proceedings of the 1st International Conference on Technologies for Music Notation & Representation (TENOR)*, Paris, France, 2015, pp. 109–15.

[12] G. Burlet, A. Porter, A. Hankinson, and I. Fujinaga, "Neon.js: Neume editor online." in *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR)*, Porto, Portugal, 2012, pp. 121–126.

[13] J. Regimbal, Z. McLennan, G. Vigliensoni, A. Tran, and I. Fujinaga, "Neon2: A Verovio-based square-notation editor," in *Proceedings of the Music Encoding Conference (MEC)*, Vienna, Austria, 2019.

[14] C. Wick and F. Puppe, "OMMR4all – A semiautomatic online editor for Medieval music notations," in *Proceedings of the 2nd International Workshop on Reading Music Systems (WoRMS)*, Delft, Netherlands, 2019, pp. 31–34.

[15] L. X. Pugin, "Editing Renaissance music: The Aruspix project," *Beihefte zu editio*, pp. 64–103, 2009.

[16] J. M. Iñesta, D. Rizo, and J. Calvo-Zaragoza, "MuRET as a software for the transcription of historical archives," in *Proceedings of the 2nd International Workshop on Reading Music Systems (WoRMS)*, Delft, Netherlands, 2019, pp. 12–15.

[17] D. Rizo, J. Calvo-Zaragoza, and J. M. Iñesta, "MuRET: A music recognition, encoding, and transcription tool," in *Proceedings of the 5th International Conference on Digital Libraries for Musicology (DLfM)*. Paris, France: ACM, 2018, pp. 52–56.

[18] J. M. Iñesta, P. J. P. de León, D. Rizo, J. Oncina, L. Micó, J. R. Rico-Juan, C. Pérez-Sancho, and A. Pertusa, "Hispamus: Handwritten Spanish music heritage preservation by automatic transcription," in *Proceedings of the 1st International Workshop on Reading Music Systems (WoRMS)*, Paris, France, 2018, pp. 17–18.

[19] D. Rizo, N. P. León, and C. S. Sapp, "White mensural manual encoding: from Humdrum to MEI," *Cuadernos de Investigación Musical*, no. 6, pp. 373–393, 2018.

[20] P. Roland, A. Hankinson, and L. Pugin, "Early music and the Music Encoding Initiative," *Early Music*, vol. 42, no. 4, pp. 605–11, November 2014.

[21] K. Desmond, A. Hankinson, L. Pugin, J. Regimbal, C. S. Sapp, and M. E. Thomae, "Next steps for Measuring Polyphony – A prototype editor for encoding mensural music [poster]," in *Proceedings of the Music Encoding Conference (MEC)*, E. De Luca and J. Flanders, Eds. Boston, MA: Humanities Commons, 2020, pp. 121–124.

[22] A. Plaksin and D. Lewis, "Merit: An interactive annotation tool for mensural rhythms," in *Proceedings of the 9th International Conference on Digital Libraries for Musicology (DLfM)*. Prague, Czech Republic: Association for Computing Machinery, 2022, pp. 55–59. [Online]. Available: https://doi.org/10.1145/3543882.3543888

[23] K. Desmond, L. Pugin, J. Regimbal, D. Rizo, C. S. Sapp, and M. E. Thomae, "Encoding polyphony from Medieval manuscripts notated in mensural notation [panel]," in *Proceedings of the Music Encoding Conference (MEC)*, S. Munich and D. Rizo, Eds. Alicante, Spain: Humanities Commons, 2021.

[24] P. Schubert, *Modal counterpoint, Renaissance style*. Oxford University Press, USA, 2008.

[25] M. E. Thomae, "The Guatemalan choirbooks: Facilitating preservation, performance, and study of the colonial repertoire," in *Christian Music Traditions in the Americas*, A. Shenton and J. Smolko, Eds. New York: Rowman & Littlefield, 2021.

A DATASET OF SYMBOLIC TEXTURE ANNOTATIONS IN MOZART PIANO SONATAS

Louis Couturier[1] **Louis Bigo**[2] **Florence Levé**[1,2]
[1] MIS, Université de Picardie Jules Verne, Amiens, France
[2] CRIStAL, UMR 9189 CNRS, Université de Lille, France
{louis.couturier, florence.leve}@u-picardie.fr, louis.bigo@univ-lille.fr

ABSTRACT

Musical scores are generally analyzed under different aspects, notably melody, harmony, rhythm, but also through their texture, although this last concept is arguably more delicate to formalize. Symbolic texture depicts how sounding components are organized in the score. It outlines the density of elements, their heterogeneity, role and interactions. In this paper, we release a set of manual annotations for each bar of 9 movements among early piano sonatas by W. A. Mozart, totaling 1164 labels that follow a syntax dedicated to piano score texture. A quantitative analysis of the annotations highlights some characteristic textural features in the corpus. In addition, we present and release the implementation of low-level descriptors of symbolic texture, that are preliminary experimented for textural elements prediction. The annotations and the descriptors offer promising applications in computer-assisted music analysis and composition.

1. INTRODUCTION

1.1 Texture and symbolic texture

Musical texture generally refers to two distinct levels of abstraction used to describe musical content [1]. On the one hand, *sound* related texture, that can be referred to as *orchestral texture*, results from orchestration, instrumentation and timbral characteristics of instruments and performances. On the other hand, *symbolic texture*, or *compositional texture*, results from the organization of notes, chords and voices in the musical score. Naturally, these notions are closely related, and Hérold studies both textural and instrumental factors of timbre [2], highlighting the impact of compositional texture on the final sound field. Symbolic texture, which is the focus of the dataset presented in this paper, can be described through high level musical concepts such as layer separation, diversity of sonic activities, layer roles, note density and interactions [3–7]. For instance, in piano music, an accompanying chord sequence can be performed in various ways, each one being identified by a specific texture contributing to a stylistic identity. Symbolic texture stands at the center of the compositional process and can sometimes be understood as a notion of style [8]. Musical style is however commonly associated with a whole piece [9] or section whereas the notion of symbolic texture considered in this work tends to describe much shorter time spans in the musical score.

1.2 Related work

Computational methods to analyze symbolic texture have been elaborated in various musical styles including classical string quartet music [10] and modern popular guitar music [11]. Given the wide range of playing modes offered by the instrument, piano music brings unprecedented challenges for the task of in-score texture identification. The present work builds on a recent formal syntax elaborated to classical piano music modeling [7]. As a crucial parameter of musical style, symbolic texture has also recently raised an important attention in the tasks of music generation [12] and style transfer [8,13], where musical texture is efficiently learned by deep neural networks but with limited perspectives of explicit categorization and musicological interpretation. Alternatively, the dataset proposed in the present work promotes a pedagogical and transparent expression of symbolic piano texture, aiming at facilitating its use to design computational tools intended to assist music pedagogy, analysis and composition.

The MIR – Music Information Retrieval – community dedicates an important part of its work effort in building musical expert annotations accompanying music datasets to facilitate training and evaluation of computational models. The classical repertoire, in particular its piano music, gives raise to a number and variety of annotation needs given the richness of its musical language. Key regions, cadences, phrases and harmonies were annotated in the Mozart piano sonatas from the New Mozart Edition (Neue Mozart Ausgabe) [14]. As other representative initiatives, the TAVERN dataset includes harmony and phrase annotations on Mozart and Beethoven's piano variations [15] and the Fugue dataset includes form annotations specific to this genre [16]. Beyond piano music, classical string quartets have also raised a number of key, harmony and structure annotation efforts including repertoires of Haydn [17], Mozart [18] and Beethoven [19, 20]. Although a corpus has been proposed in [21], symbolic texture has still rarely

© L. Couturier, L. Bigo, and F. Levé. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** L. Couturier, L. Bigo, and F. Levé, "A Dataset of Symbolic Texture Annotations in Mozart Piano Sonatas", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

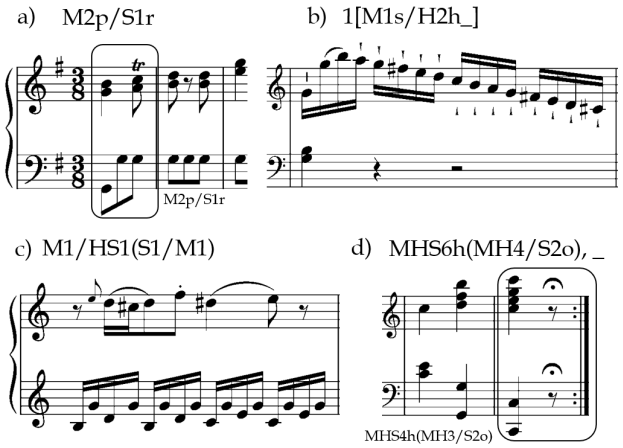


Figure 1. Examples of texture annotations using the syntax defined in [7]. a) K. 283.III m.1-2: the melody is doubled at the third (`M2p`), and moves in parallel motions (`p`) over a static layer with repeated notes (`S1r`); b) K. 279.I m.35: in addition to a melodic layer with scale motive (`M1s`), a short (sparse, '_') homorhythmic layer appears, without affecting the vertical global density whose value remains 1 on the overall measure (`1[...]`); c) K. 279.I m.6: a typical example of Alberti bass, described as `HS1` harmonic and static layer of density equal to one, and a possible division into two sublayers; d) K. 279.III m.157-158: a concluding formula with melodic (horizontal movements), harmonic (verticality) and static (regularity and emphasis), high vertical density, with an octave motion (`o`) optionally detailed in the sublayer decomposition. A comma separates the dense texture from the contrasting silence ('_') in the last measure.

been the subject of consequent corpus annotations, due to the variety of musical features it involves and the rarity of formal specification.

1.3 Motivation and outline

In order to provide to the community consistent data to study symbolic texture in Western classical piano music, we release a dataset of manual annotations describing symbolic texture at each bar of 9 movements of Mozart Piano Sonatas, totaling a set of 1164 annotated measures. The corpus and the annotation process are detailed in Section 2. Section 3 provides statistics on the textural labels annotated in the dataset. Finally Section 4 presents preliminary results of texture prediction by a machine learning model.

2. PRESENTATION OF THE DATASET

2.1 Syntax for the annotations of symbolic texture

We follow the syntax proposed in [7] to describe textural properties of piano music. More precisely, the texture of a score region is annotated by a text label expressing a set of features with the following conventions:

Diversity. The overall texture is split into independent textural layers that are described individually, separated by a `/`. They are ordered by descending register.

Example: The label `M1/H2` includes two layers, as in examples a, b and c in Figure 1.

Function. The *function* of each layer is expressed with a combination of three specific labels being `M` for *melodic* function, `H` for *harmonic* function, and `S` for *static* function like pedals and ostinati.

Example: The label `M1/H2` includes one layer with a melodic function and one layer with an harmonic function. As an other example, the label `HS1` includes one single layer having both a harmonic and a static function like a persistent arpeggio.

Density. The *density* of a layer, also called *thickness* [1], corresponds to the number of voices it includes, expressed by an integer right after the function.

Example: The label `M1/H2` includes one melodic layer with one voice and one harmonic layer including two voices.

Global density. The *global density* of a region corresponds to its global number of voices and is indicated before brackets surrounding the whole label. It is an approximation of the average number of notes perceived simultaneously. In most cases, the global density is equal to the sum of the layer's density, in which cases its notation is optional because redundant.

Example: The label `3[M1/H2]` indicates a global density of 3 and can be simplified into `M1/H2`. However, the label `2[M1/H2]`, which can occur in certain types of sparse regions, cannot be simplified.

Internal organization of a layer. Additional elements can indicate the presence of relationships between voices: homorhythmy (`h`), parallel motion (`p`), octave (`o`) or characteristic musical figures: sustained notes (`t`), repeated notes (`r`), oscillations (`b`) or scales (`s`).

Example: The label `M2p/HS3hr` can describe a melody doubled at the third accompanied by repeated three-note chords.

Sublayer decomposition. Each layer can optionally be decomposed into sublayers notated between parentheses.

Example: In the label `M1/HS1(S1/M1)` (see Figure 1.c), the second layer `HS1` itself includes one *static* sublayer and one *melodic* sublayer enabling for example the expression of a single voice accompaniment consisting in an alternation between a single repeated pitch and a moving melodic line.

Sparsity. When a layer does not last during the full measure (as in Figure 1.b), or has too low horizontal density, it is considered as *sparse* and notated with '_'. This symbol is also used to annotate empty bars.

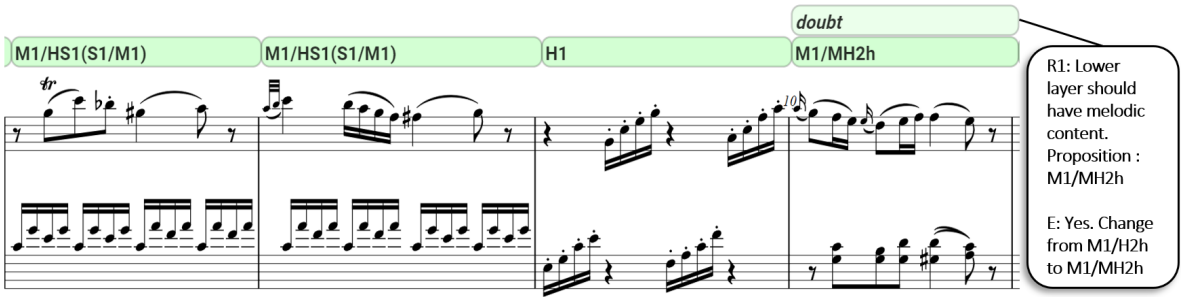


Figure 2. Excerpt of the first movement of Mozart's Piano Sonata n.2 (K. 279, m.7-10) as seen on the web application used to annotate and review the dataset. Each annotated label can be put in doubt by the reviewer: once a `feedback` is opened over one given measure, comments can be added to exchange with the annotator. Feedbacks have three states: *raised* (in white – just submitted by a reviewer), *in conflict* (red – waiting for a consensus) and *resolved* (green, as in the figure).

Corpus ID: corpus:MIR:mozartpianosonatas:texture:2022:version1.0
Raw Corpus **Definition:** existing corpus of real symbolic items. Digital Scores of Mozart's Piano Sonatas following the *Neue Mozart-Ausgabe (NME)* from Mozart Annotated Sonatas [14]. 9 annotated movements in 3 sonatas (K. 279, K. 280, K. 283). **Sampling:** Western classical style piano music, available data, all movements in Sonata Form. **Type of media diffusion:** original files in .mscx and .tsv (Tab Separated Values), annotations in .tsv, .txt and .dez (Dezrann format).
Annotations **Origin**: manual **Concepts definition**: concepts and syntax for texture annotations from [7]. **Annotation rules**: annotation guidelines provided with the dataset. **Annotators**: one annotator (expert knowledge in music and piano), 2 reviewers with the same background. **Validation/reliability**: review, one for each movement. **Annotation tools**: Dezrann web-interface [24].
Documents and Storing **Identifier and storage**: Köchel number, scores from NMA (Neue Mozart Ausgabe) reference edition, Git repository of the dataset by [14] in musescore and TSV format, annotation files on a Git repository [1] and online on Dezrann [2].

Table 1. Description of the annotated dataset.

2.2 Annotated corpus

Following the syntax presented in Section 2.1, we release manual annotations of texture at the bar level in the 3 movements of the 3 sonatas K. 279, K. 280 and K. 283 by W. A. Mozart, totaling a set of 1164 bar annotations in the 9 movements. Those early sonatas were all composed in the end of the year 1774, which presumes a style consistency and limits the shift of compositional practice that could be induced by the evolution of piano manufacture. Moreover, these movements present an interesting diversity in tonality, rhythmic signature and tempo, while covering a large variety of textures. All those movements are in Sonata Form, which opens perspectives to pursue research on the links between texture and form [22].

Table 1 synthesizes the properties of the dataset following the conventions proposed in [23].

2.3 Annotation procedure

2.3.1 Granularity of the annotations

Textural segments can have highly variable duration, which tends to complicate their annotation. As a compromise to facilitate statistics and computational processing of the annotations, we propose in this work to annotate a single texture label for each bar, as indicated in the proposition of syntax [7]. In some cases where the texture strongly shifts in the middle of a bar, a comma (`,`) is used to segment the label in two parts. In particular, this situation may occur on boundaries between musical phrases. By contrast, a texture can also remain unchanged over several consecutive bars, which will lead to a repetition of a same texture label over these bars.

2.3.2 Annotation and reviewing methodology

This section describes our annotation protocol which was inspired by [25] and illustrated in Figure 3. One expert E annotated all the measures of the 9 movements with texture labels on Dezrann [2], a web platform dedicated to the annotation of musical analyses on scores [24]. The annotations were committed to a Git repository to trace the different stages of annotation and reviews. A Python script [3] was used to systematically check that the labels were consistently formed before to proceed to the review phase, relieving the reviewers from syntax checking to concentrate on the textural decomposition. Two reviewers R_1 and R_2 (distinct from the expert), with strong musical background and piano music knowledge, as well as knowledge of the texture syntax used, reviewed those annotations on Dezrann. They added a `feedback` label each time they disagreed or were unsure of the annotator label choice, detailing the reasons of the doubt and possibly providing a new proposition for the label. The expert E then studied all the doubts, resolved the obvious ones and discussed with the two reviewers to reach a consensus on the remaining labels.

Over the 1164 initial annotations of the annotator, 31% raised a reviewer feedback and 22% (256) were updated in

[1] `http://algomus.fr/data`
[2] `http://dezrann.net`
[3] `http://algomus.fr/code`

the end. This substantial number emphasizes the importance of the reviewing phase as well as the complexity of texture annotation due to its variety and the high expressivity of the syntax. Besides correcting possible omissions or typos in the labels, the majority of conflicts ensured a more homogeneous and precise use of the syntax in the whole dataset – in each movements and between them. Hence, most of the discussions between the annotator and the reviewers involved several measures: consecutive ones sharing similar texture, cyclic resurgence of thematic materials (for example in both exposition and recapitulation of sonata form) or comparable textures across pieces. Note that in some cases (like in music analysis in general), it is possible to have different interpretations of the textural layers in a musical passage. The goal of the review here was not to blur those distinct possible analyses but to provide consistent labels that made sense for all.

2.3.3 Syntax knowledge and annotation reproducibility

The annotator followed the syntax recalled in Section 2.1 with the help of a catalog of common textural configurations provided in [7]. As shown in Section 2.3.2, the texture of a bar can sometimes be interpreted in various ways, leading to different labels. The consideration of the neighboring bars can also impact the texture estimation. To limit inconsistency in the labels, we provide annotation guidelines in addition to the dataset. These guidelines typically indicate the order of consideration of the different textural elements when estimating a label. They aim at encouraging a sense of normalization in the annotations, although such a normalization has not been strictly formalized. This aims at helping futur annotators to understand and reproduce the annotation labels. Since the web platform used allows to share several analyses of the same musical piece, divergent labels could be further used ultimately to propose alternative analyses and diversify the dataset for machine learning applications.

3. CORPUS ANALYSIS

3.1 Common textures and layers

Among the 1164 annotated bars, 16.6% include two labels instead of one due to a substantial shift of texture in the middle of the bar (see Section 2.3.1), resulting in 1357 textural configurations. From this set, we can extract 2317 non-empty textural layers. Among the most common layers, we find simple single-voice melodies (`M1`) which totaled 24.9% of all written layers, followed by melodies with scale motives (`M1s`, 7.0%, see Figure 1.b) and with parallel motions (`M2p`, 3.7%), generally doubled at the third (see Figure 1.a) or at the sixth.

Another important family of textural layers encompasses repetitive accompaniment figures like arpeggios, in which notes of the current harmony are played one by one (`HS1`, here). A famous example is the *Alberti bass*, an idiomatic pattern which alternates notes in low-high-middle-high order (see Figure 1.c and the first half of Figure 2) and is typical of Mozart's piano music. The combination

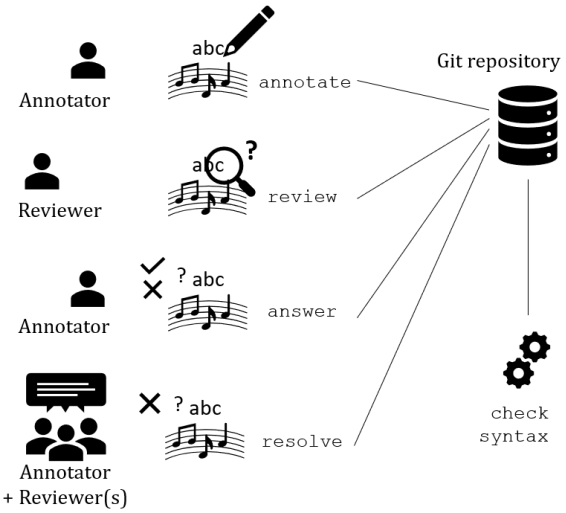


Figure 3. Annotation protocol. Once each measure of a given piece has been annotated (1), these labels are read by a reviewer (2) who can emit a 'doubt' on questionable annotations, along with a justification or a proposal of correction. Then, the annotator treats each doubt, by either resolving it or argumenting towards another solution (3). Finally, conflicts that remained unresolved are discussed between the annotator and reviewer(s) until a consensus is found (4). At each step of the process, the correctness of the syntax is automatically checked and the changes are committed to a Git repository.

of basic melody `M1` and single-voice accompaniments `HS1` is the most represented in the dataset, with 7.3% of all annotated labels.

3.2 Textural elements in labels

Symbolic texture labels are highly expressive and rich in information. We call *textural elements* the set of unary attributes, each one notated with a dedicated character, that can appear in our labels. We consider that a bar includes a textural element if it appears at least once in the whole label. For example, a bar includes the textural element `h` (homorhythmy), if the whole texture, or at least one of its layers, is homorhythmic (and therefore annotated with `h`). In the case of two successive textural configurations described in one label (`A, B`), the presence of the element on one side is sufficient to consider its presence in the whole label. The textural elements are all listed in Table 2.

Unsurprisingly, a vast majority of the dataset (94.6%[4]) contains a layer with at least one melodic function (`M`). Following on function combinations, 20.2% of labels only contain melodic layers (presence of the element `M` and absence of `H` and `S`). The coincident presence of the three functions concerns 23.5% of the labels and the most common combination is made of melodic and/or harmonic layers without any static (`S`) ones (34.1%) as it is the case in a typical melody plus accompaniment section. Finally, the proportion of measures with harmonic layers (textural

[4] The proportions of all annotated textures and textural elements are provided in the dataset repository: `http://algomus.fr/data`.

element `H`) varies between annotated movements, notably according to the tempo: this percentage is 26% higher in slower movements (the second of each full sonata) than in the average of the 6 others (84.4% versus 58.2%).

3.3 Density and diversity

We compute the diversity and the global density of each annotated textural configuration. As detailed in Section 2.1, the diversity corresponds to the number of stacked layers while the global density corresponds to the approximate number of monophonic voices that can be heard simultaneously. Figure 4 puts in relation these two values for the set of annotated labels. The diagonal is assimilated to polyphony where each voice sounds like a new distinct musical idea, an individual layer. On the contrary, the bottom row, with diversity of value 1, corresponds to monophonic texture; thus, any note is contributing to the same unique musical entity. This case is common at the end of structural parts and cadences, typically in homorhythmy (see Figure 1.d). Homophonic textures, for example combinations of a main melody and accompaniment, are found between these two areas. Among the annotated non-empty textural configurations, 29.2% lay in the $2|2$ combination of diversity|density, including the common case of melody and singe-voice accompaniment, presented earlier. Textures in $2|3$ and $2|4$ (three or four voices merged into two layers) illustrate denser homophonic variants, in which a melody may be doubled at the third or at the sixth, or an accompaniment made of simultaneous notes (in homorhythmy). The combinations above the diagonal correspond to situations where the number of layers is higher than the number of voices ($2|1$ or $3|2$). This can happen when two distinct lines alternate in antiphony (call & response): several textural layers are perceived whereas their notes do not overlap.

The diversity, which corresponds to the number of layers, rarely exceeds 2. However, Figure 4 shows that a systematic separation into two layers, which follows an intuitive organization of the score content between the two hands of the pianist, would not be sufficiently representative of the corpus.

In other repertoires, the number of voices – the vertical density – could be globally higher. Bach three-voice inventions would be mainly categorized as a continuous $3|3$ polyphony, and some works of the Romantic Era including very large chords would be extremely dense vertically. Hence, it is easy to imagine that this textural space, similarly to the one described in [4], is prone to convey strong stylistic content. Moreover, the evolution of texture throughout a single piece of music could be modeled as a trajectory in this space, which offers promising future analytical perspectives.

4. APPLICATION: PREDICTION OF TEXTURAL ELEMENTS

We present in this section preliminary experiments of texture prediction.

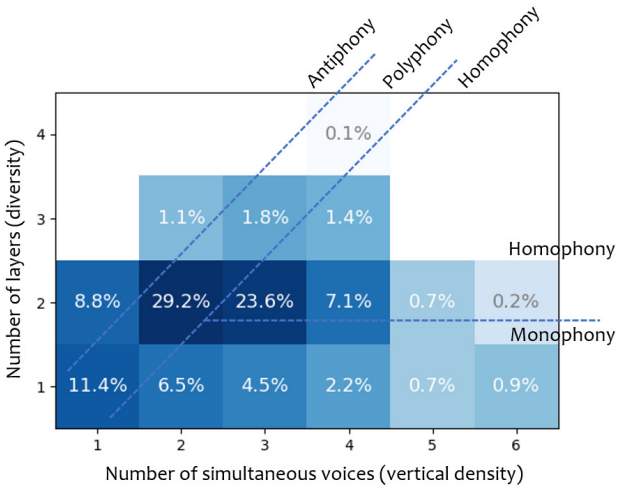


Figure 4. Repartition of textural configurations of the dataset according to their density and diversity. We divided this textural space into areas corresponding to the main texture types evoked in Section 3.3: monophony, homophony, polyphony and antiphony [4, 26]. Empty textures (silence) are not taken into account in this figure.

4.1 Symbolic textural descriptors

To facilitate the prediction of textural elements in musical scores, we propose a set of 62 high level features computed on the raw musical score. Some of these were inspired from previous works including [27] and were complemented with original ones especially elaborated for the modeling of high level textural concepts in polyphonic piano scores. The Python implementation of these descriptors is publicly released [5] . Provided code enables to compute the descriptors in *Stream* objects from the Music21 Python library [28] as well as note lists directly stored in TSV (Tab Separated Values) files, as in [14] [6] .

The selected descriptors are computed at different levels of the musical content: pitches, onsets or temporal slices. Temporal slicing is equivalent to the action of `Stream.chordify()` method in the Music21 library, also called salami-slicing [29]. For elements followed by the symbol '*', we compute average, deviation, median, and extremal values.

Pitches. We compute the total number of distinct pitches, the number of pitch classes, the notes duration*, the notes MIDI pitch* and an indicator of pitch reuse.

Onsets. We compute the total number of onsets, the number of simultaneous pitches by onset*, the regularity*, the harmonic intervals and the number of gaps* between pitches – where a *gap* is detected when the interval between two simultaneous notes is larger than a fourth [3].

Slices. We compute the number of slices, the number of simultaneous pitches*, the number of gaps*, the distance pitch ambitus*, the proportion of consonnant intervals (minor and major third, perfect fourth and perfect fifth), the proportion of rests and the longest rest.

[5] `http://algomus.fr/code`

[6] See `https://github.com/DCMLab/mozart_piano_sonatas/tree/main/notes`

Textural element	Log. Reg.	Random	All True
M (melodic)	0.912	0.665	0.973
H (harmonic)	0.744	0.545	0.799
S (static)	0.616	0.457	0.599
h (homorhythmy only)	0.673	0.396	0.453
p (parallel motions)	0.572	0.315	0.401
o (octave motions)	0.538	0.211	0.244
h+ (h or p or o)	0.810	0.538	0.708
p+ (p or o)	0.602	0.393	0.479
s (scale motives)	0.363	0.282	0.332
t (sustained notes)	0.669	0.161	0.193
b (oscillations)	0.098	0.116	0.103
r (repeated notes)	0.501	0.183	0.200
_ (sparsity)	0.587	0.258	0.306
, (sequential)	0.520	0.198	0.291

Table 2. F1-scores of the logistic regression models for the prediction of each textural element, compared to – respectively – a uniform random model and a model that always predict the presence of the textural element. F1-scores are averaged on the 9 folds of the cross-validation.

4.2 Description of the models

Different machine learning models were compared to predict the presence of textural elements presented in Section 3.2 from the set of 62 descriptors detailed in Section 4.1. Descriptor values are given to the model as vectors of 62 floats extracted from each measure. The prediction of the presence of each textural element was formulated as a binary classification task leading to a dedicated model for each of the 14 textural elements. Whereas modern data-driven machine learning approaches tend to favor neural networks for their power of abstraction, we stick to simple Logistic Regression models applied on preprocessed high level features as they seemed more adapted to the limited size of our training set and are more easily interpretable. We used Scikit-learn [30] implementation with L-BFGS solver, a maximum of 100 iterations and L2-regularization. Decision Tree classifiers and Support Vector Machine with linear kernel were also tested without significant improvements. In all our models, output classes weights are balanced with respect to their proportion in the dataset. The evaluation criterion is the F1-score, using cross-validation with *leave-one-piece-out* strategy to avoid overfitting due to similarities and repetitions inside movements.

4.3 Results and interpretations

The results are presented in Table 2. We observe that the presence of melody (`M`) is very well detected (F1-score of 0.912), despite being highly unbalanced in the dataset (94.6% of labels contains at least one layer with a melodic function). Harmonic or static functions are quite more difficult to predict. They were also more difficult to annotate: the determination of these functions also involves a part of subjectivity, despite the efforts made in the reviewing process to ensure the consistency of annotations.

The predictions of notes simultaneities and semblant motions (`h`, `p`, `o`) obtain fair results. We could have expected better from them, as well as for `t` (sustained notes) and `r` (repeated notes), since they only involved rare divergences during reviews, their determination being more straightforward. Furthermore, they are closer to the use of defined descriptors – notably those which are related, for instance, to the number of notes played at the same time or the presence of certain harmonic intervals in onsets or slices. The success of predicting `h+` compared to specific relationships `h`, `p` or `o` is meaningful: considering the fact that parallel motions are specific cases of homorhythmy, it seems more practical and sound to focus on this more general case.

The poor results of models to predict oscillation `b` can be justified by the limited proportion of positive examples in the dataset (around 5.1%). Finally, the difficulty of annotating scale motives (`s`) in practice is reflected in their prediction result (F1-score of 0.36). From slow descending sequences of neighboring pitches to cells of short-duration notes that share the same repeated contour: many variants of this element can be found in the corpus, some of which were making consensus difficult to reach between annotator and reviewers. This issue can be adressed in the definition of the syntax as the need to refine the criteria and guidelines for the annotation of targeted concepts. The scale patterns sometimes span over wider regions than the duration of a bar, making the descriptors inadequate in this case. Providing a larger context to our models therefore appears as a promising perspective to improve the detection of this textural element.

5. CONCLUSION AND FURTHER WORKS

We provide an open dataset of texture annotations that specifically handles piano classical music. This dataset contains annotations of 9 movements of Mozart's piano sonatas, all in Sonata Form, and annotation guidelines are provided in order to allow for an easy extension of the dataset by the community. It could be completed with other movements of Mozart's piano sonatas, but also with more diverse piano music from the Classical Era. An extension of the texture syntax to consider specific textures from other periods would also certainly bring new insights for the study of composition styles.

The dataset can be used for musicological purposes, for instance to study the links between texture and a variety of annotations – harmony, modulations, cadences, phrases and section boundaries – that are common subjects of interest in the MIR community. It also opens perspectives in MIR for computer-aided analysis and composition. As a first application, we implemented a set of textural descriptors and used them for the prediction of textural elements, obtaining encouraging results. An in-depth analysis of the relations between the textural descriptors and textural elements could help improving the prediction, especially on harmonic, static layer detection, or scale motives. This work will also allow to study the evolution of texture in the musical pieces and to better integrate this dimension for automatic generation of music following textural scenarios.

6. ACKNOWLEDGEMENT

We would like to thank Nathalie Hérold for fruitful discussions during the initial phase of the project. We also thank the Algomus team and the anonymous reviewers for their helpful feedback. This research is partly funded by Région Hauts-de-France.

7. REFERENCES

[1] D. Moreira, "Textural design: a compositional theory for the organization of musical texture," Ph.D. dissertation, Centro de Letras e Artes, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro), 2019.

[2] N. Hérold, "Timbre et analyse musicale : les possibilités d'intégration du timbre dans l'analyse formelle des œuvres pour piano du dix-neuvième siècle," in *L'interprétation musicale*. Delatour France, 2012, pp. 79–103.

[3] Q. R. Nordgren, "A measure of textural patterns and strengths," *Journal of Music Theory*, vol. 4, no. 1, pp. 19–31, 1960.

[4] D. Huron, "Characterizing musical textures," in *International Computer Music Conference (ICMC 1989)*, 1989, pp. 131–134.

[5] B. Duane, "Texture in eighteenth- and early nineteenth-century string-quartet expositions," Ph.D. dissertation, Northwestern University, 2012.

[6] C. P. d. Silva, "Por uma definição unificada de textura musical," Master's thesis, Escola de Música, Universidade Federal da Bahia, 2018.

[7] L. Couturier, L. Bigo, and F. Levé, "Annotating Symbolic Texture in Piano Music: a Formal Syntax," in *Sound and Music Computing Conference (SMC 2022)*, 2022.

[8] O. Cífka, U. Şimşekli, and G. Richard, "Groove2Groove: one-shot music style transfer with supervision from synthetic data," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2638–2650, 2020.

[9] C. Weiß, M. Mauch, S. Dixon, and M. Müller, "Investigating style evolution of western classical music: A computational approach," *Musicae Scientiae*, vol. 23, no. 4, pp. 486–507, 2019.

[10] L. Soum-Fontez, M. Giraud, N. Guiomard-Kagan, and F. Levé, "Symbolic textural features and melody/accompaniment detection in string quartets," in *International Symposium on Computer Music Multidisciplinary Research (CMMR 2021)*, 2021.

[11] D. Régnier, N. Martin, and L. Bigo, "Identification of rhythm guitar sections in symbolic tablatures," in *International Society for Music Information Retrieval Conference (ISMIR 2021)*, 2021.

[12] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[13] J. Zhao and G. Xia, "Accomontage: Accompaniment arrangement via phrase selection and style transfer," *arXiv preprint arXiv:2108.11213*, 2021.

[14] J. Hentschel, M. Neuwirth, and M. Rohrmeier, "The annotated mozart sonatas: Score, harmony, and cadence," *Transactions of the International Society for Music Information Retrieval*, vol. 4, no. 1, 2021.

[15] J. Devaney, C. Arthur, N. Condit-Schultz, and K. Nisula, "Theme and variation encodings with roman numerals (tavern): A new data set for symbolic music analysis," in *International Society for Music Information Retrieval Conference (ISMIR 2015)*, 2015.

[16] M. Giraud, R. Groult, E. Leguy, and F. Levé, "Computational fugue analysis," *Computer Music Journal*, vol. 39, no. 2, 2015.

[17] D. R. W. Sears, M. T. Pearce, W. E. Caplin, and S. McAdams, "Simulating melodic and harmonic expectations for tonal cadences using probabilistic models," *Journal of New Music Research*, vol. 47, no. 1, pp. 29–52, 2017. [Online]. Available: https://doi.org/10.1080/09298215.2017.1367010

[18] P. Allegraud, L. Bigo, L. Feisthauer, M. Giraud, R. Groult, E. Leguy, and F. Levé, "Learning Sonata Form Structure on Mozart's String Quartets," *Transactions of the International Society for Music Information Retrieval*, vol. 2, no. 1, pp. 82–96, 2019.

[19] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, "The Annotated Beethoven Corpus (ABC): A dataset of harmonic analyses of all Beethoven string quartets," *Frontiers in Digital Humanities*, vol. 5, p. 16, 2018. [Online]. Available: https://www.frontiersin.org/article/10.3389/fdigh.2018.00016

[20] D. Tymoczko, M. Gotham, M. S. Cuthbert, and C. Ariza, "The romantext format: A flexible and standard method for representing roman numeral analyses," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[21] M. Giraud, F. Levé, F. Mercier, M. Rigaudière, and D. Thorez, "Towards modeling texture in symbolic data," in *International Society for Music Information Retrieval Conference (ISMIR 2014)*, 2014, pp. 59–64.

[22] A. Tenkanen and F. Gualda, "Detecting changes in musical texture," in *International Workshop on Machine Learning and Music*, 2008.

[23] G. Peeters and K. Fort, "Towards a (better) definition of annotated mir corpora," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2012.

[24] M. Giraud, R. Groult, and E. Leguy, "Dezrann, a web framework to share music analysis," in *International Conference on Technologies for Music Notation and Representation (TENOR 2018)*, 2018, pp. 104–110.

[25] J. Hentschel, F. C. Moss, M. Neuwirth, and M. Rohrmeier, "A semi-automated workflow paradigm for the distributed creation and curation of expert annotations," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[26] B. Benward and M. Saker, *Music in Theory and Practice (Volume 1)*. McGraw-Hill Professional, 2008 (8th ed.).

[27] C. McKay, J. Cumming, and I. Fujinaga, "JSymbolic 2.2: Extracting features from symbolic music for use in musicological and MIR research," in *International Society for Music Information Retrieval Conference (ISMIR 2018)*, 2018, pp. 348–354.

[28] M. S. Cuthbert and C. Ariza, "music21: A toolkit for computer-aided musicology and symbolic music data," in *International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010, pp. 637–642.

[29] C. W. White and I. Quinn, "The yale-classical archives corpus," *Empirical Musicology Review*, vol. 11, no. 1, 2016.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

VIOLIN ETUDES: A COMPREHENSIVE DATASET FOR F_0 ESTIMATION AND PERFORMANCE ANALYSIS

Nazif Can Tamer **Pedro Ramoneda** **Xavier Serra**
Music Technology Group, Universitat Pompeu Fabra, Barcelona
nazifcan.tamer@upf.edu

ABSTRACT

Violin performance analysis requires accurate and robust f_0 estimates to give feedback on the playing accuracy. Despite the recent advancements in data-driven f_0 estimators, their application to performance analysis remains a challenge due to style-specific and dataset-induced biases. In this paper, we address this problem by introducing *Violin Etudes*, a 27.8-hours violin performance dataset constructed with domain knowledge in instrument pedagogy and a novel automatic f_0-labeling paradigm. Experimental results on unseen datasets show that the CREPE f_0 estimator trained on *Violin Etudes* outperforms the widely-used pre-trained version trained on multiple manually-labeled datasets. Further preliminary findings suggest that (i) existing data-driven f_0 estimators may overfit to equal temperament, and (ii) iterative re-labeling regularized by our novel *Constrained Harmonic Resynthesis* method can simultaneously enhance datasets and f_0 estimators. Our dataset curation methodology is easily scalable to other instruments owing to the quantity of pedagogical data online. It also supports a range of MIR research directions thanks to the performance difficulty labels from educational institutions.

1. INTRODUCTION

Accurate f_0 tracking is fundamental for violin performance analysis due to the prime role of intonation in violin mastery. Musicians regard intonation and pitch accuracy as the most important criteria for assessing a string performance [1], and most of the previous work on violin performance analysis also focus on vibrato and intonation [2–5]. A study on intonation patterns of artist-level violinists [4] found that highly-regarded musicians deviate significantly from equal-temperament while remaining coherent in their intonation preferences in the close vicinity of just-noticeable difference (95% confidence intervals within just 6 cents). Another study found that violinists' intonation can be better approximated by other tuning systems, e.g., Pythagorean, rather than the standard equal-temperament [2]. From an engineering perspective, intonation analysis is reliable only if the f_0 estimates are more precise than the intonation consistency of the player. Thus, these studies imply that an f_0 estimator suitable for violin performance analysis needs to conform to a frequency precision higher than 6 cents and remain consistent irrespective of the player's deviation from equal temperament.

Alongside frequency precision, an f_0 estimator has to fulfill two more requirements for reliable performance analysis: (i) robustness to octave errors that result from the complex frequency response of the violin body and (ii) high temporal precision that can handle fast string crossings common in violin performance. However, in the current paradigm, there is a trade-off in complying with these two necessities: Most of the f_0 estimators leverage temporal post-processing stages in order to eliminate octave errors at the expense of temporal precision (e.g., Viterbi for the f_0 estimator of PRAAT [6], pYIN [7], and CREPE [8]; a custom filtering for Melodia [9]). Moreover, the Viterbi implementations of some f_0 estimators are even more restrictive: pYIN does not allow a jump bigger than 2.5 semitones between consecutive frames, and that number is 2.4 semitones for CREPE. These restrictions are detrimental to violin performance analysis, as it is common to see very fast and abrupt string crossings in violin repertoire (e.g., 21 semitone jumps in Figure 2).

Despite the above-mentioned problems, monophonic f_0 estimation is considered a mature task in MIR literature, mainly owing to the high accuracies reported in benchmark datasets. Data-driven f_0 estimators claim 96-99% accuracies in datasets such as MIR-1k [10] and MDB-stem-synth [11], which are all highly restricted in terms of performance virtuosity. Although these numbers seem promising in theory, this is not what we found when we use these f_0 estimators in the real-life analysis of advanced-level violin performances. In this paper, to better address the real-world needs of performance analysis using data-driven methods, we introduce the *Violin Etudes* [1], a 27-hour large-scale monophonic dataset comprised of pedagogical violin performances by professional violin players. We also provide our methodologies for dataset curation and automatic f_0 labeling and show the strength of our approach by outperforming the pre-trained version of CREPE f_0 estimator on its own train data.

© N. C. Tamer, P. Ramoneda, and X. Serra. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: N. C. Tamer, P. Ramoneda, and X. Serra, "Violin Etudes: A Comprehensive Dataset for $\mathbf{f_0}$ Estimation and Performance Analysis ", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] *Violin Etudes* dataset is available for research purposes on https://doi.org/10.5281/zenodo.6564408

2. RELATED WORK

2.1 Monophonic f_0 Estimation

F_0 estimation is an essential step for many tasks in music and speech processing. Early f_0 estimators were handcrafted methods [6, 7, 12–19], but today data-driven methods [8, 20–24] are preferred following the general trend of deep learning. Both in handcrafted and data-driven techniques, f_0 estimation literature can be classified into two main approaches: time-domain and frequency-domain. Time-domain approaches used to utilize techniques such as auto-correlation during the times of handcrafted signal processing [6, 7, 12–15], while nowadays end-to-end deep learning methods directly use the time domain signal as their input [8, 20, 23, 24]. Frequency-domain approaches, on the other hand, were used to apply spectral analysis and select spectral peaks with signal processing [16–19]; but recently data-driven frequency-domain approaches require a spectrogram derivative as their input signal [21, 22]. A final important concept to mention here is self-supervised f_0 estimation, adopted by SPICE [22] and DDSP-inv [24]. Although the f_0 estimation strategy used in this paper is based on supervised learning, our combined procedure of iterative f_0-labeling is analogous to a self-supervision applied post-training.

2.2 Automatic f_0-labeling

To guarantee the f_0 label correctness for semi-automatically labeled datasets, Salamon et al. created an analysis-synthesis method [11] which forces the audio to represent any f_0 error in the annotation. The method was used for creating the resynthesized MDB-mf0-synth and Bach10-mf0-synth datasets. However, applying this method to unlabeled datasets is yet to be explored.

2.3 Large Scale Performance Datasets

Collecting large-scale datasets from community platforms has long been an important data source for research on action recognition and multimodal learning. The data collection procedure most often involves searching for keywords on YouTube, generally without collecting further metadata other than the query term itself. To our knowledge, the three main instrument performance datasets collected in this fashion are aimed at self-supervised source separation and spatial localization: MUSIC dataset [25] consists of solo and duet performances spanning over 11 instrument categories, including 53 solo violin performances. They extended the dataset for the task of video-to-audio synthesis with additional solo performances and constructed the MUSIC-Extra-Solo dataset [26] which include 213 solo violin performances, some of which include backing track. In a similar but more constrained scenario, the Solos dataset [27] is formed by searching for 13 classical instruments and the word 'audition' on YouTube and includes 66 solo violin performances in approximately 400 minutes. However, none of these datasets provide any label or metadata on the musical content: whether they are monophonic, include different renditions of the same score, the player or recording conditions, or the supposed difficulty of the performance. The only label is the instrument name.

A more constrained and informed large-scale data collection endeavor is the GiantMIDI-piano dataset [28] where the authors queried YouTube with the piano repertoire they collected from International Music Score Library Project (IMSLP) and later transcribed the audio into MIDI. By including more meta-data such as the composers, work titles, and style, the dataset is more suitable for musical analysis. However, it is also susceptible to MIDI transcription errors. The *Violin Etudes* dataset introduced in this work has similarities to this controlled approach, but in an even more restricted scenario: By collecting query words from the pedagogical repertoire, we have control over the performance difficulty. By keeping track of the performer and providing multiple renditions for the same etudes, we control the expressivity and recording conditions. Last but not least, by manually curating and removing the works, including double stops and chords, we ensure monophony and control the timbre.

3. VIOLIN ETUDES DATASET

From the 16^{th} century onwards, music pedagogy has been creating teaching curricula that guide the students from the very early stages of their journey to the professional level. Inspired by how humans learn an instrument, we present the *Violin Etudes* dataset, the first large-scale MIR dataset rooted in instrument pedagogy. Etudes and caprices form the backbone of the traditional violin method and are defined as study pieces presenting "a technical problem or challenge in the context of a musical setting" [29]. Alongside being vital for education, these pedagogical materials have an unparalleled potential as datasets for intelligent systems, especially MIR applications. They most often come with inherent difficulty labels and are organized in human curricula, which can be used for autonomous learning. Composers often provide textual descriptions on the purpose of the study and techniques involved (e.g., Figure 3), and they are still actively researched by instrument pedagogues on their technical content e.g. [30–32]. They are most often for solo instruments, which is favorable for signal processing. And most importantly, there are hundreds of instrument teachers actively recording their reference performances of their teaching material, which supports the much-needed data for deep learning applications.

3.1 Data Collection

While previous large-scale YouTube data collection endeavors mainly focus on data for self-supervision, we opted for a more controlled approach in curating the material and stored metadata that would be used as ground truth in many topics such as expressive performance analysis and performance difficulty analysis. The individual violin methods are selected from the standard violin curriculum by a trained violinist, but interested readers are encouraged to search for *'violin (or flute/trumpet/piano...) etudes'* to see how easily they can create similar lists, e.g., [33–36].

method	n_{unique}	n_{player}	n_{Σ}	t_{Σ}
Suzuki, Vol. 1-5	40	4	158	248.8
Dancla, Op. 84	27	2	59	131.9
Wohlfahrt, Op. 45	41	6	357	458.1
Sitt, Op. 32 Vol. 1-3	34	2	60	140.1
Kayser, Op. 20	5	8	40	81.9
Mazas, Op.36	12	4	35	100.7
Dont, Op. 37	10	3	30	68.1
Kreutzer, Études	24	4	95	229.8
Fiorillo, Op. 3	13	3	34	72.1
Rode, Op.22	7	5	35	82.5
Dont, Op. 35	7	2	14	31.7
Gavinies, Matinées	6	2	8	24.6
Total	**226**	**21**	**925**	**1670.2**

Table 1. Pedagogical methods ordered in approximate difficulty. n_{unique}: number of unique monophonic studies, n_{player}: number of distinct players, n_{Σ}: recording count per method, t_{Σ}: recording duration in minutes per method.

The most effort was spent on curating a monophonic [2] subset of these works by manually going through the scores on IMSLP and discarding all the works that include even a single double stop or chord, which corresponds to manually removing 264 of 490 studies [3]. The remaining 226 monophonic pedagogical works are summarized in Table 1 in their usual order of use in the violin curriculum.

The videos of the selected works are collected from YouTube by querying method/etude names followed by manually identifying the most reliable content creators, and then searching with queries including their names. Some videos, especially in the beginner repertoire, include different studies as multiple chapters within the same recording. These chapters are split and the audio recordings are extracted from the source videos with the highest possible quality, resulting in 925 performances.

3.2 Metadata

For each recording, we provide metadata with player ID, study No., which method/etude book it belongs to, and piece-wise difficulty rankings from multiple sources. Since difficulty is a subjective term, sources sometimes disagree on the ranking of these materials, but there are some common patterns: Suzuki Vol. 1-2 are always considered the easiest of all, with Dancla, Wohlfahrt, Suzuki Vol. 3-5 being the next. Rode, Gavinies, Fiorillo, and Dont Op. 35 form the hardest cluster. Whilst difficulty grades change from source to source, the progressive ordering within each book is universally accepted, e.g. study No. 30 from a method is always considered as harder than No. 2. Hence, *Violin Etudes* can be considered as a performance difficulty analysis dataset similar to [37].

The performers in the dataset are manually confirmed to be highly-skilled violinists, mostly violin teachers creating content for their students. As shown in Figure 1, our dataset includes performances from 21 players and is also quite skewed in favor of Players 01 and 02. Although this is a limit of the dataset from one aspect, with more than 5 hours of monophonic recordings for Player02 and 7 hours for Player01, these subsets include enough data to train violin synthesizers. In Table 1, we can see that we have a total of 925 reference performances for 226 monophonic violin studies. With some exceptions, each study has multiple renditions which allow high-level music research such as expressive performance analysis (exemplified briefly in Section 5.1), or low-level audio representation learning as exemplified in the f_0 estimation experiments in Section 5.2.

Despite its many strengths, the current state of *Violin Etudes* has two main flaws. Although the scores are included, only Mazas and Kayser etudes are in a machine-readable format, and we do not have their aligned scores. As repertoire gets harder, it gets harder to find non-commercial reference recordings. Thus, recording distribution in the dataset is limited by method popularity and difficulty, e.g., Matinées in Table 1 is only 24 minutes.

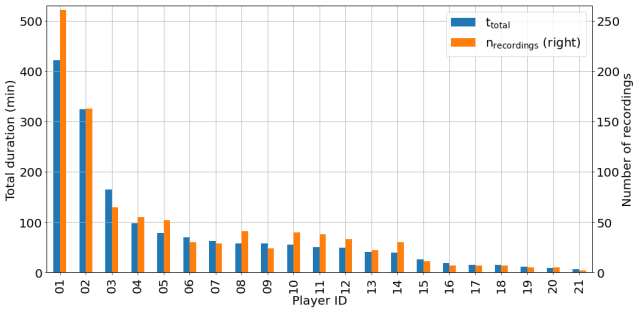


Figure 1. Total performance duration (left) and number of recordings (right) per PlayerID.

4. AUTOMATIC F_0 LABELING

The f_0 labels in the Violin Etudes dataset are automatically generated through a novel iterative f_0-labeling strategy based on two assumptions: (i) the recordings are monophonic, i.e. we can apply harmonic analysis with respect to an initial $\hat{f}_0$ estimate, and (ii) all the frames have a similar harmonic structure, i.e. violin is the sole instrument. After obtaining raw $\hat{f}_0$ estimates through a data-driven f_0 estimator, we search for harmonic peaks around the $\hat{\mathbf{f}}_{0:N}$ multiples of the $\hat{f}_0$ estimate for each frame through a modified version of Spectral Modeling Synthesis (SMS) [38]. We then force the harmonics to follow the $\hat{f}_0$ label similar to the analysis/synthesis framework of Salamon et al. [11], with additional novel constraints on instrument modeling and harmonic consistency to silence low-confidence segments. The remaining data is smaller in amount, yet it is more reliable to be used in the training of a new f_0 estimator. We then retrain the f_0 estimator using these resynthesized versions and extract new f_0' estimates, and repeat the process. Thus, as exemplified in Figure 2, both the f_0 estimator's performance and the dataset's f_0 label quality are enhanced simultaneously by interacting with one another.

[2] Here monophonic simply refers to *single note at a time*, i.e. removal of superposed notes, rather than the musicological definition.

[3] Unlike the use of violin in popular music, the classical and pedagogical violin repertoire exhaustively include double stops and chords.

4.1 Constrained Harmonic Resynthesis

First introduced by Serra et al. [38], Spectral Modeling Synthesis (SMS) has had widespread adoption in audio signal processing, including a recent revival with differential Digital Signal Processing (DDSP) [39]. We adopt the harmonic model from SMS to create constrained f_0 labels where we force the synthesis to follow the label or simply silence the frame if it does not satisfy the constraints.

4.1.1 Harmonic Analysis

Waveforms in 44.1kHz are analyzed with 1025-sample Blackman-Harris windows and hop size of 128 using `harmonicModelAnal` function from the SMS-tools [38]. The DSP-based f_0 detection algorithm of the `harmonicModelAnal` is replaced with our external $\hat{f}_0$ estimates; and harmonic amplitudes, frequencies, and phases are searched around the $\hat{\mathbf{f}}_{0:39}$ with a harmonic deviation slope of 0.001. We refer to [38] for further details.

4.1.2 Instrument-modeling Constraint (IC)

Instrument timbre defines the harmonic amplitudes $\mathbf{A}_{0:39}$ we see on top of an $\hat{f}_0$ estimate. Thus, if we know the instrument model, i.e. $P(\mathbf{A}_{0:39}|\hat{\mathbf{f}}_{0:39})$, we can assess the correctness of an estimate from harmonics. We use this idea as an additional layer of regularization of the label quality and removal of automatic-labeling artifacts. We learn an approximate model for violin resonance structure by linearly dividing the violin frequency range into N regions and fitting elliptic envelopes to the first 12 harmonic amplitudes for each of these N regions, where we experimented with $N = 50$ and 100. If this elliptic envelope model detects an anomaly in the harmonic amplitudes, we silence the frame.

4.1.3 Harmonic Consistency Constraint (HCC)

As we will show in Figure 4, an f_0 estimator is prone to overfitting problems associated with its training set distribution. To remedy this and ensure a reliable spectral distribution for our labels, we employed the Two-Way-Mismatch (TWM) procedure [19] between harmonic peaks $\hat{\mathbf{f}}_{0:39}$ and $\hat{f}_0$ candidates around the initial $\hat{f}_0$ estimate. For each voiced segment, 33 pitch candidates are selected in the range $(\hat{f}_0 - 16c, \hat{f}_0 + 16c)$ with 1 cent intervals. The candidate with the lowest Two-Way-Mismatch-error is selected to be the new $\hat{f}'_0$ estimate if its TWM-error is smaller than 5.0, and if this Harmonic Consistency Constraint is not satisfied, the frame is silenced. Finally, the constrained harmonic frequencies $\hat{\mathbf{f}}'_{\mathbf{0:39}}$ are set to the *exact multiples* of this final $\hat{f}'_0$ estimate before resynthesis.

4.1.4 Sinusoidal Synthesis

The constrained harmonics are resynthesized using the sinusoidal model from SMS [38]. Any segments shorter than 50ms are muted before synthesis to reduce artifacts. Furthermore, for each resynthesized recording, we also resynthesize its replicate with pitch shifts: both harmonics and f_0 labels are shifted with random microtonal pitch shifts in the range of 5-55 cents to ensure the statistical diversity of our labels. We found that training f_0 estimator with shifted versions of the recordings increases the stability of the estimator against equal temperament deviation.

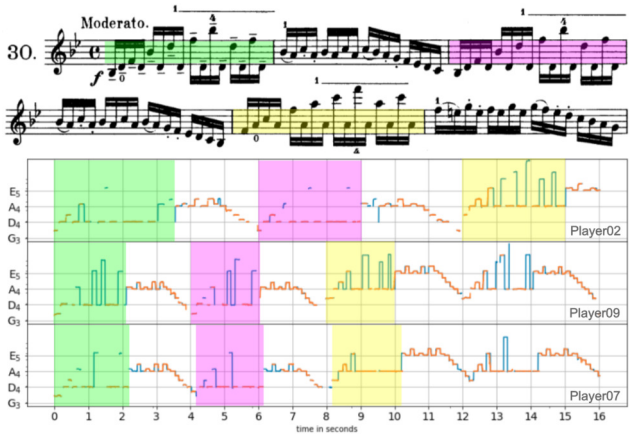


Figure 2. Iterative f_0 labeling exemplified in Kreutzer Etude No.30 performed by Players 02, 09, and 07. Constrained harmonic resynthesis acts as a barrier to wrong f_0 estimates and creates discrepancies in the initial f_0 contours (orange). Notice that most of these discrepancies are filled after the first iteration of finetuning (blue), especially in the highlighted string crossings.

4.2 Iterative Refinement of f_0 Labels

Initial f_0 tracks of the dataset are generated via the CREPE [8] convolutional f_0 estimator with 1 ms intervals and a custom Viterbi decoding to incorporate some heuristics we know about the violin repertoire. CREPE has 360 bins in its final layer with 20 cents between consecutive bin centers, i.e., states. In their implementation, they apply Viterbi with constant state observation probabilities without utilizing the confidences given by the algorithm. We replaced this by decoding with CREPE confidences as posteriors, similar to standard ANN-HMM posteriorgram decoding [40]. Using our prior knowledge of violin repertoire, we decided the Viterbi transition probabilities empirically as a weighted sum of two Gaussians to allow for fast string jumps while encouraging continuous f_0 contours: Gaussian with $\sigma_1 = 6$ semitones in Equation 1 enables fast string jumps, whilst continuous contours are encouraged by weighting the other Gaussian ($\sigma_2 = 40$ cents) with 9.

$$Pr(s_j(t+1)|s_i(t)) = \frac{1}{30\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{s_j(t+1) - s_i(t)}{30}\right)^2\right) + \frac{9}{2\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{s_j(t+1) - s_i(t)}{2}\right)^2\right) \quad (1)$$

After the initial f_0 estimates are obtained, (audio, f_0) pairs go through the Constrained Harmonic Resynthesis which silences out the wrong estimates as described in Section 4.1. The remaining (constrained audio, constrained f_0) pairs are used for finetuning of the f_0 estimator, which produces new estimates as exemplified in Figure 2. This iterative labeling process can be thought of as an Expectation-Maximization hybrid akin to ANN-HMMs [40].

	Violin		Other Inst.	
	RPA50	RPA5	RPA50	RPA5
Pretrained [8]	96.4	68.3	**96.2**	68.1
HCC	96.3	83.8	95.2	76.4
HCC + IC50	**96.7**	84.0	94.6	**76.5**
HCC + IC100	**96.7**	**84.2**	94.8	75.7
Sawtooth	68.3	49.5	56.8	37.2

Table 2. The pre-trained CREPE compared with training-from-scratch on different versions of *Violin Etudes*. Tests are conducted on the unseen URMP dataset. HCC: Harmonic Consistency Constraint. IC: Instrument-modeling Constraint. Sawtooth: Single-timbre control group.

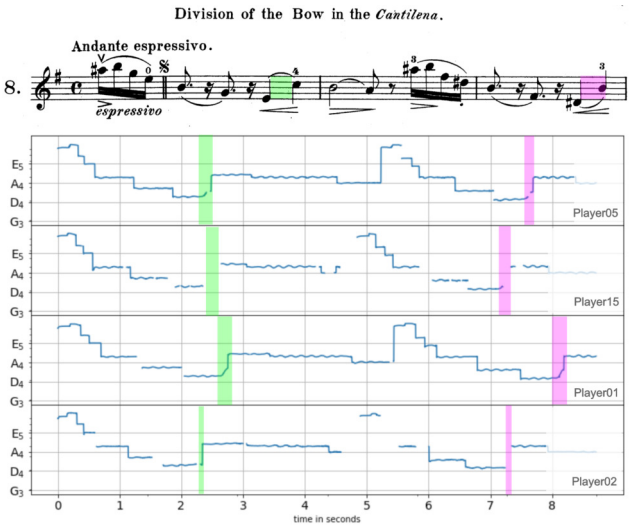


Figure 3. First line of Mazas Etude No. 8 with constrained f_0 labels of 4 performances. From the composer's words, we see that this etude targets teaching *division of the bow in the Cantilena*. f_0 tracks are automatically extracted with the method described in Section 4. Highlighted hand position changes are discussed in Section 5.1

5. EXPERIMENTS

5.1 Qualitative Performance Analysis

In this section, we give two brief examples of how *Violin Etudes* enable research on the pedagogically-motivated analysis of reference performances. In Figure 3, Mazas Op.36 No.8 with f_0 tracks of 4 different renditions can be seen: We can observe that both Players 05 and 01 have very expressive slides at the position changes indicated with green (E_4-C_5) and pink ($D_4^\sharp$-B_4), while Player02 landed into these notes directly -note that such analyses are not possible for corpora annotated with pYIN [7] or CREPE [8] since their Viterbi implementations smooth every jump. But, our method, too, has flaws: analysis of Player15 for the same section is inconclusive due to our design choice of removing unreliable automatic f_0 estimates.

Having multiple reference recordings also enable the analysis of performance tempo. In Figure 2, we see that Players 09 and 07 agree on a considerably faster pace for *Moderato* on Kreutzer Etude No.30, while Player02 play the same study a lot slower and sticks to this choice throughout the performance. Although we did not include it here, studying bowing technique analysis is also possible in *Violin Etudes*. Following the textual descriptions, players record some etudes with bowing variations. The most extreme case of this is the Player01 recording Wohlfahrt etudes with all the bowing variations, resulting in a subset of 171 recordings where some studies were repeated with up to 15 bowing variations. Moreover, the dataset allows for analyzing intonation and tuning patterns of professional violin teachers thanks to the high frequency precision of our f_0 estimates, which we will discuss in the next section.

5.2 Monophonic f_0 Estimation

Here we study the label quality of *Violin Etudes* in the context of f_0 estimator training. We train the commonly-used CREPE [8] f_0 estimator in our data, and compare it with the pre-trained version [4] trained on the following six manually-labeled and synthetic datasets: MIR-1k [10], MedleyDB [43], MDB-STEM-Synth [11, 43], Bach10-mf0-synth [11, 42], RWC-Synth [7], and NSynth [44].

We use *Violin Etudes* only for training and validation with 80/20 split, and train with a batch size of 32 and early stopping on validation accuracy. We conduct the tests on two unseen datasets where we evaluate the raw pitch accuracy (RPA, in %) with two thresholds: the conventional RPA50 quarter-tone accuracy, and our benchmark RPA5 fine-grained accuracy. Owing to the high-frequency precision requirements for performance analysis, this second metric considers the estimate accurate only if it is within 5 cents of the performance. We report frame-level accuracies without Viterbi, and separately evaluate on violin and other instruments for discussion. Since the models trained solely on violin range do not see other pitch classes, we calculate accuracies over violin range for the unseen instruments, i.e., we compute a combined accuracy from all the other instruments except violin, where the ground $f_0 \geq 190$Hz.

5.2.1 *Effect of Instrument-modeling Constraints*

In these experiments, we compare models trained on different variants of *Violin Etudes* in order to study the effectiveness of the constraints introduced in Section 4.1. Tests on the unseen URMP dataset are provided in Table 2. We see that all the three versions of *Violin Etudes* surpass the pre-trained model decisively on fine-grained accuracy RPA5 by more than 15% improvements, and increasing the number of filters from 50 to 100 in Instrument-modeling Constraint (IC) leads to marginally better results. IC also reduces the RPA50 on unseen instruments, implying that the estimator focuses excessively on the target instrument. Interestingly, one of the most conclusive results from these experiments was in the Sawtooth synthesis of the *Violin Etudes* f_0 tracks: model trained with sawtooth performs significantly better in violin compared to others, which may be due to the similarity of violin timbre to sawtooth or the effect of repertoire pitch distribution.

[4] Full CREPE model weights are taken from `https://github.com/marl/crepe/raw/models/model-full.h5.bz2`

	URMP Dataset				Bach10-mf0-synth (from train set of [8])			
	Violin		Others		Violin		Others	
	RPA50	RPA5	RPA50	RPA5	RPA50	RPA5	RPA50	RPA5
Pretrained [8]	96.4	68.3	**96.2**	68.1	98.9	89.5	99.1	**87.4**
Trained only on Violin Etudes	96.7	**84.2**	94.8	75.7	99.1	95.1	98.3	77.6
[8] finetuned on Violin Etudes	**97.0**	83.0	96.0	**77.3**	**99.2**	**95.7**	**99.2**	84.7

Table 3. Comparison of finetuning and training-from-scratch on *Violin Etudes*, tested on two instrumental datasets.

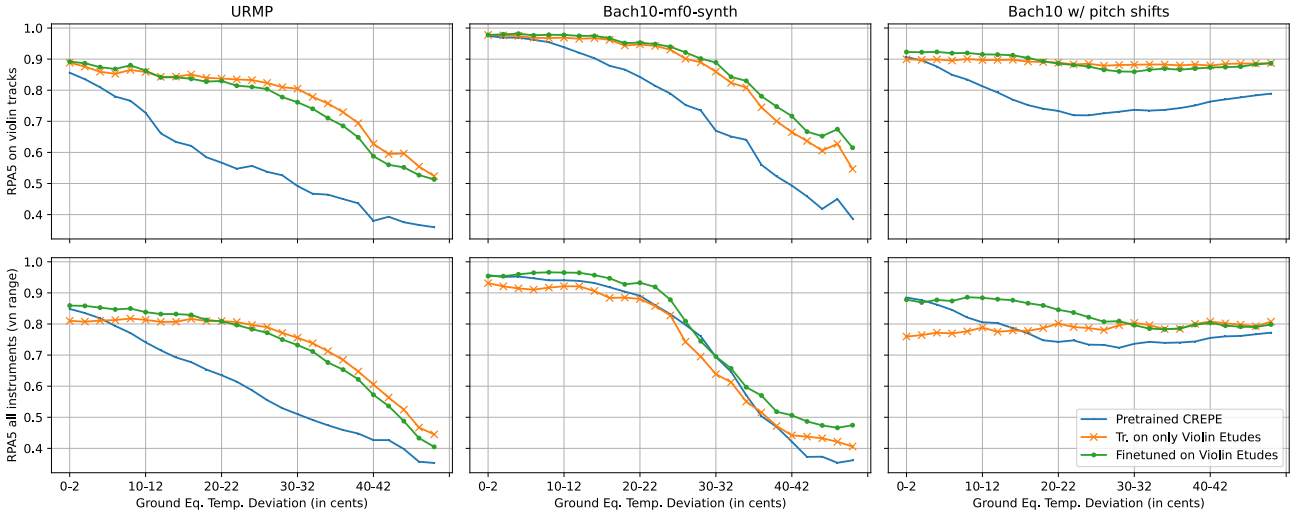


Figure 4. Fine-grained f_0 accuracy (RPA5) summarized on a grid. Music performance analysis requires precise f_0 estimates even if the player deviates from the equal temperament. However, training data-driven f_0 estimators on standard MIR datasets may induce an *auto-tuning effect*. Top: Violin-only tracks. Bottom: Over all the instruments. Left: URMP [41] dataset unseen to all the models. Center: Bach-10-mf0-synth [11, 42] from train set of [8], yet unseen to ours. Right: The conclusive experiment where we ensure the same statistics by applying microtonal pitch shifts to the Bach10 dataset.

5.2.2 *Finetuning and Generalization*

In Table 3, we study finetuning on *Violin Etudes* by testing on two datasets: alongside URMP, this time including Bach10-mf0-synth [11, 42], one of the 6 train sets of the pre-trained CREPE. We tested the estimators on the entirety of the datasets since they do not come with inherent train/test splits. We see that the finetuned model performs the best overall and remains reliable in all scenarios, including other instruments. However, interestingly, finetuning was generally more unstable and took longer to train: while the models converge after seeing around 20% of our dataset in training-from-scratch experiments, finetuning took twice as long on average. As the most interesting result of these experiments, Table 3 illustrates that our model trained on only *Violin Etudes* outperforms the pre-trained CREPE by 5.6% RPA5 on the violin tracks of its own train set.

5.2.3 *Equal Temperament and the Auto-Tuning Effect*

In Figure 4, we show that the drastic gap between the RPA5 and RPA50 performances of pre-trained CREPE can be explained by equal-temperament deviation, even on its own train data. On the other hand, our models are more robust, especially for violin-only evaluation in Figure 4. To show that this dependency is not caused by the annotation quality of the test datasets, we also experimented with microtonal pitch-shifted versions using RubberBand [5]. Yet, all our results show the importance of microtonal label distribution while training data-driven f_0 estimators. We name this phenomenon *dataset-induced auto-tuning effect*.

6. CONCLUSION

In this paper, we present *Violin Etudes*, a large-scale, comprehensive dataset for f_0 estimation and performance analysis. Our dataset curation method is easily extendable to other instruments thanks to two main novelties: 1) Reliance on pedagogy and community-driven platforms ensure abundance of material and metadata suitable for many high-level music research directions. 2) Our novel automatic f_0-labeling paradigm allows iterative refinement of labels while significantly improving the data-driven f_0 estimator as a by-product. Observing that the CREPE model converges after seeing just 20% of the *Violin Etudes* and still outperforms its pre-trained version, we argue that our dataset curation method would allow training more complex f_0 estimators that can specialize better for the needs of performance analysis. In the future, we are going to use this dataset for learning accompaniment-aware f_0 estimators and synthesizers, while further extending the *Etudes* dataset family with woodwind and brass instruments.

[5] `https://breakfastquay.com/rubberband/`

7. ACKNOWLEDGEMENTS

We would like to thank Esteban Maestre for his valuable insights on the violin resonance structure. This research was carried out under the project Musical AI - PID2019-111403GB-I00/AEI/10.13039/501100011033, funded by the Spanish Ministerio de Ciencia e Innovación and the Agencia Estatal de Investigación.

8. REFERENCES

[1] J. M. Geringer and C. K. Madsen, "Musicians' ratings of good versus bad vocal and string performances," *Journal of Research in Music Education*, vol. 46, no. 4, pp. 522–534, 1998.

[2] P. C. Greene, *Violin performance with reference to tempered, natural, and Pythagorean intonation*. University of Iowa Press, 1937.

[3] F. Loosen, "Intonation of solo violin performance with reference to equally tempered, pythagorean, and just intonations," *The Journal of the Acoustical Society of America*, vol. 93, no. 1, pp. 525–539, 1993.

[4] J. M. Geringer, "Eight artist-level violinists performing unaccompanied bach: Are there consistent tuning patterns?" *String Research Journal*, vol. 8, no. 1, pp. 51–61, 2018.

[5] J. M. Geringer, R. B. MacLeod, and J. C. Ellis, "A descriptive analysis of performance models' intonation in a recorded excerpt from suzuki violin school volume i," *String Research Journal*, vol. 4, no. 1, pp. 71–88, 2013.

[6] P. Boersma *et al.*, "Accurate short-term analysis of the fundamental frequency and the harmonics-to-noise ratio of a sampled sound," in *Proceedings of the institute of phonetic sciences*, vol. 17, no. 1193. Citeseer, 1993, pp. 97–110.

[7] M. Mauch and S. Dixon, "pyin: A fundamental frequency estimator using probabilistic threshold distributions," in *2014 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2014, pp. 659–663.

[8] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 161–165.

[9] J. Salamon, E. Gómez, D. P. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.

[10] C.-L. Hsu and J.-S. R. Jang, "On the improvement of singing voice separation for monaural recordings using the mir-1k dataset," *IEEE transactions on audio, speech, and language processing*, vol. 18, no. 2, pp. 310–319, 2009.

[11] J. Salamon, R. M. Bittner, J. Bonada, J. J. Bosch, E. Gómez Gutiérrez, and J. P. Bello, "An analysis/synthesis framework for automatic f0 annotation of multitrack datasets," in *Hu X, Cunningham SJ, Turnbull D, Duan Z. ISMIR 2017 Proceedings of the 18th International Society for Music Information Retrieval Conference; 2017 Oct 23-27; Suzhou, China.[Suzhou]: ISMIR; 2017.* International Society for Music Information Retrieval (ISMIR), 2017.

[12] J. Dubnowski, R. Schafer, and L. Rabiner, "Real-time digital hardware pitch detector," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 1, pp. 2–8, 1976.

[13] D. Talkin and W. B. Kleijn, "A robust algorithm for pitch tracking (rapt)," *Speech coding and synthesis*, vol. 495, p. 518, 1995.

[14] M. Ross, H. Shaffer, A. Cohen, R. Freudberg, and H. Manley, "Average magnitude difference function pitch extractor," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 22, no. 5, pp. 353–362, 1974.

[15] A. De Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 111, no. 4, pp. 1917–1930, 2002.

[16] A. M. Noll, "Cepstrum pitch determination," *The journal of the acoustical society of America*, vol. 41, no. 2, pp. 293–309, 1967.

[17] P. Martin, "Comparison of pitch detection by cepstrum and spectral comb analysis," in *ICASSP'82. IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 7. IEEE, 1982, pp. 180–183.

[18] A. Camacho and J. G. Harris, "A sawtooth waveform inspired pitch estimator for speech and music," *The Journal of the Acoustical Society of America*, vol. 124, no. 3, pp. 1638–1652, 2008.

[19] R. C. Maher and J. W. Beauchamp, "Fundamental frequency estimation of musical signals using a two-way mismatch procedure," *The Journal of the Acoustical Society of America*, vol. 95, no. 4, pp. 2254–2263, 1994.

[20] S. Singh, R. Wang, and Y. Qiu, "Deepf0: End-to-end fundamental frequency estimation for music and speech signals," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 61–65.

[21] K. Han and D. Wang, "Neural network based pitch tracking in very noisy speech," *IEEE/ACM transactions on audio, speech, and language processing*, vol. 22, no. 12, pp. 2158–2168, 2014.

[22] B. Gfeller, C. Frank, D. Roblek, M. Sharifi, M. Tagliasacchi, and M. Velimirović, "Spice: Self-supervised pitch estimation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 1118–1128, 2020.

[23] W. Ma, Y. Hu, and H. Huang, "Dual attention network for pitch estimation of monophonic music," *Symmetry*, vol. 13, no. 7, p. 1296, 2021.

[24] J. Engel, R. Swavely, L. H. Hantrakul, A. Roberts, and C. Hawthorne, "Self-supervised pitch detection by inverse audio synthesis," 2020.

[25] H. Zhao, C. Gan, A. Rouditchenko, C. Vondrick, J. McDermott, and A. Torralba, "The sound of pixels," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 570–586.

[26] H. Zhou, Z. Liu, X. Xu, P. Luo, and X. Wang, "Vision-infused deep audio inpainting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[27] J. F. Montesinos, O. Slizovskaia, and G. Haro, "Solos: A dataset for audio-visual music analysis," in *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2020, pp. 1–6.

[28] Q. Kong, B. Li, J. Chen, and Y. Wang, "Giantmidi-piano: A large-scale midi dataset for classical piano music," *arXiv preprint arXiv:2010.07061*, 2020.

[29] M. K. Buckles, *A structured content analysis of five contemporary etude books for the violin*. Louisiana State University and Agricultural & Mechanical College, 2003.

[30] A. R. Cutler, *Rodolphe Kreutzer's Forty-two Studies or Caprices for the Violin: Detailed analyses according to Ivan Galamian's philosophies of violin playing*. Northwestern University, 2003.

[31] M. H. Tung, *The pedagogical contributions of Rode's Caprices to violin mastery*. The University of Texas at Austin, 2001.

[32] D. Kaplunas, "Pedagogical analysis of jakob dont's 24 etudes and caprices, op. 35," Ph.D. dissertation, University of Georgia, 2008.

[33] "Violin wiki - violin etudes," Mar 2022. [Online]. Available: https://www.violinwiki.org/wiki/Violin_etudes

[34] "Violin Methods and Etudes | Violin Masterclass.com." [Online]. Available: https://www.violinmasterclass.com/p/violin-methods-and-etudes

[35] G. T. Keat, "A guide to violin etudes and studies." [Online]. Available: https://spinditty.com/learning/A-Guide-to-Violin-Etudes-and-Studies

[36] "Ecsm violin curriculum." [Online]. Available: https://www.esm.rochester.edu/community/faq/student-curriculum/violin/

[37] P. Ramoneda, N. C. Tamer, V. Eremenko, X. Serra, and M. Miron, "Score difficulty analysis for piano performance education based on fingering," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 201–205.

[38] X. Serra *et al.*, "Musical sound modeling with sinusoids plus noise," *Musical signal processing*, pp. 91–122, 1997.

[39] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "Ddsp: Differentiable digital signal processing," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=B1x1ma4tDr

[40] E. Trentin and M. Gori, "A survey of hybrid ann/hmm models for automatic speech recognition," *Neurocomputing*, vol. 37, no. 1-4, pp. 91–126, 2001.

[41] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, 2018.

[42] Z. Duan, B. Pardo, and C. Zhang, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 8, pp. 2121–2133, 2010.

[43] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research." in *ISMIR*, vol. 14, 2014, pp. 155–160.

[44] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1068–1077.

CHECKLIST MODELS FOR IMPROVED OUTPUT FLUENCY IN PIANO FINGERING PREDICTION

Nikita Srivatsan
Carnegie Mellon University
nsrivats@cmu.edu

Taylor Berg-Kirkpatrick
UC San Diego
tberg@eng.ucsd.edu

ABSTRACT

In this work we present a new approach for the task of predicting fingerings for piano music. While prior neural approaches have often treated this as a sequence tagging problem with independent predictions, we put forward a checklist system, trained via reinforcement learning, that maintains a representation of recent predictions in addition to a hidden state, allowing it to learn soft constraints on output structure. We also demonstrate that by modifying input representations — which in prior work using neural models have often taken the form of one-hot encodings over individual keys on the piano — to encode *relative* position on the keyboard to the prior note instead, we can achieve much better performance. Additionally, we reassess the use of raw per-note labeling precision as an evaluation metric, noting that it does not adequately measure the *fluency*, i.e. human playability, of a model's output. To this end, we compare methods across several statistics which track the frequency of adjacent finger predictions that while independently reasonable would be physically challenging to perform in sequence, and implement a reinforcement learning strategy to minimize these as part of our training loss. Finally through human expert evaluation, we demonstrate significant gains in performability directly attributable to improvements with respect to these metrics.

1. INTRODUCTION

While sheet music is often very specific as to *what* notes a musician must play, it can also be vague about *how* to play them. Composers generally write notation that focuses on describing the desired musical output, but leave the specifics of how to achieve this with their instrument up to the performer, as these details are outside the scope of their role in the creative process or in some cases beyond their expertise. The piano is an instrument which requires an extensive amount of decision-making on the performer's part. In order to perform a piece of music, a pianist must either consciously or intuitively select which finger to use to play every note in the song. Notes may overlap in duration or even have simultaneous onsets. Since any key on the piano can potentially be played by any finger, there are an exponential number of ways one could perform any given piece; of these however, the majority would be uncomfortable, difficult to play at full tempo, or even physically impossible [1]. The challenge of deciding the most ergonomic fingering to use can be nontrivial for less experienced pianists, who would not yet have the ability to quickly map common musical patterns to conventional fingering strategies, and might produce more consistent and accurate performances if provided with them [2].

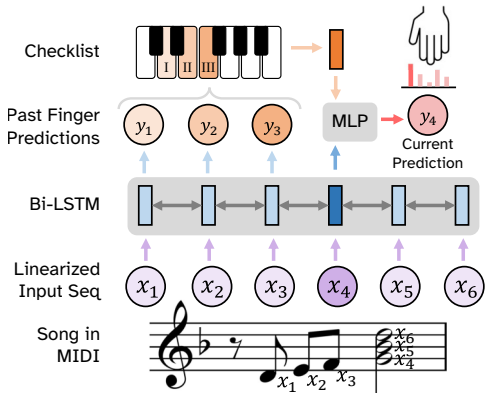


Figure 1. Visualization of the checklist model. Notes are embedded and then passed to a Bi-LSTM which outputs contextualized embeddings at each timestep. A checklist encodes where the fingers that have recently been used are located based on prior predictions. These are both fed into an MLP which predicts the next finger.

However recent work has shown that the use of machine learning techniques may be able to assist in this regard [3–6]. Automatic piano fingering prediction is the task of inferring finger assignments for each note in a symbolic representation of a piano song, given knowledge of which hand is meant to play each note. Human players generally prefer fingerings that balance physical comfort and efficiency [2], yet these criteria are difficult to quantify and highly subjective [7]. Decisions are simultaneously constrained by the current placement of the musician's fingers, and by how that decision may in turn affect possible options for the notes that follow. And while the spatial location of the notes on the keyboard is arguably the most salient factor, timing is also crucial; certain fingerings might be easy for a set of consecutive notes, but impossible if the same notes are in a chord.

© N. Srivatsan and T. Berg-Kirkpatrick. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** N. Srivatsan and T. Berg-Kirkpatrick, "Checklist Models for Improved Output Fluency in Piano Fingering Prediction", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

Prior approaches to this task have largely treated it as a sequence tagging problem, where the input at each step is simply the pitch of the note and the output is a softmax over indices corresponding to each finger, and have employed methods analogous to those used for part of speech tagging such as LSTMs and HMMs. However, these techniques are limited in their capacity to model output dependencies, and also disregard the amount of elapsed time between notes which can greatly affect a prediction's performability.

We propose an autoregressive approach, where prior predictions are fed back into the network in order to encourage the model to produce fingerings that are not just accurate on average, but also locally *fluent* and therefore playable. We do this using a checklist which indicates which fingers have either recently been or are currently in use, and where they are on the keyboard relative to the note at the current timestep. By directly exposing this information, the model can more easily learn to restrict its predictions to fingers that are actually free, and also make decisions more directly influenced by the hand's physical placement. We also experiment with an approach that only feeds back in the most recent prediction, as well as an autoregressive tagger which maintains a neural representation of prior predictions using an encoder network.

Furthermore, the evaluation metrics used in prior work do not always correlate with what a pianist might intuitively perceive as "good" predictions. In fact there are many types of desirable (or undesirable) patterns that may emerge in a model's output, which a simplistic metric such as labeling precision may not actually reflect. We demonstrate that it is possible for two models with nearly identical labeling precision to differ dramatically in the frequency of specific types of output subsequences that are physically challenging to play. Therefore, we evaluate on a battery of metrics that collectively convey a more holistic and interpretable overview of fingering quality.

Since these metrics are not always correlated with the labeling precision that a cross entropy loss optimizes, we also investigate incorporating them explicitly into our loss function at train time. While these scores are non-differentiable, we show that reinforcement learning techniques—which have previously seen success in sequence generation tasks outside of music—can successfully optimize them. This, in conjunction with our checklist approach, ensures that the model not only has direct access to the information it would need to avoid undesirable output patterns, but is also encouraged to do so.

Finally, we conduct human evaluations and qualitative analysis which confirm that our approach improves predicted fingerings in ways consistent with our modeling choices, and also suggest directions for future work.

In summary our contributions are as follows: (1) We provide a comparison of various input representations for LSTM models (2) We put forward checklist based approaches which directly incorporate information from previous decisions (3) We introduce several additional metrics which track fluency of output, and demonstrate how to train directly on them using reinforcement learning.

2. RELATED WORK

While constraint-based models for automatic piano fingering have long existed [1], the standardization as a machine learning task which we follow was formalized by [3], which introduced the PIano fingerinG (PIG) dataset, put forward LSTM and HMM baselines (following prior work using HMMs [8, 9]), and has since been followed up on by others. [4] demonstrated the value of pretraining on even a noisy automatically annotated dataset, although they focused on basic LSTM models and evaluated exclusively on labeling precision. [5] put forward the idea of representing inputs based on relative difference in pitch rather than absolute pitch, and also showed improvements from the use of a constrained transition matrix, albeit one that was built off of prior knowledge of the task. [6] recast fingering as an information retrieval problem, although they focused on the Czerny corpus instead of the PIG dataset.

Our work also draws from research into implicitly learning output structure by feeding prior decisions into the computation of future predictions. There are classic examples of graphical models that condition predictions both on inputs at the current timestep and outputs at the previous [10]. Recent work on explicit checklists in neural settings, such as by [11] found success using a structured agenda that tracked ingredient usage in conditional cooking recipe generation. Our checklists are however more transient; outputs can be removed from the checklist if enough time has passed. We also track not just the presence of outputs, but information about how they were used.

There is also much work on the use of REINFORCE [12] for sequence generation tasks. While [13] performed several tasks including summarization, translation, and image captioning, but required a critic model for stabilization, [14] developed a self-critical method for captioning that did not require this additional network. [15] applied a similar technique to abstractive news summarization. Reinforcement learning has also been applied in a musical setting [16], including the REINFORCE algorithm specifically [17, 18], but most of these have focused on music generation rather than downstream prediction of performance-oriented attributes such as fingerings.

3. MODEL

We now describe the basic layout of the models we compare, and detail the specific variations between them. First, we will formalize the basic model contract and explain the basic architecture components that most of them share.

At a high level, each model takes in a MIDI-derived representation of either the left or right hand of a complete piano song, and outputs a predicted finger for each note. More formally, we are given an input sequence of notes $\mathbf{x} = \{x_1, ..., x_N\}$, where each $x_i = (p_i, t_i^-, t_i^+)$ is a tuple consisting of a pitch $p_i \in \{1, ..., 88\}$ as well as a corresponding onset time $t_i^- \in \mathbb{R}_{\geq 0}$ and offset time $t_i^+ > t_i^-$. From $\mathbf{x}$ we must predict a corresponding sequence $\mathbf{y} = \{y_1, ..., y_N\}$ where each $y_i \in \{1, 2, 3, 4, 5\}$ is an index representing the finger used to play x_i from thumb to pinky respectively.

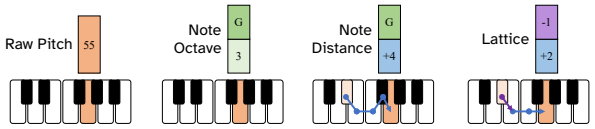


Figure 2. Examples of the input representations we compare. Note that the labels on the vectors are *indices* corresponding to sub-embeddings shared across notes.

To this end, each predictive model defines a distribution $p(\mathbf{y}|\mathbf{x};\theta)$, generally parameterized by an LSTM variant. Under our non-checklist simple Bi-LSTM baseline, the individual y_i are conditionally independent given $\mathbf{x}$: $p(\mathbf{y}|\mathbf{x};\theta) = \prod_{i=1}^{N} p(y_i|\mathbf{x};\theta)$. However for our autoregressive models, the likelihood factorizes according to chain rule: $p(\mathbf{y}|\mathbf{x};\theta) = \prod_{i=1}^{N} p(y_i|\mathbf{y_{<i}}, \mathbf{x};\theta)$

As shown in Figure 1, the notes are first embedded, and then passed to a Bi-LSTM which outputs a contextualized representation. This contextualized representation is optionally concatenated with a d dimensional vector representation of the checklist $\mathbf{c} \in \mathbb{R}^{N \times d}$ before being passed to an MLP which outputs a softmax distribution over finger indices. We can think of the checklist as a function of the past and current notes as well as the past finger predictions $c_i(\mathbf{x}_{\leq i}, \hat{\mathbf{y}}_{<i})$. The model is trained to optimize the cross entropy loss between this distribution and the true labels.

3.1 Input Representations

We now discuss four different strategies to encode the song into an input sequence of vector embeddings (illustrated in Figure 2). These each expose different information to the model which can significantly affect performance.

Raw Pitch: This approach simply represents each note by its pitch p_i, assigning a learnable embedding to each value. This was done by most previous neural implementations [3, 4], and requires learning 88 vector embeddings.

Note $\oplus$ Octave: Given the fact that a piano keyboard's layout is identical across octaves, we might believe that more important than the exact key being played is the named note itself (e.g. knowing that the next note is a `G#` might be more useful in predicting which finger to use than knowing that it is the MIDI pitch 68). To this end we also evaluate the use of an input representation that consists of an embedding corresponding to the named note concatenated with an embedding corresponding to the octave.

Note $\oplus$ Relative Distance: If finger choice is primarily a function not of the note's pitch but of its location on the keyboard relative to the other notes in the song, then directly exposing that information saves the model from having to infer it. For this representation, we concatenate an embedding for the named note with one corresponding to the change in pitch $p_i - p_{i-1}$ (see Figure 2). Since large distances between notes require physically lifting the hand from the keys, an act which eliminates any sequential constraints, we cap step sizes at 15 semitones. [5] used a similar approach without the named note embedding.

Lattice: We finally evaluate a "lattice" representation based on the one introduced by [3] (note that this was only used in that work for HMMs and not neural models). As shown in Figure 2, we can think of this as a two-dimensional encoding of relative position, where the first dimension corresponds to the number of white keys between notes (i.e. the *horizontal* distance), and the second dimension indicates if we have moved from a white key to a black key or vice versa (i.e. the *vertical* distance). These dimensions are each embedded with corresponding vector embeddings, with a horizontal cutoff of 9 steps. This is the default representation where not otherwise specified.

3.2 Checklist Formulation

To encourage the model to produce fluent predictions, rather than simply choosing the most likely finger at each timestep independently, we also consider a series of extensions that feed recent predictions into the MLP alongside the contextualized embeddings produced by the LSTM, in the form of a vector embedding $c_i(\mathbf{x}_{\leq i}, \hat{\mathbf{y}}_{<i}) \in \mathbb{R}^d$.

Autoregressive Tagger: We start with a baseline autoregressive variant of our simple Bi-LSTM model in which the forward half of the Bi-LSTM acts as an encoder over prior predictions as well as notes. In this setup, we do away with the Bi-LSTM over notes $\mathbf{x}$, and instead replace it with a forward LSTM $f(\mathbf{x}, \hat{\mathbf{y}})$, and a backward LSTM $b(\mathbf{x})$. At each timestep i, the MLP is fed the concatenated output of both LSTMs $f_i(\mathbf{x}_{<i}, \mathbf{y}_{<i}) \oplus b_i(\mathbf{x}_{\geq i})$.

Prev Finger Embedding: As a precursor to a full checklist, we also experiment with a neural variant of a Maximum Entropy Markov Model (MEMM) [10]. Under this setup, we map the predicted label of the previous timestep $\hat{y}_{i-1}$ to a corresponding vector embedding, and concatenate this with the contextualized embedding of x_i produced by the Bi-LSTM, before passing the result to the MLP. This approach only takes into account the most recently used finger and ignores timing, but also requires the fewest additional model parameters.

Binary Checklist: This checklist embedding is a 5 dimensional vector, where each dimension corresponds to a finger. The ith dimension is set to 1 if that finger has recently been predicted for a note of a higher pitch than the current, a -1 if it has recently been predicted for a lower note, or a 0 otherwise. We empirically find that "recent" is best defined as within 100 milliseconds.

Distance Checklist: This representation is a concatenation of 5 vector embeddings, one for each finger. These finger embeddings use the lattice input representation strategy, where the first half corresponds to the horizontal distance between the note that that finger was recently predicted for and the current one, and the second half corresponding to the vertical distance. If a finger has not recently been used, its section in the checklist vector will be filled by a learnable "dummy" embedding.

4. LEARNING AND EVALUATION

Next we will detail the additional metrics that we introduce, and show how by using reinforcement learning we can train our model to optimize some of them directly.

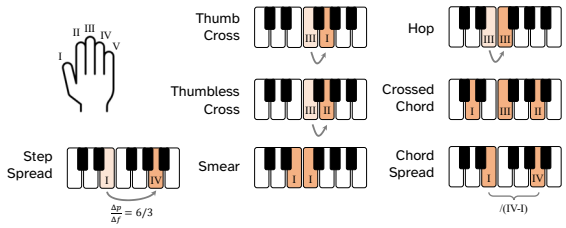


Figure 3. Fingering patterns tracked by our fluency metrics. Arrows and shading indicate sequential notes.

4.1 Fluency Metrics

We now describe the metrics we use to measure model performance. While research into pianists' fingering strategies shows that decisions are mostly motivated by minimizing ergonomically unfavorable patterns [2, 7], prior work has primarily evaluated on per-note labeling precision averaged over annotators, referred to as M_{gen} (albeit with some different strategies for handling disagreements between annotators [3]). However, by simply checking if each prediction agrees with any annotator's label for that note in isolation, we miss whether the model's predictions are coherent with one another. For instance, a model may output two adjacent fingers which each agree with an annotator's label, but not the same one, creating a pattern that is harder to play than if it had solely aligned with one of them, despite these having the same M_{gen} score. Also, a model may output a sequence that agrees with none of the annotators but is at least playable, and yet receives the same score as a physically impossible sequence. This is especially an issue for songs labeled by fewer annotators.

To address this, we expand the scope of our evaluation to include metrics that also measure how *fluent* model predictions are within themselves. To this end, we track statistics on the frequency of several types of output pattern that would increase the physical difficulty for a human performer, but may not be reflected in the raw labeling precision score. This allows us to more holistically compare models in ways that are both attentive to sustained agreement with annotators, and also expose the specific ways in which their predictions are more or less playable. Visual examples of these are provided in Figure 3 for clarity.

4-gram: Anchored 4-gram precision is the proportion of subsequences of four consecutive notes that were all predicted correctly with respect to a single annotator. This metric measures output coherence, but is still directly tied to the gold labels. Where M_{gen} scores plateau as they approach the inter-annotator agreement of 71.4, 4-gram precision has more headroom and better stratifies models.

Thumb/Thumbless Crosses and Crossed Chords: Thumb crosses are where a note lower in pitch than the previous is played with a "higher" finger or vice versa, one of those two fingers being a thumb. These shift the hand up or down the keyboard without fully lifting off of the keys. We also track crosses where neither finger is a thumb, which is much less common due to their difficulty. We refer to cases where two fingers are crossed and the notes overlap in time as crossed chords. These are only performed under very specific circumstances, as most chords are not physically playable with crossed fingers.

Hop: These are cases where one finger is used to play two different consecutive notes. While not intuitively difficult, these do not allow the player to rest their hand in a fixed spot, and make distances harder to accurately judge, limiting the maximum tempo of a piece, similar to how hunt-and-peck typing is generally slower.

Smear: In a smear, more than one note within a chord is played by a single finger. These are generally utilized when two adjacent keys are both in the chord, but because of the placement of the other notes it is hard to use two separate fingers. While these are extremely rare in our gold labels, naive baselines tend to produce them quite frequently.

Step/Chord Spread: Step spread tracks how far the player must stretch their fingers apart while playing. Specifically, it is the average number of semitones per finger separating any two adjacent but not overlapping notes. For example, if a song contains an `E` followed by the `B` a fifth above it, the step spread would be 7 for a model that predicts playing them with fingers 1 and 2 respectively, but only a 3.5 for a model that predicts 1 and 3. We measure similar cases where the notes do overlap in time as chord spread. A large average chord spread is more challenging since it requires stretching the hand over a larger distance.

4.2 Loss Functions

Having defined our model as above, we can train it on the cross entropy loss between the predicted distribution $p(\mathbf{y}|\mathbf{x};\theta)$ and the gold labels by simply backpropagating to obtain parameter gradients and taking gradient descent updates on our training set. Autoregressive models are teacher forced at train time (i.e. conditioned on gold labels from prior timesteps in lieu of the model's own previous predictions), and we decode with beam search at test time.

However, while training on cross entropy can yield strong performance on labeling precision, we find that this is not consistently correlated with other non-differentiable metrics which we may also wish to optimize with respect to. We therefore also investigate using a supplemental loss function based on the REINFORCE algorithm [12] which measures the frequency of undesirable fingering patterns in predicted outputs. While REINFORCE is traditionally associated with environment navigation, it has seen success in sequence generation tasks as well [13–15].

Under this framework, the loss function is formulated as an expected reward, which we approximate by sampling a sequence $\tilde{\mathbf{y}}$ from our output distribution (using ancestral sampling) and weighting its reward by its likelihood.

$$L = \frac{(\bar{r} - r(\mathbf{x}, \tilde{\mathbf{y}}))}{N} \sum_{i=1}^{N} \log p(\tilde{y}_i | \tilde{\mathbf{y}}_{<i}, \mathbf{x}; \theta)$$

Where $\bar{r}$ is the rolling average reward over the past 50 examples, serving as our reward baseline. In order to ensure stronger signal during training, we also calculate the REINFORCE loss over 10 note chunks at a time, rather than over an entire song as this would make the attribution of reward to specific decisions less direct. In our experi-

Model	Thumb Cross (#)	Thumbless Cross (#)	Crossed Chord (#)	Hop (#)	Smear (#)	Step Spread (s/f)	Chord Spread (s/f)	4-gram (Acc)	M_{gen} (Acc)
Gold	331	58	28	155	7	2.78	2.64	100	100
Bi-LSTM (MIDI) [3]†	228	84	47	485	148	2.63	2.56	66.7	27.8
Bi-LSTM (Note $\oplus$ Octave)	**257**	90	**36**	394	135	**2.69**	2.61	67.0	28.3
Bi-LSTM (Note $\oplus$ Rel Dist)	218	87	52	390	101	2.65	2.59	69.1	32.2
Bi-LSTM (Lattice)	217	**43**	40	**334**	**74**	2.66	**2.62**	**69.6**	**33.1**

Table 1. Results on PIG test set for Bi-LSTM model with different input representation schemes. The first row represents the value of each metric under the gold fingerings produced by human annotators. Values in bold are the closest to the gold labels' score for that metric. †This model is based specifically on the neural approach used in [3], but is our own implementation of it. It was similarly reimplemented by [4] in their work. Units for most metrics are simple frequency counts, except for spreads which are in semitones per finger, and 4-gram and M_{gen} which are accuracies.

ments, we set the reward to be a function of the number of hops and smears (see Section 4.1):

$$r(\mathbf{x}, \mathbf{y}) = \exp(-\#\{i : x_i \neq x_{i+1}, y_i = y_{i+1}\} - \#\{i : y_i = y_{i+1}, t_i^+ > t_{i+1}^-\})$$

Because training solely on the REINFORCE objective can lead the model towards degenerate solutions that trivially minimize undesirable patterns while compromising predictive accuracy, we take an approach similar to [15] and [19] where both loss functions are summed into a mixed training objective. This provides a reasonable tradeoff between the predictive signal of cross entropy, and the pressure towards fluent output given by REINFORCE. Mixed objective runs are warm-started on just the cross entropy loss to avoid degenerate solutions.

5. EXPERIMENTAL SETUP

5.1 Implementation Details

We use 2 layer LSTMs and 2 layer MLPs with a hidden size of 1024, and a dropout of 0.2 in both of these networks. We use d=256 for our input embedding size. Beam search is performed to decode autoregressive models at test time using a beam size of 10. Models are trained using the Adam optimizer [20] with a learning rate of $1e{-}4$. Our code is implemented in PyTorch `1.8.1` [21] and trains on an NVIDIA 2080ti GPU in roughly 12 hours.

5.2 Data

We train and evaluate on the PIano fingerinG dataset (PIG) [3] which contains 150 piano songs written by 24 composers. Each song has up to 6 fingerings produced by human pianists, yielding 309 annotated songs in total. Because of the imbalance in the dataset's original splits, we use alternate splits created by [4] which increase the relative size of the train and validation sets. As a baseline we compare to a reimplementation of the LSTM model of [3] (which was also reimplemented by [4] in their experiments), albeit with the same architecture of our other systems. We also show results from the third order HMM implementation of [3], although the more similar neural models are our primary point of direct comparison.

We employ similar preprocessing as prior work. Rather than modeling left and right hands separately, we reflect the pitches of all left hand parts and reverse the corresponding finger labels, thereby constructing a second "right hand part" which we treat as an independent song. This prevents overfitting by halving the label space. To handle chords, i.e. multiple notes with identical onsets, we simply arpeggiate them from lowest to highest pitch. Otherwise, notes are ordered by onset. We do however retain the original timing information for constructing accurate checklists.

6. RESULTS

6.1 Input Representation Ablation

We start by investigating the effects of input representations on a simple Bi-LSTM baseline model that outputs independent prediction probabilities at each timestep. Table 1 shows our results across all metrics. Overall the lattice representation does best, followed by note $\oplus$ octave. This matches our hypothesis that learning a separate embedding for each note on the keyboard leads to overparameterization, and that modular representations are more effective. We also see that relative distance based embeddings lead to fewer hops and smears specifically, which makes sense given that it allows the model to more easily observe when notes are or are not repeated, and therefore whether repeating the same finger would make sense.

6.2 Checklist Models

We see in Table 2 that the checklist models tend to do best overall, with Binary Checklist using REINFORCE getting the highest 4-gram score among LSTMs. All the autoregressive systems significantly outperform the Bi-LSTM baselines, but with different tradeoffs in terms of the output patterns we measure. REINFORCE minimizes the number of hops and smears compared to just cross entropy, and in doing so can sometimes boost other metrics as well.

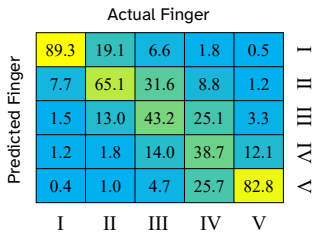
Note that M_{gen} fails to fully reflect what are at times substantial differences in model output made apparent from the other metrics. For instance, the Prev Finger Embedding produces fewer hops, smears, and crossed chords than the Autoregressive tagger, likely indicating that it would be far easier for a human to perform despite having worse M_{gen}. This is especially apparent when comparing the checklist models to the Bi-LSTM baselines. Also while

Model	Thumb Cross (#)	Thumbless Cross (#)	Crossed Chord (#)	Hop (#)	Smear (#)	Step Spread (s/f)	Chord Spread (s/f)	M_{gen} (Acc)	4-gram (Acc)
Gold	331	58	28	155	7	2.78	2.64	100	100
Bi-LSTM (MIDI) [3]†	228	84	47	485	148	2.63	2.56	66.7	27.8
HMM-3 [3]‡	220	32	31	196	84	2.84	2.72	**70.2**	**39.5**
Autoregressive Tagger	278	88	49	**168**	58	2.74	2.67	68.3	36.7
(+REINFORCE)	274	66	49	59	18	2.70	2.63	68.7	36.5
Prev Finger Embedding	271	**56**	40	127	30	**2.78**	2.67	68.1	35.5
(+REINFORCE)	283	53	54	53	**12**	2.81	2.76	68.4	36.5
Binary Checklist	227	72	**26**	332	74	2.67	2.58	69.3	35.8
(+REINFORCE)	261	49	**26**	217	42	2.69	**2.64**	69.5	37.1
Distance Checklist	274	72	37	195	37	2.80	2.69	68.8	36.6
(+REINFORCE)	**284**	78	41	134	32	2.75	2.68	69.0	36.1

Table 2. Results on PIG test set for various autoregressive setups, shown with and without using REINFORCE to minimize hops and smears (see Section 4.2). ‡ A third order HMM implementation by [3] is also included – while it ranks similarly as in prior work against LSTMs by M_{gen}, it falls behind on other metrics.

the HMM does strongly on M_{gen} and 4-gram, we only see its deficits when we look at other metrics which reveal subtle issues such as an especially high finger spread and number of smears, and low utilization of thumb crosses.

6.3 Label Confusion



This figure shows the confusion matrix for predictions by the Binary Checklist model with REINFORCE. The distribution of misclassifications is fairly different for each finger; adjacent fingers are more likely to be confused than ones that are further apart, reflecting a degree of interchangeability in how pianists use them. We also see that thumb and pinky predictions are most accurate, perhaps because being at the ends of the hand makes them less versatile.

6.4 Human Evaluation and Conclusion

Model	Physical Comfort	Mechanical Efficiency	Ease of Learning
Bi-LSTM (MIDI) [3]	1.6	1.4	1.6
Binary Checklist (+REINFORCE)	**2.8**	**2.5**	**2.9**

Table 3. Human evaluation ratings across various criteria.

Since the ultimate success criteria of our task is to produce fingerings that are well-suited to human performance, we also present results from a small scale round of human evaluation. Specifically, we recruited a college professor of piano with a doctorate in piano performance (who was not involved with the creation of the original dataset) to assess the outputs of two models—the Bi-LSTM (MIDI) [3] and Binary Checklist with REINFORCE—according to three main criteria: physical comfort, mechanical efficiency, and ease of learnability. The pianist was asked to score the models' outputs with respect to each of these categories on a scale of 1-3 for all 10 songs in the validation set (composed by Brahms, Mendelssohn, and Rachmaninoff).

We show the models' average ratings per category in Table 3. The strong contrast suggests that the checklist model is not just predicting gold label fingers more frequently (shown by M_{gen}) but also that it is producing more locally coherent outputs in a way that substantially improves performability. Furthermore, the correlation of human judgements with our proposed metrics indicates that they are meaningful measurements of output quality.

The pianist also provided qualitative observations which we summarize. He noted that both systems performed best on simpler, repetitive passages such as those found in Mendelssohn. Our model also reportedly produced fewer difficult stretches, which we expect as it is directly aware of interval distances in its input representation. Another observed issue was that the Bi-LSTM's fingerings did not consider tempo, often containing sections that would be impossible to play fast enough, especially for Rachmaninoff pieces. We suspect our model's advantage is from the checklist's dependence on absolute timing, not just the ordering of notes. The pianist also reported that our model more frequently used the same fingering for repeated notes and chords, whereas the Bi-LSTM would often change fingerings without reason, which can also likely be explained by our conditioning on recent predictions instead of decoding them independently. Quantitatively, we found that REINFORCE was able to reduce the number of hops and smears, which was confirmed by the pianist to have a significant impact on playability, emphasizing the importance of integrating these into the training loop instead of purely relying on a traditional likelihood loss.

Finally, he noted that the models particularly struggled on successive chords (as opposed to stepwise passages). Chord transitions require fingerings that are not just comfortable in isolation but also connect together. This requires the model to reason about *voices* within the same hand, something which may require moving beyond a framework that treats songs as linear sequences of individual notes, and is more sensitive to polyphony.

7. REFERENCES

[1] R. Parncutt, J. A. Sloboda, E. Clarke, M. Raekallio, and P. Desain, "An ergonomic model of keyboard fingering for melodic fragments," *Music Perception*, vol. 14, pp. 341–382, 1997.

[2] J. Sloboda, E. Clarke, R. Parncutt, and M. Raekallio, "Determinants of finger choice in piano sight-reading," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 24, pp. 185–203, 02 1998.

[3] E. Nakamura, Y. Saito, and K. Yoshii, "Statistical learning and estimation of piano fingering," *Information Sciences*, vol. 517, pp. 68–85, 2020.

[4] A. Moryossef, Y. Elazar, and Y. Goldberg, "At your fingertips: Automatic piano fingering detection," 2019.

[5] H. Zhao, X. Guan, and Q. Li, "Estimation of playable piano fingering by pitch-difference fingering match model," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2022, pp. 1–13, 2022.

[6] D. G. Barbara, Ugenio, B. Justin, and Adgerow, "Expected reciprocal rank for evaluating musical fingering advice," *Sound and Music Computing Conference*, 2021.

[7] E. Clarke, R. Parncutt, M. Raekallio, and J. A. Sloboda, "Talking fingers: An interview study of pianists' views on fingering," *Musicae Scientiae*, vol. 1, pp. 107 – 87, 1997.

[8] E. Nakamura, N. Ono, and S. Sagayama, "Merged-output hmm for piano fingering of both hands." in *ISMIR*, 2014, pp. 531–536.

[9] Y. Yonebayashi, H. Kameoka, and S. Sagayama, "Automatic decision of piano fingering based on a hidden markov models," in *IJCAI*, 2007.

[10] A. McCallum, D. Freitag, and F. C. Pereira, "Maximum entropy markov models for information extraction and segmentation," in *ICML*, 2000.

[11] C. Kiddon, L. Zettlemoyer, and Y. Choi, "Globally coherent text generation with neural checklist models," in *EMNLP*, 2016.

[12] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, pp. 229–256, 2004.

[13] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *CoRR*, vol. abs/1511.06732, 2016.

[14] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1179–1195, 2017.

[15] R. Paulus, C. Xiong, and R. Socher, "A deep reinforced model for abstractive summarization," *ArXiv*, vol. abs/1705.04304, 2018.

[16] N. Jaques, S. Gu, R. E. Turner, and D. Eck, "Generating music by fine-tuning recurrent neural networks with reinforcement learning," in *Deep Reinforcement Learning Workshop, NIPS*, 2016.

[17] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *AAAI*, 2017.

[18] G. L. Guimaraes, B. Sánchez-Lengeling, P. L. C. Farias, and A. Aspuru-Guzik, "Objective-reinforced generative adversarial networks (organ) for sequence generation models," *ArXiv*, vol. abs/1705.10843, 2017.

[19] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. R. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. S. Corrado, M. Hughes, and J. Dean, "Google's neural machine translation system: Bridging the gap between human and machine translation," *ArXiv*, vol. abs/1609.08144, 2016.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

Papers - Session V

SONUS TEXERE! AUTOMATED DENSE SOUNDTRACK CONSTRUCTION FOR BOOKS USING MOVIE ADAPTATIONS

Jaidev Shriram **Makarand Tapaswi** **Vinoo Alluri**
International Institute of Information Technology, Hyderabad
https://auto-book-soundtrack.github.io/
jaidev.shriram@students.iiit.ac.in, makarand.tapaswi@iiit.ac.in, vinoo.alluri@iiit.ac.in

ABSTRACT

Reading, much like music listening, is an immersive experience that transports readers while taking them on an emotional journey. Listening to complementary music has the potential to amplify the reading experience, especially when the music is stylistically cohesive and emotionally relevant. In this paper, we propose the first fully automatic method to build a dense soundtrack for books, which can play high-quality instrumental music for the entirety of the reading duration. Our work employs a unique text processing and music weaving pipeline that determines the context and emotional composition of scenes in a chapter. This allows our method to identify and play relevant excerpts from the soundtrack of the book's movie adaptation. By relying on the movie composer's craftsmanship, our book soundtracks include expert-made motifs and other scene-specific musical characteristics. We validate the design decisions of our approach through a perceptual study. Our readers note that the book soundtrack greatly enhanced their reading experience, due to high immersiveness granted via uninterrupted and style-consistent music, and a heightened emotional state attained via high precision emotion and scene context recognition.

1. INTRODUCTION

In 1975, a short repetitive piece of orchestral music gained notoriety for inducing fear in its listeners, putting them in a suspenseful state of impending doom. Today, it still induces the same reaction, right when the iconic shark of the movie *Jaws* appears on screen. Movie composers, such as John Williams who created the *Jaws* theme, have long been aware of the impact music has on its listeners. Well-placed music can accentuate scenes to raise the emotional stakes or foreshadow upcoming events; ill-suited music can even suggest that something evil is afoot [1]. When taken as a whole, the musical imagery afforded by such soundtracks greatly complements the movie watching experience. In this paper, we attempt to answer whether we can create a similar experience for books (*cf*. Fig. 1).

© J. Shriram, M. Tapaswi, and V. Alluri. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J. Shriram, M. Tapaswi, and V. Alluri, "Sonus Texere! Automated Dense Soundtrack Construction for Books using Movie Adaptations", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.*, Bengaluru, India, 2022.

Figure 1. We aim to transport readers to a musical universe by building a dense and coherent book soundtrack that is borrowed from movie adaptations.

Not unlike movies, reading literature can be an incredibly transportive process that puts one in a meditative state, while actively engaging their imagination and mind [2]. Apart from the lack of visuals, books share many similarities from a music composer's perspective: they have recurring themes, characters (allowing for unique *leitmotifs* [3]), and even long-drawn emotional and narrative arcs that can be teased and reinforced musically. Still, it is uncommon to see soundtracks that are tailored for books; it is up to the reader to curate their own playlist and craft their experience. Inconvenience aside, this requires the reader to preempt the type of music expected for a book's chapter and switch songs at relevant plot points. This is simply infeasible. In this work, we resolve these issues by proposing the first automatic system to build a dense soundtrack that plays throughout the entire reading duration of a book.

Specifically, we focus on building a soundtrack for books with movie adaptations. This allows us to draw on the corresponding movie soundtrack and take advantage of the composer/director's musical intents and instincts for the *same story*. However, adapting the soundtrack is far from trivial due to missing alignment between book parts and music segments. Our approach resolves this by finding scenes in the movie adaptation that match parts of the book and searching for music that plays in that movie scene. This results in high quality, narrative specific matches that amplify the reading experience. We conduct our experiments on *Harry Potter and the Philosopher's Stone*, cho-

sen due to its popularity and Academy Award nominated soundtrack. We evaluate the quality of our soundtrack by asking participants to read two music-accompanied chapters of the book followed by a semi-structured interview.

Our contributions can be summarised as follows: (i) We propose the first fully automatic approach for constructing book-long soundtracks. (ii) Our pipeline matches the story in the book and movie to lift musical cues from the movie adaptation. Gaps in this alignment are filled through an emotion-driven music retrieval system. (iii) A perceptual study of the soundtrack validates our proposed approach and provides further insights into soundtracking for books.

2. RELATED WORK

Soundtracking for books. Methods for constructing book (or story) soundtracks can be divided into two: generative or retrieval based. Generative approaches typically parse the text for concepts and provide that as input to a music generation pipeline. Topic extraction followed by sentiment analysis [4] or density estimation of emotion-related words [5] are used to generate melodies. On the other hand, retrieval based approaches, mine text for semantic concepts that are used to retrieve ambient music (sometimes with pitch correction) [6] or apply similar ideas to Twitter texts [7]. The general idea of cross-modal text-to-audio retrieval [8] is adapted for tag-based music [9] or characterises documents and music as a distribution over emotions [10]. Recently, Won *et al*. [11] propose a joint emotion-driven embedding space for story sentences and music enabling cross-modal retrieval. However, understanding emotions conveyed in a book depends on the larger narrative requiring a context of more than a few sentences. Thus, while [11] could be used to construct dense book soundtracks in theory, the potential switching of music and resulting change in emotion every few sentences may not result in a good user experience.

Unlike above approaches, we focus on retrieving music from the soundtrack of a book's movie adaptation. Our approach assembles coherent music for books by considering large chunks of the book chapter, aligning them with movie scenes, and finding matched music pairs. We fill in the gaps (only unmatched chapter segments) by relying on emotion-based matching. To the best of our knowledge, we are the first to perform narrative-specific music retrieval and *weave* a soundtrack for the entire book.

Multi-modal music recommendation systems use cues such as user location, time, and environmental information to play music in day-to-day life [12] or even traffic conditions for music in cars [13]. Perhaps closest to our work, PICASSO is a fine example of a ranking model trained on pairs of matching music and movie-clips (images, subtitles) that is used to provide music recommendations for image slideshows or audio books [14]. Instead, our work focuses on producing a *dense* book soundtrack by aligning narrative components of the book with the movie adaptation. This results in high precision narrative matches and ensures a coherent soundtrack adapted from the same story.

3. ASSEMBLING SOUNDTRACKS FOR BOOKS

A large book may take several (variable) hours for a reader to complete. Within the book, there are multiple chapters, each having sections indicated by the theme, emotion, or location. An ideal soundtrack should respect these scene boundaries and change accordingly. On the music front, while different tracks from the soundtrack are typically homogeneous, they are rarely played in their entirety in the movie at a single stretch. A track may include the score for the setup, conflict, climactic moment, and resolution for a certain storyline, which composers will selectively play at opportune moments to maximise emotional payoffs.

We use the movie adaptation as an intermediary between the two that links plot points from the book with snippets of music from the soundtrack album. We first divide each modality - the book, music soundtrack, and the movie into smaller segments (Sec. 3.1). Then, we obtain clues about the soundtrack associated with each plot point in the book by aligning both the book and soundtrack to the movie (Sec. 3.2). Finally, we weave together the music for the book by combining movie-based and emotion-based matches (Sec. 3.3). Fig. 2 illustrates this overall flow.

3.1 Segmenting the Book, Music, and Movie

We discuss methods for segmenting the book chapters $\mathcal{B}_i$ in a book $\mathcal{B} = [\mathcal{B}_1, \ldots, \mathcal{B}_L]$; identifying cohesive musical segments within each track M_j of the album $\mathcal{M} = [M_1, \ldots, M_P]$; and identifying scene boundaries for a movie $\mathcal{V} = [V_1, \ldots, V_Q]$ that facilitates the alignment.

Book narrative segmentation. We divide the text in a book chapter based on narrative-relevant factors such as theme, location, activity composition, character constellation, or even time [15]. Due to the absence of large datasets for this task, we adopt an unsupervised approach, temporally weighted hierarchical clustering (TW-FINCH) [16], recently shown to be successful on video activity segmentation. For a chapter $\mathcal{B}_i$, we encode each paragraph using a pretrained language model $\phi_{LM}(\cdot)$ (specifically MPNet [17] that performs well for sentence embeddings and semantic search [18]), and cluster semantically similar and spatially close paragraphs to produce disjoint partitions,

$$\{\mathcal{B}_i^p\}_{p=1}^{K_i} = \text{TW-FINCH}\left(\phi_{LM}(\mathcal{B}_i)\right) . \quad (1)$$

We align individual segments $\mathcal{B}_i^p$ with music segments.

We also considered a few baselines but ignored them due to inferior results. TextTiling [19] tended to uniformly partition chapters while TopicTiling [20] often resulted in over-segmentation. In contrast, our approach yielded segments that mostly respected narrative shifts.

Keystrength based music segmentation. As mentioned earlier, the entire track M_j from an album is (almost) never played in a portion of the movie. Hence, similar to book chapters, we focus on creating homogeneous emotionally-cohesive track segments that can be played continuously.

A key feature that music segments are required to reflect are emotions that the director/composer intends to convey. Since we expect homogeneity in music segment

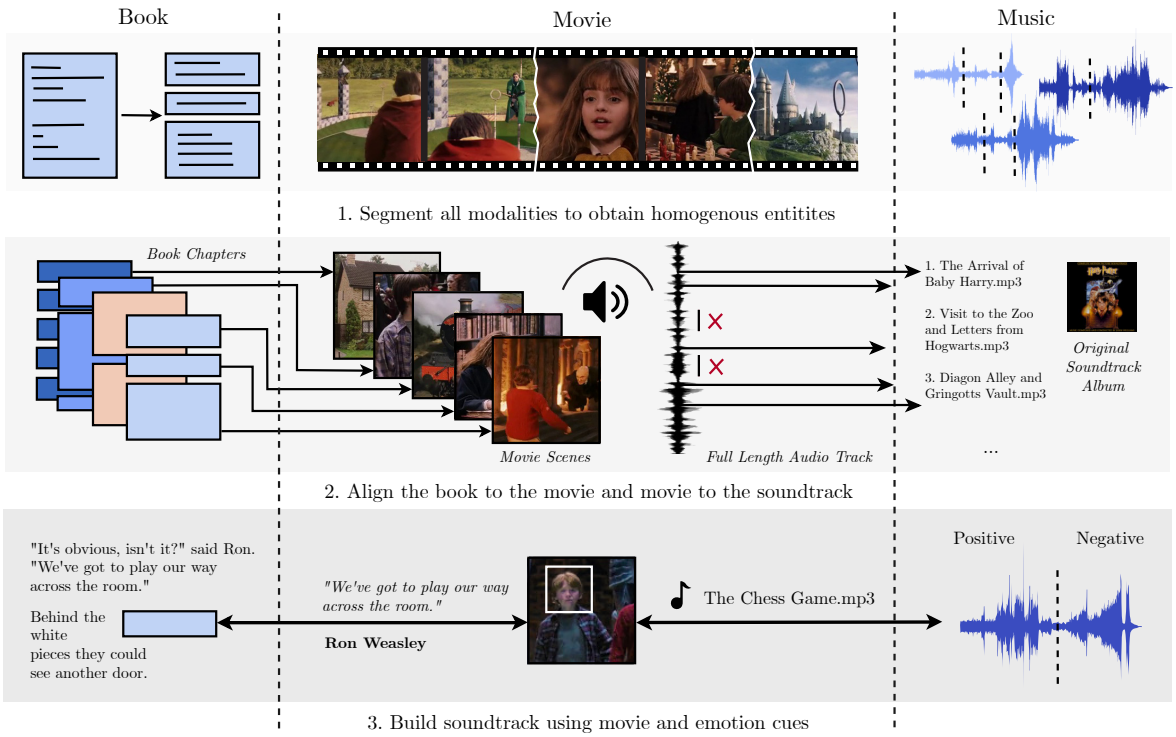


Figure 2. **Overview of approach.** We present a novel technique to build a musically rich book-length soundtrack. We accomplish this by first segmenting the book, its adaptation, and music into smaller homogenous chunks. These segments are then matched with the movie acting as an intermediary for text and music, thereby producing the final soundtrack.

emotions, *keystrength* [21] is a good attribute as it captures the tonal properties of music, and tonal changes likely suggest emotional shifts (*e.g.* major to minor mode suggesting a potential shift from positive to negative valence [22]). *Keystrength* as extracted via the MIRToolbox [23] gives us a probability for each possible key resulting in a 24 dimensional vector (12 major and 12 minor keys). We use a window size of 10 seconds with an 85% overlap, that helps ignore minor local variations in the track while retaining larger shifts. We then compute a self-similarity matrix [24] of the time-varying keystrength vector which is then used to calculate the novelty curve, with a kernel size of 64 [24]. The peaks in the novelty curve determine points of change. Finally, we use those peaks as our segment boundaries to produce L_j music segments $\{M_j^c\}_{c=1}^{L_j}$ (see Fig. 3).

Movie scene detection. As a third step, we segment the movie into narrative-coherent scenes $[V_1, \ldots, V_Q]$ using the approach described in [25]. First, the movie is divided into shots (consecutive video frames from the same camera). Then, a dynamic programming algorithm is used to find scene boundaries (a scene consists of multiple shots) so as to maximise intra-scene similarity.

Summarising, we denote $\mathcal{B}_i^p$ as the segment of book chapter $\mathcal{B}_i$; M_j^c as the music segment from track M_j; and V_q as the q^{th} scene from the movie $\mathcal{V}$.

3.2 Aligning the Book, Movie, and Music

We first align the book with the movie adaptation using a new two-stage coarse-to-fine alignment scheme. Then, we also align the movie audio to the soundtrack album.

Chapter-scene coarse alignment. We first assign a set of scenes from the movie to each book chapter. We use an approach similar to [26] where pairwise similarities are computed between a book chapter $\mathcal{B}_i$ and a video scene V_q via character histograms and matched dialogues. The chapter-scene relationship is then encoded as a graph (each node represents a chapter-scene pair), with edge weights representing similarity scores. Calculating the shortest path over this graph provides the alignment between all chapters and scenes. Additional details are provided in the Appendix.

Paragraph-scene refinement. The coarse alignment cannot be used directly for soundtracking as a chapter $\mathcal{B}_i$ contains distinct segments $\mathcal{B}_i^p$ that likely need different music. Thus, in addition to sparse dialogue matches, we compute similarities between sentences in the chapter segment $\mathcal{B}_i^p$ and frames of the video scene V_q using a pretrained vision-language model (CLIP [27]). To improve the quality of CLIP matching scores, we prune dialog and mundane sentences from the chapter segment using a TF·IDF based scoring system. This emphasises relatively rare characters and objects that are likely to give stronger matches than commonly occurring ones. We also retain sentences with a high *concreteness* index [28] that measures how likely a word can be seen or experienced (in contrast to *abstract* words). Finally, we take the top remaining sentences, encode them with CLIP, and calculate cosine similarity with all CLIP encoded video shot frames in the chapter's scenes (see Fig. 4). Scenes with a score higher than θ are assigned to the text segment using a mapping function,

$$A(\mathcal{B}_i^p) = \{V_q : \text{CLIP}(\mathcal{B}_i^p, V_q) > \theta\}\,. \tag{2}$$

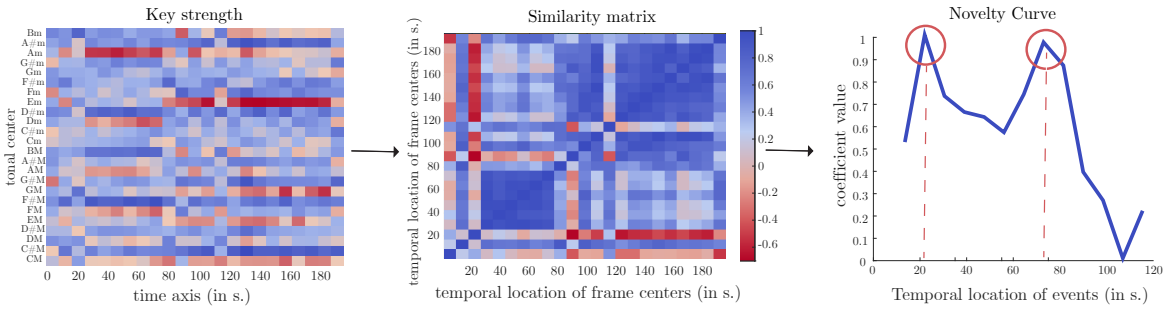


Figure 3. **Music segmentation pipeline.** We segment all tracks from the soundtrack to ensure a cohesive listening experience. We extract keystrength [21] that captures tonal properties of a soundtrack (left), compute the self-similarity matrix (center), and use that to calculate the novelty curve [24]. The peaks of this curve are used to segment the track.

Aligning the movie audio track with the soundtrack album. We identify the music in the movie by passing its audio track through Shazam [1] , a popular commercial audio-fingerprint based search application [29] using its free public API. This results in high-precision estimates for the music played every few seconds. While any audio fingerprinting approach would perhaps work, we found that Shazam was quite accurate in the presence of background noise and dialogue, which is pervasive in movies.

3.3 Weaving the Book Soundtrack

Post alignment with the movie, book chapter segments $\mathcal{B}_i^p$ belong to one of two sets: (i) those associated with at least one movie scene $\mathcal{S} = \{\mathcal{B}_i^p : |A(\mathcal{B}_i^p)| \geq 1\}$; and (ii) those without, $\bar{\mathcal{S}} = \{\mathcal{B}_i^p : A(\mathcal{B}_i^p) = \varnothing\}$. Recall, $A(\mathcal{B}_i^p)$ is the set of aligned video scenes to a book segment (*cf*. Eq. 2).

Extracting emotion labels. For all text segments $\mathcal{B}_i^p$, we classify each paragraph into positive, neutral, or negative using a BERT-based emotion classifier ϕ_{BERT}, trained on Reddit comments [30]. A majority vote across paragraphs is used to assign the emotion label to the chapter segment $E_{\text{book}}(\mathcal{B}_i^p) = \text{mode}(\phi_{\text{BERT}}(\mathcal{B}_i^p))$. For a music segment, similar to [5], we encode it's emotion as valence, $E_{\text{music}}(M_j^c)$, based on the mode of the song (major or minor). This is based on literature that indicates that tracks in minor tend to be associated with negative emotions and tracks in major with positive emotions [22]. We also tried approaches that predict emotion from audio [11, 31], but they didn't work as well for our application.

Importing music snippets from the video scene. For every text segment $\mathcal{B}_i^p \in \mathcal{S}$, we extract the movie timestamps corresponding to the matched dialogues or CLIP-based frame-sentence pairs. We use the audio-search to identify the overall track M_j being played at any of the above timestamps in the movie (in a small neighbourhood). A specific music segment M_j^c is chosen by matching emotion predictions *i.e.* $E_{\text{book}}(\mathcal{B}_i^p) = E_{\text{music}}(M_j^c)$ (we pick one randomly if there are several segments). While we can pick emotion-matching music segments without the book-movie alignment, it will likely result in spurious matches.

Emotion-based retrieval. For the chapter segments that are not aligned with any video scene, $\bar{\mathcal{S}}$, and those in $\mathcal{S}$ that did not find an emotion compatible soundtrack, we assign a random music segment among the set of emotionally compatible compositions. Note that while we can pick any music segment (even from different movies), we restrict to the soundtracks for this movie maintaining the composer's stylistic coherence.

[1] https://www.shazam.com/

4. RESULTS AND EVALUATION

Our approach is designed to be applicable to books that have mostly faithful movie adaptations in terms of few matching dialogues, a relatively similar plot, and at least some matching characters. We evaluate it on the first book and movie pair in the *Harry Potter* series, *Harry Potter and the Philosopher's Stone*.

4.1 Harry Potter: A Case Study

Our unsupervised text segmentation approach splits 17 chapters into 87 segments, with an average of 5.11 segments per chapter. Assuming a reading speed of 250 words per minute, each segment requires a ~4 minute track, for a total soundtrack duration of ~6 hours. Music segmentation

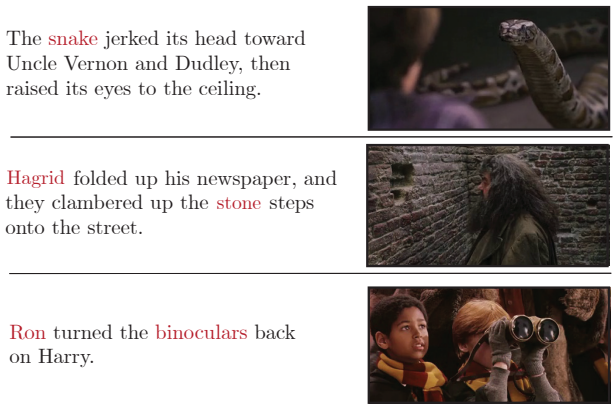


Figure 4. **CLIP-based paragraph-scene alignment.** Aligned examples of automatically selected visual sentences and their video frames in the scene using the vision-language model CLIP [27] in a zero-shot setting. Text highlights are words that we think that caused the match.

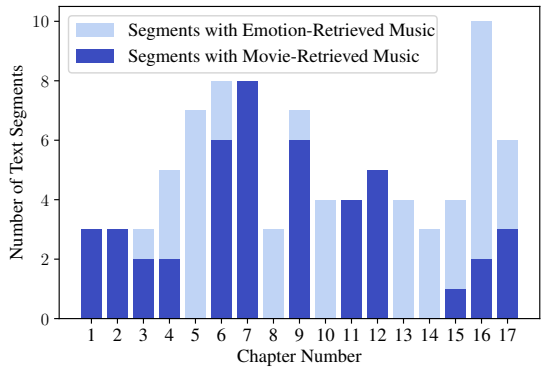


Figure 5. **Number of chapter segments with movie- vs. emotion-retrieved music.** Several chapters in the book are well represented in the movie and are predominantly soundtracked using movie cues. For others, we use an emotion-based retrieval method to build a soundtrack.

of 19 tracks from the soundtrack album results in 47 audio chunks with an average duration of ~52 seconds. For the movie, we obtain 120 scenes using the scene detection algorithm. Computing the book-movie alignment results in a strong chapter-scene alignment, with 82% accuracy, calculated as the percentage of movie shots correctly assigned to a chapter [26].

For the fine-grained alignment, 34 chapter segments have dialogue matches, while the CLIP based refinement results in a 130% improvement in the number of chapter segments associated with the movie; only 9 segments remain unmatched. After performing the final book-music alignment, 45 chapter segments have music imported from a matching video scene through the book-movie alignment. The remainder 42 segments are soundtracked using music segments fetched via emotion matching. See Fig. 5 for a chapter-wise distribution.

4.2 Experiment Design

As the primary goal of our method is to improve the reading experience, we obtain user feedback by setting up semi-structured interviews with 10 individuals, who each read two chapters of the *Harry Potter* book. We randomly picked one chapter for each person while the final book chapter was read by all. We chose the final chapter as it represents all aspects of our method: it includes music retrieved via movie cues as well as pure emotion based retrieval, apart from being a key chapter in terms of the plot. This allows us to fairly evaluate the effectiveness of our soundtrack at a book level as reading these chapters takes around an hour for each participant.

All participants are asked to read these chapters consecutively and answer a series of questions about several aspects of the reading experience. No user data is collected through this process and user consent is collected prior to recording the interviews. The study was also authorised by the author's institutional ethics committee.

Reading application. A bottleneck with playlist based approaches for soundtracking books is that they require user-input to transition at appropriate instants. We resolve this by creating an application that plays music in the background of the displayed text. Our application loops the music segment infinitely and cross-fades to the next as the reader moves on to the next text segment. This ensures complete immersiveness and lets participants with *varying* reading speeds truly enjoy the soundtrack. For readers to have a first-hand experience, we package this application for a few chapters at the project page.

Participant information. A total of 10 individuals (18-22 age range, 6 male, 4 female) volunteered for the study. Each participant had previously read the *Harry Potter* book without music and happened to be familiar with the movie as well. We also provided each participant with a small monetary reward as compensation for their time.

4.3 Findings

The soundtrack improved immersiveness. All participants reported that the soundtrack improved the immersiveness of the reading experience. This is remarkable as all participants noted that they typically do not listen to music while reading. Some participants were more receptive to the soundtrack than others and spoke glowingly about the movie-like immersion afforded by the music. These participants could recognise that the music played was similar to the score at the relevant movie scene. Others, who were positive but less movie-inclined, focused more on how the music *"set the environment"* (P4). Phrases such as *"set the mood"*, *"intensified the emotion"*, *"fit the vibe"*, and *"enhanced the experience"* were common among such participants.

> *"The music provided insight into the tone of the chapter and [...] beyond imagination, provided a soundtrack to what was read"* (P1)

Surprisingly, some participants reported that the first section of the random chapter that they read first threw them off initially, suggesting that there is an adjustment period to this experience. These participants reported that they were very comfortable with later sections (*"after the first 10-12 lines"* - P2) once they had gotten into a reading flow.

The soundtrack helped visualise the book. Some participants could recognise that the music played was similar to the score at the relevant movie scene. These participants were especially appreciative of the soundtrack and stated that it helped *"visualise the book, with movie like visuals"* (P1). Such participants typically appreciated the music's cohesiveness and were likely referring to the pleasing soundscape laid out by our soundtrack. One participant explicitly pointed out that the soundtrack helped visualise character interactions and that without, it would have just been *"two characters talking"*.

Many were also receptive to the recurring motifs and themes that appeared throughout, such as the central *Harry Potter* theme and stated that it helped imagine the fictional

world. Only one participant, P3, expressed complete disagreement with the music played in a text segment, for ironically the same reason, stating that the signature *Harry Potter* motif distracted them. Barring this, the same person spoke warmly about the remaining soundtrack, suggesting that the overtness of the motif, which permeates culture today, may be subject to individualistic preferences.

Music helped focus.

> *"I get distracted when reading so it helped me focus on certain parts"* (P5)

When describing the immersiveness of the reading experience, three participants specifically pointed out that the music helped them read continuously, without distractions that are typically present when reading in the absence of a soundtrack. It should be noted that few participants who described an aversion for listening to music during typical recreational reading specifically pointed out that they avoid pop music, suggesting that instrumental music, in general, may be better suited for reading purposes.

Moderate repetition is not a concern. Most participants noticed repetitive music in some text segments that they read but only brought it up when explicitly asked. One participant suggested playing a different, similarly composed track instead of repeating the music, but in agreement with the rest of the pool, felt that repetition did not affect the experience. These participants also said that the repetition wasn't glaringly obvious and that it did not distract or take away from the immersion. It should be noted that repetition refers to the same track being played on a cross-fade loop while a person reads a text segment, due to variable reading speeds. Despite this, it is noteworthy that our musical segments are homogeneous enough to be played repeatedly and that the text segments are narratively cohesive to warrant such repetition. When specifically asked about the diversity of the music played, all participants expressed positive opinions and noted that the tracks kept changing as and when required.

Narrative transitions mirrored in music. Many participants engaged in a conversation with the authors post-interview about how the soundtrack was built. All such participants were startled by the fact that the entire process was completely automatic. Their surprise was primarily due to the fact that the music was automatically matched with relevant areas in text and that it transitioned at appropriate narrative points.

> *"It actually made the experience better as the transition put you in the mood for the expected emotion - from melancholy to sad."* (P9)

When asked specifically about these narrative transitions during the interview, all participants agreed that it was emblematic of emotional or narrative shifts. Since the last chapter was common across all participants, there was strong consensus about the narrative shifts here especially, with participants noting that the music increased in tension as the final hero-villain clash developed and that it eased into more mellow, tender music once it was resolved.

> *"When the tension built in the plot, the music transitioned to match it."* (P1)

Some participants explicitly appreciated the foreshadowing made possible by the music, as suspenseful music at the start of a segment would precede a similar narrative plotline. Others simply elaborated on their earlier comments about the immersiveness and re-emphasised the high congruency between text segments and the matched music.

Low-arousal music preferred. We asked participants to describe the emotional suitability of the music, specifically, and in line with prior answers, and received a favourable response. The few critiques that were present revolved around the energy/arousal of the music played at certain instances. Some participants were dissatisfied with segments that were high in arousal, even though the valence matched, as it broke the immersion. They described how this music distracted them from reading and drew too much attention to the track itself. This is likely an important factor to consider for future work on soundtracking books.

Movie-based music cues stronger than emotion-based cues. At the end of the interview, participants were asked to describe their favourite and least favoured pieces of music from the two chapters, if any. The best segments were split between music retrieved via movie cues and those retrieved via emotion matching, with the former being more prominent. On the other hand, the least favoured segments, often mentioned only after the author's insistence, were typically emotion-based matches that were described in terms of their neutrality. This finding suggests that our approach can effectively retrieve narrative-specific music and that such a system is perhaps well-suited for book soundtracking, as opposed to pure emotion-based methods.

5. CONCLUSION

We presented a novel system to automatically weave a book-length soundtrack, using the music present in the relevant movie adaptation. Perceptual validation of the constructed soundtrack for the first book and movie pair of the *Harry Potter* series provided very positive feedback that validated several proposed design decisions and techniques for text, movie, and music processing. Participants in our perceptual study were particularly receptive to music that was fetched via the movie and uniformly stated that it improved the immersiveness of the reading experience, and even transported them to the fictional world.

Future work. While our method has been successful in generating a soundtrack for a book with a movie adaptation, it needs to be modified to make it work on books without adaptations. Future work can potentially use our approach to investigate common trends across books to determine new cross-narrative rules. Our approach can also be extended for human-in-the-loop collaborative soundtrack construction applications, though such a use case is beyond the scope of this work.

6. ACKNOWLEDGMENTS

We thank Siddarth Baasri for his valuable input in analysing motifs present in the *Harry Potter* soundtrack and Saravanan Senthil for illustrating Fig. 1.

7. REFERENCES

[1] M. G. Boltz, "Musical soundtracks as a schematic influence on the cognitive processing of filmed events," *Music Perception*, vol. 18, pp. 427–454, 2001. 1

[2] M. C. Pennington and R. P. Waxler, *Why reading books still matters: The Power of Literature in Digital Times*. Routledge, 2017. 1

[3] M. Chełkowska-Zacharewicz and M. Paliga, "Music emotions and associations in film music listening: The example of leitmotifs from The Lord of the Rings movies," *Annals of Psychology (Polish)*, 2019. 1

[4] J. Salas, "Generating music from literature using topic extraction and sentiment analysis," *IEEE Potentials*, vol. 37, pp. 15–18, 2018. 2

[5] H. Davis and S. M. Mohammad, "Generating music from literature," in *Workshop on Computational Linguistics for Literature @ European Association of Computational Linguistics (CLfL@EACL)*, 2014. 2, 4

[6] S. Harmon, "Narrative-inspired generation of ambient music," in *International Conference on Computational Creativity (ICCC)*, 2017. 2

[7] M. Thorogood and P. Pasquier, "Computationally created soundscapes with audio metaphor," in *International Conference on Computational Creativity (ICCC)*, 2013. 2

[8] A.-M. Oncescu, A. S. Koepke, J. F. Henriques, Z. Akata, and S. Albanie, "Audio retrieval with natural language queries," in *Interspeech*, 2021. 2

[9] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, "Multimodal Metric Learning for Tag-Based Music Retrieval," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021. 2

[10] R. Cai, C. Zhang, C. Wang, L. Zhang, and W.-Y. Ma, "MusicSense: Contextual Music Recommendation using Emotional Allocation Modeling," in *ACM International Conference on Multimedia (ACM MM)*, 2007. 2

[11] M. Won, J. Salamon, N. J. Bryan, G. J. Mysore, and X. Serra, "Emotion embedding spaces for matching music to stories," in *International Society for Music Information Retrieval (ISMIR)*, 2021. 2, 4

[12] S. Reddy and J. Mascia, "Lifetrak: Music in Tune with your Life," in *International Workshop on Human-Centered Media (HCM)*, 2006. 2

[13] L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K.-H. Lüke, and R. Schwaiger, "InCarMusic: Context-Aware Music Recommendations in a Car," in *International Conference on Electronic Commerce and Web Technologies (EC-Web)*, 2011. 2

[14] A. Stupar and S. Michel, "Picasso: Automated soundtrack suggestion for multi-modal data," in *International Conference on Information and Knowledge Management (CIKM)*, 2011. 2

[15] A. Zehe, L. Konle, L. K. Dümpelmann, E. Gius, A. Hotho, F. Jannidis, L. Kaufmann, M. Krug, F. Puppe, N. Reiter, A. Schreiber, and N. Wiedmer, "Detecting Scenes in Fiction: A new Segmentation Task," in *European Association of Computational Linguistics (EACL)*, 2021. 2

[16] S. Sarfraz, N. Murray, V. Sharma, A. Diba, L. Van Gool, and R. Stiefelhagen, "Temporally-weighted hierarchical clustering for unsupervised action segmentation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2

[17] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "MPNet: Masked and Permuted pre-training for Language Understanding," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 2

[18] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2019. 2

[19] M. A. Hearst, "Text tiling: Segmenting text into multi-paragraph subtopic passages," *Comput. Linguistics*, vol. 23, pp. 33–64, 1997. 2

[20] M. Riedl and C. Biemann, "Text Segmentation with Topic Models ," *Journal for Language Technology and Computational Linguistics (JLCL)*, vol. 27, no. 47-69, pp. 13–24, 2012. 2

[21] E. Gómez, "Tonal description of polyphonic audio for music content processing," *INFORMS J. Comput.*, vol. 18, pp. 294–304, 2006. 3, 4

[22] P. N. Juslin and J. A. Sloboda, *Handbook of Music and Emotion: Theory, Research, Applications*. Oxford University Press, 2011. 3, 4

[23] O. Lartillot, P. Toiviainen, and T. Eerola, "A matlab toolbox for music information retrieval," in *Data Analysis, Machine Learning and Applications*, 2008. 3

[24] J. Foote and M. L. Cooper, "Media segmentation using self-similarity decomposition," in *IS&T/SPIE Electronic Imaging*, 2003. 3, 4

[25] M. Tapaswi, M. Bäuml, and R. Stiefelhagen, "StoryGraphs: Visualizing Character Interactions as a Timeline," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 3

[26] M. Tapaswi, M. Bäuml, and R. Stiefelhagen, "Book2Movie: Aligning Video Scenes with Book Chapters," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3, 5

[27] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning (ICML)*, 2021. 3, 4

[28] M. Brysbaert, A. B. Warriner, and V. Kuperman, "Concreteness ratings for 40 thousand generally known english word lemmas," *Behavior Research Methods*, vol. 46, pp. 904–911, 2014. 3

[29] A. Wang, "An industrial strength audio search algorithm," in *International Society for Music Information Retrieval (ISMIR)*, 2003. 4

[30] D. Demszky, D. Movshovitz-Attias, J. Ko, A. Cowen, G. Nemade, and S. Ravi, "GoEmotions: A Dataset of Fine-Grained Emotions," in *Association of Computational Linguistics (ACL)*, 2020. 4

[31] T. Eerola, O. Lartillot, and P. Toiviainen, "Prediction of multidimensional emotional ratings in music from audio using multivariate regression models," in *International Society for Music Information Retrieval (ISMIR)*, 2009. 4

MUSIKA!
FAST INFINITE WAVEFORM MUSIC GENERATION

Marco Pasini **Jan Schlüter**
Institute of Computational Perception, Johannes Kepler University Linz, Austria
marco.pasini.98@gmail.com jan.schlueter@jku.at

ABSTRACT

Fast and user-controllable music generation could enable novel ways of composing or performing music. However, state-of-the-art music generation systems require large amounts of data and computational resources for training, and are slow at inference. This makes them impractical for real-time interactive use. In this work, we introduce Musika, a music generation system that can be trained on hundreds of hours of music using a single consumer GPU, and that allows for much faster than real-time generation of music of arbitrary length on a consumer CPU. We achieve this by first learning a compact invertible representation of spectrogram magnitudes and phases with adversarial autoencoders, then training a Generative Adversarial Network (GAN) on this representation for a particular music domain. A latent coordinate system enables generating arbitrarily long sequences of excerpts in parallel, while a global context vector allows the music to remain stylistically coherent through time. We perform quantitative evaluations to assess the quality of the generated samples and showcase options for user control in piano and techno music generation. We release the source code and pretrained autoencoder weights at github.com/marcoppasini/musika, such that a GAN can be trained on a new music domain with a single GPU in a matter of hours.

1. INTRODUCTION

Generating raw audio remains a difficult task to perform, considering the high temporal dimensionality of waveforms. Recently, a number of techniques based on deep learning architectures have been proposed: however, they often present limitations such as low generated music quality, lack of general coherence between distant time frames and slow generation speed. Regarding unconditional audio generation, autoregressive models are able to generate high quality audio with long-range dependencies; however, the sampling process is extremely slow and inefficient, which hinders possible real-world applications. On the other hand, non-autoregressive models can reach real-time generation, while they struggle to synthesize samples with satisfactory sound quality and are only able to generate samples of a fixed duration.

Considering the current shortcomings of non-autoregressive audio generation systems, in this work we propose Musika, a GAN-based system that allows fast unconditional and conditional generation of audio of arbitrary length. We achieve this by combining the following contributions:

- The use of a raw audio autoencoder which allows to encode samples into lower-dimensional invertible representations that are easier to model. We engineer the autoencoder with the specific goal of maximizing inference speed and minimizing training time, by relying on the generation of magnitude and phase spectrograms with low temporal resolution and an efficient adversarial training process
- The use of a latent coordinate system for the task of infinite-length audio generation
- The addition of a global style conditioning which allows the infinite-length generated samples to be stylistically coherent through time
- The possibility to perform both unconditional and conditional generation, with a variety of different conditioning signals, such as note density and tempo information

By avoiding auto-regression, generation can be fully parallelized and works much faster than real-time even on CPU.

2. RELATED WORK

A popular family of generative models for audio consists in autoregressive models, such as WaveNet [1], SampleRNN [2] and Jukebox [3]. WaveNet was the first model to show that autoregressive generation of raw audio waveforms is possible, and uses dilated convolutions to acquire a large receptive field over the input sample. SampleRNN is a model consisting of a hierarchical stack of recurrent units that are able to model the waveform at different resolutions, and thus capture a larger context and reduce the computational cost required to model the next sample. Jukebox uses a hierarchical VQVAE [4, 5] to encode raw samples into a sequence of discrete codes at different levels. It then uses autoregressive transformers [6] to both generate new top-level samples and upsample them to the lower

© M. Pasini, and J. Schlüter. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. Pasini, and J. Schlüter, "Musika! Fast Infinite Waveform Music Generation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

levels, accepting different features, such as lyrics, as conditioning. While autoregressive systems can achieve satisfying audio quality and long-range coherence, they suffer from extremely slow generation, as audio samples are produced sequentially. For example, Jukebox requires more than eight hours to generate one minute of audio on a V100 GPU. As an exception, RAVE [7] achieves real-time synthesis by encoding raw audio of a specific domain with a variational autoencoder [8] into a compact latent space and using a lightweight autoregressive model to generate codes: however, the short receptive field of the autoregressive model does not allow to model dependencies over distant time windows in the generated audio.

Non-autoregressive models avoid the slow sequential generation, but are mostly employed for *conditional* audio synthesis. For example, several works focus on the task of inverting a low-dimensional audio representation (often a mel-spectrogram) back to the original waveform, which constitutes a building block of modern text-to-speech (TTS) systems [9–11]. In contrast, literature on long-form non-autoregressive unconditional audio generation is scarce. Systems such as WaveGAN and SpecGAN [12], GANSynth [13], DrumGAN [14] and MP3Net [15] attempt to generate audio of a fixed length using various architectures of Generative Adversarial Networks [16] (GANs). WaveGAN and SpecGAN represent the first works in which a GAN is successfully applied to audio, in the waveform and spectrogram representations, respectively. GANSynth generates instantaneous frequency (IF) and magnitude of spectrograms with high frequency resolution of short monophonic instrument notes [17], showing that generating IFs and magnitudes instead of waveforms is advantageous for highly harmonic sounds. DrumGAN synthesizes drum sounds using the real and imaginary components of a complex STFT spectrogram and demonstrates the effectiveness of this audio representation, first introduced in [18]. Finally, MP3Net achieves minute-long coherent piano music generation using Modified Discrete Cosine Transform (MDCT) spectrograms as audio representations and Progressive GAN [19] as the model: however, the generated samples suffer from low perceived audio quality. To the best of our knowledge, UNAGAN [20] is the only non-autoregressive GAN-based system capable of generating audio of arbitrary length. The model takes a sequence of noise vectors as input and uses a hierarchical structure to achieve short-term coherence in the generated mel-spectrograms, which are then inverted to waveform using a pretrained MelGAN vocoder [9]. However, the model only generates single-channel audio and the independent sampling of noise vectors cause the generated samples to lack coherence through time.

Contrary to autoregressive models, the majority of GAN-based unconditional audio generation models are only able to synthesize audio samples of a fixed length. However, in the field of computer vision, several recent works propose models capable of generating images of arbitrary size, by synthesizing single image patches in parallel and assembling them into the final image. This process results in a fast and efficient generation of images on modern hardware. The two most notable contributions to this line of work are InfinityGAN [21] and ALIS [22]. InfinityGAN generates in parallel single patches that are coherent with each other by disentangling global appearance, local structures and textures, which are then fed into a generator, together with coordinate information, to synthesize the final patch. ALIS proposes the use of latent vectors as anchor points for the coordinate system of the model: the resulting generator is equivariant and can thus produce coherent patches from interpolations of different latent codes. However, both of the methods rely on prior knowledge regarding the particular image domain that is being generated: the experiments are conducted on a dataset of images of landscapes, where the image features of a single patch are spatially invariant on the horizontal dimension and thus permit infinite length generation along the horizontal axis.

The process of generating sequences of encoded representations has been explored in different previous works [4, 5, 23] for both image and audio data. However, these works focus on encoding samples to a discrete set of codes using vector-quantized variational autoencoders, and propose to model sequences of codes using autoregressive models. On the other hand, [24] proposes to autoencode molecules with a basic autoencoder, to then generate sequences of continuous-valued latent vectors with a GAN. This approach manages to circumvent the problematic behaviour of GANs when applied on discrete data [25], in this case molecules in the SMILES format. Similarly to this work, we propose to generate latent representations of audio with the aim of making the generation and training process faster, and achieving coherent generated samples over long time windows.

3. METHOD

Let $\mathbf{x} = \{x_1, ..., x_T\}$ be the waveform of an audio sample. We aim to encode a waveform $\mathbf{x}$ into a sequence of latent vectors $\mathbf{c} = \{c_1, ..., c_{T/r_{time}}\}$ with time compression ratio r_{time}, sampled at a lower sampling rate than the original waveform. We use an autoencoder model to perform this task, such that a reconstruction of the original waveform can be obtained from the encoded latent vectors.

We then aim to model the distribution $p(\mathbf{c})$ with a Generative Adversarial Network (GAN). We employ a latent coordinate system that is used as conditioning for the generator G to generate sequences of latent vectors of arbitrary length. We additionally condition the generator with a variety of conditioning signals, such that the generation process can be guided by human input. Finally, the generated sequence of latent vectors is inverted to a waveform with the previously trained decoder.

3.1 Audio Autoencoder

Considering the inherent high dimensionality of waveforms, generating long sequences of raw audio samples is prohibitively expensive. A frequently used audio representation in the field of speech processing and music informa-

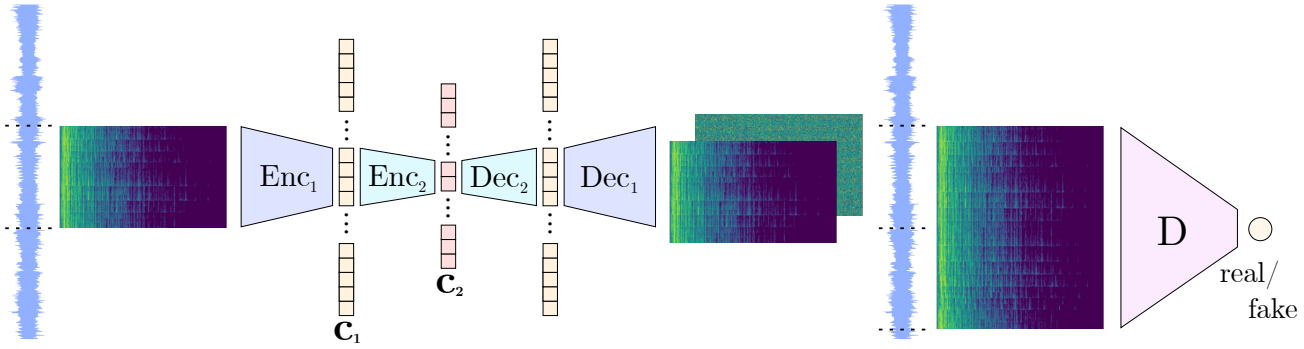


Figure 1. The proposed 2-level audio autoencoder. A log-magnitude spectrogram is used as the encoder input, while the decoder outputs magnitude and phase spectrograms which are then inverted with iSTFT to the waveform domain. A discriminator evaluates the magnitude spectrogram of two adjacent excerpts passed through the autoencoder. This training process removes both phase errors (which would manifest after the iSTFT and STFT) and boundary artifacts.

tion retrieval is the Short-Time Fourier Transform (STFT) spectrogram: while the phase component of the spectrogram is usually discarded, in case of audio synthesis applications both magnitude and phase components are necessary to perform the inverse STFT (iSTFT) and obtain a waveform.

We design an audio autoencoder with the aim of minimizing inference and training time while maximizing the compression ratio allowing to reconstruct samples with reasonable accuracy. Our proposed autoencoder takes a log-magnitude STFT spectrogram as input, and outputs magnitude and phase spectrograms which can be inverted to a waveform. Parallel to our work, iSTFTNet [26] also proposes to improve the inference speed of the model by generating magnitude and phase of a STFT spectrogram: however, they only report experiments using spectrograms with very high temporal resolution and low frequency resolution, while our proposed autoencoder reconstructs spectrograms with low temporal resolution and high frequency resolution. This should result in an even higher inference speed for similarly-sized models. In practice, we separately train two stacked autoencoders; this allows a higher compression ratio with satisfactory reconstruction quality, especially for more complex music domains. Similarly to RAVE [7], we utilize a two-step training process:

3.1.1 First training phase

We first train the model to autoencode log-magnitude spectrograms, not producing phases for now. We use a L1 loss function for the reconstruction task:

$$\mathcal{L}_{(Enc,Dec),rec} = \mathbb{E}_{s\sim p(s)}||Dec(Enc(s)) - s||_1$$

where Enc and Dec are the encoder and decoder, and s is a log-magnitude spectrogram of a waveform w.

3.1.2 Second training phase

In the second phase, we freeze the encoder weights and have the decoder produce a phase spectrogram as well, such that we can reconstruct a waveform through an iSTFT. We add an adversarial objective to aid the modeling of both the magnitudes and phases, ensuring the waveform is of perceptually satisfactory quality. Since directly modeling phase spectrograms with deep learning models is known to be difficult [13, 18], we propose to model the phases indirectly, by encouraging waveforms whose magnitude spectrogram must appear realistic. Specifically, we compute a log-magnitude spectrogram $\tilde{s}$ from the reconstructed waveform $\tilde{w}$:

$$\tilde{w} = iSTFT(Dec(Enc(s)))$$
$$\tilde{s} = \log(|STFT(\tilde{w})|^2 + \epsilon)$$

The reconstructions $\tilde{s}$ are fed to a discriminator D, using the hinge loss [27] to distinguish them from originals s:

$$\mathcal{L}_D = -\mathbb{E}_{s\sim p(s)}[min(0, -1 + D(s))] - \mathbb{E}_{s\sim p(s)}[min(0, -1 - D(\tilde{s}))]$$

The decoder is trained to fool the discriminator:

$$\mathcal{L}_{Dec,adv} = -\mathbb{E}_{s\sim p(s)}D(\tilde{s})$$

Note that we can calculate spectrograms from the reconstructed waveforms with different hop size and window length than used for the spectrograms fed to the autoencoder. We leverage this by including the multi-scale spectral distance [7, 28] in the objective of the decoder:

$$\mathcal{L}_{Dec,ms} = \mathbb{E}_{w\sim p(w)} \sum_{hop}^{N} \log(||\,|STFT_{hop}(w)| - |STFT_{hop}(\tilde{w})|\,||_1)$$

where hop indicates a choice of hop_size and fft_size.

In total, we train the discriminator with $\mathcal{L}_D$, and the decoder with a linear combination of three losses:

$$\mathcal{L}_{Dec} = \mathcal{L}_{Dec,adv} + \lambda_{rec}\mathcal{L}_{Dec,rec} + \lambda_{ms}\mathcal{L}_{Dec,ms}$$

3.2 Latent Coordinate System

We use a GAN to model sequences of latent vectors produced by a trained audio encoder. In order to generate independent audio samples that can be seamlessly concatenated with each other along the temporal axis, we condition the generator with the latent coordinate system proposed by [22], originally introduced to generate landscape

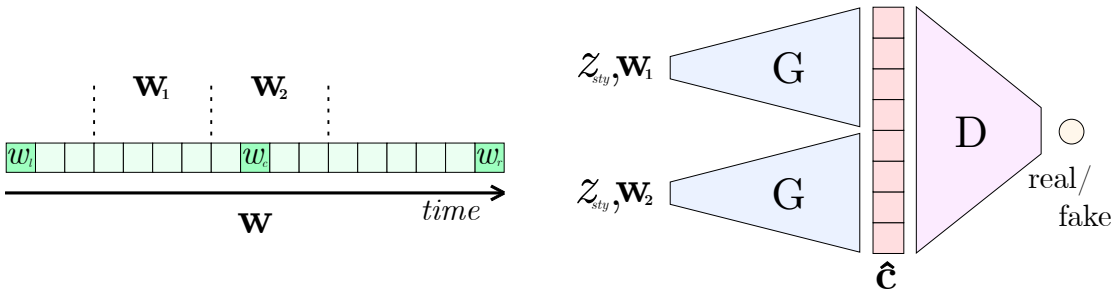


Figure 2. The proposed latent GAN training process. Two adjacent latent coordinate sequences are randomly cropped from the linear interpolation between three anchor vectors. They are then used as input to the generator, together with a shared style vector and a conditioning signal in the case of a conditional model. The discriminator takes as input the two concatenated generated sequences and a real sequence of latent vectors.

images of infinite width. Specifically, during training we sample three noise vectors w_l, w_c, w_r with dimension d that are used as anchor points (left, center, right anchors) to guide the generation process. With seq_len being the length of the sequence of latent vectors that is produced by the generator, we linearly interpolate the three anchor vectors to create a sequence of coordinate vectors of length equal to $4 \cdot seq\_len + 1$:

$$\mathbf{w} = [w_l, ..., (1-k)w_l + kw_c, ..., w_c, ..., w_r] \in \mathbb{R}^{4seq\_len+1 \times d}$$

To generate sequences that are temporally coherent with each other, we follow [22]: we randomly crop a sequence $\mathbf{w_{12}}$ of $2 \cdot seq\_len$ coordinate vectors from $\mathbf{w}$, divide it into two sequences $\mathbf{w_1}, \mathbf{w_2}$ with length seq_len, generate two patches using each sequence as conditioning, concatenate the two patches along the time axis, and feed the resulting generated sample of length $2 \cdot seq\_len$ to the discriminator. This process is illustrated in Figure 2. It allows the generator to align the sequence of latent coordinates with the generated sequence of latent vectors. Specifically, the discriminator forces the generator to learn that adjacent sequences of latent coordinates must result in adjacent sequences of latent vectors, which can be temporally concatenated resulting in a coherent final sample without artifacts at the boundaries of the generated patches.

Similarly to InfinityGAN [21], when generating adjacent sequences of latent vectors, we also condition both generations on a single random vector z_{sty}: during the learning process, this vector serves as conditioning for the global style of the generated samples. Specifically, while the latent coordinate vectors allow the generator to produce sequences of latent vectors that can be seamlessly concatenated along the temporal axis, the global style vector allows the final concatenated sequence of possibly infinite length to be stylistically coherent throughout. Without the global style vector, any temporal context available to the generator would completely change every $4 \cdot seq\_len$ samples of a sequence, resulting in a final generated sample which continuously changes style through time.

Formalized, we have

$$\hat{\mathbf{c}} = concat[G(\mathbf{w_1}, z_{sty}), G(\mathbf{w_2}, z_{sty})],$$

where $\hat{\mathbf{c}}$ is a stylistically and temporally coherent sequence of latent vectors of length $2 \cdot seq\_len$, and G is the generator model.

At inference time, a latent coordinate sequence of the desired length is created. The coordinate sequence is prepared in the same way as during the training phase, by placing a latent anchor vector at positions that are multiples of $2 \cdot seq\_len$, and by linearly interpolating these anchor vectors to calculate in-between vectors. A single random global style vector is also sampled. Each generation considers a seq_len crop and the global style vector as conditioning, and finally all generated latent vectors are concatenated together in the appropriate order. This process can be performed in a parallel manner, thus resulting in a fast generation on modern hardware.

4. IMPLEMENTATION DETAILS

4.1 Audio Autoencoder Architecture

We first train an audio autoencoder with a relatively low compression ratio, then train a second-level autoencoder that encodes the first-level latent vectors, as shown in Figure 1. During the training of the second-level autoencoder, we utilize the same training strategy and objective as explained in section 3.1, by propagating gradients through the frozen weights of the previously trained first-level decoder and adversarially discriminating between samples reconstructed by both decoders and samples reconstructed by only the first-level decoder. Both model architectures are fully convolutional, and we do not use any padding in both encoders, such that possible boundary artifacts in the encoded representations are avoided. We utilize 1d-convolutions considering the frequency bins as different channels for both encoders and decoders: this is usually not efficient regarding total number of model parameters when compared to using 2d-convolutions across the two spectrogram dimensions, but can result in a much faster inference time. We use 2d-convolutions for the discriminator, as inference time for this model is not a priority. We use $tanh$ as the activation for the bottleneck layer of both encoders. Regarding the multi-scale spectral distance loss, we use $hop\_size \in [64, 128, 256, 512]$ and we always choose $fft\_size = 4 \cdot hop\_size$, while the discriminator takes as input log-magnitude spectrograms calculated with

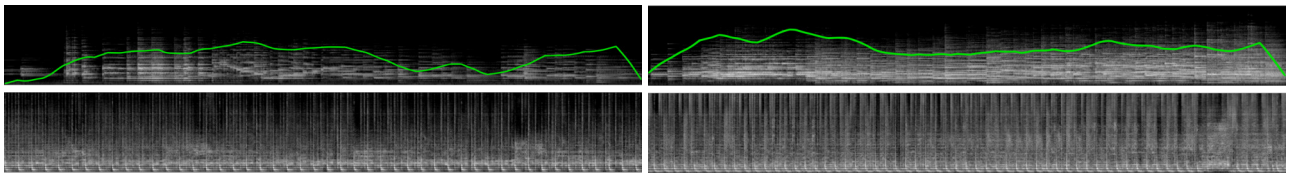


Figure 3. Log-melspectrograms of generated piano and techno samples from the conditional models. For the piano samples (top row), we indicate the corresponding note density conditioning with a green line. Note density signals were generated using a random walk algorithm. The tempo used as conditioning for the techno samples (bottom row) is 120 bpm and 160 bpm, respectively. Each sample is 23 seconds long. Visit marcoppasini.github.io/musika to listen to the examples.

$hop\_size = 256$ and $fft\_size = 6 \cdot hop\_size$. As proposed by [29, 30], two consecutive reconstructed spectrograms are concatenated along the temporal dimension and fed to the discriminator, such that concatenated reconstructions do not suffer from boundary artifacts. During training, spectrograms calculated from 0.76 s of audio are used as input to both autoencoders. We use spectral normalization [31] on the weights of the discriminator. Regarding the training loss weights, we use $\lambda_{rec} = 1$ and $\lambda_{ms} = 4$. We choose Adam [32] as the optimizer with learning rate of 0.0001 and $\beta_1 = 0.5$, and train the first-level autoencoder for 1 million iterations with batch size of 32 for both training phases, and the second-level autoencoder for 400k iterations with batch size of 32 for both training phases.

4.2 Latent GAN Architecture

We choose to adapt the FastGAN [33] architecture to our specific task. The FastGAN architecture promises fast convergence with limited amounts of data. To achieve this, it proposes a Skip-Layer channel-wise Excitation (SLE) module in the generator, for more direct propagation of gradients, and proposes to strongly regularize the discriminator with an added self-supervised reconstruction objective. We adapt the proposed architectures to use 1d-convolutions instead of 2d-convolutions and we simplify the added reconstruction objective of the discriminator, by using a single lightweight decoder which reconstructs the whole input of the discriminator. Differently from FastGAN, we do not use Batch Normalization [34] in both the generator and discriminator, while we apply the variation of Adaptive Instance Normalization [35] (AdaIN) called Spatially Aligned AdaIN (SA-AdaIN), originally proposed in [22], after each convolutional layer in the generator. To generate stereo samples, the generator produces two latent vectors at each timestep, one for each audio channel, stacked on the channel axis. We use Cross Channel Mixing (CCM), first introduced in [36], to randomly mix channels of the stereo stacked latent vectors before being fed to the discriminator. In our experience, this technique helps reducing collapses during training. Both anchor and style vectors are sampled from a normal distribution with zero mean and unit variance, and have dimension d of 64. We use R1 gradient penalty [37] as regularization, and Adam with learning rate of 0.0001 and $\beta_1 = 0.5$ as the optimizer. We train for 1.5 mio iterations with a batch size of 32 for all experiments. Training takes 23 h on a RTX 2080 Ti GPU.

Model (Faster than real-time)	**GPU**	**CPU**
Musika Uncond. Piano	972x	**40x**
Musika Cond. Piano	921x	**40x**
UNAGAN [20] Piano	28x	11x
Musika Uncond. Techno	**994x**	39x
Musika Cond. Techno	917x	39x

Table 1. Comparison of generation speed between the different models. For the Musika models, we include both the generation of the latent vectors and the decoding step to the waveform domain. We use a RTX 2080 Ti and a Ryzen 3950x as the GPU and CPU, respectively. We report the average of 100 trials.

Model	**FAD**
Musika Uncond. Piano	**1.641**
Musika Cond. Piano Rand.	2.150
Musika Cond. Piano Const. 0.15	2.584
Musika Cond. Piano Const. 0.30	3.400
Musika Cond. Piano Const. 0.45	4.389
Musika Cond. Piano Const. 0.60	4.839
Musika Cond. Piano Const. 0.75	5.434
UNAGAN [20] Piano	11.183

Table 2. FAD evaluation for generated piano music. We evaluate conditional Musika models using different constant values of note density as conditioning. We notice that FAD increases with higher note density.

5. EXPERIMENTS

Considering the relatively low compression ratio of the first autoencoder and thus its need to only encode low-level audio features, we find it possible to train a single universal model which we can later use for different music domains. As training data, we choose to use songs released and made freely available by South by SouthWest [1] (SXSW) in occasion of their yearly conference. The current collection consists of 17k songs of various genres, and for this reason it represents a fitting choice for training our universal model. We use the LibriTTS corpus [38] as additional training data, to steer the universal model into accurately synthesizing human voices, which are notoriously hard to model. Even though LibriTTS only contains speech, in-

[1] https://www.sxsw.com/festivals/music/

cluding it improves reconstructions of singing voice. We resample audio to 22.05 kHz for all experiments. We use single channel audio to train the audio autoencoders, as the latent GAN is able to generate stereo samples by using latent representations of the two mono samples stacked in the channel dimension as training data. We use $r^1_{time} = 256$ as the time compression ratio, which results in a sampling rate of the first-level latent representations of 190.22 Hz. Each of the encoded latent vectors has a dimension of 128.

5.1 Piano Music

We use the MAESTRO dataset [39], consisting of 200 hours of piano performances, to train a second-level autoencoder and a latent GAN. The final time compression ratio achieved by both autoencoders is $r_{time} = 4096$, which results in a sampling rate of the second-level latent representations of 11.89 Hz. The dimension of each latent vector is 32. We train both an unconditional and a conditional latent GAN. For both models, the generator outputs latent vectors with $seq\_len = 64$, which results in about 12 s of audio after decoding. For the conditional model, we apply the CNN-based onset detector [40] of the madmom Python library [41] to all audio files in the dataset. We then use Gaussian Kernel Density Estimation (KDE) with bandwidth of 0.004 on the detected onsets to estimate a continuous note density signal for each sample. This signal is log-scaled between 0 and 1 and serves as a conditioning signal for the conditional (and thus controllable) GAN.

5.2 Techno Music

To evaluate the performance of the system on a more musically varied domain, we scrape 10,190 songs categorized with the "techno" genre from *jamendo.com* and use them as training data. Considering the wide diversity of sounds that are present in the dataset, we train the second-level autoencoder with the same SXSW data used to train the first-level universal autoencoder. Comparing to what is achievable when training an autoencoder on a single and limited domain, such as piano music, a lower compression ratio is needed to reach a satisfactory reconstruction accuracy. However, this solution allows users to directly train a latent GAN on a new audio domain using the universal latent representations, without the need to train an autoencoder on the domain of interest. The final achieved time compression ratio is $r_{time} = 2048$, which results in a sampling rate of the second-level latent representations of 23.78 Hz. The dimension of each latent vector is 64. We train an unconditional and a conditional latent GAN model, both generating stereo latent vectors with $seq\_len = 128$, resulting in about 12 s of decoded audio. We use the Tempo-CNN framework [2] [42] to estimate the global tempo of each song in the dataset. Tempo information is then used as conditioning for the conditional model.

[2] https://github.com/hendriks73/tempo-cnn

6. RESULTS

A comprehensive collection of generated audio samples is available on marcoppasini.github.io/musika. Since current quantitative evaluation metrics are not able to assess the overall compositional and musical quality of generated music, we strongly encourage the reader to listen to the provided samples while reading the paper.

We report the generation speed of the system trained on the MAESTRO and on the techno datasets in Table 1, on both GPU and CPU. We also use the Frechét Audio Distance [43] (FAD) metric to quantitatively evaluate the quality of the generated piano samples in Table 2. A UNAGAN [20] model that was trained on the same dataset is used as comparison. While our system is capable of generating stereo audio, UNAGAN can only produce single-channel audio. The unconditional model obtains the lowest FAD, while the conditional system results in higher FADs when using more intense note density values as conditioning. This is expected, since samples with low note density are more common than samples with high note density in the MAESTRO dataset. However, considering that audio is split in short 1 s samples to calculate embeddings, FAD is not designed to evaluate overall musical and compositional quality of samples, and to the best of our knowledge there are no available quantitative metrics to evaluate these characteristics. Piano and techno samples generated by the system seem to often demonstrate long-range coherence and successfully keep a fixed general music style through time. Both conditional models successfully generate samples that are coherent with the conditioning signal, as can be seen in Figure 3.

7. CONCLUSION

We proposed Musika, a non-autoregressive music generation system that generates raw-audio samples of arbitrary length much faster than real-time on a consumer CPU. An efficient hierarchical autoencoder allows to encode audio to a sequence of low-dimensional latent vectors, from which a waveform can be reconstructed. A GAN is then used to generate new sequences of latent vectors, using a latent coordinate system that allows for generation of samples of infinite length. A style conditioning vector is introduced to force the samples to be stylistically coherent through time. We successfully use the system to generate piano and techno music, and show that the generation process can be conditioned on note density and tempo information for piano and techno music, respectively. We finally show that the system achieves lower FAD than comparable systems on piano music generation while being faster. We release the source code and pretrained models, enabling users to generate samples of different music domains and test new conditioning signals with ease and using consumer hardware. We see our system as solving an important technical challenge – real-time music generation of sufficient quality, conditioned on user input – and hope it can serve as a basis for interactive real-world applications and for research into human-AI co-creation.

8. REFERENCES

[1] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," in *The 9th ISCA Speech Synthesis Workshop*, Sep. 2016, p. 125.

[2] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. C. Courville, and Y. Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," in *5th International Conference on Learning Representations (ICLR)*, Apr. 2017.

[3] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[4] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Advances in Neural Information Processing Systems 30*, Dec. 2017, pp. 6306–6315.

[5] A. Razavi, A. van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with VQ-VAE-2," in *Advances in Neural Information Processing Systems 32*, Dec. 2019, pp. 14 837–14 847.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30*, Dec. 2017, pp. 5998–6008.

[7] A. Caillon and P. Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," *arXiv preprint arXiv:2111.05011*, 2021.

[8] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations (ICLR)*, Apr. 2014.

[9] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "MelGAN: Generative adversarial networks for conditional waveform synthesis," in *Advances in Neural Information Processing Systems 32*, Dec. 2019, pp. 14 881–14 892.

[10] J. Kong, J. Kim, and J. Bae, "HiFi-GAN: Generative adversarial networks for efficient and high fidelity speech synthesis," in *Advances in Neural Information Processing Systems 33*, Dec. 2020.

[11] I. Elias, H. Zen, J. Shen, Y. Zhang, Y. Jia, R. J. Skerry-Ryan, and Y. Wu, "Parallel tacotron 2: A non-autoregressive neural TTS model with differentiable duration modeling," in *22nd Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Aug. 2021, pp. 141–145.

[12] C. Donahue, J. J. McAuley, and M. S. Puckette, "Adversarial audio synthesis," in *7th International Conference on Learning Representations (ICLR)*, May 2019.

[13] J. H. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial neural audio synthesis," in *7th International Conference on Learning Representations (ICLR)*, May 2019.

[14] J. Nistal, S. Lattner, and G. Richard, "DRUMGAN: synthesis of drum sounds with timbral feature conditioning using generative adversarial networks," in *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, Oct. 2020, pp. 590–597.

[15] K. v. d. Broek, "Mp3net: coherent, minute-long music generation from raw audio with a simple convolutional GAN," *arXiv preprint arXiv:2101.04785*, 2021.

[16] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Dec. 2014, pp. 2672–2680.

[17] J. H. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 70, Aug. 2017, pp. 1068–1077.

[18] J. Nistal, S. Lattner, and G. Richard, "Comparing representations for audio synthesis using generative adversarial networks," in *28th European Signal Processing Conference (EUSIPCO)*. IEEE, Jan. 2020, pp. 161–165.

[19] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," in *6th International Conference on Learning Representations (ICLR)*, Apr. 2018.

[20] J. Liu, Y. Chen, Y. Yeh, and Y. Yang, "Unconditional audio generation with generative adversarial networks and cycle regularization," in *21st Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Oct. 2020, pp. 1997–2001.

[21] C. H. Lin, Y.-C. Cheng, H.-Y. Lee, S. Tulyakov, and M.-H. Yang, "InfinityGAN: Towards infinite-pixel image synthesis," in *10th International Conference on Learning Representations (ICLR)*, Apr. 2022.

[22] I. Skorokhodov, G. Sotnikov, and M. Elhoseiny, "Aligning latent and image spaces to connect the unconnectable," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2021, pp. 14 124–14 133.

[23] P. Esser, R. Rombach, and B. Ommer, "Taming transformers for high-resolution image synthesis," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Computer Vision Foundation / IEEE, Jun. 2021, pp. 12 873–12 883.

[24] O. Prykhodko, S. Johansson, P. Kotsias, J. Arús-Pous, E. J. Bjerrum, O. Engkvist, and H. Chen, "A de novo molecular generation method using latent vector based generative adversarial network," *J. Cheminformatics*, vol. 11, no. 1, p. 74, 2019.

[25] M. J. Kusner and J. M. Hernández-Lobato, "GANs for sequences of discrete elements with the Gumbel-softmax distribution," *arXiv preprint arXiv:1611.04051*, 2016.

[26] T. Kaneko, K. Tanaka, H. Kameoka, and S. Seki, "iSTFTNET: fast and lightweight mel-spectrogram vocoder incorporating inverse short-time fourier transform," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, May 2022, pp. 6207–6211.

[27] J. H. Lim and J. C. Ye, "Geometric GAN," *arXiv preprint arXiv:1705.02894*, 2017.

[28] J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: differentiable digital signal processing," in *8th International Conference on Learning Representations (ICLR)*, Apr. 2020.

[29] C. H. Lin, C. Chang, Y. Chen, D. Juan, W. Wei, and H. Chen, "COCO-GAN: generation by parts via conditional coordinating," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019, pp. 4511–4520.

[30] M. Pasini, "MelGAN-VC: Voice conversion and audio style transfer on arbitrarily long samples using spectrograms," *arXiv preprint arXiv:1910.03713*, 2019.

[31] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *6th International Conference on Learning Representations (ICLR)*, Apr. 2018.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, May 2015.

[33] B. Liu, Y. Zhu, K. Song, and A. Elgammal, "Towards faster and stabilized GAN training for high-fidelity few-shot image synthesis," in *9th International Conference on Learning Representations (ICLR)*, May 2021.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, ser. JMLR Workshop and Conference Proceedings, vol. 37, Jul. 2015, pp. 448–456.

[35] X. Huang and S. J. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017, pp. 1510–1519.

[36] A. Sauer, K. Chitta, J. Müller, and A. Geiger, "Projected GANs converge faster," in *Advances in Neural Information Processing Systems 34*, Dec. 2021, pp. 17 480–17 492.

[37] L. M. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for GANs do actually converge?" in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 80, Jul. 2018, pp. 3478–3487.

[38] H. Zen, V. Dang, R. Clark, Y. Zhang, R. J. Weiss, Y. Jia, Z. Chen, and Y. Wu, "LibriTTS: A corpus derived from LibriSpeech for text-to-speech," in *20th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Sep. 2019, pp. 1526–1530.

[39] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C. A. Huang, S. Dieleman, E. Elsen, J. H. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *7th International Conference on Learning Representations (ICLR)*, May 2019.

[40] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 6979–6983.

[41] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: A new python audio and music signal processing library," in *Proceedings of the 2016 ACM Conference on Multimedia Conference (MM)*, Oct. 2016, pp. 1174–1178.

[42] H. Schreiber and M. Müller, "A single-step approach to musical tempo estimation using a convolutional neural network," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Sep. 2018, pp. 98–105.

[43] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms," in *20th Annual Conference of the International Speech Communication Association (INTERSPEECH)*, Sep. 2019, pp. 2350–2354.

SYMPHONY GENERATION WITH PERMUTATION INVARIANT LANGUAGE MODEL

Jiafeng Liu[1⋆] **Yuanliang Dong**[1⋆] **Zehua Cheng**[2]
Xinran Zhang[1] **Xiaobing Li**[1] **Feng Yu**[1] **Maosong Sun**[1,3†]
[1] Department of Music AI and Music Information Technology, Central Conservatory of Music
[2] Department of Computer Science, University of Oxford
[3] Department of Computer Science and Technology , Tsinghua University
{jiafeng.liu, gunterdong}@mail.ccom.edu.cn

ABSTRACT

In this work, we propose a permutation invariant language model, SymphonyNet, as a solution for symbolic symphony music generation. We propose a novel Multi-track Multi-instrument Repeatable (MMR) representation for symphonic music and model the music sequence using a Transformer-based auto-regressive language model with specific 3-D positional embedding. To overcome length overflow when modeling extra-long symphony tokens, we also propose a modified Byte Pair Encoding algorithm (Music BPE) for music tokens and introduce a novel linear transformer decoder architecture as a backbone. Meanwhile, we train the decoder to learn automatic orchestration as a joint task by masking instrument information from the input. We also introduce a large-scale symbolic symphony dataset for the advance of symphony generation research. Empirical results show that the proposed approach can generate coherent, novel, complex and harmonious symphony as a pioneer solution for multi-track multi-instrument symbolic music generation.

1. INTRODUCTION

Symphony is one of the most complex and brilliant musical composition forms in human history, where many instruments are intertwined to express rich human emotions. The past decade has seen the rapid development and tremendous success of the symbolic music generation in both research and industrial field [1–3]. Most current works follow conventional text modeling and generation method by applying language model to sequences of symbolic musical events [4–6]. However, symphony modeling and generation still constitutes in itself a considerable challenge since symphony music sequences differ from text sequences in various aspects.

Natural language could be modeled as a purely linear sequence constructed strictly by a sequential order of words. Symphony scores, on the other hand, are usually viewed as two-dimensional symbolic sequences in which many notes can be played concurrently. Notes in a symphony score are **semi-permutation invariant**. More specifically, as shown in Fig. 1, the blue box indicates the musical instrument tracks, and the corresponding staves on the right side are permutation invariant. Similarly, the notes inside the red box are also permutation invariant. In contrast, notes in the upper yellow box are permutation variant since each note is played sequentially. Changes in the order of notes will impair the music itself. The yellow box at the bottom is a more complicated situation: a permutation variant note sequence in general containing permutation invariant notes. Simply flattening the score into a 1-D text-like sequence may damage the local structure of music [7]. To address this problem, we propose the Multi-track Multi-instrument Repeatable (MMR) representation with particular 3-D positional embedding in Section 3 which fully considers the properties of semi-permutation invariance in symbolic music scores.

Moreover, when comparing music scores with text, conventionally notes could be considered as characters, while intervals or chords are comparable to words. Modeling musical events at note level is a common practice [5,6,8,9]. However, this may be confronted with similar problems in char-level text generation, such as extremely long sequences and less meaningful individual tokens. Word-level tokenization suffers from large vocabulary size and out of vocabulary (OOV) problems. Byte Pair Encoding (BPE) [10, 11] subword tokenization is a tradeoff between word-level and character-level tokenization. Inspired by BPE, we propose the Music BPE algorithm in Section 4, which could automatically aggregate notes to intervals and chords as subwords without a pre-defined vocabulary and construct music sequences with richer semantics.

Generating symphony music with proper instruments for different tracks is another challenging task. Recent

⋆ The authors contributed equally to this work.
† Corresponding author. Email: sms@tsinghua.edu.cn

© J. Liu, Y. Dong, Z. Cheng, X. Zhang, X. Li, F. Yu, M. Sun. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J. Liu, Y. Dong, Z. Cheng, X. Zhang, X. Li, F. Yu, M. Sun, "Symphony Generation with Permutation Invariant Language Model", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

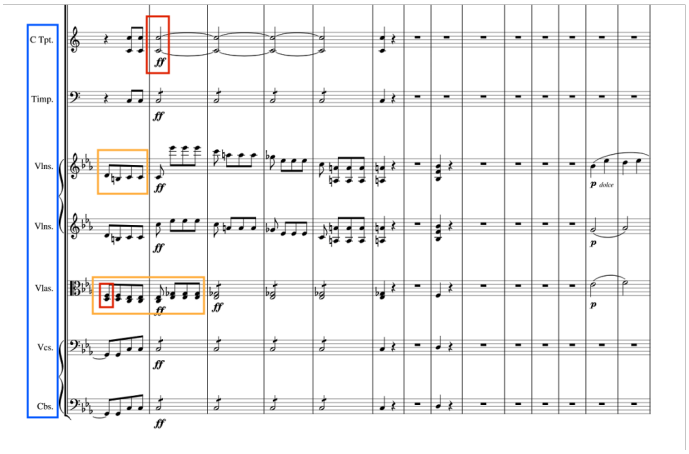


Figure 1: A simple example of Multi Instruments & Multi Tracks & Repeat Instruments symphony score.

work like Arranger [12] focuses on instrumentation by learning to separate parts from the mixture in symbolic multi-track music. However, it does not incorporate music generation task. In this paper, we present a unique linear transformer decoder architecture for instrument classification with joint-task training, which allows the model to learn auto-orchestration rather than relying on instrument information as an pre-defined input source. [5, 13–15].

The contributions of this paper are presented as below:

- We propose a novel Multi-track Multi-instrument Repeatable (MMR) representation for symphony music, including particular 3-D positional embedding designed to address the semi-permutation invariant challenge in symphony generation. Our method is also compatible with all existing symbolic music ensembles, including but not limited to piano solo, quartet and pop band music.
- We propose a novel algorithm, Music BPE, to model the symbolic music at subword-level. Furthermore, we found that our Music BPE algorithm could aggregate notes to intervals and chords, which are consistent with common chords summarized by human musicians.
- We introduce SymphonyNet, a novel music generation model with joint-task training for instrument classification based on our proposed MMR representation and Music BPE. The model can learn the proper orchestration according to the distribution of the notes.
- We collect a symphony MIDI dataset, consisting of $46,359$ high-quality MIDI files with multiple instruments and tracks to advance researches on symphony generation with deep learning.

2. RELATED WORK

We organize some existing works in Table 1 in terms of five aspects of symbolic music modeling: time unit, representation method, backbone model, music type and the ability to model music with repeat instruments. Generation works are presented above and understanding works are presented below. Pianoroll, MIDI event timeshift, and Beat-based onset and duration are the mainstream time units in music generation and understanding tasks. However, Pianoroll divides music into fixed-length grids, and MIDI format provides overprecise timeshift events, both suffering from sparsity problems, which raises another handicap for applying deep learning models in this multi-track generation. Pop Music Transformer [8] is the first attempt to introduced the beat-based REMI representation in music generation. It supports variable-length duration of notes, which is more musically inspired. Compound Word [6], derived from REMI representation, classifies the sequence of REMI into note-related or metric-related events, which are then aggregated, greatly decreasing the sequence length.. This has engendered a new trend of beat-based symbolic music generation.

Language models are now prevalent in natural language processing tasks [18]. However, applying language models to the creation of multi-track music remains challenging. MuMIDI [5] and OctupleMIDI [9] models multiple attributes of one note in one sequence step and also incorporates instrument tokens for multi-track representation. However, if one musical piece contains more than one track for the same instrument, their representation could not distinguish them in different tracks. MMM [15] introduced a MIDI-event-like representation, creating a time-ordered sequence of musical events for each track and concatenating several tracks into a single sequence. However, MMM adopts time-delta tokens and fixed positional encoding which weakens the note-level correlation and structure between tracks. MuseBert [7] proposes a permutation invariant bert-like language model with generalized relative position encoding (RPE) which, however, is not compatible with multi-track music generation.

Though various symbolic music representation strategies have been proposed, few are compatible with multi-track music with repeatable instruments or tracks, such as the symphony. Furthermore, permutation invariance of music, as is discussed in Section 1, has scarcely been considered. To our knowledge, this work proposes the first representation and tokenization method to encode music with multiple repeatable instruments and multiple repeatable tracks and designs a universal and effective strategy for generating symphony music with permutation invariant language model.

3. MULTI-TRACK MULTI-INSTRUMENT REPEATABLE REPRESENTATION

To further analyze the symphony generation task, it is crucial to understand the difference between the symphony format and other genres of music.

- **Single Instrument in Single Track**. No more than one note is played at any timestep by one instrument. Also called monophonic music. e.g., flute.
- **Multi Instruments & Each in Single Track**. Only one

Name	Time Unit	Representation	Backbone Model	Type of Music	Instruments
DeepBach [1]	Fixed-length grid	N/A	Bi-directional RNN	Chorales	Fixed Ensemble
MuseGAN [16]	Fixed-length grid	N/A	GAN	Multi-track	Fixed Ensemble
Music Transfor. [4]	MIDI-event timeshift	N/A	Vanilla Transformer	Piano	N/A
Pop MT [8]	Beat and note duration	REMI	Transformer-XL	Piano	N/A
CWT [6]	Beat and note duration	Compound Word	Linear Transformer	Piano	N/A
Musenet [13]	MIDI-event timeshift	Token Aggregation	GPT-2	Multi-track	Not Repeatable
PopMAG [5]	Beat and note duration	MuMIDI	Transformer-XL	Multi-track	Not Repeatable
LakhNES [14]	MIDI-event timeshift	Token Aggregation	Transformer-XL	Multi-track	Fixed Ensemble
MMM [15]	MIDI-event timeshift	Hierarchical	GPT-2	Multi-track	Repeatable
This work	Beat and note duration	MMR	Linear Transformer	Multi-track	Repeatable
PiRhDy [17]	Fixed-length grid	Fusion module	RNN with attention	Multi-track	Not Repeatable
MusicBert [9]	Beat and note duration	OctupleMIDI	Roberta	Multi-track	Not Repeatable

Table 1: An overview of time unit, representation, backbone model and music type in existing works, above for generation works and below for understanding works.

note for each instrument is played at any timestep. e.g., quartet singing.

- **Single Instrument in Multi Tracks**. There are multiple notes played in each timestep while only one instrument. e.g., piano.
- **Multi Instruments & Multi Tracks & No Repeat Instrument**. There are multiple notes played in each timestep. No constraint on the number of instruments and all instruments are unique. e.g., classical pop band with only drum, electric guitar and bass.
- **Multi Instruments & Multi Tracks & Repeat Instruments**. Instruments are not unique and multiple same instruments can play different notes on different tracks, e.g., symphony.

For the last case, it's a common practice to merge the same instruments into a single track in previous works. However, it may damage the intrinsic structure of symphony music. For example, this may cause a violin to play polyphonic notes, or even intermingle multiple melody lines. Our proposed Multi-track Multi-instrument Repeatable (MMR) representation models repeated instruments separately, which could capture more heuristic musical information within a single track. Since our MMR representation is aimed at symphony modeling, it is also compatible with all existing music ensembles.

3.1 Structural and Controlling Token

We consider that special tokens perform two primary functions in a symphony music generation task: 1) To represent the musical structures of notes. 2) To control the model output during the inference phase.

Score We use a pair of $[BOS]$ and $[EOS]$ tokens to designate the beginning and end of a symphony score.

Measure Different from [5, 8], we ascribe a pair of $[BOM_i]$ and $[EOM]$ to indicate the beginning and end of a measure, the character i to represent the total length of the current measure. The length of a measure is calculated by time signature, and we choose 32$^{\text{th}}$ note as the smallest unit of time. For example, a 4/4 time signature indicates four quarter notes length per measure, which is equal to the length of thirty-two 32$^{\text{th}}$ notes. In that case, character i equals 32, and the measure beginning token is $[BOM_{32}]$

Chord The chord token is a valuable indicator of how generally notes are arranged in the current measure. We pre-define 132 common types of chord token and pre-compute chord tokens with a rule-based algorithm proposed in [6], such as C major seventh chord marked as token $[C_{maj7}]$.

Track Unlike any previous works, we do not explicitly encode the track and instrument transformation in a single token. A change track token $[CC]$ only signifies the start of a new track for the latter controlling purpose. Section 5 will further explore the traits of tracks and instruments and the approaches of differentiating tracks.

Position A position token stands for the *onset* of a note within the measure, represented by the token $[POS_j]$. The following event tokens are controlled by the current position token until another position token shows up. The character j means the number of the current unit of time position. For example, a $[POS_{48}]$ indicates the 48$^{\text{th}}$ unit time position.

To summarize, structural and controlling tokens are designed to specify the general time-spatial features of notes, such as the time a note is to be played and the track it locates. In this work, these tokens are mandated with a sequential order, as a *measure* token shall be followed by a *chord* token, which altogether represents in a explicit way the measure order as shown in Fig. 3.

3.2 Note-Related Tokens

A note in music scores could be defined in five attributes: pitch, duration, onset, track and instrument. Pitch and duration are content-related and the others are position-related. The latter will be discussed in Section 5. To avoid the long-tail problem, we regard pitch and duration to be distinct note properties and construct two separate vocabularies for model input. Then we aggregate note pitches with identical duration and onset by our proposed Music

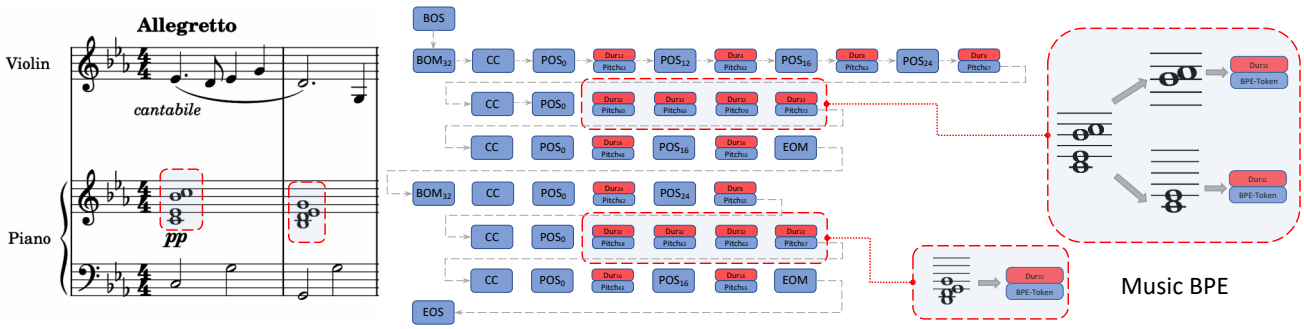


Figure 2: An example of MMR representation and illustration of Music BPE process

BPE algorithm, as will be described in the next section.

4. MUSIC BYTE PAIR ENCODING

As shown in Fig. 2, a complex chord is constructed by several notes at the same position in a measure, which can be deconstructed into two common and simple intervals. Unlike natural language, notes played at the same position are permutation invariant. Changing the order of notes in a chord does not affect its sound or meaning. For instance, a Chord C consists of ($C4$, $E4$, $G4$), which is equal to ($G4$, $C4$, $E4$). This intrinsic property may conflict with the typical natural language processing job, imposing a new constraint on the use of conventional tokenization methods such as standard BPE.

In this work, we propose a novel encoding approach, Music Byte Pair Encoding (Music BPE), for multi-track symbolic music sequence tokenization and preprocessing to exploit the semantics of music events and minimise the length of the input context from a representation standpoint. Different from the original BPE algorithm, our proposed Music BPE is based on **concurrence** of notes rather than **adjacency** of characters.

Our implementation of Music BPE is shown in Algorithm 1. As is mentioned in Section 1, a note has five attributes: *pi*tch, *du*ration, *po*sition, *tr*ack and instrument, while the instrument depends utterly on track within the same measure. Formally, in a piece of symbolic music, we define a maximum set of two or more notes

$$\{(pi, du, po, tr) \mid \text{where } du, po, tr \text{ is constant}\}$$

as a *mulpi* (multiple pitches), i.e., a maximum set of notes that have the same duration at the same global position and within the same track, equivalent to a "word" in the BPE algorithm.

We collect notes with the same global position and the same duration in the same track from each music piece to construct a bag of *mulpies*. The vocabulary list is initialized with 128 MIDI pitches, where each token represents a pitch-set containing a single pitch. Every time we locate all concurrent pairs of tokens in the bag of *mulpies*, merge the most frequent pair ('P_1', 'P_2') into a new symbol P and replace the pair with the new symbol in each *mulpi* until the vocabulary size reaches the maximum limit. A further discussion on the results of the Music BPE algorithm and its effectiveness on our symphony dataset will be presented in Section 5.

Algorithm 1 Music BPE

Input: A multi-set of *mulpies* B
Parameter: desired dictionary size n
Output: Merged dictionary V

Let $V = \{\{p\} \mid p \in [0, 128)\}$.
Let C be an empty multi-set
for all $mulpi \in B$ **do**
  $mulpi \leftarrow \{\{p\} \mid p \in mulpi\}$
  for all $\{P_1, P_2\} \subseteq mulpi$ **do**
    Insert (P_1, P_2) into C.
  end for
end for
while $|V| < n$ **do**
  Let (P_1, P_2) be the most frequent pair in C.
  $V \leftarrow V \cup \{P_1 \cup P_2\}$
  for all $mulpi \in B$ **do**
    if $\{P_1, P_2\} \subseteq mulpi$ **then**
      for all $Q \in mulpi - \{P_1, P_2\}$ **do**
        Delete $(Q, P_1), (Q, P_2)$ from C.
        Insert $(Q, P_1 \cup P_2)$ into C.
      end for
      $mulpi \leftarrow (mulpi - \{P_1, P_2\}) \cup \{P_1 \cup P_2\}$
    end if
  end for
end while
return V

5. SYMPHONYNET DETAILS

5.1 The 3-D Positional Embedding

Transformer [19] is the most used backbone for language model, which is designed *permutation invariant*: if the positional encoding is not added, disrupting the order of the inputs will yield the same output, for transformer model treats inputs as a *set* during self-attention. Therefore, considering this property of Transformer, we design a particular 3-D positional embedding to represent such a semi-permutation invariant feature as shown in Fig. 3. Event tokens follow a semi-permutation variant order on both the measure order and the note order axes. For example, notes played on the same position share the same note positional

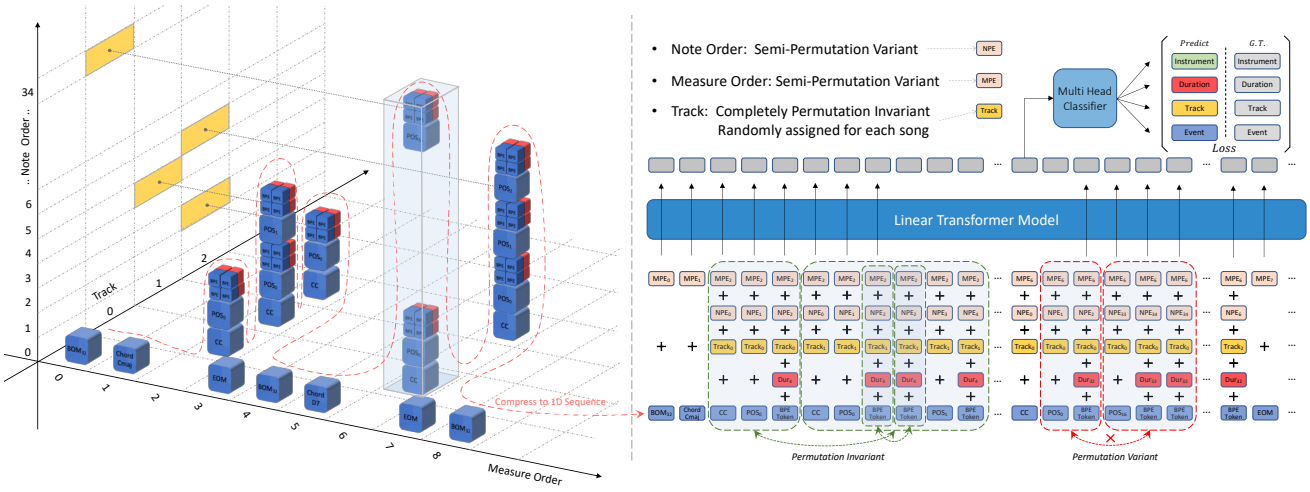


Figure 3: A illustration of the spatial and structural attributes of MMR sequence (left) and the way it is compressed and organized as model input (right).

embedding. In contrast, the track axis is entirely permutation invariant since we only need track embeddings to differentiate tracks other than a sequential order. We use the red curves to illustrate the musical moving trajectory of event tokens to better understand how we compress the spatial and structural sequence of the event tokens into one dimension and send them to the model. At last, we add all constructed embeddings vertically as the model input. To address the extraordinarily long symbolic music sequences challenge, we employ the linear transformer [20] as the backbone of our model to satisfy the length constraint. The model follows a decoder-only fashion, and we design different feed-forward heads for four attributes of musical events, which are **Instrument, Track, Duration, and Event** tokens as shown in Fig. 3.

5.2 Joint Task with Instrument Classification

We mask instrument information for every input token at the input side, and anticipate that the model will learn instrumentation from the output side with instrument loss. This will turn a succession of simple, blank notes into a fully orchestrated piece of music, analogous to colouring a black-and-white painting. This design is motivated by two primary concerns. First, we investigate the possibility if other instrument may play a certain instrument's note track. Therefore, that is a case for the model to determine to what degree the instrument fits the track's notes and how instruments interact with one another across tracks. For instance, it is allowed to substitute the piano for the marimba in some musical compositions. The intrinsic nature of a pre-assigned instrument for notes reduces the diversity of training data.

6. EXPERIMENTS AND RESULTS

This section introduces the novel symphony dataset we propose and presents two stages in the training process [1] . Secondly, we describe controllable methods to generate music under certain condition before we provide findings from Music BPE and compare them with the specific musical knowledge. Lastly, a human evaluation result analysis and scoring on several excerpts generated by different models will be presented.

6.1 Symphony Dataset

To tackle the obstacles of the symphony generation research, we gather a big corpus of symphonic music from multiple online sites and conduct a extensive data cleaning. The average duration of the 46,359 MIDI files containing multiple instruments and tracks, mostly symphony, is 4.26 minutes. The collection contains more than 279 million notes and 567 million tokens in MMR forms. Our symphony dataset is, to the best of our knowledge, the first worldwide large-scale symbolic symphonic music dataset, which might serve as a foundation for future work in multi-track music production.

6.2 Training Details

In our experiment, the model adopts 4096 as the length of input sequence. We set the embedding size for event tokens, durations, instruments and 3-D positional embedding to 512. The final size of event token vocabulary is assigned to 1000 after running Music BPE algorithm and the vocabulary size of durations, instruments, 3-D positional embeddings are derived from the dataset. The linear transformer decoder contains 12 self-attention layers and each layer consists of 16 attention heads. SymphonyNet is trained with eight 2080 Ti GPU and we use a batch size of 128 and an AdamW [21] optimizer with a learning rate of 3×10^{-4}.

6.3 Music BPE Result

After constructing a vocabulary list of length $1,000$ with Music BPE algorithm, the top-5 merged pairs shown in Fig. 4 with the highest frequency are: ($D4$, $F4$), ($C4$, $E4$),

[1] Our code, demos, dataset and further analysis can be accessed at https://symphonynet.github.io

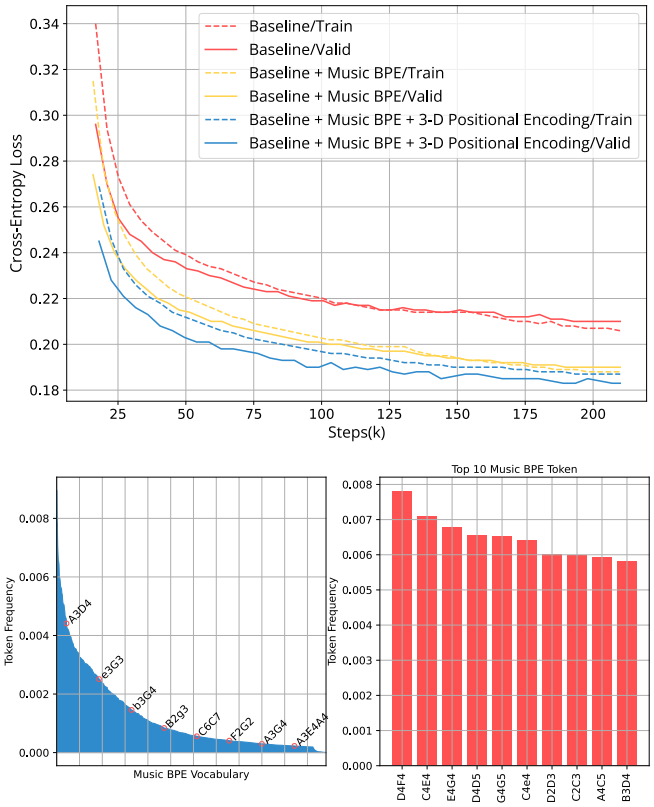


Figure 4: The training and validation curves of different models and the Music BPE note aggregation results.

	Model	C	D	H	S	O	P
Chord	Baseline	3.5	3.57	3.07	3.00	3.21	3.29
	BPE	3.64	3.64	3.14	**3.15**	3.43	3.29
	3D + BPE	**3.71**	**3.72**	**3.21**	3.07	**3.5**	**3.5**
	Human	4.43	3.43	4.14	4.36	4.14	4.14
Prime	Baseline	3.79	2.79	3.21	3.43	3.36	3.36
	BPE	**3.86**	**3.5**	**3.5**	3.5	3.64	**3.86**
	3D + BPE	**3.86**	3.14	3.43	**3.57**	**3.93**	3.64
	Human	4.36	3.57	4.36	4.00	4.36	4.36
Uncondi.	Baseline	3.52	3.46	3.04	3.07	3.11	3.07
	BPE	**3.79**	3.64	3.25	3.11	3.25	3.29
	3D + BPE	3.53	**3.93**	**3.43**	**3.32**	**3.43**	**3.32**
	Human	4.39	3.89	4.18	4.21	4.11	4.29

(a) Trained on Symphony Dataset

Model	C	D	H	S	O	P
MMM	3.20 ±0.13	2.71 ±0.12	2.51 ±0.13	2.66 ±0.12	2.80 ±0.13	2.71 ±0.11
Symph.	**3.33** ±0.15	**2.89** ±0.13	**2.76** ±0.12	**2.69** ±0.13	**2.99** ±0.12	**2.87** ±0.13

(b) Trained on Lakh MIDI Dataset

Table 2: Human evaluation results from 25 musicians and 25 non-musicians, with mean opinion scores and 95 percent confidence intervals reported.

($E4$, $G4$), ($D4$, $D5$), and ($G4$, $G5$), which are usual intervals occurring in symphony *divisi* passages. After applying Music BPE on the whole music corpus, the average token length of a $mulpi$ shortens to half (from 2.28 to 1.13), also reducing the burden of modeling ultra-long symphony sequences.

6.4 Ablation Study and Human Evaluations

We train a linear transformer decoder model with the vanilla positional encoding of GPT-3 [18] as a baseline. Then we train another model of the same architecture with the training data processed by the proposed Music BPE algorithm. Finally, we incorporate both 3-D positional embedding and Music BPE algorithm, which achieves the lowest training and validation loss after the same total training steps, as is shown in Fig. 4. The objective metric indicates that our permutation-invariant 3-D positional embedding and Music BPE algorithm could significantly improve model performance and generalization ability.

Also, we perform a human evaluation to compare the quality of generated music excerpts from different models with human composition. Participants include 25 professional musicians and 25 non-musicians. Each participant receives 16 excerpts: four excerpts conditioned on a chord progression, four excerpts conditioned on a given 4-bar prime, and eight unconditioned excerpts. The music excerpts are rated in 5 dimensions: Coherence **(C)**, Diversity **(D)**, Harmoniousness **(H)**, Structureness **(S)**, Orchestration **(O)** and Overall Preference **(P)**, in a 5-point Likert scale. As shown in Table 2a, the model with 3-D positional embedding and Music BPE beats most of the approaches. It is worth noting that excerpts generated by our models surpass the human compositions in the indicator of diversity marked by yellow color.

To further explore the model performance, we retrain SympohyNet on Lakh MIDI Dataset [22] with the same backbone model architecture as MMM [15], and carry out another human evaluation to compare with MMM. Each participant receives 10 excerpts: five generated unconditionally from MMM and the others generated unconditionally from retrained SymphonyNet. The results are presented in Table 2b, which indicate that SymphonyNet surpass MMM in all indicators. Overall, the human hearing test suggests that SymphonyNet can construct coherent, unique, complex, and harmonic symphonies.

7. CONCLUSION

In this work, we illustrate the properties of multi-track and multi-instrument music, like symphony, and propose a novel MMR representation with 3-D positional embedding for modelling it. To tokenize the ultra-long symbolic music sequence at sub-word level, we propose the Music BPE algorithm. Besides, we design a joint task for the model to learn auto-orchestration. Human evaluation results show that our suggested technique produces competitive symphonic music when compared to human compositions. In the future, we will investigate modelling long-term musical structures, since complex music, such as symphonies, often consists of numerous parts or movements.

8. ACKNOWLEDGEMENTS

Thanks for the anonymous reviewers for their valuable comments. This work is supported by High-grade, Precision and Advanced Discipline Construction Project of Beijing Universities, Major Projects of National Social Science Fund of China (Grant No. 21ZD19), and Nation Culture and Tourism Technological Innovation Engineering Project.

9. REFERENCES

[1] G. Hadjeres, F. Pachet, and F. Nielsen, "DeepBach: a steerable model for bach chorales generation," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1362–1371.

[2] L. Yang, S. Chou, and Y. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, 2017, pp. 324–331.

[3] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International conference on machine learning*. PMLR, 2018, pp. 4364–4373.

[4] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *International Conference on Learning Representations*, 2018.

[5] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "Popmag: Pop music accompaniment generation," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1198–1206.

[6] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, pp. 178–186.

[7] Z. Wang and G. Xia, "MuseBERT: Pre-training music representation for music understanding and controllable generation," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, 2021.

[8] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[9] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T. Liu, "Musicbert: Symbolic music understanding with large-scale pre-training," in *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, ser. Findings of ACL, vol. ACL/IJCNLP 2021. Association for Computational Linguistics, 2021, pp. 791–800. [Online]. Available: https://doi.org/10.18653/v1/2021.findings-acl.70

[10] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1715–1725.

[11] P. Gage, "A new algorithm for data compression," *The C Users Journal*, vol. 12, no. 2, pp. 23–38, 1994.

[12] H. Dong, C. Donahue, T. Berg-Kirkpatrick, and J. J. McAuley, "Towards automatic instrumentation by learning to separate parts in symbolic multitrack music," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, 2021.

[13] C. Payne, "MuseNet," *OpenAI Blog*, vol. 3, 2019.

[14] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. J. McAuley, "Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, 2019.

[15] J. Ens and P. Pasquier, "Mmm: Exploring conditional multi-track music generation with the transformer," *arXiv preprint arXiv:2008.06048*, 2020.

[16] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Thirty-second aaai conference on artificial intelligence*, 2018.

[17] H. Liang, W. Lei, P. Y. Chan, Z. Yang, M. Sun, and T.-S. Chua, "Pirhdy: Learning pitch-, rhythm-, and dynamics-aware embeddings for symbolic music," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 574–582.

[18] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[20] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165.

[21] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2018.

[22] C. Raffel, "Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching," Ph.D. dissertation, COLUMBIA UNIVERSITY, 2016.

MULAN: A JOINT EMBEDDING OF MUSIC AUDIO AND NATURAL LANGUAGE

Qingqing Huang[1] **Aren Jansen**[1] **Joonseok Lee**[1,2]
Ravi Ganti[1] **Judith Yue Li**[1] **Daniel P. W. Ellis**[1]
Google Research[1], Seoul National University[2]
{qqhuang, arenjansen, joonseok, gmravi, judithyueli, dpwe}@google.com

ABSTRACT

Music tagging and content-based retrieval systems have traditionally been constructed using pre-defined ontologies covering a rigid set of music attributes or text queries. This paper presents MuLan: a first attempt at a new generation of acoustic models that link music audio directly to unconstrained natural language music descriptions. MuLan takes the form of a two-tower, joint audio-text embedding model trained using 44 million music recordings (370K hours) and weakly-associated, free-form text annotations. Through its compatibility with a wide range of music genres and text styles (including conventional music tags), the resulting audio-text representation subsumes existing ontologies while graduating to true zero-shot functionalities. We demonstrate the versatility of the MuLan embeddings with a range of experiments including transfer learning, zero-shot music tagging, language understanding in the music domain, and cross-modal retrieval applications.

1. INTRODUCTION

Classifiers are generally trained to label examples with predefined and fixed class inventories, which are often manually specified as a structured ontology indicating interclass relationships. Empowered by recent advances in neural language modeling and their demonstrated transfer learning competence, researchers have begun exploring less restrictive natural language interfaces to access the categorical information underlying raw content signals. The majority of this work has been in the visual and audio event domain, where a recent series of studies have demonstrated the utility of jointly embedding media content with natural language captions [1–5]. These joint embeddings have demonstrated strong capabilities in a range of applications, including transfer learning, cross-modal retrieval, automatic captioning, and zero-shot classification.

The success of these efforts strongly depends on large-scale training resources and hefty neural network architectures that are flexible enough to model the complex, non-monotonic relationship between language and other modalities. In particular, the visual domain has greatly benefited from the availability of large amounts of captioned images available across the web [1]. However, in the general environmental audio domain, such large-scale audio-caption pairs are less readily available and related efforts have relied on small captioned datasets [6, 7]. Critically, these datasets do not span the diversity of sound-descriptive language and their success in the more difficult zero-shot setting has been lacking [3, 8, 9].

This paper considers this task of jointly embedding audio and natural language, but focuses specifically on the music domain. Our goal is to produce a flexible language interface with which any musical concept can be linked to related music audio. We face similar training data prerequisites to works listed above. However, while general environmental audio consists of background sounds that are unlikely to elicit unprompted description, music audio is often a central focus. Consequently, text associated with music videos is much more likely to relate to the underlying musical concepts that we aim to model (e.g., genres, artists, moods, structure). Thus, our strategy is to assemble a collection of textual annotations extracted from metadata, comments, and playlist data and map them to a training set of over 44 million internet music videos. As was the case with image-text model training in [1], our text data only truly refers to the musical content in a fraction of cases. Therefore, we also explore text pre-filtering using a text classifier separately trained to identify music descriptions.

We use this large-scale dataset to train MuLan, a new generation of semantically-structured music audio embedding model equipped with a natural language interface. MuLan employs a two-tower parallel encoder architecture, using a contrastive loss objective that elicits a shared embedding space between music audio and text. We demonstrate that MuLan not only leads to state-of-the-art performance in transfer learning for various music information retrieval tasks, but also enables a range of functionalities in cross-modal text-to-music retrieval, zero-shot music tagging, and music-domain language understanding.

2. RELATED WORK

Audio representation learning. Transfer learning using large-scale, task-agnostic pretraining of general-purpose content representations has become a dominant approach

© Q. Huang, A. Jansen, J. Lee, R. Ganti, J. Y. Li, and D. P. W. Ellis. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Q. Huang, A. Jansen, J. Lee, R. Ganti, J. Y. Li, and D. P. W. Ellis, "MuLan: A Joint Embedding of Music Audio and Natural Language", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

in several fields. Audio representation learning has been no exception, including both general environmental audio [10, 11] and music audio [12–15]. Different pretraining mechanisms have been explored. In supervised pretraining, an Audio Spectrogram Transformer (AST) [10], pretrained on ImageNet [16] and AudioSet [17], achieved state-of-the-art results in various tagging tasks. A strong early baseline for music audio representation learning was provided in [13], using the Million Song Database [18].

In unsupervised and self-supervised pretraining, both discriminative and generative model approaches have been demonstrated to be successful. Discriminative training was explored in [19–22] where the models tried to learn representations that assign higher similarity to audio segments extracted from the same recording compared to segments from different recordings. SSAST [11] explored similar discriminative losses, as well as generative masked spectrogram patch modeling. It was shown in [23] that the intermediate embedding of a generative model also provides a strong audio representation for downstream classification. Various forms of weak supervision, such as user interaction statistics and visual cues, have also been examined in [24–26].

Our work focuses on developing a recipe of cross-modal supervision using an abundance of text annotations that are weakly associated with the music audio. We benchmark the transfer learning capabilities of the learned representations against analogous past work, and also evaluate different audio encoder architectures.

Cross-modal contrastive learning. Spurred by the success of using contrastive learning to align image features and free-form natural language using large-scale data [1, 2], tri-modal architectures were proposed in [27] and [5] where an audio tower was introduced to the image-text model and contrastive learning is used to enforce the cross-modal alignment. Along the same line in the audio domain, [8] used contrastive learning to align the latent representation of audio and associated tags. The tags come from a fixed vocabulary of size 1K from Freesound [28], and the input to the text encoder was the multi-hot encoded tags. Follow up work in [9] uses a pretrained, non-contextual word embedding (Word2Vec) model to support generalization to new terms beyond the 1K tags. However, this still does not support generalization to free-form natural language. Contrastive learning was also explored in [29] for zero-shot audio classification, using AudioSet and ESC-50 [30] data. Our method focuses on mining a much larger scale collection of audio-text pairs specifically for the music domain. Our data scale supports using state-of-the-art Transformer-based audio and contextual language encoders, which led to a truly arbitrary zero-shot music tagging and retrieval for the first time.

Music text joint embedding models. Content-based music information retrieval requires linking the rich semantics expressible to free-form text with both broad and fine-grained musical properties. One approach is to consider a large number of text label classes and try to ground the semantics in music with a multi-label classification task. In [25], a large vocabulary of 100K n-grams was mined from noisy natural language text associated with music videos. Then, a cross entropy loss was employed to train the music audio encoder, where the softmax layer weights served as text label embeddings that were aligned with audio features by construction. The work in [31] explored various training tasks (classification, regression, metric learning) to align free-form text and music audio, relying on pre-existing emotion labels to connect the modalities.

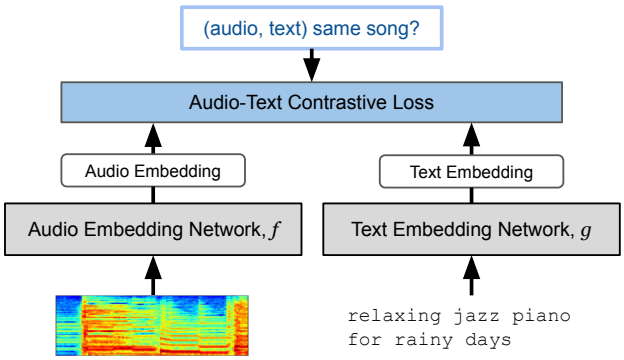


Figure 1. Learning framework diagram.

Closest to our work is MuLaP [32], where 250K audio-caption pairs were mined from a private production music library and used to train a multimodal Transformer with early fusion of the two modalities. Their choice of early fusion, as accomplished with cross-attention layers, restricts the utility of the resulting embeddings to transfer learning applications. Critically, our two-tower parallel encoder approach results in a joint embedding space that provides a natural language interface to arbitrary music audio. This opens up downstream opportunities for cross-modal retrieval, zero-shot tagging, and language understanding.

3. PROPOSED APPROACH

Our goal is to construct a shared embedding space for music audio and free-form natural language text, in which proximity is predictive of shared semantics both within and across modalities. To accomplish this, we rely on cross-modal contrastive learning and a simple two-tower architecture. This is a highly data-intensive endeavor, which we support by mining a large-scale training dataset of (audio, text) pairs. We describe these components in turn below.

3.1 Learning Framework

Figure 1 shows a high-level schematic of the learning framework. Each MuLan model consists of two separate embedding networks for the audio and text input modalities. These networks share no weights, but each terminates in ℓ_2-normalized embedding spaces with the same dimensionality, d. The audio embedding network, $f : \mathbb{R}^{F \times T} \rightarrow \mathbb{R}^d$, takes as input log mel spectrogram context windows with F mel channels and T frames. The text embedding network, $g : \mathcal{A}^n \rightarrow \mathbb{R}^d$ takes as input a null-padded text token sequence of length n over a token vocabulary $\mathcal{A}$.

Given a set of music recordings and the associated text elements for each recording, we construct a cross-modal training dataset of (audio, text) pairs as follows. For each recording, we compute an F-channel log mel spectrogram and extract a collection of T-frame context windows. We null-pad or truncate each associated text element to a fixed length n. Then, each mini-batch $\mathcal{B}$ consists of a set of B target audio-text pairs of the form $\{(\mathbf{x}^{(i)}, \mathbf{t}^{(i)})\}_{i=1}^{B}$. Here, each target pair is sampled by first selecting a random recording and sample a random spectrogram context window $\mathbf{x}^{(i)} \in \mathbb{R}^{F \times T}$ from it. Next, we randomly select one of its associated text elements $\mathbf{t}^{(i)} \in \mathcal{A}^n$. This sampling scheme means that multiple epochs are required to cover the entirety of the training audio and all the associated text. We also experimented with concatenating multiple text annotations for each example, but it did not generally work as well.

We train to minimize a batch-wise Contrastive Multiview Coding loss function [33], which is a cross-modal extension of the popular InfoNCE and NT-Xent losses [34, 35]. For each batch $\mathcal{B}$, this loss $\mathcal{L}(\mathcal{B})$ takes the form

$$\sum_{i=1}^{B} -\log \left[\frac{h[f(\mathbf{x}^{(i)}), g(\mathbf{t}^{(i)})]}{\sum_{j \neq i} h[f(\mathbf{x}^{(i)}), g(\mathbf{t}^{(j)})] + h[f(\mathbf{x}^{(j)}), g(\mathbf{t}^{(i)})]} \right],$$

where h is a critic function given by $h[\mathbf{a}, \mathbf{b}] = \exp(\mathbf{a}^T \mathbf{b}/\tau)$ for $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d$, and $\tau \in (0, 1]$ is a trainable temperature hyperparameter. For our ℓ_2-normalized embedding model outputs, the inner product is effectively cosine similarity. The critic's goal is to produce a large positive value for target audio-text pairs, and a small value close to zero for all non-target pairs constructed within the batch. Temperature values less than one function to increase the output range of h. Previous research [35, 36] demonstrated that a large batch size is beneficial to contrastive loss optimization.

3.2 Audio Embedding Network

For the audio embedding tower, f, we consider two proven audio architectures. Following its introduction to the audio machine learning community [37], the *Resnet-50* architecture has become a common and well-performing option. It is a straightforward adaptation of the original vision architecture: as in [37], we remove the stride of 2 in the first convolutional layer and apply to log mel spectrograms ($F = 64$ mel channels, 25 ms Hanning window, 10 ms step size) treated as grayscale images. Unlike the Resnet-50 model in [37] which operated on 0.96-second context windows, in order to allow the modeling of longer-term musical structure, our implementation takes as input 10-second windows (randomly selected from each training clip), in the form of $(F = 64) \times (T = 1000)$ spectrogram patches. During training, we apply SpecAugment to each spectrogram using the parameters from [10] before passing it into the embedding network. A final mean pooling operation is applied across time and mel channels, followed by a linear fully connected layer with $d = 128$ units, whose output is ℓ_2-normalized. We pretrain all but the final linear transform layer via logistic regression on AudioSet [17], including all 527 classes, and discard the final classifier layer before fine-tuning for our task.

Table 1. Text annotation examples.

Type	Examples
Short-form (SF)	tags like genre, mood, instrument, artist name, song title, album name
Long-form (LF)	'Hip-hop features rap with an electronic backing.' 'The melody is so nostalgic and unforgettable.'
Playlist (PL)	'Feel-good mandopop indie', 'Latin workout' 'Salsa for broken hearts', 'Piano for study'

Audio Spectrogram Transformer (AST) is a port of the successful Vision Transformer (ViT) base architecture and is currently the state-of-the-art in the audio event classification space [10]. AST consists of a stack of 12 Transformer blocks (hidden dimension 768, 12 self-attention heads) that are applied to a sequence of "tokens" corresponding to a flattened set of linear-transformed 16×16 (stride 10 along both axes) time-frequency patches extracted from the $(F = 128) \times (T = 1000)$ log mel spectrogram context windows. We again apply SpecAugment during training. Similar to the Transformer-based language models, trainable positional encodings are added to the sequence of patch tokens, and a `[CLS]` token is prepended to the sequence as a summary of the contextual patch embeddings. We apply a linear fully-connected layer with $d = 128$ units and ℓ_2-normalization to the final 768-dimensional encoding at the `[CLS]` token position, and this forms the output of audio embedding network f. We warm-start training for all but the final linear transform layer using the public AST checkpoint [10].

3.3 Text Embedding Network

For the text embedding model, we consider the commonly-used Bidirectional Encoder Transformer (BERT) with base-uncased architecture [38], which consists of a stack of 12 Transformer blocks (hidden dimension of 768 and 12 self-attention heads). We apply the BERT wordpiece tokenizer to convert a text input string into a sequence of tokens ($n = 512$). The output of the text embedding network is defined to be the `[CLS]` token embedding, linearly transformed to the shared audio-text embedding space of dimension $d = 128$ and subsequently ℓ_2-normalized. We warm-start our text embedding network using the publicly available checkpoint [39].

3.4 Training Dataset Mining

To assemble a large-scale collection of (audio, text) pairs needed to train our MuLan embedding models, we start with a collection of 50 million internet music videos. From the soundtrack of each video, we extract a 30-second clip starting at the 30 second mark. We then apply a pre-existing music audio detector and discard any clip that is less than half music content. After this filtering, we are left with approximately 44 million 30-second clips, which amounts to nearly 370K hours of audio.

Table 2. Statistics for text data sources. Tokens counts (in billions) are across all 44M videos. APV is the average number of text annotations (i.e. separate free-form strings) per video, including those with none.

	Pre-filter		**Post-filter**	
Type	Tokens (B)	APV	Tokens (B)	APV
Short-form	31.2	42.9	5.4	29.6
Long-form	30.7	70.7	0.2	0.4
Playlists	2.5	24.3	-	-

For each music video, we consider 3 sources of noisy text data: (i) *short-form* (SF) text including video titles and tags; (ii) *long-form* (LF) text including video descriptions and comments; and (iii) titles of 171 million *playlists* (PL) that are linked to the internet music videos in our dataset. None of these text sources is guaranteed to be referring to the musical properties of the soundtrack. In particular, comments data contains the most noise, and can be subjective or less directly related to the music content. In Table 1, we show examples that are indeed music-related to give the readers a flavor of each type of text annotation.

In observance of the highly noisy text, we experimented with training MuLan with the SF and LF text data filtered to a cleaner set of music-descriptive annotations (PL is used unfiltered). For this, we fine-tune a pre-trained BERT model with a binary classification task on a small curated set of 700 sentences, which are manually labeled to be music-descriptive or not. We then apply this text classifier to filter the sentences in the LF annotations. Separately, we apply a set of rule-based filtering heuristics to clean up the SF annotations. Table 2 shows the size and coverage of each of these text sources, both before and after filtering. Note that playlist titles and filtered long-form annotations are only available for a minority of recordings in the dataset (18M and 6.8M out of the total 44M, respectively).

We also convert AudioSet into a set of audio-text pairs, denoted below as ASET. Specifically, we include all examples for all 527 classes, using each label string attached to an example as an associated text annotation. This results in a set of approximately 2 million 10-second clips for training, each with 1.8 label annotations on average. Given the great scale imbalance of these four different data sources, which is often at odds with their linguistic richness and quality, we construct each mini-batch with a prescribed set of proportions that were chosen without any optimization: 2:2:1:1 for SF:LF:PL:ASET. This means that despite its small scale, the (e.g.) filtered LF annotations still comprise 1/3 of each mini-batch.

4. EXPERIMENTS

We evaluate MuLan using both the Resnet-50 audio encoder (M-Resnet-50) and AST audio encoder (M-AST). In both cases we use the BERT-base-uncased architecture as the text encoder. We train all models for 14 epochs on the collection of audio-text pairs mined from the 44M music recordings and the processed text labels in all categories: AudioSet (ASET), short-form tags (SF), long-form sentences (LF), playlist information (PL). We use the Adam optimizer with a step decay learning rate schedule using a decay factor 0.9 applied every 40K steps and initial values of 5×10^{-5} for M-Resnet-50 and 4×10^{-5} for M-AST. The temperature parameter is initialized to $\tau = 0.1$ for all models. M-Resnet-50 is trained with a batch size of $B = 6144$ pairs, while $B = 5120$ pairs were used for M-AST due to memory limitations. Since M-AST and M-Resnet-50 show roughly similar performance in the evaluation tasks considered, we use M-Resnet-50 throughout the text ablation study for its better training efficiency.

4.1 Evaluation Tasks

4.1.1 Zero-shot Music Tagging

Given a music clip and a set of candidate text label tags, we define each prediction score as the cosine similarity between the audio embedding of the music clip and the text embedding of each tag string. The generalization ability of the proposed method to potentially unseen target labels is achieved through (i) the use of a contextual text encoder, which provides a flexible prediction space, and (ii) the use of cross-modal contrastive learning to anchor the language semantics to an audio representation.

We conduct this evaluation with two music tagging benchmarks: MagnaTagATune (MTAT) [40] and the music related portion of AudioSet [17]. For MagnaTagATune, we consider both the well-exercised top-50 tag set, as well as the full 188 tag set. We use standard train/validation/test partitions (note that zero-shot experiments do not use train/validation) and report class-balanced area under the receiver operating characteristic curve (AUC-ROC) on the test set. The audio clips in MagnaTagATune are 29 seconds long, so we split each into 3 non-overlapping 10-second segments and average the segment-level embeddings to get the clip-level embedding. For AudioSet, we consider a 25-way genre tagging task (Gen-25) as studied in [25], and a richer 141-way tagging task (Mu-141) that includes the entire music subtree of AudioSet ontology.

It is important to note that AudioSet is included in contrastive training, and a fraction of MTAT classes overlap with the AudioSet ontology. As a result, AudioSet and (to lesser extent) MTAT evaluations are not strictly zero-shot from a label exposure perspective. However, the explicit, matched AudioSet supervision is diluted by the abundance of free-form language supervision during MuLan training. Therefore, by comparing MuLan models and conventional AudioSet classifiers, we can measure the cost of moving to a flexible natural language interface that additionally supports classes outside the AudioSet ontology.

4.1.2 Transfer Learning with Linear Probes

In addition to the zero-shot experiments introduced above, we also evaluate the audio encoder as a general purpose feature extractor for downstream tagging tasks. We again consider the two benchmarks of MagnaTagATune and AudioSet, and use the training datasets to train an independent per-class logistic regression layer on top of the frozen 128-dimensional audio embeddings. We follow the same eval-

Table 3. Text triplet evaluation examples.

Eval Set	Anchor / Positive / Negative
Ontology	Steelpan / Sounds of a tuned percussion instrument originally constructed from steel oil drums by hammering out small patches on the head to produce separate pitches. / The sound of a musical instrument that produces sound by vibration of air in a tubular resonator in sympathy with the vibration of the player's lips.
Playlist	Relaxing Korean Pop / Lets make your chill mood with a collection of easy-going sounds from Korean artists. / These fun and upbeat songs from the alternative side of the pop music spectrum will keep you energized while you exercise.

uation protocol of past transfer learning studies using these datasets, allowing for a direct comparison of performance.

4.1.3 Music Retrieval from Text Queries

Given a music search collection and a text query, MuLan provides the ability to retrieve the music clips that are closest to the query in the embedding space. This evaluation is relevant to music retrieval applications, where content features can offer finer-grained and more complete similarity information when compared with metadata-based methods [41]. We consider a proprietary collection of 7,000 expert-curated playlists, which do not overlap with the playlist information used in training. Each expert-curated playlist has a title and a description, and consists of 10-100 music recordings. The playlist titles are usually short phrases, including a mixture of genres, sub-genres, moods, activities, artist names, and compositional elements (e.g. *'Indie Pop Workout'*, *'Relaxing Korean Pop'*). Playlist descriptions consist of one or more complete sentences (see pos/neg entries of "Playlist" row of Table 3 for examples). The playlist evaluation includes approximately 100K unique recordings.

We construct two cross-modal retrieval evaluation sets from the expert-curated playlist data, one using titles as queries and the other using descriptions. For each dataset, we use the recordings belonging to the corresponding playlist as the ground truth retrieval targets, and all the 100K recordings as the pool of candidates. We report both AUC-ROC and mean average precision (mAP). We use the same embedding averaging and cosine similarity-based scoring mechanism as in the zero-shot tagging case. However, the playlist information is of substantially different nature compared to the tags involved in the music tagging benchmarks. Instead of a small vocabulary of mostly basic genres and instruments, the playlist titles and descriptions have much finer-grained information and are similar to queries that are presented to music search engines.

4.1.4 Text Triplet Classification

Compared to the conventional pre-trained BERT model, our text encoder is fine-tuned using in-domain music data and cross-modal contrastive loss. Note that there are no text-only training objectives. To measure whether our proposed method deepens the text encoder's understanding of music related text, we directly evaluate the text embeddings with a triplet classification task. Each triplet consists of 3 text strings of the form of (*anchor*, *pos*, *neg*), and it is considered correct if *pos* is closer than *neg* to *anchor* in the text embedding space. We derive two such text triplet evaluation sets. The first uses the AudioSet ontology [17]: for each of the 141 music related classes, we use its label string as the anchor text, its long-form description as the positive text, and sample 5 random class's long-form description as the negative text to construct 5 triplets. For the second set, we sample 1,000 triplets from the expert-curated playlist data in a similar fashion: we first sample a playlist, set the anchor and positive text to be its title and description, respectively, and then set the negative text to be the description of another randomly sampled playlist. Examples of both sets are shown in Table 3.

Table 4. Music tagging results reported in AUC-ROC.

	AudioSet		MTAT	
Model	**Gen-25**	**Mu-141**	**Top-50**	**All-188**
(a) Zero-shot (Trained w/ ASET + SF + LF + PL)				
M-AST	0.840	0.909	0.778	**0.776**
M-Resnet-50	0.840	0.899	**0.782**	0.772
(b) Text ablation (using M-Resnet-50 Zero-shot)				
ASET + SF + LF	0.839	0.907	0.760	0.756
ASET + SF	0.839	0.885	0.754	0.747
ASET	**0.886**	**0.942**	0.753	0.771
SF/LF Unfiltered	0.845	0.908	0.774	0.766
(c) Linear probe				
M-AST	0.906	**0.942**	0.925	0.953
M-Resnet-50	**0.910**	0.940	**0.927**	**0.954**
Baselines:				
Hybrid [25]	0.904	0.920	0.915	0.941
JukeBox [15, 23]	-	-	0.915*	-
MuLaP [32]	-	-	0.893*	-
CLMR [22]	-	-	0.866*	-
(d) End-to-end training baselines				
AST [10]	0.888	**0.949**	-	-
SC-CNN [42]	-	-	0.913*	-

* indicates that the number is brought from the original paper.

4.2 Results and Discussion

4.2.1 Music Tagging

Table 4(a) shows the zero-shot tagging metrics, where M-Resnet-50 and M-AST obtain comparable performance. Note that there can be a significant misalignment between the word sense of a label in the tagging evaluation compared to that in our training text. This can lead to a degradation in performance relative to the explicitly supervised linear probe setting where the task-expected tag semantics can be learned. The MTAT gap is substantially larger than AudioSet's, driven by particularly bad performance for (i) MTAT tags with nonspecific meaning or multiple senses, e.g. "weird" and "beats"; and (ii) MTAT tags involving simple negation (e.g. "not rock", "no piano"). This is a result of the text encoder not adequarely modeling the meaning of these negated concepts, which is a well known problem with BERT [43, 44] (the text embedding of "not rock" is similar to "rock" and performance suffers).

Table 4(b) shows the results of the text ablation study,

Table 5. Text query music retrieval evaluation results. Text ablation/unfiltered models use M-Resnet-50.

	Title		Description	
Model	AUC	mAP	AUC	mAP
M-AST	**0.933**	0.110	**0.903**	**0.090**
M-Resnet-50	0.931	0.104	0.901	0.084
Text Ablation:				
ASET+SF+LF	0.917	0.101	0.892	0.077
ASET+SF	0.913	0.089	0.867	0.060
ASET	0.626	0.005	0.688	0.009
SF/LF Unfiltered	**0.933**	**0.111**	0.897	0.081

Table 6. Text triplet classification accuracy AudioSet ontology evaluation and Playlist title to description evaluation. Text ablation/unfiltered models use M-Resnet-50.

Model	**Playlist**	**AudioSet**
M-AST	**0.959**	**0.962**
M-Resnet-50	0.945	0.951
Text Ablation:		
ASET + SF + LF	0.935	0.952
ASET + SF	0.910	0.938
ASET	0.693	0.818
SF/LF Unfiltered	0.949	0.959
Baselines:		
SimCSE [45]	**0.950**	0.938
SBERT [46]	0.942	0.889
USE [47]	0.918	**0.946**
BERT [38]	0.850	0.847

which aims to understand the benefits of different sources of text labels. Note that as we remove each dataset we maintain the same proportions described in Section 3.4. Unsurprisingly, training with AudioSet alone gets the highest AUC in AudioSet evaluation, with the text encoder learning the exact label semantics reflected in the test data. On the other hand, including more data sources in general improves performance on all other downstream tasks (MTAT, retrieval/text triplet evaluations in Tables 5 and 6) and the loss on AudioSet AUC is relatively minor. We observe that for the music tagging tasks considered, training with unfiltered data actually achieves comparable performance compared to the filtered version. That the model learns similarly useful associations without being overwhelmed by the sheer amount of noise in the raw text data came as a surprise. We speculate that our text filtering was too aggressive, having removed annotations that were not obviously music-related, but semantically important nonetheless. Since contrastive learning is highly noise tolerant, the gain from restricting to more strongly aligned audio-text pairs may have been offset by the loss of a large set of additional useful pairs.

Table 4(c) shows that when applying linear probes on MuLan audio embeddings, we achieve SOTA transfer learning performance on all tagging tasks. This demonstrates that MuLan's pretrained audio encoder continues to produce high quality general-purpose music audio embeddings, while also supporting new natural language applications. Finally, Table 4(d) lists end-to-end training baselines for 3 of these tasks. Our linear probe results exceed 2 of 3, and only slightly trails a SOTA AST AudioSet classifier.

4.2.2 Music Retrieval from Text Queries

In Table 5, we evaluate MuLan models (including with text/filter ablation) on the query retrieval evaluation tasks introduced in Section 4.1.3. Even though we start with a BERT checkpoint pretrained with massive language resources, training MuLan with only AudioSet clips and label annotations provides very limited ability to ground in-domain natural language to music. Such limited cross-modal supervision does not generalize to the rich semantics that appear in the playlist titles and descriptions, which are more in line with the complex queries that are presented to real-world music search engines. We observe significant gain after including the large-scale short-form tags mined from the internet, which helps the model learn to ground more fine-grained music concepts. There is additional gain when including comments and playlist data, where the complete sentences are helpful for grounding the more complex queries, including multi-term queries (e.g.*'instrumental action movie soundtrack'*), compositional queries (e.g. *'classical music with middle eastern influence'*), and even queries with negation (e.g. *'hard rock without vocals'*). Again, we find that training is surprisingly robust to annotation noise, achieving similar performance using unfiltered training text.

4.2.3 Text Triplet Classification

Table 6 lists triplet classification accuracy on evaluations introduced in Section 4.1.4. We compare MuLan text embedding against the following baselines: Sentence Transformer [46], SimCSE [45], Universal Sentence Embedding [47], and the average token embedding of BERT-base-uncased (this outperforms the `[CLS]` encoding by a large margin). All baselines are Transformer-based models with similar size to ours. The first three were trained with sentence-level contrastive loss, while BERT is trained with masked language prediction. We warmstart the MuLan text encoder using this same BERT baseline, but it is subsequently only trained with the cross-modal loss. We find that when including our long-form text annotations, the resulting text embedding model, which is now specialized to the music domain, outperforms the generic sentence embedding models. While it is not surprising in-domain text is helpful, it is remarkable that successful specialization is accomplished without using any text-only fine-tuning loss.

5. CONCLUSIONS

We presented a music audio and natural language joint embedding model trained with an unprecedented scale of weakly paired text and audio data. Our experiments demonstrate the versatility of the natural language interface in a range of applications. The pretrained audio embeddings also achieve SOTA transfer learning performance on music tagging benchmarks. This is a first attempt at building a free-form natural language interface for music audio and there is plenty of room for improvement. Specifically, we believe improved text filtering methods that better distinguish weak signal from absolute noise will result in better handling of rare and subtle language constructs.

6. REFERENCES

[1] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision," in *Proc. of the International Conference on Machine Learning (ICML)*, 2021.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *Proc. of the International Conference on Machine Learning (ICML)*, 2021.

[3] A. S. Koepke, A.-M. Oncescu, J. Henriques, Z. Akata, and S. Albanie, "Audio retrieval with natural language queries: A benchmark study," *IEEE Transactions on Multimedia*, 2022.

[4] K. Kilgour, B. Gfeller, Q. Huang, A. Jansen, S. Wisdom, and M. Tagliasacchi, "Text-driven separation of arbitrary sounds," *arXiv:2204.05738*, 2022.

[5] A. Nagrani, P. H. Seo, B. Seybold, A. Hauth, S. Manen, C. Sun, and C. Schmid, "Learning audio-video modalities from image captions," *arXiv:2204.00679*, 2022.

[6] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An audio captioning dataset," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[7] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating captions for audios in the wild," in *Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Human Language Technologies*, 2019.

[8] X. Favory, K. Drossos, T. Virtanen, and X. Serra, "COALA: Co-aligned autoencoders for learning semantically enriched audio representations," *arXiv:2006.08386*, 2020.

[9] ——, "Learning contextual tag embeddings for cross-modal alignment of audio and tags," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[10] Y. Gong, Y.-A. Chung, and J. Glass, "AST: Audio spectrogram transformer," *arXiv:2104.01778*, 2021.

[11] Y. Gong, C.-I. J. Lai, Y.-A. Chung, and J. Glass, "SSAST: Self-supervised audio spectrogram transformer," *arXiv:2110.09784*, 2021.

[12] P. Hamel, M. E. Davies, K. Yoshii, and M. Goto, "Transfer learning in MIR: Sharing learned latent representations for music audio classification and similarity," in *Proc. of International Conference on Music Information Retrieval (ISMIR)*, 2013.

[13] A. Van Den Oord, S. Dieleman, and B. Schrauwen, "Transfer learning by supervised pre-training for audio-based music classification," in *Proc. of International Conference on Music Information Retrieval (ISMIR)*, 2014.

[14] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proc. of International Conference on Music Information Retrieval (ISMIR)*, 2017.

[15] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv:2005.00341*, 2020.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[17] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "AudioSet: An ontology and human-labeled dataset for audio events," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[18] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proc. of International Conference on Music Information Retrieval (ISMIR)*, 2011.

[19] A. Jansen, M. Plakal, R. Pandya, D. P. Ellis, S. Hershey, J. Liu, R. C. Moore, and R. A. Saurous, "Unsupervised learning of semantic audio representations," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[20] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[21] N. Turpault, R. Serizel, and E. Vincent, "Semi-supervised triplet loss based learning of ambient audio embeddings," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

[22] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," *arXiv:2103.09410*, 2021.

[23] R. Castellon, C. Donahue, and P. Liang, "Codified audio language modeling learns useful representations for music information retrieval," in *Proc. of International Conference on Music Information Retrieval (ISMIR)*, 2021.

[24] S. Oramas, O. Nieto, F. Barbieri, and X. Serra, "Multi-label music genre classification from audio, text, and images using deep features," in *Proc. of International Conference on Music Information Retrieval (ISMIR)*, 2017.

[25] Q. Huang, A. Jansen, L. Zhang, D. P. Ellis, R. A. Saurous, and J. Anderson, "Large-scale weakly-supervised content embeddings for music recommendation and tagging," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[26] A. Ferraro, X. Favory, K. Drossos, Y. Kim, and D. Bogdanov, "Enriched music representations with multiple cross-modal contrastive learning," *IEEE Signal Processing Letters*, vol. 28, pp. 733–737, 2021.

[27] A. Guzhov, F. Raue, J. Hees, and A. Dengel, "Audioclip: Extending CLIP to image, text and audio," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[28] F. Font, G. Roma, and X. Serra, "Freesound technical demo," in *Proc. of the ACM International conference on Multimedia (MM)*, 2013.

[29] H. Xie and T. Virtanen, "Zero-shot audio classification via semantic embeddings," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1233–1242, 2021.

[30] K. J. Piczak, "ESC: Dataset for environmental sound classification," in *Proc. of the ACM International Conference on Multimedia (MM)*, 2015.

[31] M. Won, J. Salamon, N. J. Bryan, G. J. Mysore, and X. Serra, "Emotion embedding spaces for matching music to stories," in *Proc. of International Conference on Music Information Retrieval (ISMIR)*, 2021.

[32] I. Manco, E. Benetos, E. Quinton, and G. Fazekas, "Learning music audio representations via weak language supervision," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[33] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *Proc. of European Conference on Computer Vision (ECCV)*, 2020.

[34] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv:1807.03748*, 2018.

[35] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. of International Conference on Machine Learning (ICML)*, 2020.

[36] H. Lee, J. Lee, J. Y.-H. Ng, and P. Natsev, "Large scale video representation learning via relational graph clustering," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[37] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "CNN architectures for large-scale audio classification," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

[38] M.-W. C. Kenton, Jacob Devlin and L. K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, 2019, pp. 4171–4186.

[39] TensorflowHub, "https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4," 2022.

[40] E. Law and L. Von Ahn, "Input-agreement: a new mechanism for collecting data using human computation games," in *Proc. of the ACM SIGCHI Conference on Human Factors in Computing Systems*, 2009.

[41] R. Typke, F. Wiering, R. C. Veltkamp, J. D. Reiss, G. A. Wiggins *et al.*, "A survey of music information retrieval systems," in *Proc. of International Conference on Music Information Retrieval (ISMIR)*, 2005.

[42] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of cnn-based automatic music tagging models," *arXiv:2006.00751*, 2020.

[43] A. Ettinger, "What BERT is not: Lessons from a new suite of psycholinguistic diagnostics for language models," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 34–48, 2020.

[44] G. N. C. Tejada, J. Scholtes, and G. Spanakis, "A study of BERT's processing of negations to determine sentiment," in *Proc. of the Benelux Conference on Artificial Intelligence and Belgian-Dutch Conference on Machine Learning*, 2021.

[45] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 6894–6910.

[46] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 3982–3992.

[47] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar *et al.*, "Universal sentence encoder," *arXiv:1803.11175*, 2018.

MELOFORM: GENERATING MELODY WITH MUSICAL FORM BASED ON EXPERT SYSTEMS AND NEURAL NETWORKS

Peiling Lu[1] **Xu Tan**[*1] **Botao Yu**[2]
Tao Qin[1] **Sheng Zhao**[3] **Tie-Yan Liu**[1]
[1] Microsoft Research Asia, Beijing, China
[2] Nanjing University, Nanjing, China
[3] Microsoft Azure Speech, Beijing, China
{peil,xuta,taoqin,szhao,tyliu}@microsoft.com, btyu@smail.nju.edu.cn
https://github.com/microsoft/muzic

ABSTRACT

Human usually composes music by organizing elements according to the musical form to express music ideas. However, for neural network-based music generation, it is difficult to do so due to the lack of labelled data on musical form. In this paper, we develop MeloForm, a system that generates melody with musical form using expert systems and neural networks. Specifically, 1) we design an expert system to generate a melody by developing musical elements from motifs to phrases then to sections with repetitions and variations according to pre-given musical form; 2) considering the generated melody is lack of musical richness, we design a Transformer based refinement model to improve the melody without changing its musical form. MeloForm enjoys the advantages of precise musical form control by expert systems and musical richness learning via neural models. Both subjective and objective experimental evaluations demonstrate that MeloForm generates melodies with precise musical form control with 97.79% accuracy, and outperforms baseline systems in terms of subjective evaluation score by 0.75, 0.50, 0.86 and 0.89 in structure, thematic, richness and overall quality, without any labelled musical form data. Besides, MeloForm can support various kinds of forms, such as verse and chorus form, rondo form, variational form, sonata form, etc. Music samples generated by MeloForm are available via this link [1], and our code is available via this link [2].

1. INTRODUCTION

Melody is often composed of hierarchical motifs, phrases and sections with repetitions and variations given musical form [1]. This organized structure provided by musical form can help better express music ideas. For example, verse and chorus form is widely used in popular music. The repetition of verse and chorus sections helps emphasize music ideas, while the contrast between verse and chorus can create more emotional intensity. Automatic melody generation with pre-given musical form based on purely data driven technology is difficult due to the lack of labelled data on musical form. Previous work attempt to generate melody with structure information, but still suffers from the following issues: 1) They generate melodies with repetitive patterns implicitly either by learning long-term dependency [2–4] or representing the repetition structure with same harmony, rhythm patterns, etc [5–7]. However, repetitive patterns are far from the exact musical form. 2) They generate melodies with bar-level structure explicitly by learning the relationship between bars [8, 9], but bar-level structure is still not the exact musical form. 3) They generate melodies with repetitive phrases and sections either by detecting phrase labels with rules and algorithms [10] or using human labelling [11]. But it is hard for rules or algorithms to detect precise musical form, and it costs much for hiring people to label this musical form data. Furthermore, all of these work aim to generate melodies with some kind of repetitive patterns, but musical form is constructed to express music ideas by developing the hierarchical structural units (i.e., motifs, phrases and sections) with repetitions and variations. Simply repeating some fragments, or even phrases and sections, without considering the relationships among these hierarchical structures, is superficial.

Although it is difficult to collect labeled musical form data through detection algorithm or human labeling, it is much easier for expert systems to generate melodies with musical form. However, experts systems may suffer from monotonous musicality because of handcraft rules. On the other hand, neural networks are capable of creating melodies with rich expressions by learning the data distribution, but it is hard to precisely control the musical form. Considering the complementary characteristics of these two systems, we come up with a method that can leverage the advantages and make up for the shortcomings.

* Corresponding author: Xu Tan, xuta@microsoft.com
[1] https://ai-muzic.github.io/meloform/
[2] https://github.com/microsoft/muzic


© Peiling Lu, Xu Tan, Botao Yu, Tao Qin, Sheng Zhao, Tie-Yan Liu. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Peiling Lu, Xu Tan, Botao Yu, Tao Qin, Sheng Zhao, Tie-Yan Liu, "MeloForm: Generating Melody with Musical Form based on Expert Systems and Neural Networks", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

In this paper, we develop MeloForm, a system that generates melody with musical form using expert systems and neural networks. The expert system is designed to generate synthetic melodies with precise musical form. It develops the motifs to phrases then to sections, which are arranged by repetitions and variations according to the pre-given musical form. The encoder-attention-decoder Transformer based neural network is introduced to refine melodies generated by expert systems. To improve musical richness without changing musical form, we propose the refinement strategy in phrase level, the conditioning on rhythm and harmony, and the methods for differentiating sections.

MeloForm enjoys the advantages of expert systems and neural models and avoids their limitations as following: 1) Comparing with expert systems, we can generate melodies with better musical richness. 2) Comparing with the models that implicitly learn the repetitive patterns, we can generate melodies explicitly with precise musical form control. 3) Comparing with the models that depend on bar-level structures, we construct higher-level phrases and sections structures. 4) Comparing with the models using detected phrases labels or human labels, we have the labeled musical form data naturally from expert systems with zero cost and precise accuracy.

The main contributions of this work are as follows:

- We develop MeloForm, a system that generates melody with musical form using expert systems and neural networks. This system combines the best of white-box expert system and black-box neural networks for generating melodies with precise musical form and rich melodic expression without any labeled data.
- Experimental results demonstrate that MeloForm achieves precise musical form control with 97.79% accuracy without any labeled data, and outperforms baseline systems by 0.75, 0.50, 0.86 and 0.89 averagely in structure, thematic, richness and overall quality in subjective evaluation.
- MeloForm can generate melodies with various kinds of forms, such as verse and chorus form, rondo form, variational form, sonata form, etc.

2. RELATED WORK

Automatic melody generation evolves from grammar or statistical based generation [12–17] to deep learning empowered generation [2, 3, 6, 8–10, 18–31]. In this section, we introduce existing neural networks and expert systems for melody generation with musical structure.

2.1 Neural Networks for Melody Generation

Generating structured melody has attracted more attention when modeling long music sequence. Previous work address this problem as following: 1) They implicitly learn the long-term dependency or represent repetitive structure with same musical elements to generate melodies with repetitive patterns. Music Transformer [3] introduces a relative attention mechanism to capture long-term dependency. Theme Transformer [4] proposes a novel gated parallel attention module for generation with theme-based conditioning. Another work [2] presents a hierarchical recurrent neural network to model the note-beat-bar structure. Other methods [5–7] condition the model with same musical features (e.g., harmony and rhythm patterns) to represent the repetitive structure. 2) They explicitly model the bar-level structure for generating melodies one bar after another. In [9, 32], the authors leverage the bar related self-similarity matrix to model the relationship between bars for guiding the melody generation. MELONS [8] constructs a bar-level structure graph for generating melodies with clear bar-level structures. 3) They collect labeled musical data by detection algorithms for generating repetitions for melodies. Repetitive patterns from melodies are detected by music analysis algorithms in [33], while the boundaries of repetitive phrases are recognized in [10]. All of these works can help realize repetitive patterns for melody in some degree, but they still cannot model precise musical form.

2.2 Expert Systems for Melody Generation

Back to 18th century, a system called music dice game is developed for randomly generating music from precomposed options [34]. Recently, much work still investigate rule-based algorithm compositions for specific purpose. The author in [35] comes up with a rule-based algorithm to generate melody note sequence, which is constructed for comparison with the machine learning based compositions. dMelodies [36] combines the designed latent factors to create 2-bar melodies for improving data diversity in disentanglement learning. However, none of them consider the rules for generating melodies with musical form. Computoser [37] proposes a hybrid probability/rule based algorithm for music composition, which based on rules about structure, rhythm, repetition, variations, endings, etc. But it did not provide the methods about how to arrange these elements with musical form. And the method for constructing a phrase by multiple different motifs may brings about divergence from music ideas. In [38], repetition of phrases is also considered at a rhythmical level and concerning pitch intervals, but the melody in each phrase is generated one note at a time, which is different from a common composition process by developing a phrase from a motif. Comparing with these systems, the expert system in MeloForm generates melodies by considering the musical form as the hierarchical structure of motifs, phrases and sections with repetitions and variations, which can better express music ideas.

3. METHOD

To combine the advantages of precise musical form control by expert systems and musical richness learning by neural networks, we develop MeloForm that is shown in Figure 2(a), which contains two modules: 1) expert systems for generating synthetic melodies with musical forms; 2)

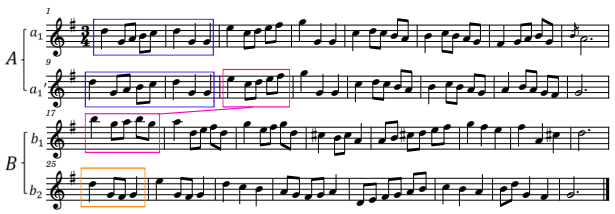


Figure 1. The score of the melody part from "*Minuet in G Major*" by the composer Christian Petzold. This is an example of a melody with musical form as $A(a_1, a_1)B(b_1, b_2)$. We use capital letters (e.g., A, B) to label sections, while using lowercase (e.g., a_1, b_1, b_2) for representing phrases. Different phrases in the same section are labeled with different numbers.

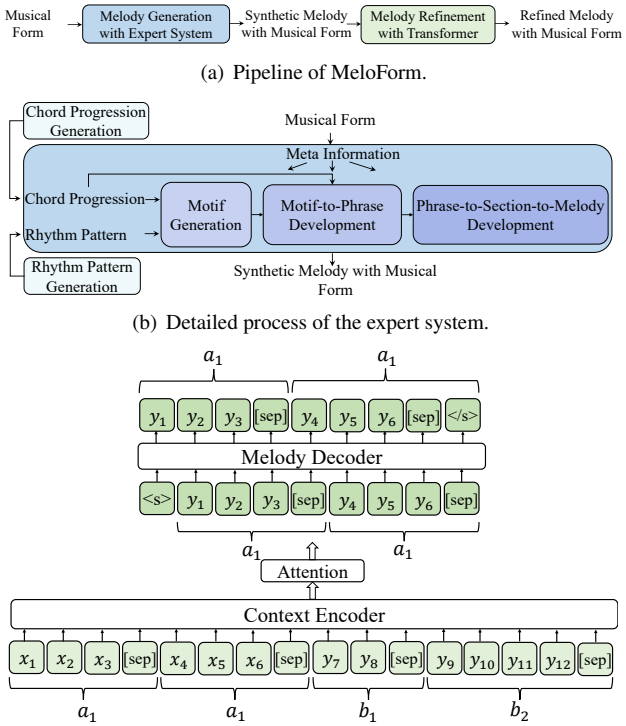
Transformer based neural networks for refining the generated melodies from expert systems.

3.1 Melody Generation with Expert System

The designed expert system with purely handcraft rules is inspired by music theory [1], and is shown in Figure 2(b). Given the musical form with hierarchical structure of sections and phrases, the expert system firstly generates the motifs based on chord progression and rhythm patterns, then develops the generated motifs to phrases. In the phrase-to-section-to-melody development module, we arrange the phrases in sections and sections in melody with repetitions and variations according to the given musical form to generate the synthetic melody with musical form. For example in Figure 1, given the musical form $A(a_1, a_1)B(b_1, b_2)$, a 2-bar motif in the blue box is developed into 8-bar phrase a_1. Section A is formed by placing repeated phrase a_1 sequentially with some variations. The similar process is implemented to form section B, in which different phrases b_1 and b_2 are developed from different motifs. Then section A and section B are placed sequentially for getting the final composition.

3.1.1 Motif Generation

Motif is the smallest structural unit for identifying music theme, so we create an initial melody in a few bars (i.e., usually 1-2 bars) to construct the motif. Before the generation process, we should define the meta information (i.e., scale, pitch range, tempo and meter) for guiding the whole generation procedure. Considering note sequences are composed of pitches and rhythm patterns, we come up with rule-based algorithms for chord progression generation to guide pitch selection and for rhythm pattern generation to create rhythm patterns for the note sequence in motif. Specifically, the initial pitch of notes is selected from tones in corresponding chords. Then we add some embellishing tones [3] to decorate the motif melody. Besides, the interval between adjacent note pitches is constrained to be not greater than seven semitones to ensure pitch consistency. The detailed algorithms of chord progression generation and rhythm pattern generation can be referred in Supplementary Materials Section 1 and 2.



(a) Pipeline of MeloForm.

(b) Detailed process of the expert system.

(c) Architecture of melody refinement neural networks. The input contains four phrases from a melody with musical form as $A(a_1, a_1)B(b_1, b_2)$. In each phrase, x represents the conditioning information (i.e., rhythm, chord and cadence) for each note, while y represents the melody information (i.e., rhythm and pitch) for each note. [sep] indicates the boundaries between phrases, while <s> and </s> indicates the beginning and end of phrases.

Figure 2. Architecture of MeloForm.

3.1.2 Motif-to-Phrase Development

A motif is a unit for identifying the music theme, but it is not a complete music expression. Thus, we need to develop it into a phrase, which is the smallest structural unit for expressing complete music ideas. In order to develop motif into phrase, we introduce three basic categories of development strategies: sequence [4], transformation and ending. These basic development strategies can be combined with each other to create compound development strategies. For example, in Figure 1, the 2-bar motif in the blue box is developed by sequence (i.e., 3-4 bars), transformation (i.e., 5-6 bars) and ending (i.e., 7-8 bars) to form 8-bar phrase a_1. The detailed description of each development strategy can be referred in Supplementary Materials Sections 3.

There may exist multiple different phrases in each section, each of which has its own motif. Too much different motifs in each section may distract listeners from the main ideas. Thus, we present another way to generate a new motif similar to already generated one for a new phrase. Specifically, we can either directly copy the already generated motif or borrow its rhythm patterns to create a new motif, which can help build some relationships between

[3] http://openmusictheory.com/embellishingTones.html

[4] https://en.wikipedia.org/wiki/Sequence_(music)

phrases in the same section. For example, in Figure 1, the rhythm patterns from the motif in the orange box of phrase b_2 is borrowed from the motif in the pink box of phrase b_1 with a few variations.

3.1.3 Phrase-to-Section-to-Melody Development

After generating all phrases, we need to arrange them with repetitions and variations to form sections, which are higher-level structural units. And sections are then ordered sequentially to finish the composition. As shown in Figure 1, given musical form as $A(a_1, a_1)B(b_1, b_2)$, repeated phrases a_1 with variations are placed in order to form section A. Phrase b_1 and b_2 are placed in order to form section B. At last, section A and section B are arranged sequentially to get the final composition.

Directly generating new motifs when building different sections is not always musically expressive, so we introduce another method for establishing not only similar but also contrastive motifs. To establish the similarity, we can borrow one of the random fragment from phrases in other sections. To form a contrast, we can adjust the rhythm patterns and pitch selection in desired section to change the tension. For example, in Figure 1, the rhythm patterns of the motif in the pink box from phrase b_1 is borrowed from that of the third bar from phrase a_1, which is not the exact motif for phrase a_1. And to form a contrast, the pitch is selected from higher range for more tension.

3.2 Melody Refinement with Transformer

Melodies generated by the expert system are guaranteed to have precise musical form, but are lack of musical richness since they are generated by handcraft rules. Thus, in this section, we introduce an encoder-attention-decoder Transformer that takes the synthetic melodies as input and outputs refined melodies with better musical richness. However, the refinement is challenging since it needs to keep the musical form unchanged while improving the musical richness. Due to the lack of controllability of neural networks, directly refining the whole melody without any constraints is easy to lose musical form information. To address this challenge, we describe several design principles in our Transformer model for refinement:

- *Refining melody phrases by phrases.* Since musical phrase is the smallest structural unit for a complete musical expression, it is better to refine the melody phrase by phrase in an iterative process, instead of refining the whole melody in a single time that fails to maintain the musical form from the expert system. However, only refining one phrase at a time is hard to model the repetitive patterns from similar phrases. Therefore, we should refine similar phrases at each iteration step.

- *Refining phrases by conditioning on rhythm and harmony.* Directly generating the refined musical phrases from scratch may lose bottom-level music structure determined by rhythm patterns and harmony. Thus, we should provide explicit condition and control with rhythm and harmony to guide the refine process.

- *Refining phrases by considering their differentiations among different sections.* When composers need to distinguish different sections (e.g., verse and chorus) in melodies, they usually change pitch distribution or rhythms patterns for building up contrasted tension. Accordingly, to maintain such contrast between sections, we need to consider these differentiations in the refinement process.

Based on these design principles, our model architecture is shown in Figure 2(c). We introduce the implementations corresponding to each design principle as follows.

3.2.1 Iterative Phrase Refinement

According to the first design principle, we refine the similar phrases in each iterative step. To train the Transformer model with iterative refinement capability, we first mask similar phrases of the melody in the training data and take the masked melody as the input of the encoder, and generate original phrases corresponding to the masked positions with the decoder, like the masked sequence to sequence (MASS) task in [39]. After model training, the Transformer model is used to refine the similar phrases in synthetic melodies from expert systems in an iterative way. Specifically, if we want to refine the melody with musical form as $A(a_1, a_1)B(b_1, b_2)$, we can refine phrases a_1 for the first iteration and replace the two original phrases with the refined versions, and then refine the phrase b_2 based on the refined melody. This iteration step is finished until all phrases are refined.

3.2.2 Condition with Rhythm and Harmony

The phrases are developed from motif by sequence strategy that results in similar rhythm patterns within some bars. Besides, harmony plays an important role to control pitch distribution in phrase development. Thus, to maintain these music structures in refinement, we condition the model with rhythm patterns, chord progression and cadence to encourage the model to only refine the pitch of the phrase. Specifically, instead of replacing the masked phrases with masked symbols in MASS [39], we replace them with the corresponding rhythm, chord and cadence symbols, shown as x in Figure 2(c).

3.2.3 Differentiation between Sections

We differentiate the phrases in different sections into aspects: 1) Controlling rhythm patterns in the expert system. Rhythm patterns can be adjusted directly in expert system by increasing/decreasing note density, prolong/shrink note length, etc. It is common that larger note density or faster tempo can bring about more intensity. 2) Controlling the pitch distribution in the Transformer model. Although expert systems can control the pitch distribution by selecting pitch from different ranges, it is hard for neural networks to control in the same way. Thus, we insert an "AVG-PITCH" token at the beginning of the condition from each target phrase to represent the average pitch of this phrase, and another "SPAN" token to indicate the pitch span (i.e., the difference between the maximum pitch with minimum

		Structure↑	Thematic↑	Richness↑	Overall↑
Dataset	LMD [40]	3.18 ($\pm$0.64)	3.07 ($\pm$0.58)	3.14 ($\pm$0.40)	3.00 ($\pm$0.57)
	POP909 [41]	4.06 ($\pm$0.49)	3.83 ($\pm$0.54)	4.00 ($\pm$0.61)	4.11 ($\pm$0.46)
Method	Music Transformer [3]	3.00 ($\pm$0.76)	3.21 ($\pm$0.74)	2.11 ($\pm$0.46)	2.32 ($\pm$0.54)
	MELONS [8]	3.18 ($\pm$0.64)	3.07 ($\pm$0.58)	2.64 ($\pm$0.64)	2.89 ($\pm$0.52)
	POP909_lm	2.61 ($\pm$0.62)	2.93 ($\pm$0.60)	2.96 ($\pm$0.45)	2.96 ($\pm$0.49)
	MeloForm	**3.68** ($\pm$**0.35**)	**3.57** ($\pm$**0.36**)	**3.43** ($\pm$**0.43**)	**3.61** ($\pm$**0.34**)

Table 1. Subjective evaluation results, with mean opinion scores and 95% confidence interval for each metric.

pitch). Higher average pitch and more pitch span can build up much more tension.

4. EXPERIMENTAL RESULTS

In this section, we first describe the dataset and system configurations. Next, we show the main results compared with baseline systems. Then we implement some method analysis to further validate the effectiveness of each design. Finally we implement some extensions to demonstrate the scalability of this system. Music samples generated by MeloForm are available via this link [1] . Our code and Supplementary materials are available via this link [2] .

4.1 Experiment Setup

Dataset. We utilize the LMD-matched MIDI dataset [40] in the training stage of the Transformer based neural networks for melody refinement. We select melodies in 4/4 time signature, and normalize them to the tonality of "C major" or "A minor". At last, we obtain 30,218 MIDI samples. This dataset contains 471,058 phrases, in which there exists 100,948 distinct set of similar phrases based on our phrase boundary detection and similarity calculation, of which the detailed algorithm can be found in Supplementary Materials Section 4.

System Configurations. The encoder-attention-decoder Transformer has 4 encoder and decoder layers, both of which contains 4 attention heads. The hidden size is set as 256. We use Adam optimizer [42] with the learning rate is 0.0005. The dropout rate is set as 0.2 during training. There contains maximum of 4,096 tokens in each batch. The dataset is randomly splitted into training/valid/test set with the ratio of 0.8, 0.1, and 0.1, respectively. We use nucleus sampling [43] with the p set as 0.9.

Subjective Evaluation Metrics. For subjective listening, we invite 10 participants to evaluate 30 samples. There are five randomly selected samples from MeloForm and baselines described in Section 4.2. For each listener, they are randomly ordered to avoid perceptual bias and habituation effect. The rating is based on five-point scale. We define the following four metrics: 1) Structure: Does this melody have complete structure with repetitions and variations? 2) Thematic: Can you figure out the musical theme? 3) Richness: Is the melody similarly melodious as human compositions? 4) Overall: Is the overall quality of the melody the same as human compositions?

4.2 Main Results

We compare MeloForm with the following baselines: 1) Music Transformer [3], an end-to-end neural network solution which introduces the relative attention to learn long-term repetitive patterns; 2) MELONS [8], a Linear Transformer and structure graph based framework for generating melody with bar-level structure; 3) POP909_lm, a language model implemented by ourself for leveraging the phrase labels in POP909 [41] to generate melodies given musical form, whose details can be found in Supplementary section 6. These baselines share the same system configurations with MeloForm for fair comparison.

Table 1 shows the subjective evaluation results in the mean opinion scores (MOS) and 95% confidence interval for these four metrics over all experimental groups. Comparing with baseline systems, MeloForm outperforms all of them in thematicness, structureness, richness and overall quality. We calculate the controlling accuracy of phrase labels for POP909_lm to verify if same phrase labels results in similar melodies. The high accuracy of 100% demonstrates it is capable of generating corresponding repetitive melodies for same phrase labels. It is worth noticing that although POP909_lm can realize the repetitions of phrases, it still fails to provide good listening experience comparing with MeloForm. This demonstrate simply repeating phrases is still deficient. Besides, the less 95% confidence interval of MeloForm provides more confidence for its better performance. We also compare MeloForm with human composed data from the LMD-matched MIDI dataset and POP909. MeloForm shows better performance compared with LMD-matched MIDI dataset, since the samples from LMD-matched MIDI dataset are collected from various kinds of publicly-available sources on the internet, which is hard to guarantee the production quality. In contrast, POP909 is constructed by hiring professional musicians for creating the samples and reviewing the whole generation process, which ensures the quality to be the same as realistic compositions. The comparing results with POP909 reveal that there still exist gaps between melodies generated by MeloForm with the human composed ones.

4.3 Method Analysis

Controllability Study. We calculate the controlling accuracy of musical form by calculating the similarity score between generated melodies from same phrases. The accu-

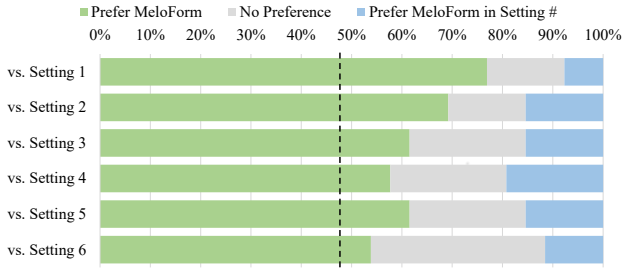


Figure 3. Preference distribution for ablation study, which compares the melodies generated by MeloForm with that from modified system (i.e., MeloForm in Setting #).

racy of 97.79% indicates MeloForm can generate melodies with precise musical form control. We also calculate the accuracy to verify if the pitch distribution is well controlled by average pitch and pitch span. The accuracy of 92.5% in average pitch controlling and 99.71% in pitch span controlling demonstrate its controllability to differentiate sections in pitch dimension.

Ablation Study. We conduct preference tests to verify the contribution of the components in MeloForm with the following settings: 1) Setting 1: w/o development strategies from expert systems. Notes are randomly arranged in expert systems without any development strategies. 2) Setting 2: w/o expert systems. Melodies are first generated for each phrase by neural networks, and then got copied directly with the given musical form for repetition in the same phrase. 3) Setting 3: w/o neural networks. We remove melody refinement neural networks. 4) Setting 4: w/o fine-grained rhythm pattern condition. The condition of rhythm patterns changed from fine-grained level to coarse-grained level. 5) Setting 5: w/o refinement strategy. We directly copy the generate melodies from each phrase to the following same phrases. 6) Setting 6: w/o section differentiation. The "AVGPITCH" and "SPAN" tokens are removed from the beginning of each phrase. The participants are required to compare the samples from MeloForm and these settings and give their preference score.

The preference distribution for the above settings is shown in Figure 3. We derived some observations in the following: 1) More preferences over setting 1 and 2 demonstrate the effectiveness of the whole expert system and the development strategies for building up phrases. 2) More preferences over setting 3, 4, 5, and 6 validate the efficacy of the whole neural refinement model, the rhythm pattern condition, the iterative refinement strategy, and the section differentiation method.

4.4 Extensions to More Musical Forms

In this section, we illustrate the principles to generate melodies with four different musical forms (i.e., Verse and Chorus, Rondo, Variational and Sonata form) based on MeloForm. The examples from these extensions are shown in the demo page.

Verse and Chorus Form is widely used in popular music. It contains two contrasting sections: verse section (i.e., A) and chorus section (i.e., B), which can be arranged in various kinds of ways, such as $AABAAB$, ABA, or $AABB$. Users can follow the method in Section 3.1 to build the verse and chorus sections, and leverage the section differentiation in Section 3.2.3 to change the tension. For example, Chorus tends to have more intensive emotional expression. To achieve this, we can control the average pitch to higher level or increase the pitch span for this section.

Rondo Form is a musical form where the refrained section alternates with contrasting sections, such as $ABACADA$. Melody of Rondo Form can be generated by leveraging the method in Section 3.1 to construct the refrained section and contrasting sections, and arrange them sequentially based on the given Rondo Form.

Variational Form is a musical form where the main section is followed by its variations. We represent Variational Form as $AA'A''$ to describe the main section and its variations. To address the challenge of deriving variations from the main section, we can treat the main section as a motif, and use the development strategies in 3.1.2 to develop this longer motif into another higher-level section. Another way is to generate melodies with same sections like AAA by expert systems, and predict the melodies from each section in different iteration steps. There are many other ways for creating the Variational Form, so users are encouraged to provide their own creative design.

Sonata Form is generally consisted of three sections: an exposition, a development, and a recapitulation, which can be represented as ABA' in our system. For the exposition, we generate two phrases with contrasting motifs, where the second phrase is a transposition of the first one by controlling the tonality token when refining the second phrase. The development section is a variation of the exposition, which can be generated by leveraging the methods for constructing variational form. The recapitulation is a repetition of the exposition, in which the second phrase should go back to the tonic key by controlling the tonality token in condition from neural networks.

5. CONCLUSIONS

In this paper, we propose MeloForm, a system to generate melody with musical form based on expert systems and neural networks. It combines the advantages from expert systems to precisely control musical form and neural networks to refine melody for better musical richness without changing musical form. Experimental results demonstrate MeloForm achieves 97.79% accuracy in musical form control, and outperforms baseline systems in structure, thematic,richness and overall quality in terms of subjective evaluation. We will release the dataset of the synthetic melodies generated by our expert system and the refined version from our Transformer model to facilitate the future research on music form modeling. Furthermore, we will explore the generation of intro, outro and bridge, and investigate the arrangement generation with musical form to accomplish a complete music composition.

6. REFERENCES

[1] W. E. Caplin and W. E. Caplin, *Analyzing classical form: an approach for the classroom*. Oxford University Press, 2013.

[2] J. Wu, C. Hu, Y. Wang, X. Hu, and J. Zhu, "A hierarchical recurrent neural network for symbolic melody generation," *IEEE transactions on cybernetics*, vol. 50, no. 6, pp. 2749–2757, 2019.

[3] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[4] Y.-J. Shih, S.-L. Wu, F. Zalkow, M. Muller, and Y.-H. Yang, "Theme transformer: Symbolic music generation with theme-conditioned transformer," *IEEE Transactions on Multimedia*, 2022.

[5] K. Chen, W. Zhang, S. Dubnov, G. Xia, and W. Li, "The effect of explicit structure encoding of deep neural networks for symbolic music generation," in *2019 International Workshop on Multilayer Music Representation and Processing (MMRP)*. IEEE, 2019, pp. 77–84.

[6] Z. Ju, P. Lu, X. Tan, R. Wang, C. Zhang, S. Wu, K. Zhang, X. Li, T. Qin, and T.-Y. Liu, "Telemelody: Lyric-to-melody generation with a template-based two-stage method," *arXiv preprint arXiv:2109.09617*, 2021.

[7] X. Zhang, J. Zhang, Y. Qiu, L. Wang, and J. Zhou, "Structure-enhanced pop music generation via harmony-aware learning," *arXiv preprint arXiv:2109.06441*, 2021.

[8] Y. Zou, P. Zou, Y. Zhao, K. Zhang, R. Zhang, and X. Wang, "Melons: generating melody with long-term structure using transformers and structure graph," *arXiv preprint arXiv:2110.05020*, 2021.

[9] J. Wu, X. Liu, X. Hu, and J. Zhu, "Popmnet: Generating structured pop music melodies using neural networks," *Artificial Intelligence*, vol. 286, p. 103303, 2020.

[10] S. Dai, Z. Jin, C. Gomes, and R. B. Dannenberg, "Controllable deep melody generation via hierarchical music structure representation," *arXiv preprint arXiv:2109.00663*, 2021.

[11] J. Zhao and G. Xia, "Accomontage: Accompaniment arrangement via phrase selection and style transfer," *arXiv preprint arXiv:2108.11213*, 2021.

[12] H. Young, "A categorial grammar for music and its use in automatic melody generation," in *Proceedings of the 5th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design*, 2017, pp. 1–9.

[13] D. Quick and P. Hudak, "Grammar-based automated music composition in haskell," in *Proceedings of the first ACM SIGPLAN workshop on Functional art, music, modeling & design*, 2013, pp. 59–70.

[14] M. Kikuchi and Y. Osana, "Automatic melody generation considering chord progression by genetic algorithm," in *2014 Sixth World Congress on Nature and Biologically Inspired Computing (NaBIC 2014)*. IEEE, 2014, pp. 190–195.

[15] A. Garay Acevedo, "Fugue composition with counterpoint melody generation using genetic algorithms," in *International symposium on computer music modeling and retrieval*. Springer, 2004, pp. 96–106.

[16] K. Wakui, Y. Hatori, and Y. Osana, "Automatic melody generation considering chord progression using genetic algorithm," *IEICE Proceedings Series*, vol. 48, no. B2L-E-7, 2016.

[17] M. Takano and Y. Osana, "Automatic melody generation considering user's evaluation using interactive genetic algorithm," in *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 2014, pp. 785–797.

[18] Z. Guo, M. Dimos, and H. Dorien, "Hierarchical recurrent neural networks for conditional melody generation with long-term structure," *arXiv preprint arXiv:2102.09794*, 2021.

[19] Y. Yu, A. Srivastava, and S. Canales, "Conditional lstm-gan for melody generation from lyrics," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 1, pp. 1–20, 2021.

[20] A. Mishra, K. Tripathi, L. Gupta, and K. P. Singh, "Long short-term memory recurrent neural network architectures for melody generation," in *Soft Computing for Problem Solving*. Springer, 2019, pp. 41–55.

[21] Y. Yu, F. Harscoët, S. Canales, G. Reddy M, S. Tang, and J. Jiang, "Lyrics-conditioned neural melody generation," in *International Conference on Multimedia Modeling*. Springer, 2020, pp. 709–714.

[22] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," *arXiv preprint arXiv:1703.10847*, 2017.

[23] S. Li, S. Jang, and Y. Sung, "Automatic melody composition using enhanced gan," *Mathematics*, vol. 7, no. 10, p. 883, 2019.

[24] Z. Sheng, K. Song, X. Tan, Y. Ren, W. Ye, S. Zhang, and T. Qin, "Songmass: Automatic song writing with pre-training and alignment constraint," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 15, 2021, pp. 13 798–13 805.

[25] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[26] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165.

[27] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs," *arXiv preprint arXiv:2101.02402*, 2021.

[28] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[29] X. Wu, C. Wang, and Q. Lei, "Transformer-xl based music generation with multiple sequences of time-valued notes," *arXiv preprint arXiv:2007.07244*, 2020.

[30] C. Zhang, L. Chang, S. Wu, X. Tan, T. Qin, T.-Y. Liu, and K. Zhang, "Relyme: Improving lyric-to-melody generation by incorporating lyric-melody relationships," *arXiv preprint arXiv:2207.05688*, 2022.

[31] A. Lv, X. Tan, T. Qin, T.-Y. Liu, and R. Yan, "Re-creation of creations: A new paradigm for lyric-to-melody generation," *arXiv e-prints*, pp. arXiv–2208, 2022.

[32] H. Jhamtani and T. Berg-Kirkpatrick, "Modeling self-repetition in music generation using generative adversarial networks," in *Machine Learning for Music Discovery Workshop, ICML*, 2019.

[33] D. Herremans and E. Chew, "Morpheus: automatic music generation with recurrent pattern constraints and tension profiles," in *Proceedings of IEEE TENCON,-2016 IEEE Region 10 Conference*. IEEE, 2016, pp. 282–285.

[34] S. A. Hedges, "Dice music in the eighteenth century," *Music & Letters*, vol. 59, no. 2, pp. 180–187, 1978.

[35] P. Wiriyachaiporn, K. Chanasit, A. Suchato, P. Punyabukkana, and E. Chuangsuwanich, "Algorithmic music composition comparison," in *2018 15th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. IEEE, 2018, pp. 1–6.

[36] A. Pati, S. Gururani, and A. Lerch, "dmelodies: A music dataset for disentanglement learning," *arXiv preprint arXiv:2007.15067*, 2020.

[37] B. Bozhanov, "Computoser-rule-based, probability-driven algorithmic music composition," *arXiv preprint arXiv:1412.3079*, 2014.

[38] A. Elowsson and A. Friberg, "Algorithmic composition of popular music," in *The 12th international conference on music perception and cognition and the 8th triennial conference of the European society for the cognitive sciences of music*, 2012, pp. 276–285.

[39] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu, "Mass: Masked sequence to sequence pre-training for language generation," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5926–5936.

[40] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

[41] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," *arXiv preprint arXiv:2008.07142*, 2020.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[43] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2019.

TOWARDS ROBUST MUSIC SOURCE SEPARATION ON LOUD COMMERCIAL MUSIC

Chang-Bin Jeon
Department of Intelligence and Information
Music and Audio Research Group (MARG)
Seoul National University
vinyne@snu.ac.kr

Kyogu Lee
Department of Intelligence and Information
Music and Audio Research Group (MARG)
Seoul National University
kglee@snu.ac.kr

ABSTRACT

Nowadays, commercial music has extreme loudness and heavily compressed dynamic range compared to the past. Yet, in music source separation, these characteristics have not been thoroughly considered, resulting in the domain mismatch between the laboratory and the real world. In this paper, we confirmed that this domain mismatch negatively affect the performance of the music source separation networks. To this end, we first created the out-of-domain evaluation datasets, *musdb-L* and *XL*, by mimicking the music mastering process. Then, we quantitatively verify that the performance of the state-of-the-art algorithms significantly deteriorated in our datasets. Lastly, we proposed *LimitAug* data augmentation method to reduce the domain mismatch, which utilizes an online limiter during the training data sampling process. We confirmed that it not only alleviates the performance degradation on our out-of-domain datasets, but also results in higher performance on in-domain data.

1. INTRODUCTION

Recent commercial music has extreme loudness compared to the past [1, 2]. Since many artists and producers want their music to be perceptually louder, it has become so trendy that it rather harms the quality of music [3] and even there exists the expression 'loudness war' [4].

A dynamic range compressor [5] is used to increase the loudness of music while keeping the digital level under 0 decibels relative to full scale (dBFS), which is the maximum possible level in the digital domain. It is a time-varying non-linear processor that adjusts the level of the signal when the level exceeds the threshold. Especially, a limiter refers to a dynamic range compressor that strongly compresses the signal with a high ratio parameter above 1:10 and increases the gain of the signal by the headroom obtained through compression.

In the last stage of music production, so-called mastering, engineers are often asked to increase the overall loudness of music. A limiter is a core tool for achieving it; it is used to make small parts of music to be louder and loud parts of music fit under 0 dBFS [7, 8]. In this process, original signals are distorted and become louder. The example of a limiter usage is depicted in Figure 1.

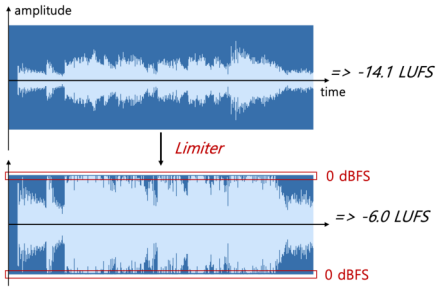


Figure 1. The short example of a limiter applied music source in our *musdb-XL* dataset. Recent commercial music has this loud volume and distorted signal characteristics.

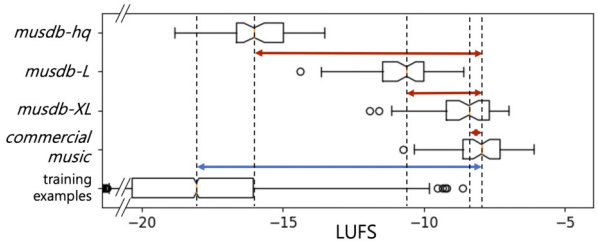


Figure 2. The boxplot representing the loudness distributions of different evaluation datasets, recent commercial music, and training examples generated from the official implementation of *Open-unmix* [6].

However, these characteristics have not yet been thoroughly considered in the music source separation. Although many data augmentation techniques have been proposed, such as random gain scaling and mixing of different instrumental stem sources [9, 10], the training examples have small overall loudness that is far from real-world commercial music [1], as can be seen in the blue line of Figure 2. This causes a huge domain shift between train and real-world domains. The term LUFS in Figure 2 is an abbreviate for loudness unit relative to full scale, which is a

© C.-B. Jeon, and K. Lee. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** C.-B. Jeon, and K. Lee, "Towards robust music source separation on loud commercial music", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] Based on the investigation of 50 songs in *'Pop Life playlist'* created by Tidal, May. 2022.

measure of music loudness [11, 12].

Furthermore, since the standard benchmark datasets for music source separation, *musdb* [13] and *musdb-hq* [14], do not reflect the characteristics of loud commercial music as shown in the red lines of Figure 2, they are unable to show performance degradation comes from the domain shift. Therefore, we conjecture that even good-performing networks based on *musdb* test subset may exhibit worse performance on real world. This problem is no exception to other datasets [15–17] for music source separation, since they do not consider the music mastering process.

Based on this question, we investigated how heavily compressed music and its loudness affect the degradation of music source separation networks. To this end, we manually built new evaluation datasets by applying a limiter to the *musdb-hq* test subset [14] to get loud and compressed music imitating the modern music mastering process. One is the *musdb-L* dataset, with increased loudness to an appropriate degree for pleasant sound following [3]. The other is the *musdb-XL* dataset, which raised the loudness to an excessive degree, as often found in recent popular music. Using our datasets, we confirmed that more dynamic range compression in test domains results in more performance degradation of the networks.

Moreover, we conducted various experiments on the training data creating methods to reduce the domain shift between train and real world data from the perspective of music loudness and heavy compression. We trained music source separation networks using the training examples constructed by *(1)* linear gain increasing, *(2)* the proposed *LimitAug* method which utilizes an online limiter during the sampling process of training examples, *(3)* simple input loudness normalization, and *(4)* the loudness normalization after applying the *LimitAug*. We confirmed that all of the methods not only showed robust performance on out-of-domain *musdb-L* and *musdb-XL* data, but also showed better performance on in-domain *musdb-hq* test dataset than the baseline.

To summarize, our contributions are three-fold.

- We built *musdb-L* and *XL* datasets [2], which have comparable overall loudness to commercial music, for evaluation of music source separation algorithms.
- Using *musdb-L* and *XL*, we quantitatively confirmed that the domain shift causes performance degradation of the state-of-the-art networks that were trained without considering loud and compressed music characteristics.
- We proposed *LimitAug* [3] data augmentation method and experimentally confirmed that it is beneficial to alleviate the domain shift between train data and the *musdb-L* or *XL*.

[2] https://github.com/jeonchangbin49/musdb-XL
[3] https://github.com/jeonchangbin49/LimitAug

dataset	Loudness [LUFS]			
	min	max	median	mean (std)
musdb-hq	-18.84	-13.52	-16.02	-15.92 (1.27)
musdb-L	-14.39	-8.61	-10.61	-10.89 (1.19)
musdb-XL	-11.93	-6.99	-8.41	-8.61 (1.17)
commercial	**-10.75**	**-6.10**	**-7.96**	**-8.05 (1.06)**

Table 1. Loudness statistics of *musdb-hq*, *musdb-L*, *musdb-XL* datasets, and the investigated commercial music.

2. MUSDB-L AND MUSDB-XL

musdb [13] and *musdb-hq* [14] have been the standard benchmark datasets on music source separation since their presentation in Signal Separation and Evaluation Challenge (SiSEC) 2018 [18]. They consist of various genres of 150 professionally produced songs — 100 songs for train, 50 songs for test — from folk, indie to electronic, rock genres. Each song has its constituting 4 stems, *vocals*, *drums*, *bass*, and the remaining as *other*. The *musdb-hq* dataset is the uncompressed version of *musdb* with the extended frequency bandwidth from 16kHz to 22.05kHz, which is a full bandwidth of 44.1kHz sample rate. We used *musdb-hq* for high-quality dataset construction.

Since the test subset of *musdb-hq* has small overall loudness compared to commercial mastering-finished music, we conjectured that it could not fully reflect the performance degradation related to heavy dynamic range compression, which prevails in recent commercial music. In addition, to the best of our knowledge, there is no dataset for music source separation that reflects these characteristics or considers the music mastering process. Therefore, we built *musdb-L* and *musdb-XL* datasets, which have loud and compressed characteristics similar to commercial music. *L* and *XL* each stands for *Loud* and *eXtremely Loud*.

2.1 Dataset construction

To reflect the characteristics of commercial music, we created datasets by imitating the music mastering process. Both datasets were made by manually applying the commercial digital limiter, iZotope Ozone 9 Maximizer [4], to the *musdb-hq* test subset. For each *musdb-L* and *musdb-XL*, we controlled the threshold parameter of a limiter so that compression is applied about 3-4 dB and 6-7 dB in loud parts of *mixture* tracks of the *musdb-hq*. As shown in Table 1, *musdb-L* and *musdb-XL* are about 5 and 7 LUFS louder than the original *musdb-hq* dataset on average, respectively. *musdb-L* has insufficient loudness compared to loud commercial music but less distorted sources. *musdb-XL* has comparable loudness to commercial music and more distorted sound than *musdb-L*.

To make the ground truth stem tracks of each mixture, we calculated the sample-wise ratio between the limiter applied mixture and the original mixture, then multiply it to the individual stems to make the ground truth stems for *musdb-L* and *XL*.

[4] https://www.izotope.com/en/products/ozone.html

3. METHODS

In general, a limiter is used as the last signal processor in music mastering of commercial music [7, 8]. Since it tends to be used excessively in modern commercial music [1, 2, 4], it should be considered in music source separation for the development of robust application.

We assumed that the key differences between the real-world mastering-finished music, i.e. a limiter applied music, and the standard training examples for music source separation are *i)* the overall amplitude scale, and *ii)* the signal distortion caused by a limiter.

Here we introduce two ways to avoid each domain mismatch problem.

3.1 Input loudness normalization

First of all, the simple loudness normalization, which is a linear gain adjustment of network inputs to the pre-defined reference level, is the easiest technique to avoid the domain shift caused by *i)* overall amplitude scale mismatch. This can be categorized into two, *(i)* input loudness normalization during both the training and evaluation stages of networks, and *(ii)* normalization only at the evaluation stage.

The method *(i)* is already used in various studies with different ways. For example, the network such as *Demucs v3* [19], uses input standardization in time domain based on the mean and standard deviation of the training data. In *Open-unmix* [6], the network implicitly normalizes the input by utilizing trainable input scaling parameters that works in time-frequency domain. However, we assumed that explicitly adjusting the gain of the inputs based on the waveform, thereby minimizing the overall amplitude scale mismatch between train and test domain, can greatly reduce the performance degradation comes from the domain shift.

The method *(ii)*, normalization only at the evaluation stage, can be considered as a readily applicable workaround for the models that were trained without considering music loudness. This method is a simple idea to reduce the overall amplitude scale difference between train and real-world domain. Since the music source separation networks are non-linear systems, we hypothesized that linear scale difference of the inputs might affect the quality of final outputs. That is, input normalization only at the evaluation stage can be a simple, yet effective trick for models that were already trained without considering the domain mismatch.

Though these methods can mitigate the domain mismatch related to *i)* amplitude scale, there needs to be another strategies that can reflect *ii)* the signal distortion caused by a limiter.

3.2 *LimitAug*

The best way to consider the characteristics of the limiter applied source is simply using the limiter during the network training. Therefore, we propose the *LimitAug* data augmentation method, which utilizes an online limiter during the training examples construction process, to reduce the domain shift between the training examples and real-world commercial music. The *LimitAug* can be considered as creating train examples that forcibly reflect the distortion that comes from a limiter, which cannot be reflected by the simple input loudness normalization technique.

```
# mix, tgt : mixture and tgt sources.
# gain(src, tgt_lufs) : calculate lufs of source,
#  then adjust its gain targeting given lufs,
#  return output and adjusted gain
# gain_adj(src, adj_gain) : gain adjustment
#  of source with given adj_gain.
tgt_lufs = random_sampled_tgt_lufs
mix_loud, _ = gain(mix, tgt_lufs)
mix_loud_limited = limiter(mix_loud)
ratio = mix_loud_limited / mix
tgt_loud = tgt * ratio
if input_loud_norm:
  mix_loud_limited, adj_gain = gain(src, tgt_lufs)
  tgt_loud = gain_adj(tgt_loud, adj_gain)
estimates = network(mix_loud_limited)
loss = objective(estimates, tgt_loud)
```

Figure 3. `numpy`-like pseudocode of the proposed *LimitAug* method.

The pseudocode for *LimitAug* is shown in Figure 3. First, calculate the LUFS of the mixture created by the data sampling process. Second, adjust the gain of the input mixture source targeting the randomly sampled LUFS value. Then, it is followed by the online limiter to fit the waveform under 0 dBFS. Lastly, calculate the sample-wise (A sample refers to an each waveform value) ratio between the limiter applied mixture source and the original mixture source, then multiply the ratio to the original target source to get the ground truth target signal of the limiter applied mixture source.

When adjusting the gain and applying the limiter to the input mixture, for example, if the input mixture had -15 LUFS and the randomly sampled target LUFS was -10, note that the gain-scaled mixture by +5 dB does not have exact -10 LUFS due to the compression of a limiter and the nature of LUFS calculation, which considers frequency weighting [11, 12].

The proposed *LimitAug* can be used with other data augmentation methods; in our study, random gain scaling, channel swap and mixing of different instrumental stem sources [9, 10] were used. Also, additional loudness normalization can be applied after the *LimitAug* as depicted in conditional statement of Figure 3, so that the overall amplitude scale mismatch between training and evaluation stages be minimized.

4. EXPERIMENTS

Here we briefly summarize our following experiments.

In Section 5.1, we quantitatively evaluated various music source separation networks that were trained without considering the loudness and heavy dynamic range compression, and confirmed the performance degradation on *musdb-L* and *musdb-XL*, compared to the origianl *musdb-hq* evaluation dataset.

network	extra train data	test data	SDR median (mean) [dB]				
			vocals	*bass*	*drums*	*other*	*avg*
Open-unmix [6]	-	*hq*	6.16 (2.54)	5.03 (2.67)	6.00 (5.46)	4.22 (3.46)	5.35 (3.53)
		L	6.33 (1.63)	4.81 (2.71)	5.82 (5.38)	4.11 (3.42)	5.27 (3.28)
		XL	5.98 (0.89)	4.76 (2.59)	4.97 (4.89)	4.04 (3.29)	4.94 (2.92)
TFC-TDF -U-net [20]	-	*hq*	7.18 (4.26)	5.59 (3.35)	5.76 (5.30)	4.04 (3.18)	5.64 (4.02)
		L	7.03 (3.65)	5.41 (3.08)	5.52 (5.09)	3.67 (3.00)	5.41 (3.71)
		XL	6.95 (3.14)	5.48 (2.90)	5.11 (4.68)	3.55 (2.82)	5.27 (3.39)
Demucs v3-A [19]	-	*hq*	**8.11 (5.22)**	**9.34** (6.21)	**8.57 (8.01)**	**5.51 (5.03)**	**7.88 (6.12)**
		L	7.54 (5.15)	9.32 **(6.22)**	8.26 (7.65)	5.51 (5.01)	7.66 (6.01)
		XL	7.30 (4.86)	9.19 (6.14)	7.62 (6.78)	5.37 (4.97)	7.37 (5.69)
Open-unmix [6]	✓	*hq*	7.02 (4.93)	5.91 (4.06)	7.18 (6.91)	4.94 (4.76)	6.26 (5.17)
		L	6.83 (5.12)	6.23 (4.09)	7.07 (6.92)	4.94 (4.78)	6.27 (5.23)
		XL	6.70 (4.77)	6.16 (3.87)	6.80 (6.48)	4.89 (4.61)	6.14 (4.93)
Spleeter [21]	25000+	*hq*	6.51 (4.42)	4.77 (3.57)	6.00 (6.09)	4.22 (4.12)	5.38 (4.55)
		L	6.18 (3.90)	4.73 (3.34)	5.67 (5.94)	4.37 (4.03)	5.24 (4.30)
		XL	6.03 (3.38)	4.80 (3.13)	5.55 (5.52)	4.24 (3.91)	5.15 (3.98)
Demucs v3-B [19]	200+ including *musdb-hq* test set	*hq*	**9.24 (7.05)**	**11.65 (9.58)**	**11.73 (11.34)**	**7.83 (8.03)**	**10.11 (9.00)**
		L	9.05 (6.91)	11.61 (9.55)	11.05 (10.27)	7.83 (7.91)	9.88 (8.66)
		XL	8.76 (6.41)	11.56 (9.29)	9.22 (8.78)	7.52 (7.51)	9.26 (8.00)

Table 2. Performance of various music source separation networks trained with *musdb-hq* [14] on the test using *musdb-hq*, *musdb-L* and *musdb-XL*.

In Section 5.2, we simply normalized the network inputs in the evaluation stage, to observe the performance degradation caused by the signal distortion caused by a limiter, not an overall amplitude mismatch between the training and the evaluation data. Then, we carefully analyzed the results on *Demucs v3* [19], which took the 1st place in the 2021 Sony Music Demixing (MDX) Challenge leaderboard A and the 2nd place in the leaderboard B [22]. The leaderboard A was the competition that only allowed training on the only *musdb-hq* train subset, and the leaderboard B allowed the extra training data.

In Section 5.3, we conducted a comparative study on various training data construction strategies. Unfortunately, we were unable to train the current state-of-the-art *Demucs v3*, due to the constraint of our GPU experimental environments. Considering the training efficiency and reasonable performance, we used *TFC-TDF-U-Net* [20] — a backbone architecture of KUIELab-MDX-Net [23] with slight modifications, which took the 2nd place in the 2021 MDX Challenge leaderboard A — in this experiment.

4.1 Training

In Section 5.1 and Table 2, we used official pre-trained weights of each model except *TFC-TDF-U-Net* since there were no official weights for the 4 stems of *musdb-hq*. For fair comparison in Section 5.3, we trained *TFC-TDF-U-Net* for 300 epochs with early stopping, based on official training framework of *Open-unmix* [6] and official network implementation of *TFC-TDF-U-net*. We used default network hyper-parameters introduced in its webpage [5] .

For the *LimitAug*, we used the limiter implemented in `pedalboard` [24]. The threshold parameter was set to 0 dBFS, and the release parameter was randomly sampled from uniform distribution of (30, 200) millisecond in data sampling process. Also, we used `pyloudnorm` [25] for loudness calculation.

[5] https://github.com/ws-choi/ISMIR2020_U_Nets_SVS

4.2 Evaluation

In all of the evaluations, Signal-to-Distortion Ratio (SDR) [26] was calculated using `museval` python library [18]. Also, in following experimental results, both *median* and *mean* SDR scores were presented for the detailed comparison. Note that we only used the *musdb-hq* train subset for training the networks. *musdb-L* and *XL* are only for evaluation.

In the original *musdb-hq* test subset, as stated in the official webpage [6] , 'PR - Oh No' track's mixture signal is panned to the right channel, which causes the inconsistency between linear summation of stems and mixture source. Since this causes the limiter to be operated in unmusical way while the construction of *musdb-L* and *XL*, which also can be hardly found in popular music, we used the linear summation of stems as a mixture only for this track. This may results in the slight difference of SDR scores between the Table 2 and the official scores of each network.

5. RESULTS

5.1 Performance degradation on *musdb-L* and *XL*

Here we quantitatively evaluated the performance of state-of-the-art networks on *musdb-hq* [14], *L* and *XL* datasets and confirmed that the domain shift in perspective of music loudness and compression negatively affect the performance, indeed. As shown in the Table 2, all networks showed significant performance degradation on the evaluations with *musdb-L* and *musdb-XL* datasets. The amount of decrease on *Demucs v3* [19] was slightly larger than the others. Overall, we concluded that every networks are somewhat overfitted to the *musdb-hq* data, making the networks vulnerable to loud and heavily compressed music. Therefore, it is highly required to consider these characteristics for the robust music source separation. Note that

[6] https://sigsep.github.io/datasets/musdb

network	extra train data	test data	SDR median (mean) [dB]				
			vocals	*bass*	*drums*	*other*	*avg*
Demucs v3-A [19]	-	*hq*	**8.11** (5.22)	**9.34 (6.21)**	**8.57 (8.01)**	5.51 (**5.03**)	**7.88 (6.12)**
		L	8.05 (**5.23**)	9.25 (6.20)	8.47 (7.92)	**5.53** (5.02)	7.82 (6.09)
		XL	7.93 (5.03)	9.27 (5.92)	7.74 (7.44)	5.55 (4.91)	7.62 (5.82)
Demucs v3-B [19]	200+ including *musdb-hq* test set	*hq*	**9.24 (7.05)**	**11.65 (9.58)**	**11.73 (11.34)**	**7.83 (8.03)**	**10.11 (9.00)**
		L	9.19 (7.04)	11.64 (9.55)	11.68 (11.21)	7.82 (8.02)	10.08 (8.95)
		XL	9.13 (6.90)	11.56 (9.33)	11.32 (10.75)	7.74 (7.95)	9.94 (8.73)

Table 3. Performance of *Demucs v3* [19] trained with *musdb-hq* [14]. On the test using *musdb-L* and *musdb-XL*, each of the input sources were loudness normalized targeting the LUFS of its original *musdb-hq* sources.

network	extra train data	SDR median [dB]		
		hq	*L*	*XL*
Open-unmix [6]	-	5.35	5.32	5.25
TFC-TDF-U-Net [20]	-	5.64	5.62	5.51
Demucs v3-A [19]	-	**7.88**	**7.82**	**7.62**
Open-unmix [6]	✓	6.26	6.25	6.18
Spleeter [21]	✓	5.38	5.33	5.21
Demucs v3-B [19]	✓	**10.11**	**10.08**	**9.94**

Table 4. Performance of networks with loudness normalized input at the evaluation stage. Each score represents the average score across the 4 stems. Note that the networks were trained without considering music loudness or dynamic range compression explicitly.

we stated the scores on different test datasets in each block to emphasize the performance degradation between the domains.

5.1.1 Extra training data

Though the extra training data was of help, it did not work as the fundamental solution to the domain shift. *Open-unmix* [6] with extra training data showed more robust performance than that of the network without extra data, but performance degradation on *Demucs v3-B* was more significant than that of *Demucs v3-A*. Since *Demucs v3-B* was trained with extra training data including *musdb-hq* test subset, it is reasonable to guess that the network is highly overfitted to the *musdb-hq* data. Nevertheless, the amount of performance decrease especially on *drums* comparing the scores between *musdb-hq* and *XL*, 2.5 dB, is somewhat high. Considering *Demucs v3* uses an input standardization technique based on mean and standard deviation of waveform values both at training and evaluation stages, this implies that there needs to be another solution for the robust music source separation.

5.1.2 Performance degradation on drums and vocals

It is noteworthy that the degradation on *drums* and *vocals* were more significant than the others. Due to the percussive nature of *drums*, in general, they have the biggest momentary energy in music. Also, since *vocals* are important ingredients in modern commercial music, they usually consist of not only a single singing voice but also doubling, harmony and chorus. Therefore, we assumed *drums* and *vocals* are most affected by the limiter, which is activated when loud input sources that are above the threshold are given. To quantitatively confirm the signal distortion by the limiter, we calculated Scale-invariant Signal-to-Distortion Ratio (SI-SDR) [27] between *musdb-hq* and *musdb-XL* for each stem. As a result, *drums* and *vocals* scored each 19.97 and 23.69 dB, on the other hand, *bass* and *other* scored each 25.12 and 25.48 dB on average. That is, the signal distortion caused by a limiter is more significant on *drums* and *vocals*.

Unfortunately, since the networks in the Table 2 have never seen these kinds of distorted *drums* or *vocals* as training examples, we assumed that the degradation on *drums* and *vocals* were significant compared to the rest. This result strongly supports the necessity of considering the heavy dynamic range compression from the training stage, i.e. the *LimitAug*, which forcibly makes the distorted and compressed training examples for training purposes, thereby minimizing the domain shift.

5.2 Analysis on the input normalization at the evaluation stage

If the key differences between the real-world music and the training examples of music source separation networks are *i)* overall amplitude scale, and *ii)* the signal distortion caused by a limiter, as stated in Section 3, then what if we give the loudness normalized *musdb-L* or *XL* data as inputs to the networks in Table 2? Due to the non-linear nature of deep neural networks, we assumed that simply normalizing the amplitude scale of the networks on the evaluation stage may affect the performance.

The inference was conducted with following procedures, *(i)* reducing the loudness of *musdb-L* or *XL* input so that its loudness becomes same with that of the corresponding original *musdb-hq* data, *(ii)* inference with loudness normalized input, and *(iii)* increasing the scale of output as much as reduced in step *(i)*.

Comparing the scores between Table 2 and Table 4, we confirmed that the performance degradation was greatly alleviated by just the simple loudness normalization of the inputs only at the evaluation stage. Note that the networks were not trained with loudness normalized inputs. This result shows that the input loudness normalization at the evaluation stage can be a quick and easy solution to get robust results from the pre-trained music source separation networks.

Especially, on *drums* of *Demucs v3-B*, it should be noted that the amount of decrease on *median* SDR score between the test using *musdb-hq* and *XL* was sharply reduced from 2.5 dB in Table 2 to 0.4 dB in Table 3. This implies that the network is overfitted not only to the contents of the signal, but also to the scale or loudness, especially

network	methods	linear gain increase	*LimitAug*	input loud-norm	target LUFS	SDR median (mean) [dB]			
						hq	*L*	*XL*	*avg*
TFC-TDF -U-Net [20]	baseline	-	-	-	-	5.64 (4.02)	5.41 (3.71)	5.27 (3.39)	5.44 (3.71)
	(1)	✓	-	-	$\mathcal{N}(\mu_L, \sigma_L^2)$	**5.90** (4.31)	5.86 (4.33)	5.73 (4.15)	5.83 (4.26)
					$\mathcal{N}(\mu_{XL}, \sigma_{XL}^2)$	5.32 (3.43)	5.36 (3.62)	5.28 (3.49)	5.32 (3.51)
	(2)	-	✓	-	$\mathcal{N}(\mu_L, \sigma_L^2)$	5.79 (4.30)	**5.90 (4.41)**	5.74 (4.25)	5.81 (4.32)
					$\mathcal{N}(\mu_{XL}, \sigma_{XL}^2)$	5.69 (3.93)	5.72 (4.22)	5.57 (4.15)	5.66 (4.10)
	(3)	-	-	✓	-14	5.89 **(4.38)**	5.87 (4.35)	**5.82 (4.25)**	**5.86 (4.33)**
	(4)	-	✓	✓	$\mathcal{N}(\mu_L, \sigma_L^2)$, -14	5.87 (4.25)	5.85 (4.21)	5.76 (4.16)	5.83 (4.21)
					$\mathcal{N}(\mu_{XL}, \sigma_{XL}^2)$, -14	5.78 (4.27)	5.78 (4.26)	5.73 (4.20)	5.76 (4.24)

Table 5. Performance of *TFC-TDF-U-Net* [20] trained with various training data construction strategies. Each score represents the average score across the 4 stems.

network	methods	target LUFS	test data	SDR median (mean) [dB]				
				vocals	*bass*	*drums*	*other*	*avg*
TFC-TDF -U-Net [20]	baseline	-	*hq*	7.18 (4.26)	5.59 (3.35)	5.76 (5.30)	4.04 (3.18)	5.64 (4.02)
			L	7.03 (3.65)	5.41 (3.08)	5.52 (5.09)	3.67 (3.00)	5.41 (3.71)
			XL	6.95 (3.14)	5.48 (2.90)	5.11 (4.68)	3.55 (2.82)	5.27 (3.39)
	(3) loud-norm	-14	*hq*	7.35 **(4.76)**	**5.93** (3.61)	**5.91 (5.37)**	4.39 (3.79)	**5.89 (4.38)**
			L	7.32 (4.72)	5.91 (3.61)	5.85 (5.29)	4.39 (3.78)	5.87 (4.35)
			XL	7.26 (4.64)	5.91 **(3.62)**	5.68 (4.99)	4.42 (3.78)	5.82 (4.25)
	(4) LimitAug, loud-norm	$\mathcal{N}(\mu_L, \sigma_L^2)$, -14	*hq*	**7.59** (4.64)	5.75 (3.25)	5.63 (5.28)	4.50 (3.82)	5.87 (4.25)
			L	7.58 (4.61)	5.69 (3.21)	5.62 (5.22)	4.50 (3.82)	5.85 (4.21)
			XL	7.48 (4.55)	5.67 (3.29)	5.36 (4.99)	**4.51 (3.82)**	5.76 (4.16)

Table 6. Stem-wise performance of *TFC-TDF-U-Net* [20] trained with the method *(3)* input loudness normalization, and *(4)* input loudness normalization after the proposed *LimitAug*.

on *drums*. Note that there was no distinction between the given input sources to the networks except the linear gain difference.

Although the performance degradation was reduced by the input loudness normalization, still there exists the performance degradation due to the signal distortion caused by a limiter. Similar to Section 5.1.2, this result also supports the necessity of the *LimitAug* for robust music source separation.

5.3 Analysis on various training strategies

In this section, we trained the *TFC-TDF-U-net* [20] with various training data creating methods; *(1)* linear gain increasing, *(2)* the proposed *LimitAug*, *(3)* input loudness normalization, and *(4)* input loudness normalization after the *LimitAug*. Of course, the methods *(3)* and *(4)* includes the input loudness normalization at the evaluation stage for the consistency between train and test domains. We compared the results to check which one is the most powerful way for training robust music source separation networks. For the input normalization, we chose the target reference LUFS value as -14.

In Table 5, we confirmed that all of the methods were effective for robust music source separation; every methods showed relatively robust performance on *musdb-L* and *XL*, compared to the baseline. Especially, except the method *(1)* targeting LUFS of a normal distribution following statistics of *musdb-XL*, $\mathcal{N}(\mu_{XL}, \sigma_{XL}^2)$, all methods showed greater performance on *musdb-hq* than the baseline. This result implies that these methods are useful not only for the domain shift, but also for the standard benchmark data.

Furthermore, the methods *(3)* and *(4)*, which prevent the domain shift caused by overall amplitude scale mismatch by input loudness normalization, showed slightly better performances than others. In the stem-wise analysis as presented in Table 6, though we expected that the *LimitAug* would be of help for *vocals* and *drums*, the method *(4)* was better at *vocals* and *other* than the method *(3)*.

Overall, it seems obvious that considering the music loudness and heavy dynamic range compression from the training stage is beneficial for robust music source separation. For real world applications, it is highly recommended that not just using the single method we experimented, but using various training methods on different stems or using a bag of models trained with various methods.

6. CONCLUSIONS

In this study, we questioned the domain shift between the research and the real-world data for music source separation, from the viewpoint of music loudness and heavy dynamic range compression. To answer this, We first built new evaluation datasets, *musdb-L* and *musdb-XL*, which reflect dynamic range compressed music characteristics and heavy loudness. Then, we confirmed the significant performance degradation of state-of-the-art networks on our datasets. To alleviate this, we conducted various experiments on training data construction strategies, including the proposed *LimitAug* method, and confirmed that the methods using the input loudness normalization only or with the *LimitAug* greatly improved the robustness. We hope that our proposed methods and evaluation datasets could contribute to future music source separation research and application.

7. ACKNOWLEDGEMENTS

We appreciate Zafar Rafii, the author of *musdb*, for allowing us to reprocess the original musdb data. We thank Antoine Liutkus, also the author of *musdb*, for giving the creative suggestion on the distribution of our proposed datasets. We are grateful to Ben Sangbae Chon, Keunwoo Choi, and Hyeongi Moon from GaudioLab for their fruitful discussions on our proposed methods. Last but not least, we thank Donmoon Lee, Juheon Lee, Jaejun Lee, Junghyun Koo, and Sungho Lee for helpful feedbacks.

8. REFERENCES

[1] S. Dredge, "Pop music is louder, less acoustic and more energetic than in the 1950s," *The Guardian*, 25 Nov. 2013.

[2] G. Milner, "They really don't make music like they used to," *The New York Times*, 7 Feb. 2019.

[3] N. B. Croghan, K. H. Arehart, and J. M. Kates, "Quality and loudness judgments for music subjected to compression limiting," *The Journal of the Acoustical Society of America*, vol. 132, no. 2, pp. 1177–1188, 2012.

[4] E. Vickers, "The loudness war: Background, speculation, and recommendations," in *Audio Engineering Society Convention 129*. Audio Engineering Society, 2010.

[5] E. F. Stikvoort, "Digital dynamic range compressor for audio," *Journal of the Audio Engineering Society*, vol. 34, no. 1/2, pp. 3–9, 1986.

[6] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix - a reference implementation for music source separation," *Journal of Open Source Software*, 2019. [Online]. Available: https://doi.org/10.21105/joss.01667

[7] B. Katz and R. A. Katz, *Mastering audio: the art and the science*. Butterworth-Heinemann, 2003.

[8] L. Bregitzer, *Secrets of recording: Professional tips, tools & techniques*. Routledge, 2008.

[9] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 261–265.

[10] L. Prétet, R. Hennequin, J. Royo-Letelier, and A. Vaglio, "Singing voice separation: A study on training data," in *ICASSP 2019-2019 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2019, pp. 506–510.

[11] R. ITU-R, "Itu-r bs. 1770-2, algorithms to measure audio programme loudness and true-peak audio level," *International Telecommunications Union, Geneva*, 2011.

[12] R. EBU-Recommendation, "Loudness normalisation and permitted maximum level of audio signals," *European Broadcasting Union*, 2011.

[13] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[14] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "Musdb18-hq - an uncompressed version of musdb18," Aug. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3338373

[15] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 45–49.

[16] A. Liutkus, F.-R. Stöter, Z. Rafii, D. Kitamura, B. Rivet, N. Ito, N. Ono, and J. Fontecave, "The 2016 signal separation evaluation campaign," in *International conference on latent variable analysis and signal separation*. Springer, 2017, pp. 323–332.

[17] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research." in *ISMIR*, vol. 14, 2014, pp. 155–160.

[18] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation: 14th International Conference, LVA/ICA 2018, Surrey, UK*, 2018, pp. 293–305.

[19] A. Défossez, "Hybrid spectrogram and waveform source separation," *arXiv preprint arXiv:2111.03600*, 2021.

[20] W. Choi, M. Kim, J. Chung, D. Lee, and S. Jung, "Investigating u-nets with various intermediate blocks for spectrogram-based singing voice separation," in *21th International Society for Music Information Retrieval Conference, ISMIR, Ed*, 2020.

[21] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020.

[22] Y. Mitsufuji, G. Fabbro, S. Uhlich, and F.-R. Stöter, "Music demixing challenge 2021," *arXiv preprint arXiv:2108.13559*, 2021.

[23] M. Kim, W. Choi, J. Chung, D. Lee, and S. Jung, "Kuielab-mdx-net: A two-stream neural network for music demixing," *arXiv preprint arXiv:2111.12203*, 2021.

[24] Spotify, "pedalboard : A python library for manipulating audio." 2022. [Online]. Available: https://github.com/spotify/pedalboard

[25] C. J. Steinmetz and J. D. Reiss, "pyloudnorm: A simple yet flexible loudness meter in python," in *150th AES Convention*, 2021.

[26] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[27] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "Sdr–half-baked or well done?" in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 626–630.

TOWARDS QUANTIFYING THE STRENGTH OF MUSIC SCENES USING LIVE EVENT DATA

Michael Zhou
Columbia University
mgz2112@columbia.edu

Andrew McGraw
University of Richmond
amcgraw@richmond.edu

Douglas R. Turnbull
Ithaca College
dturnbull@ithaca.edu

ABSTRACT

There are many benefits for a community when there is a vibrant local music scene (e.g., increased mental & physical well-being, increased economic activity) and there are many factors that contribute to an environment in which a live music scene can thrive (e.g., available performance spaces, helpful government policies). In this paper, we explore using an estimate of the *live music event rate* (LMER) as a rough indicator to measure the strength of a local music scene. We define LMER as the number of music shows per 100,000 people per year and then explore how this indicator is (or is not) correlated with 28 other socioeconomic indicators. To do this, we analyze a set of 308,051 music events from 2019 across 1,139 cities in the United States. Our findings reveal that factors related to transportation (e.g., high walkability), population (high density), economics (high employment rate), age (high proportion of individuals age 20-29), and education (bachelor's degree or higher) are strongly correlated with having a high number of live music events. Conversely, we did not find statistically significant evidence that other indicators (e.g., racial diversity) are correlated.

1. INTRODUCTION

Certain American cities such as Los Angeles and New York City are famous for having great music scenes. However, there are hundreds of small, medium, and large cities around the country that support vibrant music scenes. These cities often have well-known venues like the Grand Ole Opry in Nashville, TN, or put on large music festivals such as SXSW in Austin, TX. They can be associated with artists who obtain some level of regional, national, or international fame such as Minneapolis, MN with Prince and Asbury Park, NJ with Bruce Springsteen. They sometimes become historically connected with specific genres as in New Orleans, LA with jazz, Seattle, WA with grunge, and Asheville, NC with bluegrass.

There have been many studies that detail the ways in which music scenes can benefit their communities by enhancing social bonding, improving emotional well-being [1,2], and increased economic activity [3–15]. Researchers have also studied factors that can help foster an environment in which a local music scene can develop and thrive [4, 5, 8, 9, 11, 12, 16–22]. These factors include having a rich music history, having strong support for music education, and government regulations that are favorable for live performance (see Section 2 for details).

Investment by government and non-government organizations (e.g., Chambers of Commerce, Arts Councils) are associated with strong music scenes ("music havens") to both further develop these havens as well as help cities with underdeveloped music scenes ("music deserts") [11]. Many organizations have produced extensive reports [19] about "music cities" based on interviews and surveys of cities around the world (e.g., Austin, USA [3], London, UK [23], Victoria, AU [24]). While these reports produce valuable and transferable knowledge, they tend to be narrow in their geographic focus (i.e., one city or region). To complement this body of work, we propose a quantitative approach that uses the live music event rate (LMER) to estimate the *relative* strength of a local music scene. We argue that this simple statistic is straightforward to calculate, easy to interpret, and useful.

In this work, we introduce a music event dataset that contains information for 308,051 live music events that took place in 1,139 American cities during 2019. Here we consider an event as a live performance by one or more artists at a venue (e.g., bar, concert hall, festival) on a given date. This dataset was collected for music event recommendation application and combines event information that was collected from BandsInTown [1] and Facebook [2]. While this dataset has a number of limitations (e.g., only music events with a digital footprint, data collected using snowball sampling), it allows us to calculate a rough estimate of LMER for each city to enable comparison.

In this paper, we explore how our estimated LMER is (or is not) correlated with 28 socioeconomic indicators across 6 different categories: transportation, population, economics, age, education, and race. These indicators are closely related to some of the factors (e.g., transportation convenience, population density) that other researchers have suggested are important factors for fostering healthy music scenes.

In Section 2, we explore existing research on the ben-

© M. Zhou, A. McGraw, and D. Turnbull. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: M. Zhou, A. McGraw, and D. Turnbull, "Towards Quantifying the Strength of Music Scenes using Live Event Data", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] https://www.bandsintown.com/
[2] https://www.facebook.com/

Benefits of Music Communities	Factors that Create a Strong Music Community
Job creation [3, 4, 11, 12]	Availability of performance spaces and venues [11]
Increased consumer spending [6, 8, 9]	Financial affordability [11, 20]
Patronages & sponsorships [5, 11, 12]	Music tourism [8, 9, 11]
Increased financial investments [3, 11]	Music education resources [11, 12, 22]
Lower poverty rates [25]	Music heritage and history [4, 11]
Strengthening social bonding [11]	Demographics (e.g. student populations) [4, 11, 12, 17]
Improved mental/physical health [1, 2]	Government support/regulation [5, 11, 12, 16]
Lower crime rates [7, 13, 15]	Transportation convenience [11, 20]
Better community reputation [3, 11]	Population density [19]
	Population growth [18, 21]

Table 1: Table summarizing the benefits of a music community and factors that stimulate a music community.

efits of and factors that support a strong music scene. In Section 3, we introduce our dataset of music event information and describe how we estimate LMER. Section 4 explores how LMER is correlated with different socioeconomic indicators some of which have been identified as being important factors in healthy music scenes. We then conclude in Section 5 with a discussion of how LMER can be useful for identifying potential music deserts.

2. RELATED WORK

In this section, we review both academic and music industry research that focuses on the following two questions:

1. What are the benefits of having a strong music scene?
2. What factors help foster a strong music scene?

A summary of our review is provided in Table 1.

2.1 Benefits of a Strong Music Community

Vibrant music scenes support local economies [3,4,11,12]. A live music event involves working musicians, booking and ticketing agents, sound and lighting technicians, bartenders, security guards, etc. Consumers spend money [6, 8, 9] on concert tickets, food and beverages, and artist merchandise. If a music community is strong enough, it also attracts tourism, where people from outside the community generate revenue for the local economy [8, 9, 11] through hotel stays and other local attractions (museums, parks, etc.). Live music scenes include many small businesses (music and record stores, recording studios, and private music teachers) and large institutions (music conservatories, theaters, and academic institutions). Finally, a vibrant music scene can generate economic knock-on benefits through patronage, sponsorship, cross-promotion, and cross-pollination relationships with other economic sectors [3, 5, 11, 12]. Such cultural-economic interactions can enhance a location's reputation and quality of life, attracting new residents and new participants to a scene.

The social and individual benefits of music are also well-studied. Music strengthens the social fabric of a community by "building bridges between cultures and languages, connecting people within a city, a region and across borders," as music "touches human beings" and "engages people" [11]. Vibrant music scenes are a component of the "social infrastructure" of healthy communities [26], spaces in which community members of various backgrounds can engage and interact with one another. They are physical spaces that afford and encourage face-to-face social interaction, the development of dense social networks, and the cultivation of shared values.

Many of these spaces gather together people across the social divisions of class, race, ethnicity, and faith. They are contexts in which people of different backgrounds can encounter one another in a common and often public space. According to the sociologist Robert Putnam, in the context of large pluralistic societies such as the USA, social activities such as music cultivate "bridging social capital." These are contexts in which we connect to people potentially unlike ourselves and through this cultivate a sense of "civic virtue" that widens "our awareness of the many ways in which our fates are linked" [27].

Empirical analyses of large social datasets suggest that engagement in arts events is associated with increased happiness and satisfaction measures [2]. Ethnographic case studies of particular music scenes suggest that participation in vibrant scenes is associated with subjective well-being [1, 11]. Industry analyses and censuses associate strong scenes with enhanced civic pride and cultural reputation [3, 11]. Our ongoing work will explore possible correlations between live, public music events across America and a wide range of well-being indicators, including political participation and the experience of belonging, trust, and reciprocity.

2.2 Factors that Create a Strong Music Community

Terrill et al. [11] identifies five essential elements of strong music scenes: a large number of active musicians, a community that supports diverse musical offerings, a variety of spaces for rehearsing, recording, and performance, along with a receptive and engaged audience, and a variety of music-related business. Other helpful factors include multi-level government support [5, 12, 16] (e.g., cultural zones, and accommodating noise ordinances and liquor laws [28]), city infrastructure (transportation, affordable housing), and music education [12, 22] from public school

music programs, to private lessons, to music conservatories [29]. A vibrant music scene also requires equitable access to funding, the fair enforcement of city ordinances, and the cultivation of shared social spaces [28]. Are the small business loans needed to support music infrastructure equitably available across demographic sectors? Are noise ordinances and venue regulations enforced and policed evenly across neighborhoods? Does the local government sponsor arts festivals and programs available to and welcoming of all citizens?

Demographics play a key role in establishing and invigorating a music community; for example, young people such as university students are more likely to go out to music events [4, 11, 12]. Denser populations often mean stronger social networking, associated with vibrant music scenes [19]. Lastly, as the population increases, the demand for live music also often increases, which can strengthen a music community [18, 21].

3. MUSIC EVENT DATA

To obtain a rough estimate of the strength of a city's music scene, we calculated its *live music event rate* (LMER). Specifically, we calculate the number of music events per year per 100,000 residents in a given geographic area.

We compiled the LocalifyMusicEvents-USA-2019 dataset [3] that consists of information for 308,051 music events from 1,139 cities in the United States, each with a population of 10,000 or more residents. The music event information was collected using custom web scrapers in late 2018 and all of 2019. We developed two web scrapers, one for BandsInTown, a comprehensive music event website, and one for Facebook, a popular social network. The BandsInTown scraper relied on snowball sampling [30] to build growing lists of artist and venue pages. Each page was scraped for events every 30 days. New artists and venues were continually added based on similarities between artists, artists playing at different venues, and artists playing events with other artists. By contrast, our Facebook scraper collected music event information by cycling repeatedly over a large list of cities continually.

To calculate LMER, we first count the total number of music events listed in the city during 2019 by matching the city and state names. We then divided by the city population according to the US Census population estimates for 2019.

Our dataset also contains 28 city-level socioeconomic indicators obtained from three different websites: Census Reporter [4], DataUSA [5], and Census Quickfacts [6]. These indicators are partitioned into six groups (transportation, population, economics, age, education, race) and are summarized in Table 2.

[3] Data can be found at `https://github.com/JimiLab/LocalifyMusicEventData`.

[4] https://censusreporter.org/

[5] https://datausa.io/

[6] https://www.census.gov/quickfacts/

3.1 Music Havens

We divide the cities into three subsets grouped by population: small (pop. 10K-100K), medium (pop. 100K-500K), and large (pop. 500K or more). For each subset, the top ten "music havens" (cities with the largest LMER) are shown in Table 3.

Size	Principal City	Pop.	LMER
Small	South Burlington, VT	19162	0.036
	Steamboat Springs, CO	12928	0.020
	Asheville, NC	92859	0.019
	Santa Cruz, CA	64605	0.019
	Key West, FL	24843	0.017
	Burlington, VT	42545	0.017
	Ithaca, NY	30569	0.016
	Fredericksburg, TX	11245	0.016
	Lahaina, HI	12776	0.014
	Rutland, VT	15398	0.013
Medium	Salt Lake City, UT	200546	0.023
	Berkeley, CA	121353	0.016
	New Orleans, LA	390144	0.015
	Richmond, VA	230436	0.015
	Cambridge, MA	118925	0.013
	Boulder, CO	105670	0.010
	Orlando, FL	287435	0.009
	Fort Collins, CO	170245	0.009
	Minneapolis, MN	429605	0.009
	Charleston, SC	143151	0.008
Large	Denver, CO	727211	0.016
	Washington, DC	705749	0.011
	Austin, TX	979263	0.011
	Atlanta, GA	506804	0.010
	Seattle, WA	753655	0.010
	Portland, OR	653467	0.009
	Las Vegas, NV	651297	0.008
	San Francisco, CA	881549	0.007
	Philadelphia, PA	1584064	0.005
	Dallas, TX	1343565	0.005

Table 3: Top 10 "Music Havens" for Small (pop. 10K-100K), Medium (pop. 100K-500K), and Large Cities (pop. 500K+) ranked by Live Music Evert Rate (LMER)

In these three lists, we find several popular tourist towns, e.g., Steamboat Springs, CO, Key West, FL, Las Vegas, NV, as well as college towns, e.g., Burlington & South Burlington, VT (U. of Vermont), Ithaca, NY (Cornell U.), Berkeley, CA (U. of California). We also find several cities, like Austin, TX, Asheville, NC, Atlanta, GA, and New Orleans, LA which all frequently appear on lists of top destinations for "music tourism" in the United States. [7]

Many of the medium and large cities that we identify as music havens also appear on a list of American cities with the most concerts per capita based on data from Seat-

[7] https://www.thrillist.com/travel/nation/best-cities-for-live-music-new-york-memphis-asheville-and-austin

Indicator	Description	Source
	Transportation	
Mean Travel Time	Mean travel time to work (minutes)	Census Reporter
Public Transit	% of population who took public transit (e.g., buses)	Census Reporter
Bicycle	% of population who biked	Census Reporter
Walkability	% of population who walked	Census Reporter
Public Transit+Bicycle+Walkability	% of population who took public transit, biked, or walked	Census Reporter
	Population	
Population Density	Population per square mile	Census Reporter
10 Year Population Growth	Population growth from 2010 (acc. to Census Quickfacts, as of April 1, 2010) to 2019 (acc. to ACS 2019).	Census Quickfacts, Census Reporter
Migration Rate Since Previous Year	Geographic mobility - % of population who moved to city since last year.	Census Reporter
	Economics	
Per Capita Income	Average income per person in city ($)	Census Reporter
Median Household Income	Median income per household in city ($)	Census Reporter
Poverty Rate	% of city population below poverty line	Census Reporter
Median Property Value	Median value of owner-occupied housing units ($)	Census Reporter
Employment Rate	Number of people employed (DataUSA) divided by 2019 population (Census Reporter)	DataUSA, Census Reporter
Median Gross Rent	Median gross rent ($), 2015-2019	Census Quickfacts
Median Owner Cost With Mortgage	Median selected monthly owner costs -with a mortgage ($), 2015-2019	Census Quickfacts
Median Owner Cost Without Mortgage	Median selected monthly owner costs -without a mortgage ($), 2015-2019	Census Quickfacts
Owner-Occupied Housing Unit Rate	Owner-occupied housing unit rate (%), 2015-2019	Census Quickfacts
	Age	
Median Age	Median age of city	Census Reporter
Percent Under 18	Percentage of population under age 18	Census Reporter
Percent 18-29	Percentage of population between ages 18 and 29	Census Reporter
Percent Under 30	Percentage of population under age 30	Census Reporter
Percent 20-29	Percentage of population between ages 20 and 29	Census Reporter
Percent 10-29	Percentage of population between ages 10 and 29	Census Reporter
Age Diversity Index	Simpson's diversity index for age (0-9, 10-19, ..., 70-79, 80+)	Census Reporter
	Education	
High School Or Higher	Percentage of population that are high school grads or higher	Census Reporter
Bachelor Or Higher	Percentage of population with Bachelor'\s degree or higher	Census Reporter
Postgrad Degree	Percentage of population with post-grad degree	Census Reporter
	Race	
Race Diversity Index	Simpson's diversity index for ethnicity (White, Black, Native, Asian, Islander, Other, Two+, Hispanic)	Census Reporter

Table 2: A list of all 28 city-level socioeconomic indicators used and their corresponding descriptions.

Geek [8], a large ticket reselling site [31]. Their top cities include Las Vegas, NV, Nashville, TN, Austin, TX, and Denver, CO. This overlap gives us some confidence in our approach but it should be noted that the SeatGeek reports only examines large events from the top 100 grossing artists in the top 100 market areas (i.e., cities).

3.2 Music Deserts

There were 87 cities (7.6% of 1,139 cities) for which there were no events in the LocalifyMusicEvents-USA-2019 dataset. The three "music desert" cities with the largest populations were Renton, WA (pop. 101,747), Deltona, FL (pop. 92,752), and Newton, MA (pop. 88,411). When we examine all three using simple Google web searches, it is clear that there are music events taking place in these cities, as they are listed on their corresponding local websites [9] but not found when we scraped event information from our two data sources (BandsInTown, Facebook). This suggests that our dataset is incomplete due to the imperfect nature of our scraping procedure and our limited set of data sources. We will further discuss these limitations in Section 5.1.

[8] https://seatgeek.com/

[9] Renton, WA: https://rentondowntown.com/happenings/summer-concert-series/, Deltona, FL: https://www.deltonafl.gov/parks-recreation-department/events/22814, Newton, MA: https://patch.com/massachusetts/newton/newton-porchfest-2019-what-know

4. CORRELATION WITH SOCIOECONOMIC INDICATORS

In this section, we explore correlations between LMER and the 28 different socioeconomic indicators using statistical testing. If we assume that LMER is a rough indicator of the strength of a local music scene, we can use it to study how music scenes are related to other aspects of our society. We have grouped the 28 socioeconomic indicators into six categories: Transportation, Population, Economics, Age, and Education & Race. The 28 indicators we sampled, as well as the sources they were collected from, are shown in Table 2.

We use a Bonferroni correction when determining statistical significance since we are conducting multiple hypothesis statistical tests [32]. Usually, in significance testing, we perform one statistical test, and obtain the correlations r and p-values p; an indicator is deemed significant if $p < \alpha$, where α is the p-value threshold. The Bonferroni correction, however, deems an indicator significant if $p < \alpha/I$, where I is the number of statistical tests (i.e., one per indicator). In our experiment, there are $I = 28$ indicators and we set α to 0.05; thus, a correlation is significant if the p-value is less than $\alpha/I = 1.8\text{e-}3$.

Table 4 shows the correlation coefficients (or r-values) and the probability of observing the data assuming no correlation (p-value) for each of these indicators grouped in their categories, with the p-values (p) ranked from smallest (most significant) to largest (least significant) for each category.

4.1 Transportation

As shown in Table 4, we find that there is a strong positive correlation between LMER and the first four transportation-based indicators (percentage of people who bike, take public transit, walk, or all three together). Good public transportation reduces the overall cost of attending a show [20] and enables more people to drink alcohol (more safely) at night, which is very often associated with live music events. This finding supports the claim by Terrill et al. [11] that supportive urban infrastructure is an important factor for a strong local music scene. We also considered the mean travel time to work, but did not find it to have a statically significant correlation.

4.2 Population

We observe that population density as well as one-year and ten-year population growth are all positively correlated with LMER. Van der Hoeven and Hitters [19] argue that "density and diversity provide the critical mass of participants that alternative [music] scenes need to thrive." When people are clustered together, it is easier to engage and interact with one another. This idea also relates to population growth; as people with diverse backgrounds immigrate to the city, there is an increase in opportunities for musicians to influence one another in novel ways [18].

4.3 Economics

As we discussed in Section 2.2, many researchers have suggested that having a strong local music scene is good for the local economy [6, 8, 9, 11]. This is consistent with our findings that employment rate, housing cost, per capita income, and median property values are all positively correlated with LMER. We found it interesting that the owner-occupied housing rate is negatively correlated which suggests that there are more live music events when there is a higher proportion of renters relative to homeowners. This is consistent with the observation that home affordability has dropped in the United States, making it harder for young people to own their homes [33], and as we observe in the next subsection, having a larger percentage of younger adults is positively correlated with LMER.

Concerning poverty, Harrison [25] examines how "music projects develop the skills, education levels, incomes, or occupational possibilities of participants living in material poverty, which in turn can enhance their socio-economic status." Accordingly, we might expect a lower poverty rate where there was more live music but we did not find a statistically significant correlation between poverty rate and LMER. We assume therefore that the dynamic between poverty and music is too complex to be measured in a simple quantitative analysis.

4.4 Age

Our results show that cities with a high proportion of young people between the ages of 18 and 29 tend to have a large live music event rate. Conversely, cities with a large population of people under 18 are negatively correlated with LMER. This may be because many music venues (i.e., bars) tend to have 18-and-up and 21-and-up policies due to laws related to serving alcohol. These two results, along with the fact that neither median age nor the percentage of the population under 30 is correlated with LMER, suggest that cities with many young adults (college students, young professionals, aspiring young artists) are places where we expect to have many live music events. Conversely, cities with a relatively large percentage of families are less likely to have a high rate of music events. As mentioned in the previous subsection, age is also positively correlated to home ownership, as decreasing home affordability in the United States has caused fewer young people to own a home; this entails a negative correlation between owner-occupied housing rate and LMER, as higher housing ownership rates means fewer young people, an indicator of a lower LMER.

4.5 Education

Terrill et al. [11] suggests that "cities such as Toronto, Adelaide, Austin, and Berlin point to their large student populations as helpful factors in generating engaged audiences." Our results support this in that we find a positive correlation between LMER and cities with a high proportion of people with undergraduate or graduate degrees. This is also reflected by the fact that many of the top Music Havens in Table 3 are college towns as was discussed in Section 3.1.

4.6 Race

We did not find a significant correlation between our racial diversity index and the LMER. This might be surprising considering the associations suggested between healthy music scenes and demographic diversity [34]. In that, we only explore one indicator that explicitly relates to race, a more thorough analysis is required to explore the complex relationship between race and the strength of music scenes.

5. DISCUSSION

In this paper, we explore how using the live music event rate (LMER) is straightforward to estimate, easy to interpret, and correlated with a large and diverse set of socio-economic indicators. We found a strong positive correlation between LMER and the percentage of people who take public transit, walk, or bike. Education is also strongly correlated with LMER. We also observed that many economic indicators (e.g., employment rate, per capita income, median property value) are also correlated with LMER. Finally, there is a high live music event rate when there is a high proportion of late teens and individuals in their twenties. We did not find a significant correlation between LMER and our race diversity index. This is not to say that no relationship exists, but rather our simplistic analysis did not reveal a statistically significant correlation.

Indicator	r	p
Transportation		
Bicycle	**0.39**	**5.1e-43**
Public Transit+Bicycle+Walkability	**0.34**	**1.1e-32**
Public Transit	**0.26**	**1.8e-18**
Walkability	**0.24**	**7.7e-17**
Mean Travel Time	0.04	1.5e-01
Population		
Population Density	**0.21**	**1.5e-12**
10 Year Population Growth	**0.18**	**9.8e-10**
Migration Rate Since Previous Year	**0.13**	**1.7e-05**
Economics		
Employment Rate	**0.26**	**4.7e-19**
Owner-Occupied Housing Unit Rate	**-0.21**	**3.3e-13**
Median Owner Cost w/ Mortgage	**0.19**	**2.4e-10**
Median Owner Cost w/o Mortgage	**0.17**	**2.9e-09**
Median Property Value	**0.17**	**1.8e-08**
Per Capita Income	**0.16**	**3.7e-08**
Median Gross Rent	**0.15**	**2.7e-07**
Median Household Income	0.06	3.1e-02
Poverty Rate	-0.00	8.2e-01
Age		
Percent Under 18	**-0.24**	**7.6e-16**
Percent 20-29	**0.17**	**2.0e-08**
Percent 18-29	**0.14**	**1.8e-06**
Age Diversity Index	**0.11**	**1.1e-04**
Percent 10-29	**0.10**	**4.5e-04**
Median Age	-0.06	4.1e-02
Percent Under 30	0.02	4.1e-01
Education		
Bachelor Or Higher	**0.27**	**6.6e-20**
Postgrad Degree	**0.24**	**8.1e-16**
High School Or Higher	**0.12**	**6.6e-05**
Race		
Race Diversity Index	-0.07	1.9e-02

Table 4: Table showing the correlation coefficient (r) and p-values (p) for all 28 socioeconomic indicators across all 1,139 cities. Statistically significant indicators are in **bold** font.

5.1 Limitations

As mentioned in Section 3.2, the music events dataset was not collected for this research, but rather for our Localify.org [10] music events recommendation application. As a result, the scraping was limited in the number of data sources (Facebook & BandsInTown), collected in an ad-hoc manner (snowball sampling), only covers one year of data (2019), and only looks at one country (United States). It is probable that many active artists, venues, and events were not found using our music event data collection process.

To increase our coverage and add redundancy, we have since added additional scrapers (e.g., SongKick, Google, Eventbrite) but the music event data we have scraped in 2020 and 2021 is problematic since such a high percentage of scheduled music events were canceled due to COVID-19. However, even with a large number of scrapers, our approach necessarily ignores music events that do not have a digital footprint. This includes many underground (private house parties, DIY shows), music in religious spaces (e.g., churches), and impromptu performances (e.g., busking). Our future work will be to explore how our approach to scraping music event information might be incomplete and/or biased by taking a detailed census of music events in individual cities using more principled ethnographic techniques (e.g., observation, interviews, historical records).

As described in Section 3, the 28 socioeconomic indicators for each city were collected using three different web sources, rather than one unified source. A more systematic approach is to use one unified source for our data collection, such as ESRI datasets [11] . For certain indicators addressed in Section 2 and Table 1, such as mental/physical health, cultural heritage, and local government regulations on music, we could not find any available quantitative data sources to measure them.

5.2 Future Work

Both popular culture and academic research tend to focus on "music havens" like Nashville, TN and Seattle, WA. These are cities that are known to have a great local music scene and they reap economic, social, and cultural rewards because of it. We, by contrast, are especially interested in identifying cities with underdeveloped music scenes. We plan to study these "music deserts"' and explore how various strategies could be used to help them strengthen their local music scenes. For example, Terrill et al. [11] outlines a number of potential strategies which include developing music-friendly government policies, creating music-focused offices and advisory boards, investing in audience development, and creating music tourism plans. We argue that cities with low local music event rates (LMER) may be good initial candidates for future research involving the study of music deserts.

6. REPRODUCIBILITY

To make our research both fully reproducible and transparent, our full LocalifyMusicEvents-USA-2019 dataset and the associated data processing code (in the form of Jupyter Notebooks) can be found at `https://github.com/JimiLab/LocalifyMusicEventData`.

7. ACKNOWLEDGEMENTS

Doug Turnbull, Tim Clerico, John Hunter, and Emmett Barry all contributed to the Localify.org music event scraper code. This research was supported by NSF grant IIS-1901330/1901168 and NEH grant HAA-280975-21. Finally, we would like to thank the anonymous ISMIR reviewers for their highly constructive feedback.

8. REFERENCES

[1] S. Baker, R. Nowak, P. Long, J. Collins, and Z. Cantillon, "Community Well-Being, Post-Industrial Music

[10] https://localify.org/

[11] https://www.esri.com/en-us/arcgis/products/arcgis-open-data

Cities and the Turn to Popular Music Heritage," in *Music Cities*. Springer, 2020, pp. 43–61.

[2] D. Wheatley and C. Bickerton, "Subjective Well-Being and Engagement in Arts, Culture and Sport," *Journal of Cultural Economics*, vol. 41, no. 1, pp. 23–45, 2017.

[3] T. M. G. (Firm), *The Austin Music Census: A Data-Driven Assessment of Austin's Commercial Music Economy*. Titan Music Group, LLC, 2015. [Online]. Available: https://books.google.com/books?id=fStOjwEACAAJ

[4] A. J. Baker, "Algorithms to Assess Music Cities: Case Study — Melbourne as a Music Capital," *SAGE Open*, vol. 7, no. 1, pp. 1–12, 2017.

[5] S. Frith, M. Brennan, M. Cloonan, and E. Webster, "'Analysing Live Music in the UK' Findings One Year into a Three-Year Research Project," *IASPM Journal*, vol. 1, no. 1, pp. 1–30, 2010.

[6] F. Holt, "The Economy of Live Music in the Digital Age," *European Journal of Cultural Studies*, vol. 13, no. 2, pp. 243–261, 2010.

[7] D. Siegel and F. Bovenkerk, *Crime and Music*. Springer, 2021.

[8] "Economic Contribution of the Venue-Based Live Music Industry in Australia," *Australasian Performing Right Association*, vol. 20, 2011.

[9] S. Hallam, I. Cross, and M. Thaut, *Oxford Handbook of Music Psychology*. Oxford University Press, 2011.

[10] A. J. Ferrer, "The Effect of Live Music on Decreasing Anxiety in Patients Undergoing Chemotherapy Treatment," *Journal of Music Therapy*, vol. 44, no. 3, pp. 242–255, 2007.

[11] A. Terrill, D. Hogarth, A. Clement, P. Assistant, and R. Francis, *The Mastering of a Music City: Key Elements, Effective Strategies and Why It's Worth Pursuing*. Music Canada, 2015.

[12] E. Webster, M. Brennan, A. Behr, M. Cloonan, and J. Ansell, "Valuing Live Music: The UK Live Music Census 2017 Report," 2018.

[13] A. Williams and A. Lal, "Correlations between Crime Rates in US Cities, and the Popularity of Rap and Hip-Hop Music," 2015.

[14] S. Oakes and G. Warnaby, "Conceptualizing the Management and Consumption of Live Music in Urban Space," *Marketing Theory*, vol. 11, no. 4, pp. 405–418, 2011.

[15] C. Pinkney and S. Robinson-Edwards, "Gangs, Music and the Mediatisation of Crime: Expressions, Violations and Validations," *Safer Communities*, 2018.

[16] S. Brown and U. Volgsten, *Music and Manipulation: On the Social Uses and Social Control of Music*. Berghahn Books, 2005.

[17] H. Shobe and D. Banis, "Music Regions and Mental Maps: Teaching Cultural Geography," *Journal of Geography*, vol. 109, no. 2, pp. 87–96, 2010.

[18] C. Gibson, "Rural Transformation and Cultural Industries: Popular Music on the New South Wales Far North Coast," *Australian Geographical Studies*, vol. 40, no. 3, pp. 337–356, 2002.

[19] A. Van der Hoeven and E. Hitters, "The Social and Cultural Values of Live Music: Sustaining Urban Live Music Ecologies," *Cities*, vol. 90, pp. 263–271, 2019.

[20] P. E. Earl, "Simon's Travel Theorem and the Demand for Live Music," *Journal of Economic Psychology*, vol. 22, no. 3, pp. 335–358, 2001.

[21] M. Brennan, "Live Music History," *The SAGE Handbook of Popular Music*, pp. 207–222, 2015.

[22] K. Swanwick, *Music, Mind and Education*. Routledge, 2003.

[23] M. Davyd, "GRASSROOTS MUSIC VENUES (GMVs)–DEFINITION," 2016.

[24] D. A. Economics, "The Economic, Social and Cultural Contribution of Venue-based Live Music in Victoria," *Melbourne: Arts Victoria*, 2011.

[25] K. Harrison, "Music, Health, and Socio-Economic Status: A Perspective on Urban Poverty in Canada," *Yearbook for Traditional Music*, vol. 45, p. 58–73, 2013.

[26] E. Klinenberg, *Palaces for the People: How Social Infrastructure Can Help Fight Inequality, Polarization, and the Decline of Civic Life*. Crown, 2018.

[27] R. D. Putnam, "Bowling Alone: America's Declining Social Capital," in *Culture and Politics*. Springer, 2000, pp. 223–234.

[28] A. McGraw, "Mapping Sonic and Affective Geographies in Richmond, Virginia," *Sonic Identity at the Margins*, p. 19, 2022.

[29] L. K. Richerme, "Equity via Relations of Equality: Bridging the Classroom-Society Divide," *International Journal of Music Education*, vol. 39, no. 4, pp. 492–503, 2021.

[30] L. A. Goodman, "Snowball Sampling," *The Annals of Mathematical Statistics*, pp. 148–170, 1961.

[31] "Which U.S. Cities Get the Most Concerts?" https://seatgeek.com/tba/music/which-u-s-cities-get-the-most-concerts/.

[32] C. Y. Wijaya, "Multiple Hypothesis Testing Correction for Data Scientist," Oct 2021. [Online]. Available: https://towardsdatascience.com/multiple-hypothesis-testing-correction-for-data-scientist-46d3a3d1611d

[33] G. Paz-Pardo, "This is Why Owning a Home is More Difficult than Ever for Young People," Jan 2022. [Online]. Available: https://www.weforum.org/agenda/2022/01/younger-generations-homeownership-housing-market-wealth-inequality/

[34] C. Ballico and A. Watson, *Music Cities: Evaluating a Global Cultural Policy Concept*. Springer, 2020.

LEARNING MULTI-LEVEL REPRESENTATIONS FOR HIERARCHICAL MUSIC STRUCTURE ANALYSIS

Morgan Buisson[1] **Brian McFee**[2,3] **Slim Essid**[1] **Hélène C. Crayencour**[4]
[1] LTCI, Télécom Paris, Institut Polytechnique de Paris, France
[2] Music and Audio Research Laboratory, New York University, USA
[3] Center of Data Science, New York University, USA
[4] L2S, CNRS-Univ.Paris-Sud-CentraleSupélec, France

ABSTRACT

Recent work in music structure analysis has shown the potential of deep features to highlight the underlying structure of music audio signals. Despite promising results achieved by such representations, dealing with the inherent hierarchical aspect of music structure remains a challenging problem. Because different levels of segmentation can be considered as equally valid, specifically designed representations should be optimized to improve hierarchical structure analysis. In this work, unsupervised learning of such representations using a contrastive approach operating at different time-scales is explored. The proposed system is evaluated on flat and multi-level music segmentation. By leveraging both time and the hierarchical organization of music structure, we show that the obtained deep embeddings can encode meaningful patterns and improve segmentation at various levels of granularity.

1. INTRODUCTION

Common approaches for music structure analysis can usually be broken down into two main steps: segmentation and structural grouping [1]. The segmentation task aims at determining the boundary locations between consecutive musical sections while the grouping step consists in assigning labels to each of the retrieved segments based on certain musical similarities. Traditional algorithms for structural segmentation use different hand-crafted features [2] and their combinations to detect abrupt changes of particular musical characteristics or repetitions of certain patterns throughout the song. However, recent progress in deep learning has given rise to new systems automatically producing more robust representations which manage to combine several acoustic characteristics to enhance the recognition of musical sections [3–5]. While these representations have consistently improved downstream segmentation methods, their performance is mostly evaluated on *flat* structural annotations and metrics. However, musical structure naturally exhibits a hierarchical organization where a variety of cues can trigger boundaries between segments of different length [6], depending on the time scale at which they are observed [1]. At the lowest temporal level, short segments might only last a few measures. Coarser annotation levels are generally composed of longer segments, grouping various shorter fragments into larger musically meaningful units (ex: chorus, verse ...). This nested organization of musical events at different levels holds crucial information about music structure [7]. While the original task of music structure analysis is commonly performed at a pre-defined level of granularity (i.e. *flat* segmentation), the problem of *hierarchical* structural analysis consists in predicting a set of segmentation candidates called hierarchy, ordered by their amount of detail (from the coarsest to the most refined level). Recent efforts have been made to compile datasets with multi-level structural annotations [2, 8], which greatly facilitates the study of musical structure in a hierarchical manner. Although a few methods have been proposed for such task, the role of hierarchy in music structure has never been explicitly considered while building better-suited representations of audio music signals prior to segmentation.

1.1 Our contributions

In this work, we propose a deep unsupervised hierarchical metric learning approach for music structure analysis. We show that leveraging both time information and the hierarchical structure of music can help building efficient representations for music segmentation at different levels without requiring any supervision from structural annotations. We demonstrate the effectiveness of these representations for both flat and multi-level segmentation and show that they can accommodate structural annotations of varying styles and levels.

1.2 Related work

The method proposed here builds upon recent work in music structure analysis devoted to finding efficient representations using deep learning methods to improve already existing downstream algorithms. The work by McCallum [3]

© M. Buisson, B. McFee, S. Essid and H. C. Crayencour. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. Buisson, B. McFee, S. Essid and H. C. Crayencour, "Learning Multi-Level Representations for Hierarchical Music Structure Analysis", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

proposes an unsupervised method to learn deep features using a triplet-based approach. It relies on the assumption that frames temporally close to each other are more likely to belong to the same musical section than those separated by a certain amount of time. Therefore, triplets are sampled in such a way that the temporal distance between the anchor point and the positive example is smaller than the distance separating the anchor from the negative example. Wang et al. [4] adopt a similar approach by using structural annotations in a supervised fashion to mine informative sets of frames.

One of the main challenges in estimating the structure of a musical piece is to account for the different temporal levels at which it can be decomposed. Up to now, only a few approaches have been proposed for the task of multi-level segmentation. McFee and Ellis [9] use spectral clustering to decompose an enhanced self-similarity matrix and produce segmentations at different temporal levels. This approach is later improved by Tralie and McFee [10] where the input self-similarity matrix is obtained by combining different features using Similarity Network Fusion. Salamon et al. [5] further extend this method by employing two types of deep embeddings along with CQT features. They capture local timbral patterns with few shot-learning and long-term similarities with disentangled deep metric learning [11]. While these works demonstrate the advantage of combining multiple representations of a same signal to extract meaningful structural patterns, our approach shows instead that these can be directly encoded into the representations using time proximity and the hierarchy of music structure.

2. HIERARCHICAL REPRESENTATIONS

The method introduced here constructs deep representations which allow for structural segmentations at various time-scales. To facilitate the decomposition of a song at different levels, these representations should provide strong discriminative capabilities for time frames belonging to different musical sections and separated by a large amount of time. Conversely, they should be more homogeneous for frames belonging to the same section and happening within a short time interval. As section lengths might vary from one annotator to another due to the ambiguity of the task [1], the aforementioned constraint is imposed at different temporal scales. Additionally, most datasets for music structure analysis come with only one level of annotations, which motivates us to learn such representations in an unsupervised fashion, taking advantage of large quantities of unlabelled data. A base convolutional neural network is used to output embeddings which are divided into multiple sub-regions. Each of them is optimized independently using specific triplets of frames efficiently sampled to encode the temporal structure of the song at different levels. We show that each level of the final representations can model frames proximity with its own amount of granularity.

2.1 Sampling

The objective of the sampling method introduced by McCallum [3] is to build triplets of frames where the anchor and the positive example belong to the same musical section, while the anchor and the negative example are labelled differently. The method proposed here can be viewed as its multi-level extension. A hierarchy is defined as a set of L levels of structural segmentations ordered from the coarsest to the most refined. For each level $\ell \in \{0; \dots; L-1\}$ in the hierarchy, triplets of beat indices are sampled using a specific set of parameters $\delta = \{\delta^\ell_{p,min}, \delta^\ell_{p,max}, \delta^\ell_{n,min}, \delta^\ell_{n,max}\}$. Intuitively, they rule how "close" or "far away" from the anchor the positive and negative examples will be sampled throughout the song. More specifically, for a given anchor beat index i_a, positive and negative examples respectively located at beat indices i_p and i_n are uniformly sampled from the interval I^ℓ_p defined by $\delta^\ell_{p,min}$ and $\delta^\ell_{p,max}$ and I^ℓ_n specified by $\delta^\ell_{n,min}$ and $\delta^\ell_{n,max}$. An example for an arbitrary level ℓ is shown in Figure 1. The δ parameters actually define a notion of temporal distance d_ℓ between frames at a given level of the hierarchy (analogous to a notion of dissimilarity). Therefore, the set of triplets T_ℓ at level ℓ can be expressed as:

$$T_\ell = \{(i_a, i_p, i_n; l) \mid d_\ell(x_a, x_p) < d_\ell(x_a, x_n)\} \quad (1)$$

where x_k is the input feature patch observed at beat index i_k, $i^\ell_p \sim \mathcal{U}(I^\ell_p)$ and $i^\ell_n \sim \mathcal{U}(I^\ell_n)$.

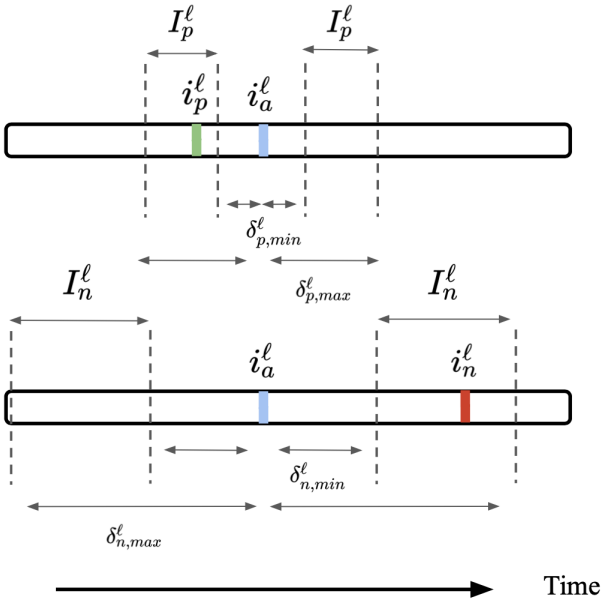


Figure 1. Initial triplet sampling method at level ℓ.

In order for the learned hierarchy levels to remain consistent with one another, monotonicity is encouraged by modifying the initial triplet mining technique. In addition to the time constraint imposed on triplets of the same level, their probability of being sampled is restricted from one level in the hierarchy to the next. For a randomly sampled anchor index at level $\ell = 0$, a complete triplet (i^a_0, i^p_0, i^n_0) is built only using time proximity (*i.e.* δ parameters). Then

for each level $\ell \in \{1; \ldots; L-1\}$, the positive example is sampled closer and closer to the same anchor (*i.e.* $\delta^{\ell}_{p,min}$ and $\delta^{\ell}_{p,max}$ decrease), whereas the negative is obtained by selecting the positive example from level $\ell - 1$. This way, going deeper into the hierarchy means that the representations get more refined to detect short-term musical patterns. The modified sampling method is summarized in Figure 2. The process is then repeated by starting over from level 0, going down the hierarchy with the same anchor index, transferring the negative example from the current to the next level, and uniformly sampling the positive ones using the right δ parameters. At the end, the whole training set for all levels of the hierarchy is given by combining every set of triplets T_ℓ level-wise: $T = \{T_\ell\}_{\ell=0}^{L-1}$.

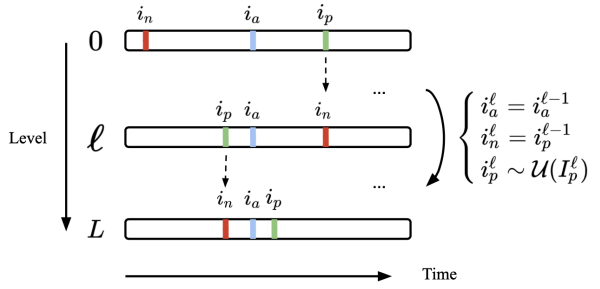


Figure 2. Modified triplet sampling, moving downwards in the hierarchy.

2.2 Disentangled hierarchy levels

During training, the model is shown triplets sampled at different hierarchy levels and should optimize the corresponding sub-regions of the output embeddings. We adapt the method introduced by Veit et al. [12], called Conditional Similarity Networks. This method has already proven to be efficient in the context of multi-dimensional music similarity learning [11], where a joint model learns compact representations of music audio signals complying with different similarity criteria, namely genre, mood, instrumentation and tempo. We propose to extend it to the hierarchical case: to model the different temporal distances d_ℓ, a set of L masking functions $m_\ell \in \{0,1\}^n$ that are applied to the embedding space of size n is defined. Each mask can be interpreted as an element-wise gating function selecting the relevant dimensions of the embedding corresponding to a particular level of the hierarchy. For a given triplet (x_a, x_p, x_n) at level ℓ, the training objective becomes:

$$\mathcal{L}(x_a, x_p, x_n) = [D_\ell(x_a, x_p) - D_\ell(x_a, x_n) + \alpha]_+, \quad (2)$$

$$D_\ell(x_i, x_j) = \| \, m_\ell \circ [f(x_i) - f(x_j)] \, \|_2^2 \quad (3)$$

where $\circ$ is the Hadamard product, $[.]_+$ denotes the Hinge loss, α the margin parameter and $f(x)$ is the projection of x into the embedding space by the convolutional neural network. An example is illustrated in Figure 3, where $L = 3$ and $\ell = 1$. Since going deeper into the hierarchy results in triplets of frames getting temporally closer to each other, it is unnecessary for the model to separate samples by the same distance margin at all levels. Therefore, margin values were evenly distributed within the range $[0.05, 0.1]$ so that for each level $\ell \in \{0; \ldots; L-2\}$, we have $\alpha_\ell > \alpha_{\ell+1}$.

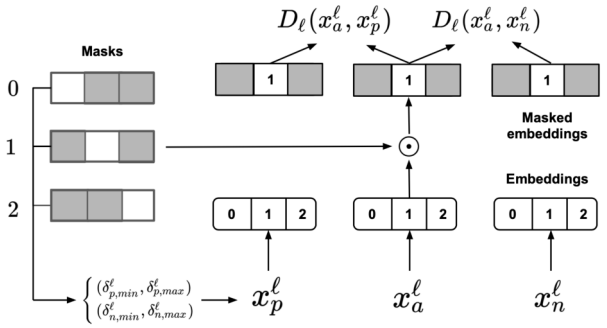


Figure 3. Training pipeline for $\ell = 1$ and $L = 3$. At each iteration, the current hierarchy level defines the set of δ parameters to sample the positive example. The mask here conserves the sub-region corresponding to level $\ell = 1$.

3. EXPERIMENTS

The evaluation of our method is divided into three distinct parts. First, we consider the problem of boundary detection on flat annotations using the SALAMI dataset. Second, we verify if the learned hierarchical representations improve multi-level segmentation predictions on that same dataset using the two-level structural annotations available. We finally demonstrate the flexibility of our approach and provide additional results on other commonly used datasets for music structure analysis where their original flat annotations have been automatically expanded beforehand [13].

3.1 Datasets

We use five different datasets in our evaluation:

SALAMI: the Structural Annotations for Large Amounts of Music Information (SALAMI) [8] is the most substantial dataset for music structure analysis. It contains $1,359$ tracks ranging from classical, jazz, popular to world and live music. Each track is provided with two levels of structural annotations. We use a subset of 884 songs labelled by two different annotators. Therefore, for each track contained in this subset, we end up with a total of 4 segmentation ground-truths (2 annotators $\times$ 2 levels of granularity). In the rest of this work, this subset is referred as SALAMI.

BeatlesTUT: a revised version of 174 annotated Beatles songs, originally released in the Isophonics dataset [14] and corrected by researchers from Tampere University of Technology.

RWC-Pop: the Popular subset of the RWC dataset [15] contains 100 songs with section annotations. Note that two versions of these annotations are available online; here the ones originally provided by the authors (AIST) are used.

RWC-Jazz: the Jazz subset of the RWC dataset [15] is composed of 50 songs from various Jazz sub-genres such as Vocal, Big Band, Modal, Funky, Free or Fusion Jazz.

JAAH: the Audio-aligned jazz harmony dataset (JAAH) [16] is composed of 113 tracks selected from "The Smithsonian Collection of Classic Jazz" and "Jazz: The Smithsonian Anthology", covering various performers, sub-genres and historical periods.

3.1.1 Obtaining multi-level annotations:

For all datasets but SALAMI, we apply automatic hierarchy expansion [13] before evaluating multi-level segmentation. As can be seen in the descriptive statistics from Table 1, both the distributions of section labels and segment durations vary from one dataset to another. This difference can either be explained by the style of annotations (*i.e.* label taxonomy, desired level of detail...) or the music genre. As a consequence, the average number of levels obtained after automatic hierarchy expansion is dependent on the repetition of section labels and their semantic structure, which varies with the annotation process as well.

Dataset	N	Uni	Seg	Dur	Levels
SALAMI^0 (upper)	884	5.3	10.9	63.5	2.0
SALAMI^0 (lower)	884	10.0	33.1	18.4	2.0
SALAMI^1 (upper)	884	5.0	11.2	61.1	2.0
SALAMI^1 (lower)	884	9.2	34.1	18.0	2.0
BeatlesTUT	174	5.6	10.1	36.1	2.5
RWC-Pop	100	8.9	16.4	28.5	2.9
RWC-Jazz	50	14.2	19.9	32.1	2.8
JAAH	113	6.2	8.0	63.1	2.0

Table 1. Datasets descriptive statistics. N: number of annotated songs. Uni: average number of unique section labels per song. Seg: average number of segments per song. Dur: average duration of each section per song (in beats). Levels: average number of annotation levels per song after automatic hierarchy expansion. SALAMI^i: ith annotator.

3.1.2 Training data:

Since this work falls under the scope of unsupervised learning, a non annotated external audio collection is used for training. It is composed of $23,725$ tracks, spanning various musical genres such as rock, popular, rap, jazz, electronic or classical. These were retrieved from publicly available playlists and the audio obtained from Youtube. Care has been taken to discard any track from this external collection also present in one of the testing datasets.

3.2 Evaluation metrics

3.2.1 Flat segmentation:

For boundary detection, we report the F-measure [1] of the trimmed boundary detection hit-rate with a 3-second tolerance window (F_3) on the original annotations. We also report the F-measure of frame pairwise clustering [18] (F_{pairwise}), which gives another view on flat segmentation performance in terms of frame-wise section assignment.

[1] All evaluations are done using the mir_eval package [17].

3.2.2 Multi-level segmentation:

The second part of the evaluation on multi-level segmentation is carried out using the L-measure [7]. This metric allows for comparing hierarchies of segmentations operating at different scales. First, the reference hierarchy H^R is decomposed into a finite number of time instants (*i.e.* frames). Then, the set $A(H^R)$ of all triplets of frames (i, j, k) such that i and j receive the same label deeper in the hierarchy than i and k is retrieved. The same process is repeated with the same set of time instants for the estimated hierarchy H^E to obtain $A(H^E)$. Finally, the L-precision, L-recall and L-measure are derived by comparing $A(H^R)$ against $A(H^E)$. As noted in previous work [5, 10], hierarchies estimated with greater depth than reference annotations can make the L-precision metric unreliable. Therefore, our evaluation focuses on the L-recall, indicating how much of the reference hierarchy is retrieved in the estimated one. For this part of the evaluation, the expanded version of each dataset is used except for SALAMI, where for comparison purposes, the reference hierarchy only comprises both of the original annotation levels provided by each annotator (*upper* and *lower*).

3.3 Input features

All tracks are resampled at 22.05 kHz. Previous work has demonstrated that homogeneous regions and sharp changes of timbral content can be a good indicator of section transitions [19]. Therefore, we use log-scaled mel-spectrograms, with a window and hop size of 2048 and 256 respectively. We compute 60 mel-bands per frame. Beats are estimated for all tracks using the Librosa [20] implementation of the beat tracking algorithm from Ellis [21]. For both feature types, patches of 512 frames ($\simeq 5.94$s) are observed, centered at each detected beat location.

3.4 Implementation details

3.4.1 Network architecture:

We use a basic convolutional neural network architecture composed of 3 convolutional blocks, each comprising a convolutional and a max-pooling layer and Relu activation, followed by two fully-connected layers with Relu activations and a third fully-connected layer with linear activation. All convolutional layers use a kernel size of $(6, 4)$. A common practice in contrastive learning is to constrain the learned representations to lie within the unit hypersphere [22]. Therefore, the output embeddings are L2-normalized prior to distance calculations. The models were implemented with Pytorch 1.7.1 [23]. The RMSProp optimizer with default parameters is used. All models are trained on the non-annotated external audio collection described in Section 3.1 for a maximum of 200 epochs. The learning rate is set to 10^{-4} and dropout [24] is applied with probability 0.1 after each convolutional block and 0.2 after each fully-connected layer. All models [2] return embeddings of dimension $n = 128$.

[2] Code: `github.com/morgan76/HE`

3.4.2 Masks design:

In previous work, it was found beneficial to learn the masks during training to promote information sharing across similarity dimensions [12]. As in the method proposed by Lee et al. [11], we found that this did not bring any major improvement. Since information is already shared implicitly among the different hierarchy levels by the sampling strategy detailed in Section 2.1, the masks are kept disjoint from one another with equal length. After some preliminary experiments, the number of hierarchy levels $L = 4$ has been found as a good compromise between the diversity of triplets at each level and the temporal scale between the top and bottom ones.

3.4.3 Batch sampling scheme:

During training, mini-batches of size 120 are composed of 10 anchor points uniformly sampled from one song, and from which 12 triplets are derived (3 for each level). To choose good sampling parameters, we used both annotation levels of the held-out subset of SALAMI and measured the amount of true positive and true negative examples while varying $\delta_{p,min}$ and $\delta_{p,max}$ of level $\ell = 0$. It was found that setting $\delta_{p,min} = 32$ and $\delta_{p,max} = 64$ provided a good balance between the true positives rate at level $\ell = 0$ and the true negatives rate at level $\ell = 1$. For the case where $L = 4$, the rest of the parameters were set such that each level spans the same duration in beats (i.e. 16 beats) under the maximum value of 64 beats. All sampling parameters δ used for each level are summarized in Table 2.

L	ℓ	$\delta_{p,min}$	$\delta_{p,max}$	$\delta_{n,min}$	$\delta_{n,max}$
1	0	1	16	1	128
4	0	48	64	64	128
	1	32	48	48	64
	2	16	32	32	48
	3	1	16	16	32

Table 2. Sampling parameters (in beats) used in our experiments for $L = 1$ and $L = 4$ hierarchy levels.

3.5 Downstream algorithms and baselines

A common way of evaluating deep representations for music structure analysis is to measure the improvement made when combined with downstream segmentation methods. While there exists a variety music segmentation algorithms in the literature [1, 2], the one employed in these experiments was chosen to facilitate comparison against previous work. Boundary detection and section grouping on flat annotations as well as multi-level segmentation are performed with spectral clustering [9], as it remains the only unsupervised method that can output multiple levels of segmentation while being competitive. Additionally, it appears as a well-suited downstream method for hierarchical features since it operates on a graph decomposition of the audio signal. The proposed triplet sampling method forces the learned features to discriminate frames temporally close to one another at different levels in the hierarchy. Consequently, each sub-region in the embeddings learns one possible decomposition of the song. Applied on each of these sub-regions, spectral clustering can take advantage of the graph sub-structures proper to each level in order to efficiently retrieve the overall structure of the song. The original algorithm [9] takes two distinct audio features as input (MFCC and CQT), here, both features are replaced by the representations proposed in this work. Results obtained with the whole embedding matrices are denoted by HE (Hierarchical Embeddings). As an upper-bound of the proposed system, section grouping and multi-level segmentation are also performed using each individual sub-region of the embeddings (*i.e.* hierarchy levels), and the best results obtained across levels (denoted by HE_{best}) are reported. In a use case scenario, this can be seen as selecting the most adapted level of representation for each track in the testing set given a desired amount of granularity. For SALAMI, boundary detection is performed per annotator. For each, the scores obtained on both annotation levels (*upper* and *lower*) are computed both separately and combined together (best score between both levels per annotator is kept, noted *combined*). As an example, "$\text{HE}_{0,\text{best}}$" corresponds to the score obtained for the first annotator, selecting for each track the embedding level which maximizes the metric considered. In addition to results from previous work [5,9], those obtained here are compared against the method proposed by McCallum [3] (which comes down to setting $L = 1$ as described in Table 2), it is denoted as FE (Flat Embeddings).

4. RESULTS

4.1 Flat segmentation

Flat segmentation results on SALAMI are given in Table 3. The representations proposed in this work yield competitive results against the reported baselines on all the metrics considered. This trend is accentuated when the best embedding sub-region is selected. For *lower* annotations, the learned representations improve over traditional features. However, they do not perform better than those from McCallum [3], since this method uses sampling parameters that are more adapted to this level of annotation. The best-level scenario shows that the smallest temporal scales used during training (levels $\ell = 2, 3$) allow for the detection of very small regions of homogeneous timbral content, which helps detecting section changes at this level of annotation. The higher pairwise clustering scores indicate that these small detected regions are homogeneous enough to be identically labelled with spectral clustering (k-means step).

For the *upper* annotations, the results for boundary detection and pairwise clustering constantly improve over the reported baselines, indicating that for higher levels in the hierarchy, the proposed representations improve homogeneity inside annotated sections. Long-term similarities are implicitly captured by the highest embedding levels ($\ell = 0, 1$), yielding discriminative features able to separate consecutive musical sections at that level.

Finally, for both annotation levels combined, all models

perform better than when considering each level independently. The fact that difficult examples at the *lower* level are better managed at the *upper* one and vice-versa indicates that the representations learned are not specific to any particular annotation level. The very small performance gap across annotators also shows that these same representations capture relevant structure characteristics that are shared between them.

Level	Method	F_3	$F_{pairwise}$
lower	LSD [9]	0.525 ± 0.19	0.561 ± 0.16
	FE_0 [3]	0.624 ± 0.14	0.561 ± 0.14
	HE_0	0.611 ± 0.16	0.580 ± 0.15
	$HE_{0,best}$	0.643 ± 0.15	0.580 ± 0.15
	FE_1 [3]	0.611 ± 0.14	0.563 ± 0.14
	HE_1	0.600 ± 0.15	0.581 ± 0.14
	$HE_{1,best}$	0.635 ± 0.15	0.580 ± 0.14
upper	SNF [10]	0.456	0.567
	DEF [5]	0.564	0.600
	LSD [9]	0.579 ± 0.15	0.652 ± 0.13
	FE_0 [3]	0.568 ± 0.17	0.694 ± 0.14
	HE_0	0.597 ± 0.18	0.714 ± 0.14
	$HE_{0,best}$	0.627 ± 0.16	0.719 ± 0.14
	FE_1 [3]	0.559 ± 0.17	0.697 ± 0.14
	HE_1	0.595 ± 0.18	0.718 ± 0.14
	$HE_{1,best}$	0.625 ± 0.16	0.720 ± 0.14
combined	HE_0	0.665 ± 0.13	0.730 ± 0.14
	$HE_{0,best}$	0.711 ± 0.12	0.733 ± 0.14
	HE_1	0.662 ± 0.13	0.731 ± 0.14
	$HE_{1,best}$	0.707 ± 0.12	0.731 ± 0.14

Table 3. Boundary detection and section grouping results on SALAMI.

4.2 Multi-level segmentation

The results obtained for multi-level segmentation are reported in Table 4. When employing the full embedding representation, the performance on multi-level segmentation is competitive in terms of L-recall with previous work. As well as for boundary detection, selecting the best embedding sub-region leads to even further improvement. The importance of keeping inter-annotator agreement as a reference for comparison in multi-level segmentation has previously been argued [10]. It is found that the proposed representations result in multi-level segmentations that adapt to both annotators, within the range of the inter-annotator agreement reported by Tralie and McFee [10].

Method	L-precision	L-recall	L-measure
Inter-annot	0.664	0.664	0.654
LSD [7]	0.419	0.636	0.498
SNF [10]	0.431	0.668	0.517
DEF [5]	0.435	0.673	0.520
FE_0 [3]	0.412 ± 0.10	0.677 ± 0.13	0.505 ± 0.11
HE_0	0.413 ± 0.11	0.680 ± 0.13	0.507 ± 0.11
$HE_{0,best}$	0.432 ± 0.11	0.694 ± 0.13	0.527 ± 0.11
FE_1 [3]	0.413 ± 0.10	0.663 ± 0.12	0.503 ± 0.10
HE_1	0.418 ± 0.11	0.671 ± 0.13	0.509 ± 0.11
$HE_{1,best}$	0.423 ± 0.11	0.686 ± 0.13	0.517 ± 0.11

Table 4. Multi-level segmentation results on SALAMI. Inter-annot denotes the inter-annotator agreement.

4.3 Additional evaluation

In Table 5, the results obtained for boundary detection, section grouping and multi-level segmentation on additional datasets using the whole embedding matrices are summarized. The boundary detection scores obtained for BeatlesTUT and RWC-Pop fall within the same range, where more specifically, the score on RWC-Pop is higher than the one obtained by Wang et al. [4] with representations learned via supervised contrastive learning. However, a significant drop is observed for the two remaining datasets: RWC-Jazz and JAAH. If the music genre might play a role in this performance gap, it is also worth considering some statistics of these datasets summarized in Table 1. For RWC-Jazz, the high number of unique section labels compared to the total number of segments might cause some errors during the section grouping step done at the frame level with k-means (last step of the spectral clustering method). Regarding the JAAH dataset, given the low average number of segments per track, the segmentation method returns more boundaries than those originally annotated, therefore reducing the hit-rate precision and F-measure. For all metrics considered, other experiments have shown that hierarchical representations also performed better than their flat counterparts [3], of which due to space constraints, the results are not reported here.

Dataset	F_3	$F_{pairwise}$	L-P	L-R	L-M
BeatlesTUT	71.77	72.25	49.32	75.25	59.37
RWC-Pop	68.07	65.35	47.02	77.06	58.30
RWC-Jazz	55.05	58.51	32.89	81.80	45.76
JAAH	55.57	76.72	46.49	81.18	58.55

Table 5. Boundary detection, section grouping and multi-level segmentation results on additional datasets (in percentage) with the whole embedding matrix. L-P: L-precision, L-R: L-recall, L-M: L-measure.

The L-recall values obtained across datasets remain within the same range, regardless of the performance achieved on flat segmentation or section grouping. The temporal notion induced during sampling helps adapting to different musical genres or annotation sources. Even though the learned representations may not always fit with one specific level in the annotations, most of the reference structure hierarchies are captured and more refined levels of segmentation are discovered.

5. CONCLUSION

In this work, unsupervised contrastive learning of deep representations for music structure analysis at different time-scales has been explored. By leveraging time information and the hierarchical aspect of music structure, the resulting representations facilitate single and multi-level segmentation while being robust against different types of annotations. Future work includes searching for better-suited architectures to detect musical patterns at different time scales and automatically combine them to accommodate specific annotation styles or levels.

6. REFERENCES

[1] O. Nieto, G. J. Mysore, C.-i. Wang, J. B. Smith, J. Schlüter, T. Grill, and B. McFee, "Audio-based music structure analysis: Current trends, open challenges, and applications," *Transactions of the International Society for Music Information Retrieval*, 2020.

[2] O. Nieto and J. P. Bello, "Systematic exploration of computational music structure research." in *ISMIR*, 2016.

[3] M. C. McCallum, "Unsupervised learning of deep features for music segmentation," in *ICASSP*, 2019.

[4] J.-C. Wang, J. B. L. Smith, W. T. Lu, and X. Song, "Supervised metric learning for music structure features," in *ISMIR*, 2021.

[5] J. Salamon, O. Nieto, and N. J. Bryan, "Deep embeddings and section fusion improve music segmentation," in *ISMIR*, 2021.

[6] J. B. Smith and E. Chew, "Using quadratic programming to estimate feature relevance in structural analyses of music," in *Proceedings of the 21st ACM international conference on Multimedia*, 2013.

[7] B. McFee, O. Nieto, M. M. Farbood, and J. P. Bello, "Evaluating hierarchical structure in music annotations," *Frontiers in psychology*, 2017.

[8] J. B. L. Smith, J. A. Burgoyne, I. Fujinaga, D. De Roure, and J. S. Downie, "Design and creation of a large-scale database of structural annotations." in *ISMIR*, 2011.

[9] B. McFee and D. Ellis, "Analyzing song structure with spectral clustering." in *ISMIR*, 2014.

[10] C. J. Tralie and B. McFee, "Enhanced hierarchical music structure annotations via feature level similarity fusion," in *ICASSP*, 2019.

[11] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Disentangled multidimensional metric learning for music similarity," in *ICASSP*, 2020.

[12] A. Veit, S. Belongie, and T. Karaletsos, "Conditional similarity networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

[13] B. McFee and K. M. Kinnaird, "Improving structure evaluation through automatic hierarchy expansion," in *ISMIR*, 2019.

[14] M. Mauch, C. Cannam, M. Davies, S. Dixon, C. Harte, S. Kolozali, D. Tidhar, and M. Sandler, "Omras2 metadata project 2009," in *ISMIR*, 2009.

[15] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "Rwc music database: Popular, classical and jazz music databases." in *ISMIR*, 2002.

[16] V. Eremenko, E. Demirel, B. Bozkurt, and X. Serra, "Audio-aligned jazz harmony dataset for automatic chord transcription and corpus-based research." in *ISMIR*, 2018.

[17] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *ISMIR*, 2014.

[18] M. Levy and M. Sandler, "Structural segmentation of musical audio by constrained clustering," *IEEE transactions on audio, speech, and language processing*, 2008.

[19] F. Kaiser and G. Peeters, "A simple fusion method of state and sequence segmentation for music structure discovery." in *ISMIR*, 2013.

[20] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, 2015.

[21] D. P. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, 2007.

[22] T. Wang and P. Isola, "Understanding contrastive representation learning through alignment and uniformity on the hypersphere," in *ICML*, 2020.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, 2019.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, 2014.

MULTI-INSTRUMENT MUSIC SYNTHESIS WITH SPECTROGRAM DIFFUSION

Curtis Hawthorne **Ian Simon** **Adam Roberts** **Neil Zeghidour**
Josh Gardner* **Ethan Manilow*** **Jesse Engel**
Google Research, Brain Team
{fjord,iansimon,adarob,neilz,jesseengel}@google.com
jpgard@cs.washington.edu, ethanm@u.northwestern.edu

ABSTRACT

An ideal music synthesizer should be both interactive and expressive, generating high-fidelity audio in realtime for arbitrary combinations of instruments and notes. Recent neural synthesizers have exhibited a tradeoff between domain-specific models that offer detailed control of only specific instruments, or raw waveform models that can train on any music but with minimal control and slow generation. In this work, we focus on a middle ground of neural synthesizers that can generate audio from MIDI sequences with arbitrary combinations of instruments in realtime. This enables training on a wide range of transcription datasets with a single model, which in turn offers note-level control of composition and instrumentation across a wide range of instruments. We use a simple two-stage process: MIDI to spectrograms with an encoder-decoder Transformer, then spectrograms to audio with a generative adversarial network (GAN) spectrogram inverter. We compare training the decoder as an autoregressive model and as a Denoising Diffusion Probabilistic Model (DDPM) and find that the DDPM approach is superior both qualitatively and as measured by audio reconstruction and Fréchet distance metrics. Given the interactivity and generality of this approach, we find this to be a promising first step towards interactive and expressive neural synthesis for arbitrary combinations of instruments and notes.

1. INTRODUCTION

Neural audio synthesis of music is a uniquely difficult problem due to the wide variety of instruments, playing styles, and acoustic environments. Among current approaches, there is often a trade-off between interactivity and generality. Interactive models, such as DDSP [3,4], offer realtime synthesis and fine-grained control, but only for specific types of instruments with domain-specific training information. On the other hand, models like Jukebox [5] are much more general and allow for training on any type of music, but are several orders of magnitude slower than realtime and offer more limited and global control, such as lyrics and global style (though with some early MIDI conditioning experiments).

Natural language generation and computer vision have seen dramatic recent progress through scaling generic encoder-decoder Transformer architectures [6, 7]. MT3 [8, 9] demonstrated that such an approach can be adapted to Automatic Music Transcription, transforming spectrograms to variable-length sequences of notes from arbitrary combinations of instruments. This generic approach enables training a single model on a wide variety of datasets: anything with paired audio and MIDI examples.

In this paper, we explore the inverse problem: finding a simple and general method to transform variable-length sequences of notes from arbitrary combinations of instruments into spectrograms. By pairing this model with a spectrogram inverter, we find that we are able to train on a wide variety of datasets and synthesize audio with note-level control over both composition and instrumentation.

To summarize, our contributions include:

- Demonstrating that the generic encoder-decoder Transformer approach used for multi-instrument transcription can likewise be adapted for multi-instrument audio synthesis.
- Realtime synthesis enabling interactive note-level control over both composition and instrumentation, by pairing a MIDI-to-spectrogram Transformer model with a GAN spectrogram inverter, and training on a wide range of paired audio/MIDI datasets (synthetic and real) with diverse instruments.
- Adapting DDPM decoding for arbitrary length synthesis without edge artifacts, through additional autoregressive conditioning on segments ($\sim$5 seconds) of previously generated audio.
- Quantitative metrics and qualitative examples demonstrating the advantages of segment-wise diffusion decoders over frame-wise autoregressive decoders for continuous spectrograms.

* Work done as a Google Brain Student Researcher.

© C. Hawthorne, I. Simon, A. Roberts, N. Zeghidour, J. Gardner, E. Manilow, and J. Engel. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** C. Hawthorne, I. Simon, A. Roberts, N. Zeghidour, J. Gardner, E. Manilow, and J. Engel, "Multi-instrument Music Synthesis with Spectrogram Diffusion", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

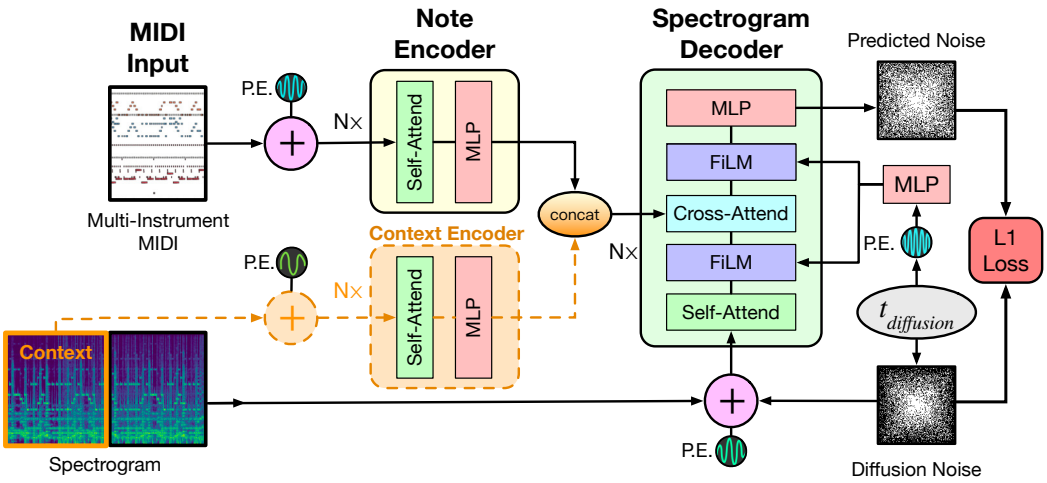


Figure 1. Training configuration for our spectrogram diffusion model. The architecture is an encoder-decoder Transformer that takes a sequence of note events as input and outputs a spectrogram. We train the decoder stack as a Denoising Diffusion Probabilistic Model (DDPM) [1], where the model learns to iteratively refine Gaussian noise into a target spectrogram (Figure 2). We generate ∼5 second spectrogram segments, and to ensure a smooth transition between these segments we (optionally) encode the previously generated segment in a second encoder stack. At inference time, a generated spectrogram is inverted to a waveform using a model similar to MelGAN [2]. "P.E." means Positional Encoding.

- Source code and pretrained models [1] .

2. RELATED WORK

Neural audio synthesis first proved feasible with autoregressive models of raw waveforms such as WaveNet and SampleRNN [10, 11]. These models were adapted to handle conditioning from latent variables [12, 13] or MIDI notes [14–17], but suffer from slow generation due to needing to run a forward pass for every sample of the waveform. Real-time synthesis often requires the use of specialized CPU and GPU kernels [18, 19]. These models also have limited temporal context due to the high sample rate of audio (e.g., 16kHz or 48kHz), where even the largest receptive fields (thousands of timesteps) can equate to less than a second of audio.

Approaches to overcome these speed limitations of waveform autoregression have focused on generating audio directly with a single forward pass. Architectures commonly either use GANs [20–23], controllable differentiable DSP elements such as oscillators and filters [3,4,24–27], or both [28, 29]. These models often focus on limited domains such as generating a single instrument/note/voice at a time [4, 15, 16, 30]. Here, we focus our search on architectures capable of both realtime synthesis, note-level control, and synthesizing mixtures of multiple polyphonic instruments at once.

Researchers have also overcome temporal context limitations of waveform autoregression by adopting a multi-stage approach, first creating coarser audio representations at a lower sample rate, and then modeling those representations with a predictive model before decoding back into audio. For example, Jukebox and Soundstream [5,31,32] use a Transformer to autoregresively model the discrete vector-quantized codes [33] of a base waveform autoencoder.

Tacotron architectures [34–36] have demonstrated that straightforward spectrograms can be an effective audio representation for multi-stage generation, first autoregressively generating continuous-valued spectrograms, and then synthesizing waveforms with a neural vocoder. The success of this approach led to a flurry of research into spectrogram inversion models, including streamlined waveform autoregression, GANs, normalizing flows, and Denoising Diffusion Probabilistic Models (DDPMs) [1, 2, 18, 37–39]. Our approach here is inspired by Tacotron. We use an autoregressive spectrogram generator paired with a GAN spectrogram inverter as a baseline, and further improve upon it with a DDPM spectrogram generator.

DDPMs and Score-based Generative Models (SGMs) have proven well-suited to generating audio representations and raw waveforms. Speech researchers have demonstrated high-fidelity spectrogram inversion [38, 39], text-to-speech [40, 41], upsampling [42], and voice conversion [43, 44]. Musical applications include singing synthesis of individual voices [45, 46], drums synthesis [47], improving the quality of music recordings [48], symbolic note generation [49], and unconditional piano generation [50]. Similar to singing synthesis, here we investigate DDPMs for spectrogram synthesis, but focusing on arbitrary numbers and combinations of instruments.

3. ARCHITECTURE

We approach the problem of audio synthesis using a two-stage system. The first stage consists of a model that produces a spectrogram given a sequence of MIDI-like note events representing any number of instruments. We then use a separate model to invert those spectrograms to au-

[1] https://github.com/magenta/music-spectrogram-diffusion

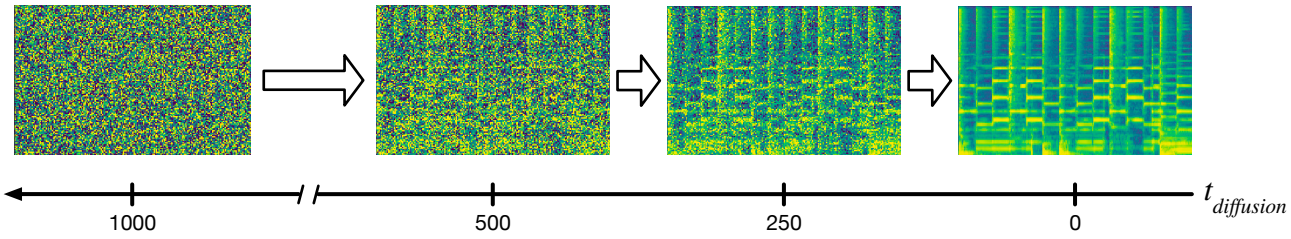


Figure 2. Our decoder stack is trained as a Denoising Diffusion Probabilistic Model (DDPM) [1]. The model starts with Gaussian noise as input and is trained to iteratively refine that noise toward a target, conditioned on a sequence of note events and the spectrogram of the previously rendered segment. This figure illustrates the diffusion process for one example segment.

dio samples. Our model for the first stage is an encoder-decoder Transformer architecture, shown in Figure 1. The encoder takes in a sequence of note events and, optionally, a second encoder uses an earlier part of the spectrogram as context. These embeddings are passed to a decoder, which generates a spectrogram corresponding to the input note sequence. Here, we explore training the decoder autoregressively (Section 3.1) or as a Denoising Diffusion Probabilistic Model (DDPM) [1] (Section 3.2).

Our architecture uses the encoder-decoder Transformer from T5 [51] with T5.1.1 [2] improvements and hyperparameters. We use the same note event vocabulary and note sequence encoding procedure as MT3 [8]. We find that training on a full song is prohibitive in terms of memory and compute due to the quadratic scaling of self-attention with sequence length, so we split note sequences into segments. Specifically, we train models with 2048 input positions for note events and 256 output positions for spectrogram frames. Each spectrogram frame represents 20 ms of audio, so each segment is 5.12 seconds.

Rendering audio segments independently introduces the problem of how to ensure smooth transitions between the segments once they are eventually concatenated together to form a full musical piece. We address this problem by providing the model with the spectrogram from the previously rendered segment, meaning that this model is autoregressive at the segment level. This context segment has its own encoder stack with 256 input positions. As seen in Figure 1, the outputs of both encoder stacks are concatenated together as input for cross-attention in the decoder layers.

We use sinusoidal position encodings, as in the original Transformer paper [52]. However, we found better performance if we decorrelated the position encodings of each network, as they have different meanings. We decorrelate the encodings by applying a unique random channel permutation and phase offset for the sinusoids used in each of the encoder and decoder stacks.

3.1 Autoregressive Decoder

As an initial approach, we took inspiration from Tacotron [34, 35], and trained the decoder as an autoregressive model on the continuous spectrograms. In this setting, standard causal masking was applied throughout the self-attention layers. The inputs and outputs are continuous spectrogram frames, and the model was trained with a mean-squared error (MSE) loss on those frames. Mathematically, this is equivalent to training a continuous autoregressive model with a Gaussian output distribution with isotropic and fixed variance. We also tried training models using mixtures of several Gaussians, but they tended to be unstable during sampling.

For sampling we use a fixed variance (0.2 in units of log magnitude) that is added to the outputs of every inference step. In practice, this "dithering" reduced strong audio artifacts due to overly smooth outputs.

Despite this, these baseline models still produce spectrogram outputs that tend to be blurry and contain jarring artifacts in some frames. We hypothesized an approach enabling incremental, bidirectional refinement, and model dependencies between frequency bins, could help resolve these issues.

3.2 Diffusion Decoder

Taking inspiration from recent successes in image generation such as DALL-E 2 and Imagen [53, 54], we investigated training the decoder as a Denoising Diffusion Probabilistic Model (DDPM) [1].

DDPMs are models that generate data from noise by reversing a Gaussian diffusion process that turns an input signal $\mathbf{x}$ into noise $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. This *forward* process occurs over diffusion timesteps $t \in [0, 1]$ (not to be confused with time for the audio) resulting in noisy data $\mathbf{x_t} = \alpha_t \mathbf{x} + \sigma_t \epsilon$, where $\alpha_t \in [0, 1]$ and $\sigma_t \in [0, 1]$ are noise schedules that blend between the original signal and the noise over the diffusion time. In this work, the decoder, ϵ_θ with parameters θ, is trained to predict the added noise given the noisy data by minimizing an L1 objective,

$$\mathop{\mathbb{E}}_{\mathbf{x},\mathbf{c},\epsilon,t} w_t ||\epsilon_\theta(\mathbf{x_t}, \mathbf{c}, t) - \epsilon||_1^1 \quad (1)$$

where w_t is a loss weighting for different diffusion timesteps and $\mathbf{c}$ is additional conditioning information for the decoder. Schedules for w_t, α_t and ϵ_t are hyperparameters chosen to selectively emphasize certain steps in the reverse diffusion process.

[2] https://github.com/google-research/text-to-text-transfer-transformer/blob/main/released_checkpoints.md#t511

During sampling, we follow the *reverse* diffusion process by starting from pure independent Gaussian noise for each frame and frequency bin and iteratively use noise estimates to generate new spectrograms with gradually decreasing noise content. Full details of this process can be found in Ho et al. [1] and source code is provided for clarity [3]. A visualization of the forward and reverse diffusion process can be seen in Figure 2.

Because the diffusion process expects inputs to be in the range $[-1, 1]$, we scale spectrograms used for training targets and audio context to that range before using them. During inference, we scale model outputs back to the expected spectrogram range before converting them to audio.

During training, we used a uniformly-sampled noise time step in $[0, 1]$ for each training example and run the diffusion process forward to that step with randomly sampled Gaussian noise. The noisy example is then used for input, and the model is trained to predict the Gaussian noise component with Equation (1). During inference, we run the diffusion process in reverse using 1000 linearly spaced steps in $[0, 1]$.

Based on the implementation of Imagen [54], we also incorporated several recent DDPM improvements, including using logSNR [55, 56], a cosine noise schedule [57] of $\cos(t * \pi/2)^2$, and Classifier-Free Guidance [58] during sampling. In particular, we found that using Classifier-Free Guidance during sampling led to samples that were less noisy and more crisp sounding. After a coarse sweep of values, we train with a conditioning dropout probability of 0.1 and sample with a conditioned sample weight of 2.0.

Many diffusion models use a UNet [59] architecture, but in order to stay close to the architecture used in MT3, we used a standard Transformer decoder with no causal masking. We incorporate the noise time information by first converting the continuous time value to a sinusoid embedding using the same method as the position encodings in the original Transformer paper [52] followed by an MLP block. That embedding is then incorporated at each decoder layer using a FiLM [60] layer before the self-attention block and after the cross-attention block. The outputs of note events and spectrogram context encoder stacks are concatenated together and incorporated using a standard encoder-decoder cross-attend layer.

3.3 Spectrograms to Audio

To translate the model's magnitude spectrogram output to audio, we use a convolutional spectrogram inversion network as proposed in MelGAN [2]. In particular, we base our implementation on the more recent SEANet [61] and SoundStream [31]. This model first applies a 1D-convolution with kernel size 3 and stride 1. It then cascades four blocks of layers. Each block is composed of a transposed convolution followed by three residual units, applying a dilated convolution with kernel size 3 and dilation rate 1, 3, and 9 respectively. ELUs [62] are used after each convolution. As in [31], the model is trained with a mix of three loss functions. The first is a multi-scale spectral reconstruction loss, inspired by [63]. The second and third losses are an adversarial loss and a feature matching loss, computed with two discriminators, one STFT-based and one waveform-based. We train this spectrogram inverter on an internal dataset of 16k hours of uncurated music with Adam [64] for 1M steps with a batch size of 128.

The spectrograms used for training both the spectrogram inverter and the synthesis models used audio with a sample rate of 16 KHz, an STFT hop size of 320 samples (20 ms), a frame size of 640, and 128 log magnitude mel bins ranging in frequency from 0 to 8 KHz.

4. DATASETS

Our architecture is flexible enough to train on any dataset containing paired audio and MIDI examples, even when multiple instruments are mixed together in the same track (e.g., individual instrument stems are unavailable). Thus, we are able to use all the same datasets used to train the MT3 transcription model: MAESTROv3 [14] (piano), Slakh2100 [65] (synthetic multi-instrument), Cerberus4 [66] (synthetic multi-instrument), Guitarset [67] (guitar), MusicNet [68] (orchestral multi-instrument), and URMP [69] (orchestral multi-instrument).

We use the same preprocessing of these datasets as MT3, including the data augmentation strategy used for Slakh2100 where 10 random subsets of instruments from each track were selected to increase the number of tracks. We also use the same train/validation/test splits. Our coarse hyperparameter sweeps (e.g., for selecting Classifier-Free Guidance weighting) were done using only validation sets. Results are reported on the test splits, except for Guitarset and URMP which do not have a test split, and so the validation split was used for final results.

The datasets used in these models contain a wide variety of instruments, both synthetic and real. In order to simplify training, we map all MIDI instruments to the 34 Slakh2100 classes plus drums. These mappings cover all General MIDI classes other than "Synth Effects", "Other", "Percussive", and "Sound Effects". As a result, the trained model is capable of synthesizing arbitrary MIDI files while retaining clear instrument identity.

We use the same note event vocabulary as MT3, which is similar to MIDI events. Specifically, there are events for Instrument (128 values), Note (128 values), On/Off (2 values), Time (512 values), Drum (128 values), End Tie Section (1 value), and EOS (1 value). A full description of these events can be found in Section 3.2 of the MT3 paper [8]. Because not all datasets include velocity information, we do not currently include velocity events (as in MT3) and rely on the model's decoder to produce natural sounding velocities given the music context.

Training on such a diverse set of examples gives the model flexibility during synthesis. For example, we can synthesize a track with realistic piano sounds learned from MAESTRO, orchestral instruments from URMP, and synthesizers and drum beats from Slakh.

[3] https://github.com/magenta/music-spectrogram-diffusion

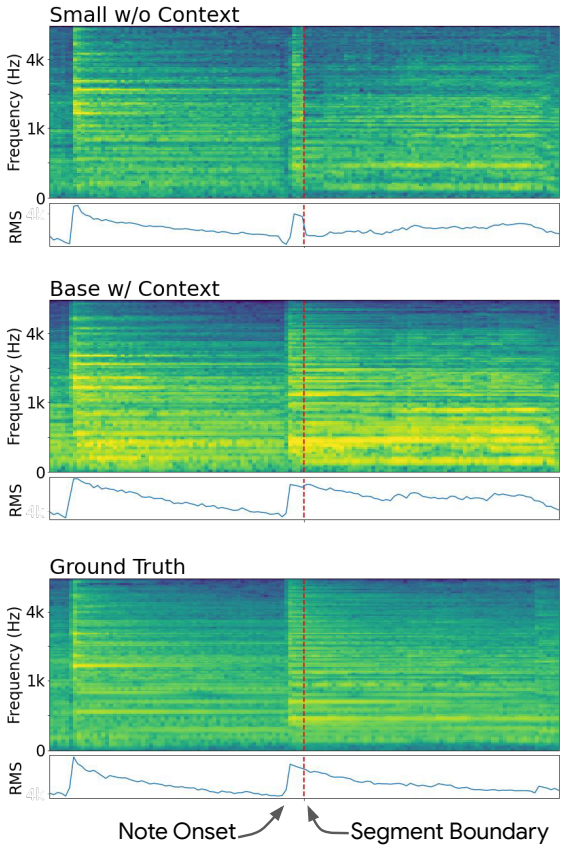


Figure 3. The "Small w/o Context" model (top) does not receive the previously rendered segment as conditioning input and exhibits clear artifacts at segment boundaries (note the dip in RMS after the segment boundary). The "Base w/ Context" model (middle) has access to the previous segment as conditioning information and achieves smooth transition between segments. By repeating the process of rendering a segment and then feeding that segment in as context for the next segment, this model is capable of rendering arbitrary length pieces.

5. EXPERIMENTS

We used the MT3 codebase as a starting point [72], and experiments were implemented using `t5x` for model training and `seqio` for data processing and evaluation [6].

We used the T5.1.1 "small" or "base" Transformer hyperparameters. For "small", there are 8 layers on the encoder and decoder stacks, 6 attention heads with 64 dimensions each, 1024 MLP dimensions, and 512 embedding dimensions. For "base", there are 12 layers on the encoder and decoder stacks, 12 attention heads with 64 dimensions each, 2048 MLP dimensions, and 758 embedding dimensions. We used float32 activations for all models.

Using the datasets described above, we trained four versions of the synthesis model: an 85M parameter "small" autoregressive model with no spectrogram context, an 85M parameter "small" diffusion model with no context, a 104M parameter "small" diffusion model with context, and a 412M parameter "base" diffusion model with context. Training used a batch size of 1024 on 64 TPUv4s with a constant learning rate of $1e^{-3}$ for 500k steps with the Adafactor [73] optimizer. Depending on model size and hardware availability, training took 44–134 hours.

5.1 Metrics

We evaluate model performance on the following metrics:

Reconstruction Embedding Distance For this metric, we pass an individual audio clip and a synthesis model's reconstruction of that audio into a classification network and calculate the distance in the classification network's embedding between the two signals. Here, we report numbers from both VGGish [70] and TRILL [71]. To compute the distance we use the Frobenius norm of the network's embedding layer, averaged over time frames. VGGish outputs 1 embedding per input audio second, and we use the model's output layer as the embedding layer. TRILL outputs $\sim$5.9 embeddings per input audio second, and we use the network's dedicated "embedding" layer.

Fréchet Audio Distance (FAD) FAD measures the perceptual similarity of the distribution of all of the model's output over the entire evaluation set of note events to the distribution of all of the ground truth audio [74].

MT3 Transcription This metric measures how well the synthesis model is reproducing the notes and instruments specified in the input data. We pass the synthesis model output through a pretrained MT3 transcription model and compute an F1 score using the "Full" metric from the MT3 paper as calculated by `mir_eval` [75]. A note is considered correct if has a matching note onset within ± 50 ms, an offset within $0.2 \cdot$ reference_duration or 50 ms (whichever is greater), and has the exact instrument program number as the input data.

Realtime (RT) Factor This measures how fast synthesis is compared to the duration of the audio being synthesized. For example, an RT Factor of 2 means that 2 seconds of audio can be created with 1 second of wall time compute. Inference is performed on a single TPUv4 with a batch size of 1. Here, we include both spectrogram inference and spectrogram-to-audio inversion.

The reconstruction and FAD metrics require embedding both model output and ground truth audio using a model sensitive to perceptual differences. Following the original FAD formulation, we use the VGGish model [70]. To avoid any biases from that particular model, we also compute embeddings using the TRILL model [71]. Pretrained models for computing these embeddings were obtained from the VGGish [4] and TRILL [5] pages on TensorFlow Hub. Due to the large size of the datasets and the high compute and memory requirements for these metrics, we limit metrics calculation to the first 10 minutes of audio for each example.

[4] https://tfhub.dev/google/vggish/1
[5] https://tfhub.dev/google/nonsemantic-speech-benchmark/trill/3

Model	VGGish [70]		TRILL [71]		Transcription F1 ↑	RT Factor ↑
	Recon. ↓	FAD ↓	Recon. ↓	FAD ↓		
Small Autoregressive	3.70	1.03	7.27	0.39	0.31	**10.55**
Small w/o Context	3.49	1.07	6.16	0.46	0.31	2.82
Small w/ Context	3.56	1.10	6.40	0.53	0.25	3.07
Base w/ Context	**3.13**	**1.00**	**2.74**	**0.27**	**0.36**	1.05
Ground Truth Encoded	2.14	0.51	1.54	0.05	0.36	12.25
Ground Truth Raw	0.00	0.00	0.00	0.00	0.63	–

Table 1. Synthesis model experiment results. Metrics are the mean across all evaluation datasets. "Small" and "Base" refer to model capacity (Section 5). "Ground Truth Encoded" refers to the ground truth audio encoded by the spectrogram inverter (Section 3.3) and represents an upper limit on synthesis model performance. "Ground Truth Raw" refers to the unprocessed ground truth audio and represents an upper limit on transcription model performance. Metrics are fully described in Section 5.1 and results are discussed in Section 5.2.

5.2 Results

We first evaluated a "small" autoregressive model (Section 3.1). Initial results were promising, but quantitative and qualitative evaluation of results made it clear that this approach would not be sufficient.

We then investigated using a diffusion approach (Section 3.2). Results from a "small" model were impressive (nearly maxing out the Transcription metric), but we noticed abrupt timbre shifts and audible artifacts at segment boundaries, as illustrated in Figure 3. This makes sense because the synthesis problem from raw MIDI is underspecified: input to the model specifies notes and instruments, but the model has been trained on a wide variety of sources, and the audio corresponding to a given note and instrument combination could just as likely be synthetic or real, played in a dry or reverberant room, played with or without vibrato, etc.; the model has to sample from this large space of possibilities independently for each segment.

To address this issue, we added a context spectrogram input encoder to encourage the model to be consistent over time. We first experimented with adding this context input to a "small" model, but got poor results, possibly due to insufficient decoder capacity for incorporating the additional encoder output. We then tried scaling up to a "base" model that included context. The additional model capacity resulted in generally higher audio quality and also did not have segment boundary problems. This is reflected in the metrics, where this model achieves the best scores by far. Also, even with the larger model size, the synthesis process is still slightly faster than realtime.

Qualitatively, we find that the model does an especially impressive job rendering instruments where the training data came from real audio recordings as opposed to synthetic instruments. For example, when given a note event sequence from the Guitarset validation split, the model not only accurately reproduces the notes but also adds fret and picking noises. Sequences rendered for orchestral instruments such as the ones found in URMP add breath noise and vibrato. The model also does a remarkably good job of rendering natural-sounding note velocities, even though no velocity information is present in our note vocabulary.

Results for these experiments are presented in Table 1. Additional results, including audio examples demonstrating the ability to render out-of-domain MIDI files and modify their instrumentation, are in our online supplement [6] . Results on a per-dataset basis are in Appendix A.

These results are encouraging, but there is still plenty of room for improvement. Particularly, the synthesized audio has occasional issues with inconsistent loudness and audio artifacts, and its overall fidelity does not match the training data. However, in all our experiments, we observed no overfitting on the validation sets, so we suspect that even larger models could be trained for higher fidelity audio.

Another limiting factor of our approach is the spectrogram inversion process, which represents an upper bound on audio quality. This is apparent in the Transcription F1 score, where our models have already achieved the maximum score possible, even though that score is only a little over half of what is achievable with raw audio.

6. CONCLUSION

This work represents a step toward interactive, expressive, and high fidelity neural audio synthesis for multiple instruments. Our flexible architecture allows incorporating any training data with paired audio and MIDI examples. Improved automatic music transcription systems such as MT3 [8] point toward being able to generate high quality MIDI annotations for arbitrary audio, which would greatly expand the available training data and range of instruments and acoustic settings. Using generic Transformer encoder stacks also presents the possibility of utilizing conditioning information beyond note events and spectrogram context. For example, we could add finer grained conditioning such as control over note timbre, or more global controls such as free text descriptions. This model is already slightly faster than realtime, but ongoing research into diffusion models shows promising directions for speeding up inference [56], giving plenty of room for larger and more complicated models to remain interactive. This may be especially important as we explore options for higher quality audio output than our current spectrogram inversion process enables.

[6] Online supplement and examples: `https://bit.ly/3wxSS4l`

7. ACKNOWLEDGEMENTS

We would like to thank William Chan, Mohammad Norouzi, Chitwan Saharia, and Jonathan Ho for help with the diffusion implementation and Sander Dieleman for helpful discussions about diffusion approaches.

8. REFERENCES

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 6840–6851. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf

[2] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brébisson, Y. Bengio, and A. C. Courville, "Melgan: Generative adversarial networks for conditional waveform synthesis," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/6804c9bca0a615bdb9374d00a9fcba59-Paper.pdf

[3] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=B1x1ma4tDr

[4] Y. Wu, E. Manilow, Y. Deng, R. Swavely, K. Kastner, T. Cooijmans, A. Courville, C.-Z. A. Huang, and J. Engel, "Midi-ddsp: Detailed control of musical performance via hierarchical modeling," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=UseMOjWENv

[5] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[6] A. Roberts, H. W. Chung, A. Levskaya, G. Mishra, J. Bradbury, D. Andor, S. Narang, B. Lester, C. Gaffney, A. Mohiuddin *et al.*, "Scaling up models and data with `t5x` and `seqio`," *arXiv preprint arXiv:2203.17189*, 2022.

[7] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan, "Flamingo: a visual language model for few-shot learning," 2022. [Online]. Available: https://arxiv.org/abs/2204.14198

[8] J. P. Gardner, I. Simon, E. Manilow, C. Hawthorne, and J. Engel, "MT3: Multi-task multitrack music transcription," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=iMSjopcOn0p

[9] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel, "Sequence-to-sequence piano transcription with transformers," *arXiv preprint arXiv:2107.09142*, 2021.

[10] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in *9th ISCA Speech Synthesis Workshop*, 2016, pp. 125–125.

[11] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "SampleRNN: An unconditional end-to-end neural audio generation model," *arXiv preprint arXiv:1612.07837*, 2016.

[12] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with wavenet autoencoders," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1068–1077.

[13] N. Mor, L. Wolf, A. Polyak, and Y. Taigman, "A universal music translation network," Tech. Rep., 2018. [Online]. Available: https://ytaigman.github.io/musicnet

[14] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=r1lYRjC9F7

[15] R. Manzelli, V. Thakkar, A. Siahkamari, and B. Kulis, "Conditioning deep generative raw audio models for structured automatic music," *arXiv preprint arXiv:1806.09905*, 2018.

[16] J. W. Kim, R. Bittner, A. Kumar, and J. P. Bello, "Neural music synthesis for flexible timbre control," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 176–180.

[17] E. Cooper, X. Wang, and J. Yamagishi, "Text-to-speech synthesis techniques for midi-to-audio synthesis," 2021. [Online]. Available: https://arxiv.org/abs/2104.12292

[18] N. Kalchbrenner, E. Elsen, K. Simonyan, S. Noury, N. Casagrande, E. Lockhart, F. Stimberg, A. van den Oord, S. Dieleman, and K. Kavukcuoglu, "Efficient neural audio synthesis," in *arXiv*, 2018, pp. 2415–2424.

[19] L. H. Hantrakul, J. Engel, A. Roberts, and C. Gu, "Fast and flexible neural audio synthesis." in *ISMIR*, 2019.

[20] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=ByMVTsR5KQ

[21] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial neural audio synthesis," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=H1xQVn09FX

[22] G. Greshler, T. R. Shaham, and T. Michaeli, "Catch-a-waveform: Learning to generate audio from a single short example," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://openreview.net/forum?id=XmHnJsiqw9s

[23] M. Morrison, R. Kumar, K. Kumar, P. Seetharaman, A. Courville, and Y. Bengio, "Chunked autoregressive gan for conditional waveform synthesis," in *International Conference on Learning Representations (ICLR)*, April 2022.

[24] X. Wang and J. Yamagishi, "Neural harmonic-plus-noise waveform model with trainable maximum voice frequency for text-to-speech synthesis," Tech. Rep., 2019.

[25] J. Engel, R. Swavely, L. H. Hantrakul, A. Roberts, and C. Hawthorne, "Self-supervised pitch detection by inverse audio synthesis," in *ICML 2020 Workshop on Self-supervision in Audio and Speech*, 2020. [Online]. Available: https://openreview.net/forum?id=RlVTYWhsky7

[26] R. Castellon, C. Donahue, and P. Liang, "Towards realistic midi instrument synthesizers," in *NeurIPS Workshop on Machine Learning for Creativity and Design (2020)*, 2020.

[27] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter waveform models for statistical parametric speech synthesis," *arXiv preprint arXiv:1904.12088*, 2019.

[28] A. Caillon and P. Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," *arXiv preprint arXiv:2111.05011*, 2021.

[29] ——, "Streamable neural audio synthesis with non-causal convolutions," 2022.

[30] A. Défossez, N. Zeghidour, N. Usunier, L. Bottou, and F. Bach, "Sing: Symbol-to-instrument neural generator," *Advances in neural information processing systems*, vol. 31, 2018.

[31] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2021.

[32] C. Hawthorne, A. Jaegle, C. Cangea, S. Borgeaud, C. Nash, M. Malinowski, S. Dieleman, O. Vinyals, M. Botvinick, I. Simon *et al.*, "General-purpose, long-context autoregressive modeling with perceiver ar," *arXiv preprint arXiv:2202.07765*, 2022.

[33] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, "Neural discrete representation learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. USA: Curran Associates Inc., 2017, pp. 6309–6318. [Online]. Available: http://dl.acm.org/citation.cfm?id=3295222.3295378

[34] Y. Wang, R. J. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, and Others, "Tacotron: Towards end-to-end speech synthesis," in *Proc. Interspeech*, 2017, pp. 4006–4010.

[35] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, R. A. Saurous, Y. Agiomvrgiannakis, and Y. Wu, "Natural TTS Synthesis by Conditioning Wavenet on MEL Spectrogram Predictions," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2018-April, 2018, pp. 4779–4783. [Online]. Available: https://google.github.io/tacotron/publications/tacotron2.

[36] R. J. Weiss, R. Skerry-Ryan, E. Battenberg, S. Mariooryad, and D. P. Kingma, "Wave-tacotron: Spectrogram-free end-to-end text-to-speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5679–5683.

[37] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 3617–3621.

[38] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "Diffwave: A versatile diffusion model for audio synthesis," in *International Conference on Learning Representations*, 2020.

[39] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "Wavegrad: Estimating gradients for waveform generation," in *International Conference on Learning Representations*, 2020.

[40] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov, "Grad-TTS: A diffusion probabilistic model for text-to-speech," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8599–8608.

[41] M. Jeong, H. Kim, S. J. Cheon, B. J. Choi, and N. S. Kim, "Diff-TTS: A denoising diffusion model for text-to-speech," *arXiv preprint arXiv:2104.01409*, 2021.

[42] J. Lee and S. Han, "Nu-wave: A diffusion probabilistic model for neural audio upsampling," *Proc. Interspeech 2021*, pp. 1634–1638, 2021.

[43] H. Kameoka, T. Kaneko, K. Tanaka, N. Hojo, and S. Seki, "Voicegrad: Non-parallel any-to-many voice conversion with annealed langevin dynamics," *arXiv preprint arXiv:2010.02977*, 2020.

[44] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, M. Kudinov, and J. Wei, "Diffusion-based voice conversion with fast maximum likelihood sampling scheme," *arXiv preprint arXiv:2109.13821*, 2021.

[45] J. Liu, C. Li, Y. Ren, F. Chen, P. Liu, and Z. Zhao, "Diffsinger: Diffusion acoustic model for singing voice synthesis," *CoRR*, vol. abs/2105.02446, 2021. [Online]. Available: https://arxiv.org/abs/2105.02446

[46] S. Liu, Y. Cao, D. Su, and H. Meng, "Diffsvc: A diffusion probabilistic model for singing voice conversion," *arXiv preprint arXiv:2105.13871*, 2021.

[47] S. Rouard and G. Hadjeres, "Crash: Raw audio score-based generative modeling for controllable high-resolution drum sound synthesis," *arXiv preprint arXiv:2106.07431*, 2021.

[48] N. Kandpal, O. Nieto, and Z. Jin, "Music enhancement via image translation and vocoding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 3124–3128.

[49] G. Mittal, J. Engel, C. Hawthorne, and I. Simon, "Symbolic music generation with diffusion models," *arXiv preprint arXiv:2103.16091*, 2021.

[50] K. Goel, A. Gu, C. Donahue, and C. Ré, "It's raw! audio generation with state-space models," *arXiv preprint arXiv:2202.09729*, 2022.

[51] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: http://jmlr.org/papers/v21/20-074.html

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[53] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022. [Online]. Available: https://arxiv.org/abs/2204.06125

[54] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," *arXiv preprint arXiv:2205.11487*, 2022.

[55] D. P. Kingma, T. Salimans, B. Poole, and J. Ho, "Variational diffusion models," in *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://openreview.net/forum?id=2LdBqxc1Yv

[56] T. Salimans and J. Ho, "Progressive distillation for fast sampling of diffusion models," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=TIdIXIpzhoI

[57] A. Q. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. Meila and T. Zhang, Eds., vol. 139. PMLR, 18–24 Jul 2021, pp. 8162–8171. [Online]. Available: https://proceedings.mlr.press/v139/nichol21a.html

[58] J. Ho and T. Salimans, "Classifier-free diffusion guidance," in *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. [Online]. Available: https://openreview.net/forum?id=qw8AKxfYbI

[59] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[60] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[61] M. Tagliasacchi, Y. Li, K. Misiunas, and D. Roblek, "SEANet: A multi-modal speech enhancement network," in *Interspeech*, 2020, pp. 1126–1130.

[62] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://arxiv.org/abs/1511.07289

[63] A. A. Gritsenko, T. Salimans, R. van den Berg, J. Snoek, and N. Kalchbrenner, "A spectral energy distance for parallel speech synthesis," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/9873eaad153c6c960616c89e54fe155a-Abstract.html

[64] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[65] E. Manilow, G. Wichern, P. Seetharaman, and J. Le Roux, "Cutting music source separation some Slakh: A dataset to study the impact of training data quality and quantity," in *2019 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 45–49.

[66] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 771–775.

[67] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, "GuitarSet: A dataset for guitar transcription." in *ISMIR*, 2018, pp. 453–460.

[68] J. Thickstun, Z. Harchaoui, and S. M. Kakade, "Learning features of music from scratch," in *International Conference on Learning Representations (ICLR)*, 2017.

[69] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications," *Trans. Multi.*, vol. 21, no. 2, p. 522–535, feb 2019. [Online]. Available: https://doi.org/10.1109/TMM.2018.2856090

[70] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, New Orleans, USA, Mar. 2017.

[71] J. Shor, A. Jansen, R. Maor, O. Lang, O. Tuval, F. de Chaumont Quitry, M. Tagliasacchi, I. Shavitt, D. Emanuel, and Y. Haviv, "Towards learning a universal non-semantic representation of speech," in *Interspeech*, 2020, pp. 140–144.

[72] "MT3: Multi-task multitrack music transcription," https://github.com/magenta/mt3, accessed: 2022-05-01.

[73] N. Shazeer and M. Stern, "Adafactor: Adaptive learning rates with sublinear memory cost," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4596–4604. [Online]. Available: https://proceedings.mlr.press/v80/shazeer18a.html

[74] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet audio distance: A reference-free metric for evaluating music enhancement algorithms." in *INTERSPEECH*, 2019, pp. 2350–2354.

[75] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*. Citeseer, 2014.

DDX7: DIFFERENTIABLE FM SYNTHESIS OF MUSICAL INSTRUMENT SOUNDS

Franco Caspe **Andrew McPherson** **Mark Sandler**
Centre for Digital Music, Queen Mary University of London, UK
{f.s.caspe, a.mcpherson, mark.sandler}@qmul.ac.uk

ABSTRACT

FM Synthesis is a well-known algorithm used to generate complex timbre from a compact set of design primitives. Typically featuring a MIDI interface, it is usually impractical to control it from an audio source. On the other hand, Differentiable Digital Signal Processing (DDSP) has enabled nuanced audio rendering by Deep Neural Networks (DNNs) that learn to control differentiable synthesis layers from arbitrary sound inputs. The training process involves a corpus of audio for supervision, and spectral reconstruction loss functions. Such functions, while being great to match spectral amplitudes, present a lack of pitch direction which can hinder the joint optimization of the parameters of FM synthesizers. In this paper, we take steps towards enabling continuous control of a well-established FM synthesis architecture from an audio input. Firstly, we discuss a set of design constraints that ease spectral optimization of a differentiable FM synthesizer via a standard reconstruction loss. Next, we present Differentiable DX7 (DDX7), a lightweight architecture for neural FM resynthesis of musical instrument sounds in terms of a compact set of parameters. We train the model on instrument samples extracted from the URMP dataset, and quantitatively demonstrate its comparable audio quality against selected benchmarks.

1. INTRODUCTION

Sound generation and transformation tools are ubiquitous in music composition and production. The electronic synthesizer has enabled musicians to access forms of timbre beyond the capabilities of acoustic or amplified instruments. Chowning's FM Synthesis [1] is a widely used technique that is flexible to create complex, harmonic and inharmonic spectra from a reduced set of controls. Its long-standing presence in the audio industry has shaped traditional and contemporary sound design techniques [2–4] across musicians and producers, with a wide number of current commercial synthesizer keyboards, modules and software plugins featuring it.

© F. Caspe, A. McPherson, and M. Sandler. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** F. Caspe, A. McPherson, and M. Sandler, "DDX7: Differentiable FM Synthesis of Musical Instrument Sounds", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

For most of its history, the digital synthesizer has been inextricably associated with the MIDI keyboard. Several historical technical and social factors contributed to making the keyboard the de-facto control interface for the synthesizer [5]. The legacy of this association continues to steer synth design, favouring triggers and envelope generators over continuous control strategies.

Developing alternative interfaces for controlling digital synthesis remains an active area of research [6]. Many such digital musical instruments (DMIs) are based on mappings between sensor data and synthesis parameters [7, 8]. An alternative approach uses features extracted from the audio signal of an acoustic instrument to control a digital synthesis process. In that context, previous works employed audio signals from musical instrument as oscillators [9], in a configuration that can be seen as a special application of Adaptive Digital Audio Effects [10, 11].

More recently, Neural Audio Synthesis (NAS) algorithms have employed Deep Neural Networks (DNNs) to map audio features to synthesizer parameters, in tasks such as realistic audio synthesis from a compact set of control signals [12–14], timbre transfer from one instrument to another [15, 16] and enhancement of symbolic musical expression [17]. While the results are impressive, we observe that the employed synthesis architectures are overly complex, featuring dozens of time-evolving parameters that hinder intervention, with users defaulting to indirect methods to intervene with the synthesis process, such as network bending [18].

We are interested in an audio-based, continuous control technique for a well-established synthesis architecture, that can potentially offer musicians similar sound design primitives and outcomes available to keyboard players. Furthermore, we want the model to be compatible with live use, therefore it should be able to work in real-time.

In this work, we take steps towards enabling interpretable sound design controls for NAS algorithms, and present Differentiable DX7 (DDX7), a causal and lightweight DNN architecture that maps continuous audio features to the synthesis parameters of a well-known FM synthesizer. We train DDX7 in single instrument datasets and evaluate its resynthesis performance against selected benchmarks. We provide full source code, and an online supplement [1] with audio examples and a preliminary analysis of the model's real-time execution capabilities.

[1] https://fcaspe.github.io/ddx7

2. BACKGROUND

2.1 Linear FM Synthesis

Linear FM modulation for audio signals, originally described by Chowning [1] in the early 1970s, is a well-known sound design technique that powered the first massively available and commercially successful digital synthesizer, the Yamaha DX7, defining an era in music production [2]. Since then, it that has retained a fair amount of attention by the research community [19], being present in a variety of topics from timbre semantic analysis [13], to empirical [4] and computer assisted sound design [20], and adaptive effects [11]. Commercially, manufacturers continue to release as of today a sizable number of new instruments based on this technology [21, 22].

Linear FM synthesis is actually a phase modulation technique aimed at generating different timbres with few parameters and low computational resources. Expressed in term of sine waves, the instantaneous FM modulated signal can be written as shown in Equation 1, where f_c is the carrier frequency, f_m is the modulator frequency and I is the modulation index. In this work, we denote this particular linear phase modulation technique simply as FM Synthesis.

$$y(t) = sin(2\pi f_c t + Isin(2\pi f_m t)) \quad (1)$$

The side-bands of an FM signal are equally spaced around the carrier frequency, and their separation determined by f_m, while the number of harmonics depends on the modulation index and follows a Bessel function of the first order (Equation 2).

$$y(t) = \sum_{n=-\infty}^{n=+\infty} J_n(I) sin(2\pi(f_c + nf_m) \cdot t) \quad (2)$$

By controlling the modulation indexes and ratios, users can generate rich and complex timbre with a small set of oscillators modulated in frequency. For instance, the simple modulator-carrier setup can be extended with a new carrier that is modulated by the output of the previous pair. This *stacked* arrangement spreads the bands of the initial modulator-carrier pair across another carrier, adding a new layer of complexity to the timbre. Furthermore, outputs from many stacks can be mixed in an *additive* approach. Next, by setting the frequency ratio r in such a way that $f_c = r \cdot f_m$, with $r \in \mathbb{Q}$, FM generates harmonic spectra.

2.2 The DX7 Synthesizer

The Yamaha DX7 synthesizer is probably the best-known FM synthesizer, and it features a linear FM synthesis architecture that has been previously used by other synth models as well, making it an excellent candidate for integration to a continuous control strategy.

The sound of the DX7 is generated by six frequency-modulated sinusoidal oscillators, and it is programmed by means of setting up a *patch*. For each oscillator, a patch describes their routing (i.e. how the oscillators are interconnected in a stacked or additive fashion), the Attack-Decay-Sustain-Release (ADSR) envelope generator parameters and the frequency ratios with respect to the note that is being played. Fixed frequencies can also be assigned to the oscillators. The ADSR parameters control the *output levels* of each oscillator; affecting their output volume or modulation index depending on their interconnection. Finally, in the DX7, the oscillators' frequency ratios and routing remain *fixed* during audio rendering. Sound dynamics are generated mainly by the ADSR envelopes controlling the volume or modulation index of the carriers and modulators respectively.

2.3 Sound Matching

Sound matching is the process of estimating a set of synthesis parameters that approximate a given audio signal as best as possible. The task of matching an audio input to synthesizer controls is not new especially for FM, where evolutionary methods have been used to optimize a patch for a particular given sound [20, 23].

More recently, DNNs have been studied for supervised sound matching from an annotated dataset of patches and synthesized audio excerpts. The approaches include classification [24], where the network predicts a patch based on an input spectrogram, variational inference [25], with the DNN learns an invertible mapping between a dataset of audio and patches, and multi-modal analysis [26] that employs multiple aggregated audio features for prediction.

While these methods are effective, they usually process long audio windows on the order of seconds, required to estimate the values of the ADSR envelope generator. Furthermore they can only estimate parameters from audio excerpts at specific pitch values. These approaches are not suitable for continuous control of a synthesizer from an audio signal.

2.4 Neural Audio Synthesis

DNNs can capture complex relations from a dataset. This can be exploited in a generative approach to produce realistic sounding audio. Neural Audio Synthesis (NAS) algorithms employ DNNs as powerful synthesizers that can capture structure and nuances from a corpus of music and produce new audio with similar characteristics during inference. Furthermore, these algorithms can be conditioned at train or inference time, modifying the output on-the-fly.

Since the introduction of the seminal autoregressive architectures Wavenet [27] and SampleRNN [28], the NAS field has included generation approaches such as Generative Adversarial Neural Networks (GANs) [29–31], probabilistic models [32, 33] and style transfer methods [15, 34]. Furthermore, a branch of the field has focused on controllable music generation, in an effort to bring interaction possibilities to users, with NAS algorithms supporting control inputs such as MIDI [17], timbre descriptors [29] and pitch and loudness signals [12–14].

2.5 Differential Digital Signal Processing

Differential Digital Signal Processing (DDSP) [12] is one approach for efficient, high quality audio rendering from

a compact set of input controls. It biases a DNN towards generating and processing audio by the insertion of signal processing elements, such as oscillators and filters, in the network structure. These are implemented using differentiable operators from a neural network training framework and therefore can back-propagate gradients during training.

DDSP methods have been successfully employed in resynthesis and tone transfer [16] tasks, where they are conditioned on a set of time-evolving inputs, including the fundamental frequency of the target signal, and generate audio on a frame-by-frame basis. Some approaches such as the Neural Source Filter (NSF) [14, 35] or the Neural Waveshaper [13] learn to control a non-linear filter that shapes a harmonic source towards a particular target sound. Other DDSP architectures [12, 17] directly drive a Harmonic plus Noise (HpN) synthesizer [36], effectively learning a mapping between the input controls and the synthesizer parameters that generate the output.

While the previous resynthesis architectures can generate realistic tone transfer, there is little users can do to manipulate the resulting audio other than controlling the pitch and loudness inputs. The DNN controls spectral modelling algorithms, which do not have musically meaningful parameters. In this work, we present a differentiable FM synthesizer module that features a compact set of well-known sound design controls driven by a DNN, enabling a potential user intervention into the synthesis process.

While the idea of differentiable FM is not new, we have not been able to find published literature with details and evaluations of systems employing it. We highlight two web repositories, one involving a 2-oscillator FM optimization strategy of audio excerpts [37] and another presenting an extension of the original DDSP project with a Differentiable FM synthesizer [38], where the experiments fail to reproduce musical instrument sounds. Our approach imposes a set of constraints on the FM synth that allows a DNN to generate instrument sounds.

2.6 Training objective for DDSP resynthesis

The training process for the DDSP algorithms usually involves a multi-scale spectral (MSS) loss function that includes the L_1 distance of the amplitude spectrograms of a synthesized and a target audio excerpt, in linear and logarithmic form [12]. This function is used as a reconstruction loss, with the DDSP model aiming to replicate the target spectra during training.

Despite its widespread use, the MSS loss and more generally, spectral-based distance metrics present pitch-based failure modes that can conspire against the generalization capabilities of NAS algorithms when trained with gradient descent, as demonstrated by Turian et. al. [39]. The authors show how such functions fail to propagate informative gradients for fine-tuning oscillator frequencies towards a frequency target due to fine grained ripple on the loss surface. Furthermore, they indicate that for the MSS distance, jointly optimizing amplitude and frequency generates misleading gradients for both tasks; this loss function can match spectral amplitudes only when the harmonics of target and prediction are aligned in frequency.

We argue that the MSS loss works well for DDSP because of two main reasons: Firstly, these models drive highly parameterized spectral modelling synthesizers with fine control on the output, either in the form of filter-distortion DNNs [14, 35], multiple parallel waveshapers [13], or the HpN synthesizer [12]. Secondly, and most importantly, all DDSP resynthesis architectures require as conditioning the fundamental frequency from the target signal, extracted with an estimator [40]. This ensures a harmonic alignment between the target and the prediction, and effectively avoids the problem of having to optimize pitch using gradient descent and the MSS loss during training.

A loss function that cannot propagate informative gradients cannot be employed to train a DNN. This is particularly disadvantageous for the case of FM generation, where the synthesis parameters include the modulation index I and the frequency ratios r that determine distance between the side-bands and the carrier. A small mismatch on ratio estimation can generate an unwanted vibrato-like effect, due to small frequency differences between oscillators. A big mismatch can hinder the joint optimization of the harmonics' positions and amplitudes. A DNN that does not learn how to precisely control the ratios could very easily incur either of those problems.

3. METHOD

In this section, we propose a set of constraints that ease training of a DNN with a differentiable FM synthesizer. Next, we present DDX7, a NAS architecture for FM synthesis controlled by a Temporal Convolutional Network (TCN) [41]. We train the DDX7 model for FM resynthesis of musical instrument sounds. This effectively results in a DX7 patch that is playable by an arbitrary audio input. A diagram of the architecture is shown in Figure 1.

3.1 Differentiable DX7

Our aim is to provide continuous control possibilities for a well-known FM synthesizer. We choose the Yamaha DX7 for its lasting influence on musical practice.

Considering the DX7 patch design, we observe that when fixing the frequency ratios and the routing of the oscillators, all the harmonics and overtones that can be generated take a fixed position in the spectrum. This *patch constraint* can allow the synthesis model to propagate informative gradients from the MSS reconstruction loss, provided the FM partials are pre-aligned with the ones of the target audio by means of pitch conditioning, as in the case of DDSP resynthesis.

We propose a control scheme for an FM synthesizer where a DNN controls the modulation indices and volume of the oscillators. The oscillator routing and frequency ratios remain fixed. One problem may arise when considering the maximum values that the modulation index I can take. For different ranges of I, the Bessel functions can create local minima during spectral optimization due to

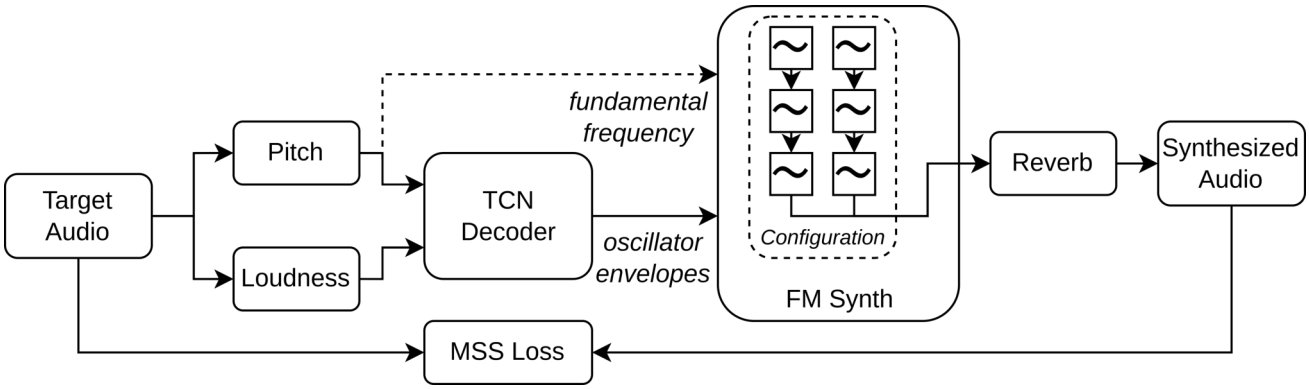


Figure 1. The DDX7 architecture employs a TCN decoder conditioned on a sequence of pitch and loudness frames to drive the envelopes of a few-oscillator differentiable FM synthesizer that features a fixed FM configuration with fixed frequency ratios, effectively mapping continuous controls of pitched musical instruments to a well-known synthesis architecture.

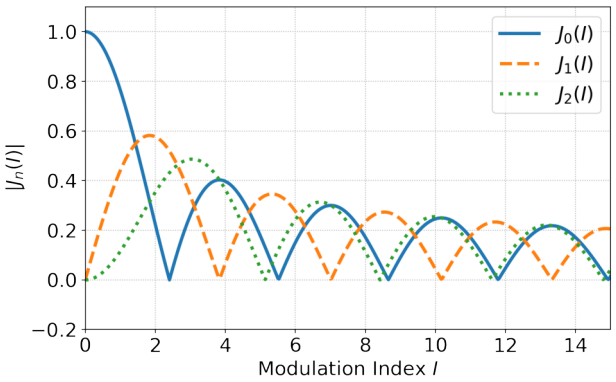


Figure 2. Absolute amplitudes of the first three harmonics generated by FM in function of I.

their oscillatory nature. In the DX7, the modulation index envelopes can take values of as much as 4π [42], but only for $I < 1.83$, are the Bessel functions strictly monotonic, with the carrier just exchanging energy with the sidebands, as shown in Figure 2. We analyze the effect of the maximum ranges through experiments in Section 5.

We implement the FM modulation algorithm in Pytorch, and adapt it manually for the different *FM configurations* that are tried in this work. Each configuration defines a fixed oscillator routing and a set of frequency ratios.

3.2 TCN Decoder

TCNs have been successfully employed a number of sequential modelling tasks, including for audio processing and generation [14, 27, 43]. These are fully-convolutional networks that employ 1-dimensional convolutions and exponentially growing dilations to efficiently model long sequences within their receptive field [41]. We choose TCNs for our DDX7 implementation for their fast training capabilities and good sequential modelling performance.

In our FM resynthesis problem, we aim to map a set of synchronous input sequences of pitch $f_{01}, ..., f_{0T} \in \mathbb{R}$ and loudness $ld_1, ..., ld_T \in \mathbb{R}$ to the controls of our constrained synthesizer, i.e. the output levels $ol_1, ..., ol_T \in \mathbb{R}^6$ of the six oscillators. We define the parameterized mapping function f_θ as shown in Equation 3, where the conditioning sequence $c_1, ..., c_T \in \mathbb{R}^2$ is obtained by concatenating along a new dimension both pitch and loudness sequences, $\sigma(.)$ is the sigmoid activation function and A_{max} is the maximum output level value that the envelopes can take for that oscillator, as shown in Equation 4, where I_{max} is a hyperparameter describing the maximum modulation index range that the system can realize.

$$\hat{ol}_1, ...\hat{ol}_T = A_{max} \cdot \sigma(f_\theta(c_1, ...c_T)) \tag{3}$$

$$A_{max} = \begin{cases} 1 & \text{if carrier} \\ I_{max} & \text{otherwise} \end{cases} \tag{4}$$

We implement our mapping function f_θ with a simple, causal TCN architecture following [41], with 2 input and 6 output channels, processed by 5 TCN residual blocks with skip connections and 128 hidden channels each. Each residual block features two convolutional layers of kernel size 3, and the dilation increases by a factor of 2 in each block. Weight normalization, dropout with probability of 0.5 and ReLU activation functions are used throughout the network, with the exception of the output layer that features a sigmoid layer. This yields a relatively lightweight decoder, with about 400k parameters, and a receptive field T of 125 pitch and loudness frames.

3.3 Learnable Reverb

We employ a differentiable reverb module, similar to the one employed for the DDSP decoder [12], featuring learnable mix and decay parameters, and a trainable impulse response of 1 second length. This is used to estimate the room response of the dataset recordings, decoupling it from the FM sound generation block. It is applied directly to the FM synthesizer output and it is jointly optimized with the DNN during training.

4. TRAINING

4.1 Loss function

The DDX7 architecture can be trained with a corpus of audio from a musical instrument as supervision, employing stochastic gradient descent on minibatches with a spectral reconstruction objective. We employ the MSS reconstruction loss shown in Equation 5, where S_i and $\hat{S}_i$ are the magnitude spectrograms of the target and synthesized audio respectively, $||.||_1$ denotes the L_1 norm, and i is a particular Fourier transform analysis window on which the spectrogram is computed. We use $i \in \{64, 128, 256, 512, 1024, 2048\}$ with an overlap of 75% between windows.

$$L = \sum_i (||S_i - \hat{S}_i||_1 + ||logS_i - log\hat{S}_i||_1) \quad (5)$$

4.2 Dataset

We train our DDX7 models on audio samples of instruments extracted from the University of Rochester Music Performance (URMP) dataset [44], an audio-visual dataset that contains classical pieces. We select the separated audio stems for violin, flute and trumpet as our training data, down-sample them to 16 kHz, remove silences and crop the audio files to instances of 4 seconds. We extract the A-weighted loudness [45] and fundamental frequency employing the CREPE [40] pitch estimator. We discard all instances that yield a mean pitch confidence smaller than 0.85, with the exception of the flute corpus, for which we relax the requirement down to 0.80 due to its short length. We further normalize pitch and loudness values within a range between 0 and 1.

We process each annotation with a hop size of 64 samples, yielding pitch and loudness sequences of 1000 frames for each 4-second instance. The selection of hop size and sample rate results in our TCN model featuring a receptive field of 0.5 s, and dictates the frame rate at which it drives the oscillators, 250 Hz. We linearly interpolate the envelope frames before feeding them into the oscillators. Finally, we separate the dataset into train, validation and test sets with 0.75 / 0.125 / 0.125 splits respectively.

4.3 FM configurations

For each target instrument, we select a different FM configuration extracted from the original patch set of the Yamaha DX7, which we retrieve from the web. [2] Then, we load the patches in Dexed [46], a DX7 emulator, and audit them searching for most similar to the target instruments. We select "STRINGS 1" for violin, "FLUTE 1" for flute and "BRASS 3" for the trumpet. We deploy the oscillator routing with the frequency ratios rounded up to one decimal point (to avoid vibrato-like effects) as differentiable FM modules, as shown in Figure 3. We do not deploy the oscillator feedback feature of the DX7 in our implementation, as it cannot be computed in parallel and we find it is very slow to render using a standard for loop in Python.

[2] http://bobbyblues.recup.ch/yamaha_dx7/

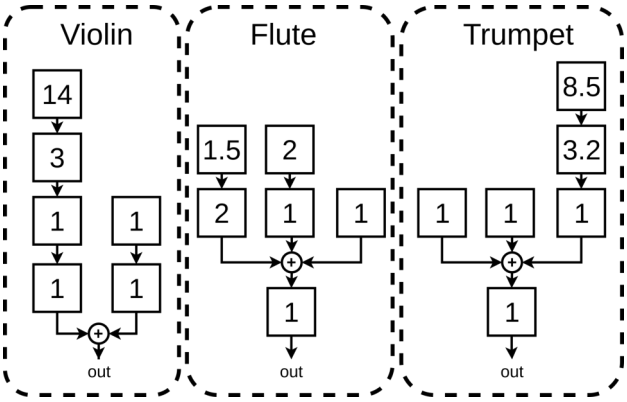


Figure 3. FM configurations used for training. Squares indicate sinusoidal oscillators and their frequency ratios.

4.4 Training process

Our models are trained for 120k steps, with the Adam optimizer, set with an initial learning rate of 3e-4, decreasing with a factor of 0.98 each 10k steps. We clip gradients to a maximum norm of 2, and employ a batch size of 16.

5. EVALUATION

We evaluate the performance of DDX7 on the resynthesis task, training the model on the flute, trumpet and violin corpus, and evaluate its performance on selected benchmarks with the Fréchet Audio Distance (FAD).

5.1 Fréchet audio distance

The Fréchet Audio Distance (FAD) presented by Kilgour et. al. [47] serves as a quality metric for audio enhancement, that correlates better with human listeners than SDR (signal-to-distortion ratio) or spectral differences such as the MSS loss. This is in line with other deep neural features used in Computer Vision that are found to outperform heuristic metrics by great margins [48]. It has been used to assess synthesis quality in previous works [13, 29, 31]. The FAD computes the Fréchet distance between multivariate Gaussian distributions inferred from the embeddings of a pre-trained VGGish model [49]. The compared distributions are generated from the embeddings of a corpus of audio for evaluation and a "background" corpus of high quality audio as reference.

5.2 Benchmarks

5.2.1 Maximum modulation index

We are interested into knowing which are the best modulation index limits for which our model can successfully render the instrument audio. Taking into account that optimizing for a particular spectra may be difficult for the original maximum modulation index range of the DX7 of 4π, we train our model for each instrument with three different maximum modulation index ranges for the oscillators: $I_{max} = \{4\pi, 2\pi, 2\}$, including the original DX7 range, a halved one and one limited at two, which includes the

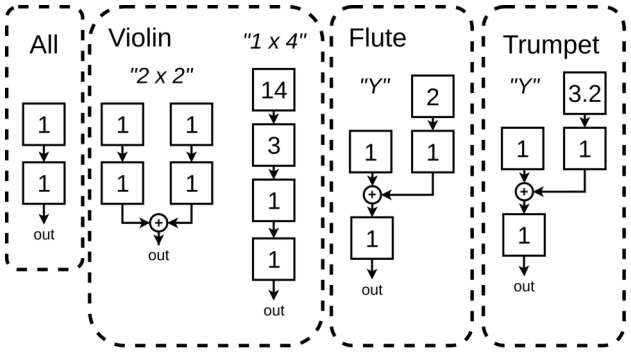


Figure 4. Ablated FM configurations.

global maximum of the first harmonic $J_1(I)$, but limits the other Bessel functions to a space where they are strictly monotonic. Finally, we load the corresponding configuration for each instrument selected, and we train three versions of DDX7, to account for each of the values of I_{max}.

Our FM model learns to control the envelopes of 6 oscillators frame-wise. As a baseline, we assess the performance of the common Harmonic plus Noise spectral modelling synthesizer employed for resynthesis tasks. We train a baseline pitch and loudness decoder that controls 121 frame-wise parameters of an HpN synthesizer, similar to the original solo instrument DDSP Decoder [12], but implemented in the same training framework as DDX7, to ensure that we use exactly the same dataset, preprocessing and frame rate. The model is roughly 11 times bigger than DDX7, with about 4.5 M parameters. We train it for 120k steps for parity with our model, with a batch size of 16, and an initial learning rate of 1e-4, decreasing to a factor of 0.98 for each 10k steps.

For evaluation with FAD, we generate background embedding distributions of the complete audio corpus of each instrument. Next, we generate embeddings distributions from the resynthesized excerpts of the test set for each model and instrument. We compute the FAD of these and the original test set against their corresponding background distribution. Results are shown in Table 1. We observe that the HpN baseline outperforms our model in violin and trumpet, but controlling 121 parameters instead of 6, and at a higher computational cost. Surprisingly, DDX7 outperforms the baseline on flute, suggesting that our *patch constraint* approach can bias a DNN towards a usable FM timbral space. We observe the best DDX7 performance is obtained with $I_{max} = 2$ for flute and violin and $I_{max} = 2\pi$ for trumpet. This may be correlated with the different spectra of the instruments, suggesting that the trumpet may require a bigger I_{max} for an improved reconstruction. Finally, for some configurations of I_{max} the models sound unnatural and fail at the estimation of the room response. We leave further analysis of this issue for future work.

5.2.2 Oscillator ablation test

We want to assess if our model is effectively optimized to leverage all of the modulators for the reconstruction task. We propose ablated versions of the previous patches with two and four oscillators, as shown in Figure 4. We train the DDX7 models on the ablated patches using the optimal I_{max} found with the previous benchmark and compare their resynthesis quality with the FAD following the same previous procedure. The results shown in Table 2 suggest that the violin and flute model benefit from the extra degrees of freedom present with more oscillators. On the other hand, the trumpet model works best with the smallest configuration, possibly due to an incorrect patch selection that hindered the optimization process. Finally, the 2-oscillator trumpet model outperforms the HpN baseline, suggesting that good results can be achieved with a small number of frequency-modulated oscillators.

	Fréchet Audio Distance		
Model	*Flute*	*Violin*	*Trumpet*
Test Data	2.074	0.577	1.069
HpN Baseline	4.326	**0.795**	**2.486**
DDX7 ($I_{max} = 2$)	**2.731**	1.618	4.941
DDX7 ($I_{max} = 2\pi$)	3.281	2.148	3.326
DDX7 ($I_{max} = 4\pi$)	2.938	1.637	3.853

Table 1. FAD of resynthesis results for all models computed against the background embedding distributions for each instrument complete corpus. Best results are in bold and best I_{max} configurations are underlined.

Instrument	**FM Configuration**				
	6	*4 "Y"*	*4x1*	*2x2*	*2*
Flute	**2.731**	3.246	-	-	3.364
Violin	**1.618**	-	1.877	5.620	8.270
Trumpet	3.326	2.943	-	-	**1.674**

Table 2. FAD for complete and ablated patches on DDX7.

6. CONCLUSION

We presented DDX7, an approach for FM resynthesis of musical instrument sounds that yields good reconstructions controlling few parameters, with relatively smaller models. We have shown that FM with a *patch constraint* can perform comparably well to a more complex baseline with just 6, and even less oscillators; we hope this motivates further research along this line, including for instance sound matching techniques to find suitable configurations.

Current resynthesis architectures feature synthesizers that are difficult to intervene in a musically meaningful way. In contrast, DDX7 learns to control an FM synthesizer that is common in the sound design practice. It replaces the ADSR generator of the original DX7 with a TCN that infers the envelopes from continuous control inputs. At runtime, it is possible to manipulate the timbre on-the-fly, either by re-shaping the spectrum with the ratios, altering dynamics on the envelopes, or by re-routing the oscillators. Finally, the small model size and causal temporal dependency make DDX7 an interesting candidate for real-time implementation. We leave an exploration of these affordances and possibilities for future work.

7. ACKNOWLEDGEMENTS

We would like to thank the ISMIR reviewers for their valuable feedback. Also, we would like to thank our colleagues Ben Hayes and Rodrigo Diaz for their advice and many compelling discussions about audio rendering and DNN optimization. This work was supported by UK Research and Innovation [grant number EP/S022694/1]. AM's contributions are supported by the Royal Academy of Engineering under the Research Chairs and Senior Research Fellowships scheme.

8. REFERENCES

[1] J. M. Chowning, "The synthesis of complex audio spectra by means of frequency modulation," *Journal of the Audio Engineering Society*, vol. 21, no. 7, pp. 526–534, 1973.

[2] M. Lavengood, "What makes it sound '80s? The Yamaha DX7 electric piano sound," *Journal of Popular Music Studies*, vol. 31, no. 3, pp. 73–94, 2019.

[3] E. Miranda, *Computer Sound Design: Synthesis techniques and programming*. Routledge, 2012.

[4] B. Stevens, *Teaching Electronic Music: Cultural, Creative, and Analytical Perspectives*. Routledge, 2021.

[5] T. Pinch and F. Trocco, "The social construction of the early electronic music synthesizer," *ICON*, pp. 9–31, 1998.

[6] A. R. Jensenius and M. J. Lyons, *A NIME reader: Fifteen years of New Interfaces for Musical Expression*. Springer, 2017, vol. 3.

[7] T. West, B. Caramiaux, S. Huot, and M. M. Wanderley, "Making mappings: Design criteria for live performance," *New Interfaces for Musical Expression conference (NIME)*, 5 2021.

[8] J. Regimbal and M. M. Wanderley, "Interpolating audio and haptic control spaces," in *New Interfaces for Musical Expression conference (NIME)*. PubPub, 2021.

[9] C. Poepel and R. B. Dannenberg, "Audio Signal Driven Sound Synthesis," in *International Computer Music Conference*, 2005.

[10] V. Verfaille, U. Zolzer, and D. Arfib, "Adaptive digital audio effects (a-dafx): a new class of sound transformations," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1817–1831, 2006.

[11] V. Lazzarini, J. Timoney, and T. Lysaght, "Adaptive FM Synthesis," in *DAFX-07 the 10th Int. Conference on Digital Audio Effects*, September 2007.

[12] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable Digital Signal Processing," in *8th International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

[13] B. Hayes, C. Saitis, and G. Fazekas, "Neural waveshaping synthesis," *Proceedings of the 22th International Society for Music Information Retrieval Conference*, 2021.

[14] M. Michelashvili and L. Wolf, "Hierarchical timbre-painting and articulation generation," *Proceedings of the 21th International Society for Music Information Retrieval Conference*, 2020.

[15] O. Cifka, A. Ozerov, U. Şimşekli, and G. Richard, "Self-Supervised VQ-VAE for One-Shot Music Style Transfer," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 96–100.

[16] M. Carney, C. Li, E. Toh, P. Yu, and J. Engel, "Tone transfer: In-browser interactive neural audio synthesis," in *Joint Proceedings of the ACM IUI 2021 Workshops*, 2021.

[17] Y. Wu, E. Manilow, Y. Deng, R. Swavely, K. Kastner, T. Cooijmans, A. Courville, C.-Z. A. Huang, and J. Engel, "MIDI-DDSP: Detailed control of musical performance via hierarchical modeling," *International Conference on Learning Representations (ICLR) 2022*, 2022.

[18] M. Yee-King and L. McCallum, "Studio report: Sound synthesis with DDSP and network bending techniques," *Proceedings of the 2nd Conference on AI Music Creativity*, 2021.

[19] K. Nielsen, "Practical linear and exponential frequency modulation for digital music synthesis," *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20), Vienna, Austria, September 8–12, 2020-21*, 2020.

[20] N. Masuda and D. Saito, "Quality diversity for synthesizer sound matching," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx20in21)*, 2021.

[21] Sound On Sound Magazine. (2020) Korg Opsix. [Online]. Available: https://www.soundonsound.com/reviews/korg-opsix

[22] Max Kuehn, for Fidlar Music. (2022) Best FM Synth 2022. [Online]. Available: https://fidlarmusic.com/best-fm-synth/

[23] A. Horner, J. Beauchamp, and L. Haken, "Machine tongues XVI: Genetic algorithms and their application to FM matching synthesis," *Computer Music Journal*, vol. 17, no. 4, pp. 17–29, 1993.

[24] M. J. Yee-King, L. Fedden, and M. d'Inverno, "Automatic Programming of VST Sound Synthesizers Using Deep Networks and Other Techniques," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 2, pp. 150–159, Apr. 2018.

[25] G. Le Vaillant, T. Dutoit, and S. Dekeyser, "Improving synthesizer programming from variational autoencoders latent space," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx20in21)*, 2021.

[26] Z. Chen, Y. Jing, S. Yuan, Y. Xu, J. Wu, and H. Zhao, "Sound2Synth: Interpreting sound via FM synthesizer parameters estimation," *arXiv preprint arXiv:2205.03043*, 2022.

[27] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *The 9th ISCA Speech Synthesis Workshop*, 2016.

[28] S. Mehri, K. Kumar, I. Gulrajani, R. Kumar, S. Jain, J. Sotelo, A. Courville, and Y. Bengio, "SampleRNN: An Unconditional End-to-End Neural Audio Generation Model," in *5th International Conference on Learning Representations*, Toulon, France, 2017.

[29] J. Nistal, S. Lattner, and G. Richard, "DrumGAN: Synthesis of Drum Sounds With Timbral Feature Conditioning Using Generative Adversarial Networks," in *Proceedings of the 21th International Society for Music Information Retrieval Conference*, Montréal, Aug. 2020.

[30] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial Neural Audio Synthesis," in *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, p. 17.

[31] A. Lavault, A. Roebel, and M. Voiry, "StyleWaveGAN: Style-based synthesis of drum sounds with extensive controls using generative adversarial networks," *arXiv preprint arXiv:2204.00907*, 2022.

[32] P. Esling, N. Masuda, A. Bardet, R. Despres, and A. Chemla-Romeu-Santos, "Flow Synthesizer: Universal Audio Synthesizer Control with Normalizing Flows," *Applied Sciences*, vol. 10, no. 1, p. 302, 2020.

[33] A. Caillon and P. Esling, "RAVE: A variational autoencoder for fast and high-quality neural audio synthesis," *arXiv preprint arXiv:2111.05011*, 2021.

[34] S. Huang, Q. Li, C. Anil, S. Oore, and R. B. Grosse, "TimbreTron A WaveNet(CycleGAN(CQT(Audio))) Pipeline for Musical Timbre Transfer," in *7th International Conference on Learning Representations*, New Orleans, LA, USA, 2019, p. 17.

[35] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter waveform models for statistical parametric speech synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 402–415, 2019.

[36] Y. Stylianou, J. Laroche, and E. Moulines, "High-quality speech modification based on a harmonic+ noise model," in *Fourth European Conference on Speech Communication and Technology*, 1995.

[37] A. Jansson, "Implicit neural differentiable FM synthesizer," https://github.com/andreasjansson/fmsynth, 2022.

[38] J. Alonso, "DDSP-FM: differentiable FM synthesis," https://juanalonso.github.io/ddsp_fm/, 2021.

[39] J. Turian and M. Henry, "I'm Sorry for Your Loss: Spectrally-Based Audio Distances Are Bad at Pitch," *arXiv:2012.04572 [cs, eess]*, Dec. 2020, I Can't Believe It's Not Better! (ICBINB) NeurIPS 2020 Workshop.

[40] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A Convolutional Representation for Pitch Estimation," in *2018 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2018 - Proceedings*. Institute of Electrical and Electronics Engineers Inc., Sep. 2018, pp. 161–165.

[41] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling," *arXiv:1803.01271 [cs]*, Apr. 2018, arXiv: 1803.01271.

[42] D. Bristow and J. Chowning, "FM Theory and Applications: By Musicians for Musicians," *Yamaha Music Foundation*, 1986.

[43] C. J. Steinmetz and J. D. Reiss, "Efficient neural networks for real-time modeling of analog dynamic range compression," in *152nd AES Convention*, 2022.

[44] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a Multitrack Classical Music Performance Dataset for Multimodal Music Analysis: Challenges, Insights, and Applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, Feb. 2019.

[45] B. C. Moore, B. R. Glasberg, and T. Baer, "A model for the prediction of thresholds, loudness, and partial loudness," *Journal of the Audio Engineering Society*, vol. 45, no. 4, pp. 224–240, 1997.

[46] P. Gauthier, "Dexed - FM Plugin Synth," https://github.com/asb2m10/dexed, 2022.

[47] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "Fréchet Audio Distance: A Reference-Free Metric for Evaluating Music Enhancement Algorithms," in *Interspeech 2019*. ISCA, Sep. 2019, pp. 2350–2354.

[48] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, Jun. 2018, pp. 586–595.

[49] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "CNN architectures for large-scale audio classification," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. New Orleans, LA: IEEE, Mar. 2017, pp. 131–135.

SINGING BEAT TRACKING WITH SELF-SUPERVISED FRONT-END AND LINEAR TRANSFORMERS

Mojtaba Heydari **Zhiyao Duan**
Department of Electrical and Computer Engineering
University of Rochester, 500 Wilson Blvd, Rochester, NY 14627, USA
mheydari@ur.rochester.edu zhiyao.duan@rochester.edu

ABSTRACT

Tracking beats of singing voices without the presence of musical accompaniment can find many applications in music production, automatic song arrangement, and social media interaction. Its main challenge is the lack of strong rhythmic and harmonic patterns that are important for music rhythmic analysis in general. Even for human listeners, this can be a challenging task. As a result, existing music beat tracking systems fail to deliver satisfactory performance on singing voices. In this paper, we propose singing beat tracking as a novel task, and propose the first approach to solving this task. Our approach leverages semantic information of singing voices by employing pre-trained self-supervised WavLM and DistilHuBERT speech representations as the front-end and uses a self-attention encoder layer to predict beats. To train and test the system, we obtain separated singing voices and their beat annotations using source separation and beat tracking on complete songs, followed by manual corrections. Experiments on the 741 separated vocal tracks of the GTZAN dataset show that the proposed system outperforms several state-of-the-art music beat tracking methods by a large margin in terms of beat tracking accuracy. Ablation studies also confirm the advantages of pre-trained self-supervised speech representations over generic spectral features.

1. INTRODUCTION

Music tempo and beat detection are two of the core and well-defined MIR topics, and scholars have proposed many approaches to addressing different aspects of them for various music genres. For instance, some works such as [1–4] proposed some unsupervised approaches to detect music beat and tempo by using some low-level features like onset strengths. More recently, several more recent approaches employ neural networks to address the mentioned tasks [5–8]. Extracting singing rhythmic parameters makes it possible to address several MIR problems such as automatic instrumental and singing tracks alignment, automatic music mix, and remix, interactive content creation on social media platforms, etc.

Many of the proposed music rhythmic analysis approaches [1, 5, 7] perform in an offline fashion while others [6, 9, 10] are capable of extracting music rhythmic features causally and even in real-time.

In contrast to all of the mentioned models, beat and tempo detection for singing voice is an untapped MIR task. There are significant inherent differences between the natures of complete music and singing voices. One of those differences is that complete music pieces usually contain rich percussive and harmonic profiles while singing tracks usually lack such beneficial parameters making their rhythmic analysis more demanding compared to the former group. Their other important difference is that music beat tracking models usually only use acoustical clues such as magnitude spectrogram as input features while singing tracks are more similar to speech signals where the models may require to deal with para-linguistics, semantic, and phonemic level inputs in addition to acoustical features.

Our contributions in this paper are as follows:

1- We introduce singing beat tracking as a novel MIR task. We propose two strategies to tackle the lack of annotated data for this task by leveraging pre-existing datasets and beat tracking and source separation techniques. We also propose a new evaluation scheme that can account for phase ambiguities of beat annotation for vocal music.

2- We propose two neural models for the singing beat tracking task. These models leverage pre-trained speech self-supervised models to extract feature embeddings, which are then fed into a linear transformer network to output beat predictions.

3- We evaluate the proposed models on hundreds of vocal tracks with diverse genres. Experiments show that the proposed models outperform three representative baselines designed or trained for general music beat tracking by a large margin. An ablation study is also performed to investigate the effect of speech self-supervised models over commonly used generic spectral features, and results again show an outperformance by a large margin.

The remainder of the paper is organized as follows: In Section 2, we describe the proposed vocal beat tracking task with dataset curation and a new evaluation scheme. In Section 3, we describe our proposed models with details. In Section 4, we present experimental comparisons with baselines and an ablation study on the feature extraction

© M. Heydari and Z. Duan. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. Heydari and Z. Duan, "Singing beat tracking with Self-supervised front-end and linear transformers", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

front end. Finally, Section 5 concludes the paper.

2. THE PROPOSED TASK

We propose singing beat tracking as a new MIR task, which aims to design algorithms that can track beats of singing voices in online or offline fashion. In this section, we describe the two cornerstones for this task: datasets and evaluation metrics.

2.1 Singing Data and Label Generation

Data for training and evaluating singing beat tracking algorithms require both singing voice recordings and their beat annotations. As there is no existing datasets that meet this requirement, we need to design a mechanism to collect such data. Instead of recording singing voices and annotating their beats manually, we propose to leverage existing MIR datasets and techniques to collect data for singing beat tracking, as illustrated in Figure 1.

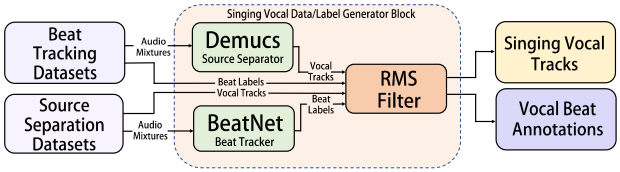


Figure 1. Singing data and label generation pipeline.

The first strategy we propose is to use pre-existing music beat tracking datasets in which the beat annotations are available. The idea is to obtain their singing tracks through music source separation. State-of-the-art (SOTA) singing voice separation methods have been shown to achieve outstanding results on popular genres of music, and the separated singing voice contains little interference from the background music. For this task, we employ Demucs Hybrid [11], one of the superior music source separation models. It uses a waveform-to-waveform convolutional autoencoder with a U-Net structure and bidirectional LSTM layers, and is designed to separate the music mixture into four sources including bass, drums, vocals, and others. We use the pre-trained model [1] which was trained on all 150 songs of MusDB [12] dataset. It is noted that source separation is used in some related works e.g. [13–15] to improve music beat tracking while in this work, we utilize it as one of the suggested systematic ways to generate singing vocal beat tracking datasets.

The second strategy we propose leverages pre-existing music source separation datasets, in which the isolated vocal tracks are available as a part of those datasets. To obtain beat annotations, we apply BeatNet [6] offline version on the complete songs (i.e., music mixtures). BeatNet is a SOTA online beat tracking method that uses a convolutional recurrent neural network (CRNN) and a hidden Markov model (HMM) decoder to extract music beats and downbeats. We modify the HMM decoder from particle filtering inference to Viterbi algorithm to improve its performance in the offline scenario. The reason that we run BeatNet on music mixtures instead of the separated vocal tracks is because it, the same as all other beat tracking methods, is trained on complete music pieces. While BeatNet has been shown to be fairly accurate on many songs with different genres, there are still annotation errors. To fix them, we also perform a manual revision by listening to the separated vocal track together with synthesized beeping sounds of the beat annotations and correcting errors.

While the beat annotations are obtained for the entire song, the ground-truth or separated vocal tracks show long chunks of silence due to the inactivity of singing. These long silent chunks are not useful and even are distracting for singing rhythmic analysis. Therefore, we normalize the energy of each vocal track using the root-mean-square (RMS), calculate a frame-wise RMS, and set a threshold to detect long silent chunks and split the vocal track into vocal segments. The RMS threshold is set to 0.01, and silent chunks shorter than 8 seconds are kept.

We apply these two strategies to a total of 8 existing beat tracking and music source separation datasets to obtain a total of 2248 vocal excerpts with beat annotations. The entire length of the vocal segments is 34h 35m. The datasets are summarized in Table 1.

2.2 Evaluation Metrics

For evaluation metrics, we adopt the commonly used F-measure in beat tracking. A beat is considered correctly detected if it is matched with a ground-truth beat with a time deviation smaller than 70 ms. In addition, we employ three additional metrics including P-score, Cemgil and Goto to provide a more detailed evaluation. Details about these three metrics can be found in [3].

For many music pieces, vocal tracks can align with the offbeat position (i.e., middle point between two adjacent ground-truth beats) rather that the beats. In fact, it also can be quite natural for humans to clap on the 180-degree phase shifted positions While this inherent ambiguity also exists in some instrumental music, it is much more common for singing voices. Apparently, the off-beat predictions would

Dataset	*# Number of vocal excerpts*	*Total Length*
Ballroom [16, 17] *	452	2 h 38 m
GTZAN [18, 19] *	741	5 h 44 m
Hainsworth [20]*	154	1 h 47 m
MUSDB18 [12]†	263	6 h 21 m
Rock Corpus [21]*	315	9 h 23 m
RWC pop [22, 23]*	188	5 h 06 m
RWC Royalty free [22,23]*	29	19 m
URSing [24]†	106	3 h 17 m

Table 1. Datasets collected and adapted for singing beat tracking. $*$ denotes beat tracking datasets and † denotes music separation datasets.

[1] https://github.com/facebookresearch/demucs

be evaluated poorly against the ground-truth beat annotations for all of the abovementioned metrics.

To address this problem, we propose an additional Phase Inclusive (PI) evaluation scheme with existing metrics. In this scheme, a metric is computed twice, one comparing the predicted beat positions with the ground-truth beat annotations and the other comparing with the 180-degree shifted ground-truth. The maximum is reported as the final metric under the PI evaluation scheme.

3. THE PROPOSED METHOD

To address the singing beat tracking task, we propose two models that take advantage of pre-trained WavLM and DistilHuBERT SSL representations respectively to extract the input features. Then we build the same linear multi-head self-attention network on top of them to fine-tune the models for our task. Finally, a hidden Markov model decoder is used to infer the singing beat positions using the activations provided by the respective neural networks. The source code and system demos are available [2] .Figure 2 demonstrates the overall structure of all proposed models and Figure 3 demonstrates the neural network structures of all proposed models.

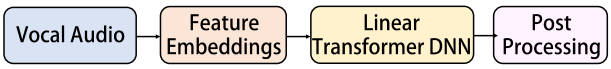


Figure 2. General pipeline of the proposed method.

3.1 Feature Embedding

Neural networks benefit from large quantities of labeled training data. However, in many cases including that of our task, labeled data is much harder to obtain than unlabeled data. Self-supervised learning has emerged as a paradigm to learn general data representations from unlabeled examples and fine-tuning the model on labeled data [25]. On the other hand, as demonstrated in [6], selecting appropriate input features plays an important role in music rhythmic analysis tasks. Due to acoustic and linguistic similarities between singing voices and speech, our main assumption is that self-supervised speech representations are helpful as feature embeddings in singing rhythmic analysis.

SSL achieves great success in speech-related tasks such as Automatic Speech Recognition, Phoneme Recognition and Emotion Recognition. Each task can be addressed by fine-tuning a universal pre-trained self-supervised model or training a neural network on top of the contextualized self-supervised representation of the input. In recent years, several pre-trained self-supervised models (e.g. [25–30]) attempt to provide such universal speech representations. The most successful models such as [25, 26] usually encode speech audio through a multi-layer convolutional neural network and then mask some chunks of the resulting latent speech representations. Then, a contrastive learning step is performed by feeding the latent representations to a transformer network to distinguish true latents [25].

WavLM: WavLm [26] is one of the recent universal self-supervised pre-trained models on 94k hours of data to solve full-stack downstream speech tasks. It jointly predicts masked speech and performs denoising in the pre-training stage. The denoising module helps to enhance the potential for non-ASR tasks. Moreover, it uses gated relative position bias in its transformer structure to improve capturing sequence ordering of input speech. WavLM has three different versions, Base, Base+, and Large. WavLM Base and WavLM Base+ include 12 encoder layers, 768-dimensional hidden states, and 8 attention heads. The WavLM Large contains 24 encoder layers, 1024-dimensional hidden states, and 12 attention heads. According to the SUPERB [31], which is an evaluation benchmark designed to provide a standard and comprehensive testbed for pre-trained models on several speech tasks, WavLM outperforms the other counterparts for several speech downstream tasks.

Since WavLM model has demonstrated promising results for several downstream speech tasks, we employ it as the front-end model for our first proposed approach to prepare contextualized input embeddings. Among three pre-trained options, we chose the Base+ model, becasue of its better performance than the Base model and similar performance with the Large model on many downstream tasks.

According to some works on the layer-wise analysis of self-supervised models (e.g. [32, 33]), using their last layer's output is not necessarily the best option to leverage the SSL model's maximum capability. This is because the desired feature aspects of the input signal are distributed differently through internal encoder layers. For example, [33] demonstrated that in Wav2Vec2 self-supervised model, for some properties like word meaning, intermediate layers play a much more important role than the output layer, while for acoustical features, early layers carry the most important information. Therefore, in order to capture different aspects of the input signal, we use a weighted sum of all encoder layers of WavLM in which each weight is a learnable parameter during the training phase.

DistilHuBERT: Although WavLM demonstrated the most successful results for several downstream tasks, it is a large model which requires large memory and high pre-training and inference costs imposing a speed bottleneck for the system. DistilHuBERT [29], is a novel multi-task learning framework to distill hidden representations from a HuBERT [27] model directly. It is initialized with the HuBERT's parameters. Then, its prediction heads learn to generate the teacher's hidden representations by minimizing the loss function. DistilHuBERT reduces the model size by 75% and increases the inference speed by 73% from HuBERT while retaining most performance in ten different speech tasks. Moreover, it requires little training time and data, opening the possibilities of pre-training personal and on-device SSL models for speech.

Therefore, we propose a second model that switches the heavy WavLM front-end model with DistilHuBERT,

[2] https://github.com/mjhydri/singing-vocal-beat-tracking

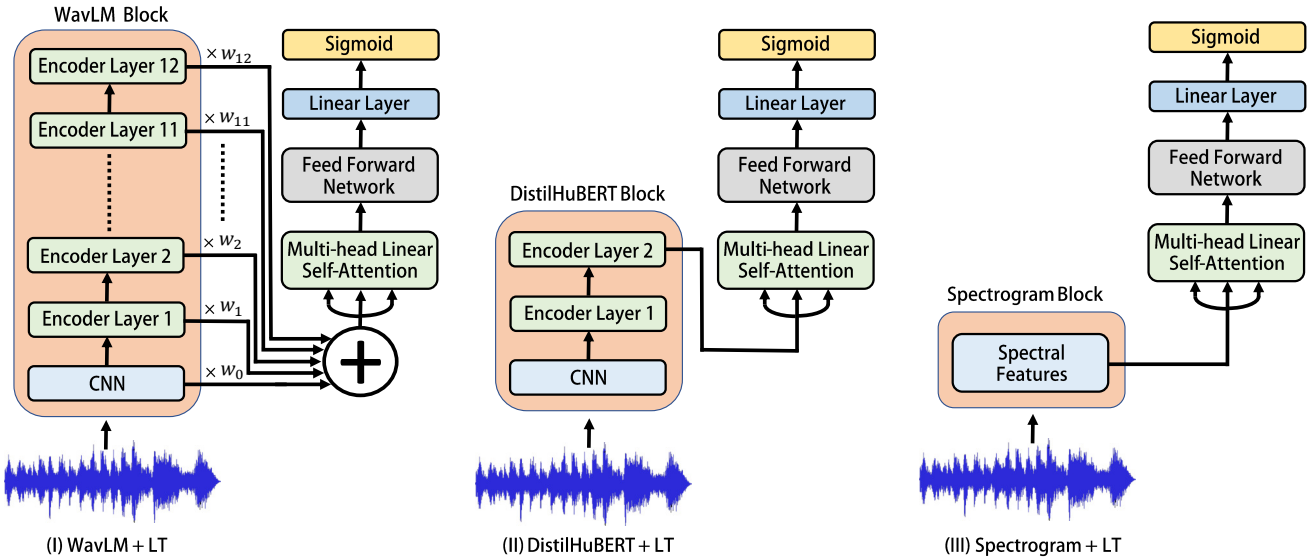


Figure 3. Neural network structures of the proposed models. (I), (II) and (III) use WavLM, DistilHuBERT and Spectrogram front-ends blocks, respectively, followed by the same linear transformer network.

a much lighter model, leading to a much faster inference process. DistilHuBERT includes a CNN network and two self-attention layers. According to its evaluation results [29], DistilHuBERT delivers comparable performance to the teacher HuBERT model with much less computational cost. It is noted that as the training objective of the DistilHuBERT student model is to learn directly from several internal encoder layers of the teacher model, the weighted sum strategy is not needed for this model and its last layer's output is used as the feature embeddings.

Spectrogram: We also perform an ablation study to evaluate the effect of using self-supervised models on the system performance by replacing the self-supervised pre-trained feature embeddings with spectral features that are commonly used in SOTA music beat tracking methods [6, 7]. These spectral features include log-magnitude mel-frequency spectrogram with three window sizes of 1024, 2048, and 4096 samples and their first-order time differences.

3.2 Linear Transformer Network

Transformers achieve remarkable performance in many tasks, but due to their quadratic complexity with respect to the input length, they are prohibitively slow for long sequences [34]. To address this limitation and to improve transformers' computational cost, several approaches are proposed. For instance, Linformer [35] is a model that reduces the transformer complexity in both time and space from $O(N^2)$ to $O(N)$ by approximating the self-attention mechanism by a low-rank matrix. It is noted that N is the sequence length. Another model [34]reduces the complexity to linear by expressing the self-attention as a linear dot-product of kernel feature maps and making use of the associativity property of matrix products. It achieves similar performance to vanilla transformers and they are up to 4000x faster on auto-regressive prediction of very long sequences. Another recent model [36] removes the softmax in self-attention by using the Gaussian kernel function to replace the dot-product similarity without further normalization. It enables a full self-attention matrix to be approximated via low-rank matrix decomposition.

In our proposed method, we build a linear multi-head self-attention layer using [34] on top of the feature embedding blocks for all three proposed models.

Our encoder comprises a self-attention layer with four attention heads with query and value dimensions of 192 and a feed-forward network with the size of 1024. The encoder network is followed by a linear layer and a sigmoid layer. The final output can be viewed as the beat salience for the input audio frame.

To train the proposed models, we use binary cross-entropy with logits between model output and the ground-truth beat annotations as the loss function. Only the linear transformer and its following layers are trained, while the self-supervised front ends are pre-trained and fixed.

4. POST-PROCESSING

In order to decode the beat positions, we employed a DBN approximated using HMM from Madmom library [37] with the default parameters. It is noted that the mentioned inference block is shared between all reported models in the evaluation table except BeatRoot [2].

5. EXPERIMENTS

5.1 Experimental Setup

We use all of the datasets collected in Section 2 except GTZAN for training the models with 80% of the songs randomly allocated for training and 20% for validation. Following several previous music beat tracking works, we keep the entire GTZAN dataset as the test set. It contains

1000 music excerpts covering 10 different music genres, out of which 741 of them contain vocal tracks. It ensures genre diversity in the test set and such a training/test dataset partition reduces the possibility of overfitting as well.Moreover, it is unseen in the training blocks of all compared methods. However, one drawback of it is that GTZAN's vocal tracks are separated from the whole songs. Hence, they may contain some information leakage from other musical instruments, which may boost the system performance compared to testing on isolated vocal tracks. To address this issue, we took the following steps. First, We used one of the best-performing supervised source separation models [11] and listened through the separated vocal tracks to ensure that they do not contain obvious music signal leakage. Second, when tuning the RMS filter, we discarded some sparse, abrupt and short signal segments that are more likely to be percussive sounds instead of vocal signals. Third, we compared with several high performing music beat tracking methods in the evaluation section; The leakage of musical signals into the separated tracks, if any, would improve the performance of these comparison models as well.

For the mentioned reason and given that singing beat tracking is a novel MIR task with no specifically design existing models, in this paper, we compare our proposed models against general music beat tracking models including two supervised models, BeatNet [6] and Böck [7] and an unsupervised signal-processing model, BeatRoot [2]. BeatNet is the SOTA online joint beat, downbeat, and meter tracking model that uses convolutional recurrent neural networks and particle filtering. It can also operate in an offline fashion by switching its particle filtering inference block with an offline hidden Markov model decoder. Böck uses a recurrent neural network to predict beat and downbeat saliences of each audio frame, and uses a Dynamic Bayesian Network (DBN) to infer beat and downbeat positions from the salience function. In our implementation, we use the same HMM decoder as that in BeatNet to replace the DBN. BeatRoot is a unsupervised method with a multiple agent architecture that simultaneously considers several different hypotheses of beat positions and tempi.

5.2 Training Details

To train the proposed models, we freeze the self-supervised front-end blocks and train the rest of the model including the multi-head self-attention layer and its following feed-forward and linear layers. For the WavLM+LT model, in addition to the abovementioned items, the 12 weights to combine different encoder layer's output are also trained.

The mentioned weights are initialized as ones and the rest of the weights and biases are initialized randomly. We use the Adam optimizer with a learning rate of 5×10^{-5} and a batch size of 10. The batches comprise 15-second long excerpts randomly sampled from all audio files in the training set. As the audio files have different length, to ensure a fair distribution, we sample with a probability that is proportional to the length of the files.

5.3 Results and Discussions

To assess the computational cost of the models, we report the inference speed for all of them. The reported numbers are the average computational time for all 741 excerpts with the average duration of 28 seconds each. Note that we measure the speed of all methods using CPU processing on the same Windows machine with an AMD Ryzen 9 3900X CPU and 3.80 GHz clock. Table 2 demonstrates the evaluation results.

Table 2 shows the evaluation results of the baselines and the proposed models using the four metrics presented in Section 2.2. It also includes the Phase Inclusive (PI) evaluation scheme for the proposed models. Several interesting observations can be made. First, all of the three proposed models outperform the three baselines by a large margin. This difference is especially pronounced for more strict metrics such as Goto [3]. This confirms our hypothesis that vocal tracks show very different patterns from complete songs, and the baseline models that are trained on complete songs do not perform as well as the proposed models that are trained on the vocal tracks. This hypothesis is further validated by the fact that BeatRoot outperforms BeatNet and Böck, while the latter two models are shown to outperform BeatRoot in previous studies on complete songs [6, 7]. It is noted that BeatNet and Böck are data-driven approaches and are trained on complete songs, while BeatRoot is a signal processing model without training. The mismatch between complete songs and vocal tracks does impose a strong negative bias to the data-driven baselines.

Second, the two proposed models that leverage speech SSL feature embeddings (i.e., WavLM+LT and DistilHuBERT+LT) improves over Spectrogram+LT by a large margin, and this improvement is even bigger than that from the best baseline to Spectrogram+LT. This shows the significant advantage of using feature embeddings learned on speech data and suggests that this advantage is even more significant than training on the vocal tracks. As the spectrogram features used in Spectrogram+LT are the same as those in BeatNet and Böck and transformers have been shown to outperform RNNs in various tasks, it is reasonable to believe that even if BeatNet and Böck are trained on the vocal tracks, their performance would be only comparable to that of Spectrogram+LT.

Third, WavLM+LT delivers the best scores for all of the evaluation metrics except the computation time, showing that stronger feature embeddings lead to better vocal beat tracking accuracy. However, the attention layers' complexity in WavLM is quadratic to the length of input; while the average computational time (4.09 seconds) is acceptable for vocal segments (28 s long on average) in this experiments, as the length increases, the computational time will soon become prohibitive. Figure 4 shows the learned weights of different encoder layers of the WavLM SSL model. It can be seen that earlier layers generally contribute more to forming the feature embeddings compared to latter layers. Layers 3, 2 and 1 contribute the most. Note that Layer 0 represents the CNN network output.

	Method	F-Measure	P-Score	Cemgil	Goto	Comp. Time
Baseline	BeatNet	0.243	0.327	0.173	0.003	0.13 (s)
	BeatRoot	0.301	0.394	0.22	0.066	**0.03 (s)**
	Böck	0.171	0.195	0.122	0.009	1.56 (s)
Proposed	WavLM + LT	**0.733**	**0.704**	**0.618**	**0.560**	4.09 (s)
	DistilHuBERT + LT	0.703	0.668	0.593	0.516	1.83 (s)
	Spectrogram + LT	0.454	0.438	0.367	0.223	0.32 (s)
Proposed (PI Results)	WavLM + LT	0.745	0.715	0.627	0.574	4.09 (s)
	DistilHuBERT + LT	0.721	0.684	0.608	0.537	1.83 (s)
	Spectrogram + LT	0.489	0.477	0.391	0.265	0.32 (s)

Table 2. Average performance and speed across segments of several methods on the GTZAN separated vocal tracks.

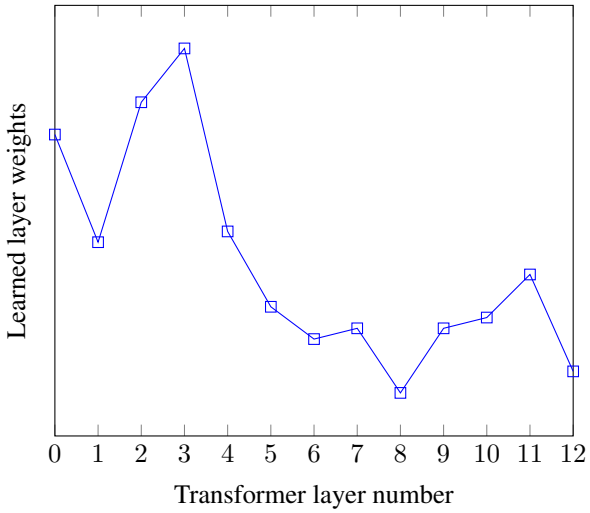


Figure 4. Illustration of the learned weights for different encoder layers of WavLM in the WavLM+LT model.

Fourth, with slight performance drop, the second proposed model which employs LT and DistilHubert delivers the next best performance among all models. DistilHubert uses a much lighter structure with fewer parameters than WavLM, making it ideal for time-sensitive inferences.

Finally, the PI evaluation scheme shows a slight improvement of all of the proposed models on all accuracy metrics. In particular, the improvement of Spectrogram+LT is greater than that of the other models. This suggests that the SSL pretrained speech embedding features are helpful to reduce the phase shift ambiguity.

Figure 5 shows the performance of our proposed WavLM + LT model for different music genres. According to the table, the best and worst performances belong to *disco* and *blues* respectively. One reason for that may be the stronger and punchier stresses in *disco* vocal tracks comparing to that of *blues* tracks. Another important reason may be the different syncopation ratio among vocal track of different music genres. Syncopation is a musical term that defines the placement of rhythmic stresses or accents where they would not normally occur, making part or all of a tune or piece of music off-beat [38]. Therefore, a higher syncopation ratio leads to a more difficult beat tracking process. Nate that in Figure 5 we only report the evaluation results of the genres that contain at least 80 separate singing tracks. It leaves out the *classical* and *jazz* genres which have only two and zero separated tracks.

6. CONCLUSION

In this paper we introduced singing beat tracking as a novel MIR task. We proposed two approaches to obtain labeled data for the mentioned task and introduced two methods to accomplish the mentioned task using pre-trained speech self-supervised models and multi-head linear self-attention networks. We also conducted an ablation study to investigate the effect of speech self-supervised models on the system performance. Experiments on a collection of vocal tracks with diverse genres show that the proposed models that combine self-supervised models and transformers outperform three representative baselines designed or trained for beat tracking for general music. The ablation study also shows that the self-supervised speech embeddings outperform generic spectral features that are commonly used in music beat tracking. In the future we will extend this research to cover real-time applications and more challenging singing tracks such as human humming and operas.

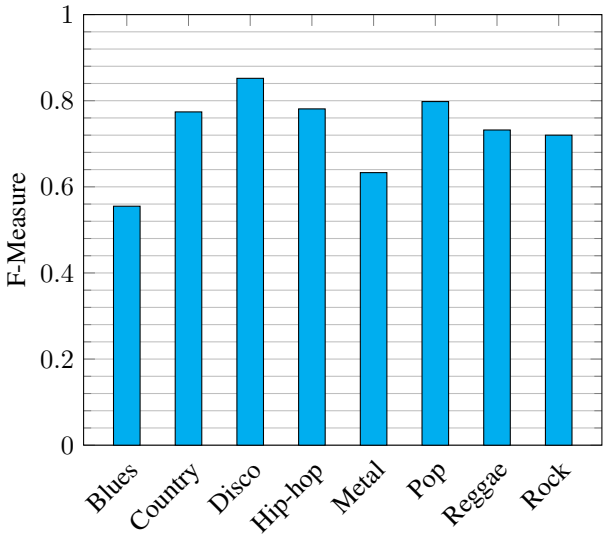


Figure 5. F-measure performance of WavLM + LT model on the GTZAN separated vocal tracks for different genres.

7. ACKNOWLEDGEMENT

This work was supported by National Science Foundation grant No. 1846184.

8. REFERENCES

[1] D. P. Ellis, "Beat tracking by dynamic programming," *Journal of New Music Research*, vol. 36, no. 1, pp. 51–60, 2007.

[2] S. Dixon, "Evaluation of the audio beat tracking system beatroot," *Journal of New Music Research*, vol. 36, no. 1, pp. 39–50, 2007.

[3] M. E. Davies, N. Degara, and M. D. Plumbley, "Evaluation methods for musical audio beat tracking algorithms," *Queen Mary University of London, Centre for Digital Music, Tech. Rep. C4DM-TR-09-06*, 2009.

[4] A. Mottaghi, K. Behdin, A. Esmaeili, M. Heydari, and F. Marvasti, "Obtain: Real-time beat tracking in audio signals index terms—onset strength signal, tempo estimation, beat onset, cumulative beat strength signal, peak detection," *International Journal of Signal Processing Systems*, pp. 123–129, 2017.

[5] S. Böck and M. E. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR), Montreal, QC, Canada*, 2020, pp. 12–16.

[6] M. Heydari, F. Cwitkowitz, and Z. Duan, "Beatnet: Crnn and particle filtering for online joint beat downbeat and meter tracking," 2021.

[7] S.Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *In Proc. of the 7th Intl. Conf. on Music Information Retrieval (ISMIR)*, 2016.

[8] M. Heydari and Z. Duan, "Don't look back: An online beat tracking method using RNN and enhanced particle filtering," in *In Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2021.

[9] J. L. Oliveira, F. Gouyon, L. G. Martins, and L. P. Reis, "Ibt: A real-time tempo and beat tracking system." in *ISMIR*, 2010, pp. 291–296.

[10] M. Heydari, M. McCallum, A. Ehmann, and Z. Duan, "A novel 1d state space for efficient music rhythmic analysis," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 421–425.

[11] A. Défossez, "Hybrid spectrogram and waveform source separation," in *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, 2021.

[12] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[13] D. Desblancs, R. Hennequin, and V. Lostanlen, "Zero-note samba: Self-supervised beat tracking," 2022.

[14] C.-Y. Chiu, J. Ching, W.-Y. Hsiao, Y.-H. Chen, A. W.-Y. Su, and Y.-H. Yang, "Source separation-based data augmentation for improved joint beat and downbeat tracking," in *2021 29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021, pp. 391–395.

[15] A. Gkiokas, V. Katsouros, G. Carayannis, and T. Stajylakis, "Music tempo estimation and beat tracking by applying source separation and metrical relations," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 421–424.

[16] F. Gouyon, A. P. Klapuri, S. Dixon, M. Alonso, G. Tzanetakis, C. Uhle, and P. Cano., "An experimental comparison of audio tempo induction algorithms," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, 2006.

[17] F. Krebs, S. Böck, and G. Widmer., "Rhythmic pattern modeling for beat and downbeat tracking in musical audio," in *Proc. of the 14th Int. Society for Music Information Retrieval Conf.*, 2013.

[18] U. Marchand and G. Peeters, "Swing ratio estimation," in *In Proc. of the 18th Int. Conference on Digital Audio Effects (DAFx)*, 2015.

[19] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing,*, vol. 10, no. 5, 2002.

[20] S. W. Hainsworth and M. D. Macleod, "Particle filtering applied to musical tempo tracking," *EURASIP Journal on Advances in Signal Processing*, vol. 2004, no. 15, pp. 1–11, 2004.

[21] T. de Clercq and D. Temperley., "A corpus analysis of rock harmony," *Popular Music*, vol. 30, no. 1, pp. 47–70, 2011.

[22] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "Rwc music database: Popular, classical and jazz music databases." in *Ismir*, vol. 2, 2002, pp. 287–288.

[23] M. Goto *et al.*, "Development of the rwc music database," in *Proceedings of the 18th international congress on acoustics (ICA 2004)*, vol. 1, 2004, pp. 553–556.

[24] B. Li, Y. Wang, and Z. Duan, "Audiovisual singing voice separation," *arXiv preprint arXiv:2107.00231*, 2021.

[25] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 449–12 460, 2020.

[26] S. Chen, C. Wang, Z. Chen, Y. Wu, S. Liu, Z. Chen, J. Li, N. Kanda, T. Yoshioka, X. Xiao *et al.*, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," *arXiv preprint arXiv:2110.13900*, 2021.

[27] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[28] P. Peng and D. Harwath, "Fast-slow transformer for visually grounding speech," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7727–7731.

[29] H.-J. Chang, S.-w. Yang, and H.-y. Lee, "Distilhubert: Speech representation learning by layer-wise distillation of hidden-unit bert," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 7087–7091.

[30] S. Ling and Y. Liu, "Decoar 2.0: Deep contextualized acoustic representations with vector quantization," *arXiv preprint arXiv:2012.06659*, 2020.

[31] S.-w. Yang, P.-H. Chi, Y.-S. Chuang, C.-I. J. Lai, K. Lakhotia, Y. Y. Lin, A. T. Liu, J. Shi, X. Chang, G.-T. Lin *et al.*, "Superb: Speech processing universal performance benchmark," *arXiv preprint arXiv:2105.01051*, 2021.

[32] L. V. Prasad, A. Seth, S. Ghosh, and S. Umesh, "Analyzing the factors affecting usefulness of self-supervised pre-trained representations for speech recognition," *arXiv preprint arXiv:2203.16973*, 2022.

[33] A. Pasad, J.-C. Chou, and K. Livescu, "Layer-wise analysis of a self-supervised speech representation model," *arXiv preprint arXiv:2107.04734*, 2021.

[34] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165.

[35] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[36] J. Lu, J. Yao, J. Zhang, X. Zhu, H. Xu, W. Gao, C. Xu, T. Xiang, and L. Zhang, "Soft: Softmax-free transformer with linear complexity," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[37] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: a new Python Audio and Music Signal Processing Library," in *Proceedings of the 24th ACM International Conference on Multimedia*, Amsterdam, The Netherlands, 10 2016, pp. 1174–1178.

[38] M. Hoffman, "Syncopation," *National Symphony Orchestra. NPR. Retrieved*, vol. 13, 2009.

ENSEMBLESET: A NEW HIGH QUALITY SYNTHESISED DATASET FOR CHAMBER ENSEMBLE SEPARATION

Saurjya Sarkar **Emmanouil Benetos** **Mark Sandler**
Centre for Digital Music, Queen Mary University of London, UK
{saurjya.sarkar, emmanouil.benetos, mark.sandler}@qmul.ac.uk

ABSTRACT

Music source separation research has made great advances in recent years, especially towards the problem of separating vocals, drums, and bass stems from mastered songs. The advances in this field can be directly attributed to the availability of large-scale multitrack research datasets for these mentioned stems. Tasks such as separating similar-sounding sources from an ensemble recording have seen limited research due to the lack of sizeable, bleed-free multitrack datasets. In this paper, we introduce a novel multitrack dataset called EnsembleSet generated using the Spitfire BBC Symphony Orchestra library using ensemble scores from RWC Classical Music Database and Mutopia. Our data generation method introduces automated articulation mapping for different playing styles based on the input MIDI/MusicXML data. The sample library also enables us to render the dataset with 20 different mix/microphone configurations allowing us to study various recording scenarios for each performance. The dataset presents 80 tracks (6+ hours) with a range of string, wind, and brass instruments arranged as chamber ensembles. We also present our benchmark on our synthesised dataset using a permutation-invariant time-domain separation model for chamber ensembles which produces generalisable results when tested on real recordings from existing datasets.

1. INTRODUCTION

Audio source separation aims to extract individual sound sources from a digital audio mixture. Based on the constituents of the input mixture and the target output, the problem definition can be further refined to specific audio separation tasks like speech separation, speech enhancement, and music source separation [1]. While specific subtasks in the speech-domain like speech denoising, multi-speaker separation and dereverberation have been thoroughly explored, music separation research has largely been focused on the demixing challenge [2] aided by the popular MUSDB dataset [3]. The demixing challenge is targeted at solving the problem of separation of vocals, bass and drums from mixed and mastered pop songs. This has greatly benefited the field by demonstrating that source separation is indeed possible at a commercial scale with state-of-the-art deep learning based architectures. Unfortunately, this also has resulted in the research towards this specific task to dwarf other problems that would also fall under the umbrella of music source separation, to the extent that music source separation has become synonymous with the task of separating vocal, drums and bass stems from mastered songs.

In this paper, we explore a different area in music source separation with a focus on separation of chamber ensembles, where the target sources are harmonised and have very high spectral overlap. The music demixing challenge has shown successful separation of instruments with distinct spectro-temporal cues like vocals, drums and bass. Separating monotimbral ensembles is an inherently challenging task as they combine challenging aspects of both speech and music separation. In chamber ensembles we find the sources to occupy similar frequency ranges, may have label ambiguity [4, 5] due to multiple sources belonging to the same instrument family meanwhile being temporally and harmonically correlated, due to their musical structure which further increases their spectral overlap.

To address this challenge, we present a novel chamber ensemble dataset titled EnsembleSet [6]. EnsembleSet was synthesised using a realistic sample library Spitfire BBC Symphony Orchestra (BBCSO) [7] utilising the MIDI transcriptions from the RWC Classical Music Database [8] and lilypond scores from Mutopia [9] (refer to Section 3.4 for details). In Section 4 we utilise EnsembleSet to train a source separation model based on the Dual-path Transformer architecture (DPTNet) [10, 11] for separating mixtures of chamber ensembles. We achieve very good and generalisable separation performance which we exhibit through cross-dataset performance evaluation in Section 5. Other applications of EnsembleSet may include topics such as multi-instrument transcription [12], instrument recognition [13], score-informed source separation [14], microphone translation [15], automatic mixing [16] and other tasks that may benefit from score-aligned multitrack multi-instrument data.

2. BACKGROUND

Although the term "Music Source Separation" sounds like an umbrella-term for all applications of source separation

© S. Sarkar, E. Benetos, and M. Sandler. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: S. Sarkar, E. Benetos, and M. Sandler, "EnsembleSet: a new high quality synthesised dataset for chamber ensemble separation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

in music, in research the use of this term is largely used to describe the specific task of vocals/drums/bass instrument stem separation from mastered tracks [2, 17]. A music source separation task which is relatively unexplored is the challenge of separating similar sounding instruments from a mixture. This problem has two significant differences from the aforementioned task. Firstly, if the sources in the mixtures are similar sounding (e.g., mixture of a strings section), it results in high spectral energy overlap. This is further compounded by the fact that such sources often play in a very synchronised fashion while harmonising each other. Secondly, often in such mixtures we may have multiple sources of the same type present, which makes it an unsuitable problem to be solved using *class-based* separation methods [4]. We define the task of separating such mixtures with constituent sources suffering from label ambiguity and high timbral similarity as *monotimbral ensemble separation*.

2.1 Datasets

Training supervised source separation models typically requires datasets which provide the clean target sources as a reference for the deep learning models to learn from. While the majority of popular music can be recorded in separate takes for different performers, with a reference metronome or a backing track, ensembles are usually recorded together in the same take. This is due to the fact that ensemble performers rely on being able to hear each other during performance to be able to synchronise perfectly. This raises the problem that the majority of stems available from recording projects of monotimbral ensembles contain bleed [1] from non-target sources [18, 19]. This becomes problematic for training models for source separation as we do not have the clean ground truth as a target result for the model. This lack of clean and sizeable datasets for ensembles has affected the amount of research seen in this domain.

The URMP dataset [20] addresses this problem by making the performers record isolated takes and then subsequently re-align, dereverberate and downmix them to be present in a physical space. Another work [21] presented a dataset recorded by minimising the amount of bleed using soundproofing across booths while recording multiple performers in the same take. Bach10 [22] also presents multitrack recordings of chamber ensembles where each song consists of four parts (Soprano, Alto, Tenor and Bass) which were performed by violin, clarinet, saxophone and bassoon, respectively. The TRIOS dataset [23] consists of 5 bleed-free multitracks and synchronised MIDI files of 4 classical music pieces and 1 jazz piece.

2.2 Prior work

There are some tasks which fit our definition of monotimbral separation that have been explored recently. One is vocal harmony separation [11,24–26]. While the label ambiguity problem does exist for this task, some approaches have circumvented it by looking at the problem in a class-based separation fashion by categorising the constituent sources based on their registers i.e. alto, soprano, bass and tenor. One method of solving the label ambiguity problem is to tackle this problem in a score-conditioned fashion as in [25]. Another method of tackling this problem is called permutation invariant training (PIT) [27] which has been the preferred solution to tackle the label ambiguity problem for speech separation research [10, 28, 29]. PIT has been utilised for choral separation in [11]. Another approach has been to use multi-task learning by utilising score-information to simultaneously separate and transcribe mixtures of 2-source chamber ensembles which has shown some success for scenarios with small datasets [30].

3. DATASET

To overcome the challenge of bleed-free real recorded datasets for ensembles, we introduce a novel dataset “EnsembleSet", which utilises a highly realistic orchestral sample library by Spitfire Audio called "BBC Symphony Orchestra" (BBCSO) [7]. We use this sample library to render digital chamber ensemble scores from MIDI and MusicXML format to 18 unique multi-mic recordings and 2 professional mixes. For this work, we utilised the RWC Classical Music Database [8] and Mutopia [9] to source our chamber ensemble MIDI and MusicXML (converted from lilypond) scores. It must be noted that MIDI data are not ideal to capture string, wind and brass instrument scores as they do not encapsulate articulation information. On the other hand lilypond scores contain minimal dynamics (velocity) information, which is essential for realistic rendering using virtual instruments. In order to address these challenges, we utilise expression maps provided by Dorico [31], a scorewriter software which allows us to determine the articulation mode for each note in the piece.

3.1 Collecting Digital Music Scores

3.1.1 RWC Classical Music Database

The RWC Classical Music Database [8] consists of 50 public-domain classical pieces performed by musicians and then manually transcribed to MIDI with high-quality tempo and velocity mapping. Since the database only provides the final mix of these performances, its applications are limited especially in the context of source separation. We choose a subset of these pieces which contain chamber ensembles which can be rendered using our method. Our 9 selected pieces (1h 3m 34s) [2] consist of 4 string quartets, 2 clarinet quintets, 2 piano trios and 1 piano quintet. It must be noted that for the piano trios and quintet, we only render the string instrument parts. Because MIDI files lack articulation information, we modify the MIDI files using Dorico to automatically add it using keyswitches, which are then subsequently rendered as multitracks on Reaper [32].

[1] Sound picked up by a microphone from a source other than that which is intended.

[2] Rendered duration in dataset.

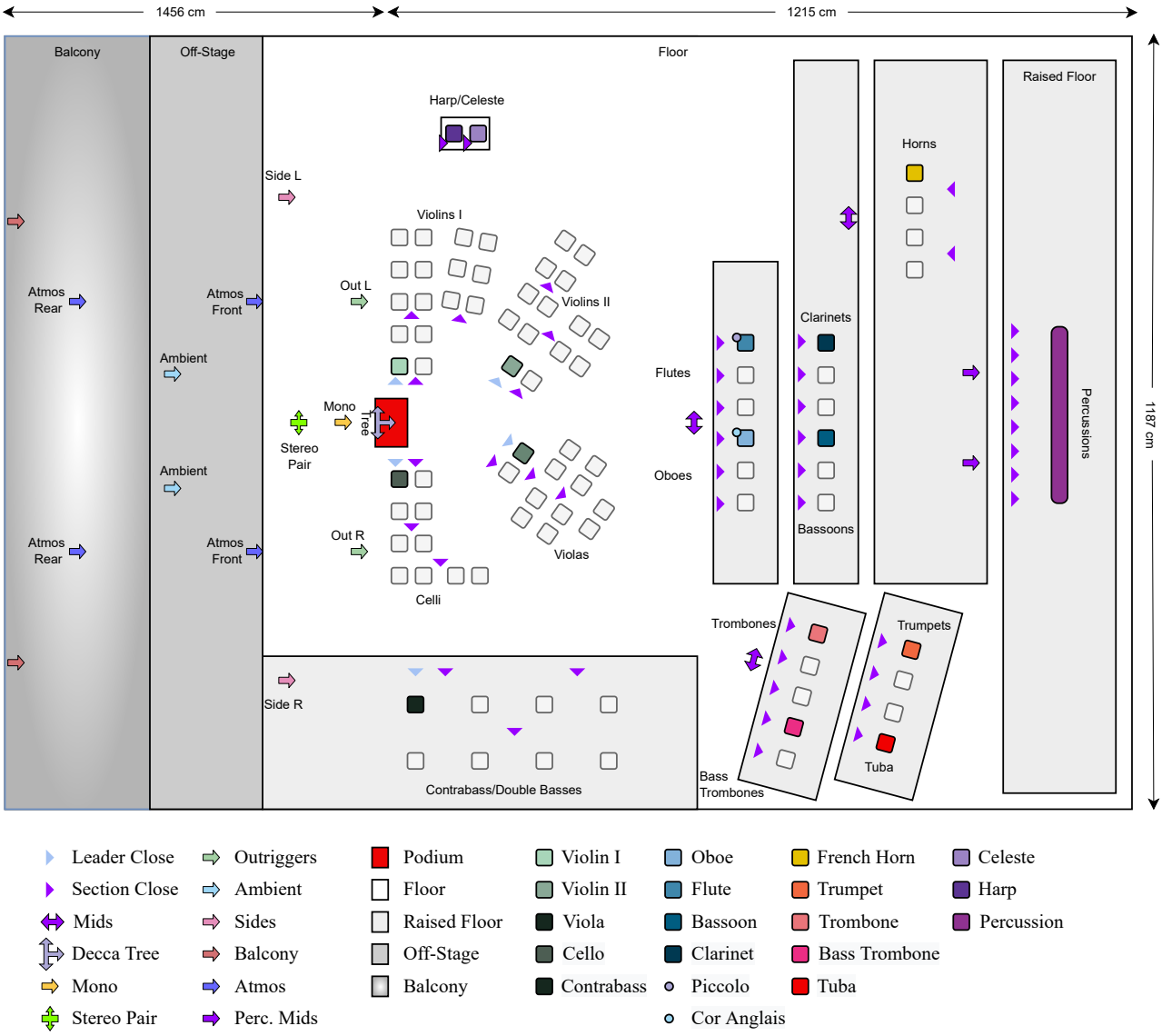


Figure 1. Recording configuration for the Spitfire Audio BBC Symphony Orchestra sample library depicting the placement of individual microphones, performers. The scale of the image is not uniform as the off-stage elements are placed further than they appear.

3.1.2 Mutopia

The Mutopia project [9] is a publicly sourced and manually verified free content sheet music library. The sheet music scores are manually annotated from old scores that are now public domain, and digitally archived using the lilypond format which can be converted to MusicXML and MIDI. This library has a large collection of string ensembles, of which we utilise 71 pieces (5h 5m 35s)[2] comprising a variety of chamber ensembles primarily composed of string quartets but also including other instruments such as Trumpet, Horn, Oboe, Clarinet, Flute and Bassoon. Although all the lilypond files come with their standard MIDI conversions, we utilise the lilypond to MusicXML conversion python library [33], to preserve the articulation information present in the lilypond files. For the files which we are able to convert to MusicXML, we import them to Dorico, where these articulations are translated to keyswitches (described in Table 2) and rendered to MIDI format which can then be utilised by the BBCSO plugin when rendered on Reaper [32].

3.1.3 Data Cleaning

Many of the scores used to render our dataset contain instruments that are absent (e.g., piano, vocals) in our sample library. Since the intent of EnsembleSet is to generate realistic renders of instruments performing and being recorded in the same physical space, we chose to remove the incompatible instruments as rendering them using other plugins will not be consistent. While converting Mutopia based files using the lilypond to MusicXML conversion tool, many files resulted in erroneous MusicXML files. For the corrupted conversions, we used the MIDI files directly from the database and were unable to preserve articulation for those pieces. For these pieces we use the same

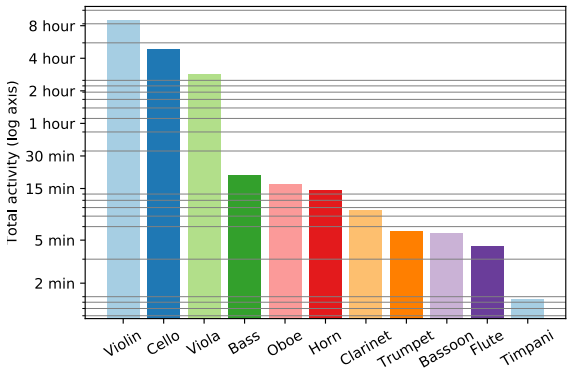


Figure 2. Instrument wise activity duration in EnsembleSet

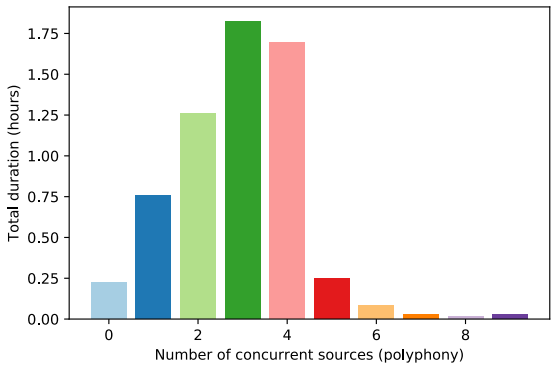


Figure 3. Polyphony distribution across EnsembleSet

articulation generation pipeline as the RWC sourced files. For some other files where the errors were minor (incorrect timing and track assignments), we used their corresponding MIDI files from the library to manually inspect and correct these MusicXML files.

3.2 BBC Symphony Orchestra Sample Library

This library was developed in partnership between BBC Studios and Spitfire Audio, by capturing a full orchestra as sections as well as individual section leaders. Each instrument was recorded for each note in a variety of articulation modes using multiple microphones placed at different positions in the room. For shorter notes, multiple iterations were recorded which are rendered in a round-robin fashion to simulate microtiming variations of real performers. The sample library was recorded in the same fashion as a film score would be recorded in a studio with a multi-microphone setup that enables the capture of each performer from different perspectives in the room. This is what allows us to simulate high quality recordings of chamber ensemble pieces from digital music scores, rendered using individual section leaders.

No.	Render Name	Type	# Mics	Pan
1	Mono	Bidirectional	1	Mono
2	Leader	Unidirectional	1	Stage Pan
3	Decca Tree	Omnidirectional	3	Stereo
4	Outriggers	Omnidirectional	2	Stereo
5	Ambient	Omnidirectional	2	Stereo
6	Balcony	Omnidirectional	2	Stereo
7	Stereo Pair	Coles 4038	2	Stereo
8	Mids	Omnidirectional	2	Stereo
9	Sides	Omnidirectional	2	Stereo
10	Atmos Front	Omnidirectional	2	Stereo
11	Atmos Rear	Omnidirectional	2	Stereo
12	Close	Unidirectional	1	Stage Pan
13	Close Wide	Unidirectional	1	Mono
14	Spill String	Unidirectional	15	Stage Pan
15	Spill Brass	Unidirectional	11	Stage Pan
16	Spill Woodwind	Unidirectional	12	Stage Pan
17	Spill Percussion	Unidirectional	10	Stage Pan
18	Spill Full	Unidirectional	48	Stage Pan
19	Mix 1	Mix	12	Stage Pan
20	Mix 2	Mix + FX	12	Stage Pan

Table 1. List of available renders in EnsembleSet. It must be noted that the Leader microphone is only available for string instruments.

3.2.1 Microphone Renders

The BBCSO sample library provides an entire portfolio of recording stems/microphones that a mix engineer would rely on while producing orchestral scores including traditional mixing setups like decca tree, outriggers, ambient microphones, far balcony mics, side mics and more modern setups including Atmos front and back mics placed at a height in the room. The placement of the individual microphones and each performer is shown in Figure 1. These recorded samples not only preserve the timbral changes that occur for a source recorded at various positions based on their distance, microphone type and directionality, but also preserves the phase shifts that occur across these different microphones placed at different distances w.r.t. the sources. The recorded samples are rendered without any time-correction, which implies that different mic renders would have different time delays and phase shifts based on the distance between the source and the mic. The renders also realistically reflect the timing adjustments performers for different sections make based on their instruments dynamics and position on stage.

It must be noted that in EnsembleSet, the positions for the close microphones are unique for each instrument, but the remaining room microphones are common across all instruments. Thus to simulate a realistic microphone bleed scenario, one can simply render each source at any given room microphone and downmix the resulting instrument stems. On the other hand, downmixing the close microphones would simulate the more typical scenario of music separation from a mixed song. Further details about individual microphone/mix setups can be found in Table 1.

3.2.2 Mixes

Apart from the individual microphone stems, the plugin also provides two professionally mixed stems. Mix 1 is a

Switch	Strings	Horn & Trumpet	Flute & Clarinet	Oboe & Bassoon
C-1	Legato	Legato	Legato	Legato
C#-1	Long	Long	Long	Long
D-1	Long Con Sordino	Staccatissimo	Trill Major 2nd	Trill Major 2nd
D#-1	Long Flautando	Marcato	Trill Minor 2nd	Trill Minor 2nd
E-1	Spiccato	Long Cuivre	Staccatissimo	Staccatissimo
F-1	Staccato	Long Sforzando	Tenuto	Tenuto
F#-1	Pizzicato	Long Flutter	Marcato	Marcato
G-1	Col Legno	Multi-tongue	Long Flutter	Multi-tongue
G#-1	Tremolo	Trill Major 2nd	Multi-tongue	-
A-1	Trill Major 2nd	Trill Minor 2nd	-	-
A#-1	Trill Minor 2nd	Long (muted)	-	-
B-1	Long Sul Tasto	Staccatissimo (muted)	-	-
C0	Long Harmonics	Marcato (muted)	-	-
C#0	Short Harmonics	-	-	-
D0	Bartok Pizzicato	-	-	-
D#0	Marcato	-	-	-

Table 2. List of keyswitch-articulation mappings for different instruments.

general starting point for a mix engineer with a good balance of the commonly used microphones like Decca Tree, Outriggers, Ambient, Balcony, Mids and Close mics. Mix 2 provides a more intense sound with some added compression, EQ and reverb. These stems are ideal to simulate the typical music separation scenario as the mixes provided present a good simulation of an unmastered and a mastered mix for an orchestral ensemble.

3.3 Articulation Automation

The BBCSO plugin allows rendering each note in a variety of articulations that are particular to each instrument. We use Dorico which in case of importing scores as MusicXML files, is capable of mapping articulations from MusicXML to keyswitches in the -1 octave in MIDI. Alternatively if articulations are unavailable, as is the case for importing scores as MIDI files, Dorico automatically selects either staccato or long articulation based on individual note lengths with a crossover at 187.5ms (16th note at 80bpm). The list of keyswitches and articulation mappings for each of the instruments available in EnsembleSet is shown in Table 2.

3.4 Dataset Contents

EnsembleSet contains a total of 6 hours and 9 minutes of multi-instrument, multi-mic data and is available on Zenodo [3] . The resulting total active duration of each instrument in EnsembleSet can be seen in Figure 2. The dataset presented is focused around string ensembles, and each of the 80 tracks presented in the dataset contains at least one string instrument, while the majority of pieces comprise string quartets. EnsembleSet also contains other woodwind and brass instruments, although their distribution is rather sparse. The overall polyphony distribution across the dataset is shown in Figure 3. Each song is also paired with its accompanying MIDI file which was used to generate the renders, and also contains the articulation information. Our implemented data preprocessing (described in section 4.2), data augmentation pipeline and other metadata related to the tracks such as song title, author, instrumentation and audio examples are available online [4] .

3.5 Limitations

While we have tried our best to make the synthesised recordings sound as realistic as possible, the achieved quality was still limited by the amount of information available in the source MIDI/lilypond files. All of the 9 tracks sourced from the RWC database have very good dynamics and realistic tempo variations in the renders, but since the source data was MIDI, the articulations are limited to long, staccato and pizzicato. For the 71 songs sourced from Mutopia, we were able to render from MusicXML for 30 of them, thus these are the only songs that are able to map to all possible articulations present in the source sheet music. For the remaining 41 tracks which were rendered from MIDI, the articulations are similarly limited to long, staccato and pizzicato. For all the songs sourced from Mutopia, the dynamics mapping available was limited due to limitations of the source lilypond format, thus resulting in each note having only one of 3 levels of velocity. While the instrument names in the renders have been standardised across the dataset, the accompanying MIDI files provided with each of the renders do not have standardised track names as they were preserved from the original track names from the source MIDI/Lilypond files.

4. EXPERIMENTS

To exhibit the value of our synthesised dataset, we use EnsembleSet to train a source separation model that is able to separate any chamber ensemble duet as explored in [30]. While we are training our model exclusively on our generated data, we evaluate on real-world data from the URMP dataset [20]. We make use of the multi-mic renders that are available in EnsembleSet as a form of data augmentation by randomising the mix/mic(s) presented to the

[3] https://zenodo.org/record/6519024

[4] http://c4dm.eecs.qmul.ac.uk/EnsembleSet/

Model	Train	Eval	SDR	SI-SDR
MSI [30]	URMP	URMP	+6.33 dB	-
DPTNet	URMP	ES	+6.29 dB	+4.37 dB
DPTNet	ES	URMP	+11.37 dB	+9.06 dB
DPTNet	ES	ES	+14.17 dB	+12.87 dB

Table 3. 2-source Chamber Ensemble Separation results.

model at each epoch. In addition, we use other augmentations including pitch shift and gain modulation to help the model generalise better to unseen source/microphone configurations. We utilise the same architecture as presented in [11] which is based on [10], modified to accommodate for 44.1kHz sample rate audio.

4.1 Model

We utilise the Dual-path Transformer (DPTNet) [10] based architecture using PIT [27] and modify the filterbank, scheduler and other network parameters to accommodate input segments at a sampling rate of 44.1kHz. Our model takes 2.97 second input frames (131072 samples) with 8 repeating separator units. We define the 1-D encoder filterbank to have a filter length of 32 samples with a hop size of 4 samples which resulted in best results in our experiments. Utilising a PIT loss for monotimbral ensembles is particularly well suited, as this enables our model to be able to separate any two monotimbral instruments regardless of their class.

4.2 Data

We train the model using all possible combinations of chamber ensemble duets playing simultaneously from EnsembleSet (ES) amounting to about 53 hours of data. To achieve this we implemented a novel dataloader that measures instrument activity confidence for each instrument track and identifies pairs of instrument segments where both the sources have some overlapping activity in all possible combinations (for eg: a string quartet piece for 2 source separation can be used as 6 different pairs of string duets). We used the URMP dataset (URMP) [20] to generate real examples for cross-validation and testing in a similar fashion resulting in 4.5 hours of 2 source mixtures. We utilise torch-audiomentations [34,35] for data augmentation such as gain modulation, channel swap and pitch-shifting by up to +/- 2 semitones. We also use the multi-mic renders of each instrument track as data augmentation by randomly choosing one of the 20 renders for each instrument for each iteration. It must also be noted that we maintain temporal and harmonic integrity of the mixtures through all the data augmentations. This is unlike the typical music separation data augmentation pipeline where the constituent parts of the mixtures are randomised across different songs at every epoch during training [36].

4.3 Training

We train the models for 100 epochs with early stopping patience of 10 epochs. We start with a learning rate of $5 \times e^{-3}$ with a scheduler that halves the learning rate if the validation loss does not improve for 3 epochs. We train the models on 4 x NVIDIA A100 GPUs using a distributed data parallel back-end. Each epoch in our experiments took 40 minutes with a batch size of 1 per GPU.

5. RESULTS

We present our baseline results based on the experiments described above and compare it to previous experiments conducted for a similar task as described in [30]. The results from [30] are based on a zero-shot learning + multi-task source informed (MSI) separation model designed to tackle the limitation of a very small training dataset. We compare our model's cross-dataset evaluation performance between the URMP Dataset [20] and EnsembleSet (ES) with the experiments from [30] as shown in Table 4.1. We find that our model trained on URMP and tested on ES reports similar separation quality as the MSI experiments from [30], although the test sets were not identical. The same model trained on ES and tested on URMP reports an improvement of 5dB in separation quality.

6. CONCLUSION

In this paper we introduced a new dataset constructed using digitised chamber ensemble scores and a professional orchestral sample library to address the lack of multitrack chamber ensemble datasets. We described our data generation process and data augmentation methods to enable generalisable deep learning solutions using the same. We provided a baseline for the task of separating 2 monotimbral instruments playing simultaneously and are able to show that models trained exclusively on our synthesised dataset are able to generalise to real world data for the same task. This outcome emphasises the strong dependence of the performance of deep learning on training regimes, in particular the quality of the training dataset.

The presented dataset not only contains high quality multi-microphone renders of various instruments, but is also accompanied by the MIDI files that were utilised for generating this dataset. This paired data can be utilised for various tasks including multi-instrument transcription [12], instrument recognition [13], score-informed source separation [13], microphone simulation [15], and automatic mixing [16].

While PIT is well suited for monotimbral ensemble separation as it can separate any 2 sources regardless of their instrument class, it is limited by polyphony where a model only works for mixtures with a fixed number of sources. In the future we intend to explore source conditioned separation models which would enable separating any particular source from a mixture. Although the efficacy of such solutions in the case of mixtures with multiple instances of same instruments has to be tested. In our current work we perform the separation on single channel audio, but we would like to extend our model to be capable of handling multi-channel audio input and utilise spatial information implicitly during separation.

7. ACKNOWLEDGMENTS

Special thanks to Jake Jackson, Michael Krause, Dave Foster, and Mary Pilataki-Manika for insightful discussions and advice on generating this dataset. S. Sarkar is a research student at the UKRI Centre for Doctoral Training in Artificial Intelligence and Music, supported jointly by UK Research and Innovation [grant number EP/S022694/1] and Queen Mary University of London. This work was supported by a Turing Fellowship for E. Benetos under the EPSRC grant EP/N510129/1. The computations described in this research were performed using the Baskerville Tier 2 HPC service (https://www.baskerville.ac.uk/), funded by the EPSRC and UKRI through the World Class Labs scheme (EP/T022221/1) and the Digital Research Infrastructure programme (EP/W032244/1) and is operated by Advanced Research Computing at the University of Birmingham.

8. REFERENCES

[1] E. Vincent, T. Virtanen, and S. Gannot, *Audio source separation and speech enhancement*. John Wiley & Sons, 2018.

[2] Y. Mitsufuji, G. Fabbro, S. Uhlich, and F.-R. Stöter, "Music demixing challenge at ISMIR 2021," *arXiv e-prints*, pp. arXiv–2108, 2021.

[3] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "MUSDB18-HQ - an uncompressed version of MUSDB18," Dec. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3338373

[4] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 31–35.

[5] C. Weng, D. Yu, M. L. Seltzer, and J. Droppo, "Deep neural networks for single-channel multi-talker speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 10, pp. 1670–1679, 2015.

[6] S. Sarkar, E. Benetos, and M. Sandler, "EnsembleSet," May 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6519024

[7] SpitfireAudio, "User Manual BBC Symphony Orchestra Professional," 2019. [Online]. Available: https://d1t3zg51rvnesz.cloudfront.net/p/files/product-manuals/4126/1648649726/BBCSOPro_Manual_v2.0.pdf

[8] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "RWC music database: Popular, classical and jazz music databases." in *Proceedings of the 2nd International Society for Music Information Retrieval Conference (ISMIR)*, vol. 2, 2002, pp. 287–288.

[9] E. Praetzel, "Mutopia project: Free sheet music for everyone," 2000. [Online]. Available: https://www.mutopiaproject.org/index.html

[10] J. Chen, Q. Mao, and D. Liu, "Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation," *arXiv preprint arXiv:2007.13975*, 2020.

[11] S. Sarkar, E. Benetos, and M. Sandler, "Vocal Harmony Separation Using Time-Domain Neural Networks," in *Proc. Interspeech 2021*, 2021, pp. 3515–3519.

[12] Y.-T. Wu, B. Chen, and L. Su, "Multi-instrument automatic music transcription with self-attention-based instance segmentation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2796–2809, 2020.

[13] H. F. Garcia, A. Aguilar, E. Manilow, and B. Pardo, "Leveraging hierarchical structures for few-shot musical instrument recognition," *arXiv preprint arXiv:2107.07029*, 2021.

[14] S. Ewert, B. Pardo, M. Muller, and M. D. Plumbley, "Score-informed source separation for musical audio recordings: An overview," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 116–124, 2014.

[15] A. Mathur, A. Isopoussu, F. Kawsar, N. Berthouze, and N. D. Lane, "Mic2mic: using cycle-consistent generative adversarial networks to overcome microphone variability in speech systems," in *Proceedings of the 18th international conference on information processing in sensor networks*, 2019, pp. 169–180.

[16] J. D. Reiss, "Intelligent systems for mixing multichannel audio," in *2011 17th International Conference on Digital Signal Processing (DSP)*, 2011, pp. 1–6.

[17] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 2018, pp. 293–305.

[18] S. Rosenzweig, H. Cuesta, C. Weiß, F. Scherbaum, E. Gómez, and M. Müller, "Dagstuhl choirset: A multitrack dataset for mir research on choral singing," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.

[19] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "MedleyDB: A multitrack dataset for annotation-intensive mir research." in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, vol. 14, 2014, pp. 155–160.

[20] B. Li, X. Liu, K. Dinesh, Z. Duan, and G. Sharma, "Creating a multitrack classical music performance dataset for multimodal music analysis: Challenges, insights, and applications," *IEEE Transactions on Multimedia*, vol. 21, no. 2, pp. 522–535, 2018.

[21] C. Böhm, D. Ackermann, and S. Weinzierl, "A multi-channel anechoic orchestra recording of beethoven's symphony no. 8 op. 93," *Journal of the Audio Engineering Society*, vol. 68, no. 12, pp. 977–984, 2021.

[22] Z. Duan and B. Pardo, "Soundprism: An online system for score-informed source separation of music audio," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 6, pp. 1205–1215, 2011.

[23] J. Fritsch and M. D. Plumbley, "Score informed audio source separation using constrained nonnegative matrix factorization and score synthesis," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 888–891.

[24] D. Petermann, P. Chandna, H. Cuesta, J. Bonada, and E. Gómez Gutiérrez, "Deep learning based source separation applied to choir ensembles," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[25] M. Gover and P. Depalle, "Score-informed source separation of choral music," Ph.D. dissertation, McGill University, 2019.

[26] P. Chandna, H. Cuesta, D. Petermann, and E. Gómez, "A Deep-Learning Based Framework for Source Separation, Analysis, and Synthesis of Choral Ensembles," *Frontiers in Signal Processing*, vol. 2, 2022. [Online]. Available: https://www.frontiersin.org/article/10.3389/frsip.2022.808594

[27] D. Yu, M. Kolbæk, Z.-H. Tan, and J. Jensen, "Permutation invariant training of deep models for speaker-independent multi-talker speech separation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 241–245.

[28] Y. Luo and N. Mesgarani, "Tasnet: time-domain audio separation network for real-time, single-channel speech separation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 696–700.

[29] Y. Luo, Z. Chen, and T. Yoshioka, "Dual-path rnn: efficient long sequence modeling for time-domain single-channel speech separation," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 46–50.

[30] L. Lin, Q. Kong, J. Jiang, and G. Xia, "A unified model for zero-shot music source separation, transcription and synthesis," in *Proceedings of 21st International Conference on Music Information Retrieval, ISMIR*, 2021.

[31] Steinberg, "Dorico: Music notation software | steinberg," 2016. [Online]. Available: https://www.steinberg.net/dorico/

[32] Cockos, "Reaper: Digital audio workstation," 2006. [Online]. Available: https://www.reaper.fm/

[33] LilyPond, "python-ly: Python library containing various python modules to parse, manipulate or create documents in lilypond format," 2016. [Online]. Available: https://github.com/frescobaldi/python-ly

[34] I. Jordal, "torch-audiomentations: Audio data augmentation in pytorch," 2021. [Online]. Available: https://github.com/asteroid-team/torch-audiomentations

[35] M. Pariente, S. Cornell, J. Cosentino, S. Sivasankaran, E. Tzinis, J. Heitkaemper, M. Olvera, F.-R. Stöter, M. Hu, J. M. Martín-Doñas *et al.*, "Asteroid: the pytorch-based audio source separation toolkit for researchers," *arXiv preprint arXiv:2005.04132*, 2020.

[36] S. Uhlich, M. Porcu, F. Giron, M. Enenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, "Improving music source separation based on deep neural networks through data augmentation and network blending," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 261–265.

END-TO-END LYRICS TRANSCRIPTION INFORMED BY PITCH AND ONSET ESTIMATION

Tengyu Deng[1] **Eita Nakamura**[1] **Kazuyoshi Yoshii**[1,2]
[1] Graduate School of Informatics, Kyoto University, Japan
[2] PRESTO, Japan Science and Technology Agency (JST), Japan
deng@sap.ist.i.kyoto-u.ac.jp, {eita.nakamura, yoshii}@i.kyoto-u.ac.jp

ABSTRACT

This paper presents an automatic lyrics transcription (ALT) method for music recordings that leverages the framewise semitone-level sung pitches estimated in a multi-task learning framework. Compared to automatic speech recognition (ASR), ALT is challenging due to the insufficiency of training data and the variation and contamination of acoustic features caused by singing expressions and accompaniment sounds. The domain adaptation approach has thus recently been taken for updating an ASR model pre-trained from sufficient speech data. In the naive application of the end-to-end approach to ALT, the internal audio-to-lyrics alignment often fails due to the time-stretching nature of singing features. To stabilize the alignment, we make use of the semi-synchronous relationships between notes and characters. Specifically, a convolutional recurrent neural network (CRNN) is used for estimating the semitone-level pitches with note onset times while eliminating the intra- and inter-note pitch variations. This estimate helps an end-to-end ALT model based on connectionist temporal classification (CTC) learn correct audio-to-character alignment and mapping, where the ALT model is trained jointly with the pitch and onset estimation model. The experimental results show the usefulness of the pitch and onset information in ALT.

1. INTRODUCTION

Automatic lyrics transcription (ALT) refers to a task that aims to estimate the sung texts from music recordings, typically under the presence of accompaniment sounds. Since music composition and sharing have become very popular among non-professional people (e.g., YouTube), the number of non-annotated music data without lyrics transcriptions has been increasing rapidly. ALT has thus gained a lot of attention from the music information retrieval (MIR) community because of its usefulness in karaoke subtitle generation and text-based indexing.

Considering the similarity between ALT and automatic speech recognition (ASR), most studies on ALT have attempted to use ASR techniques, with some modifications if necessary. The hybrid approach based on a hidden Markov model (HMM) enhanced by a deep neural network (DNN), for example, has been used, where only the acoustic model was optimized for ALT [1]. Another way of ALT is to take the end-to-end approach that directly learns a sequence-to-sequence (audio-to-text) mapping. While the connectionist temporal classification (CTC) [2] and/or the attention mechanism [3] have widely been used for ASR [4, 5], the CTC has mainly been used for ALT [6, 7], in conjunction with the attention mechanism [8]. One reason is that the CTC considers only the monotonic audio-to-text alignment, which is relatively easier to infer from a limited amount of training data.

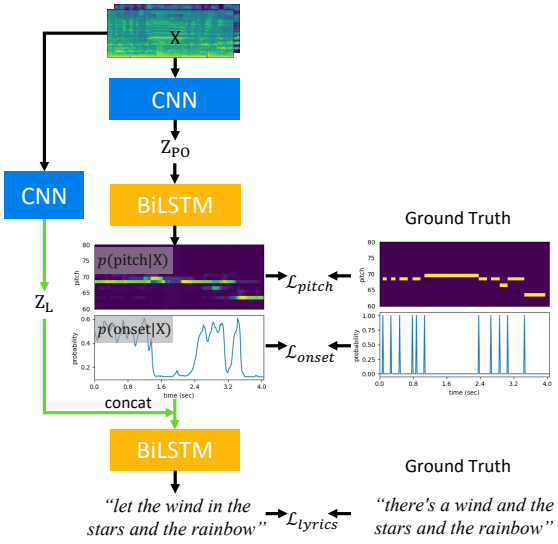


Figure 1. The proposed lyrics transcription method that estimates the pitches and onsets of singing voice and then uses them for character-level lyrics transcription.

The audio-to-text alignment plays a key role in end-to-end ALT and still remains a challenging problem. Firstly, the acoustic characteristics of singing voice vary over time in various ways according to the underlying sung notes and singing expressions. While the phones of speech tend to have particular durations and pitches specific to the speaker, those of singing voice may have time-stretched durations and semi-stepwisely time-varying pitches determined by the singer and the score. Secondly, the acoustic features of singing voice are contaminated by accompaniment sounds

© T. Deng, E. Nakamura, and K. Yoshii. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** T. Deng, E. Nakamura, and K. Yoshii, "End-to-End Lyrics Transcription Informed by Pitch and Onset Estimation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

or distorted by singing voice separation. It is, however, difficult to collect as training data a sufficient amount of music recordings with aligned lyrics annotations that cover a wide variety of recording conditions and singing styles.

To solve these problems, we make effective use of the framewise pitches and onset times of sung notes as acoustic features exclusive for ALT. In musical scores, lyrics are typically described synchronously with notes, i.e., words or characters tend to coincide with notes. This implies that the note onset information can be used for guiding an end-to-end ALT model to efficiently find the correct audio-to-lyrics alignment from a limited amount of training data. Multi-conditional training with the note pitch information is also expected to make the ALT model robust against the pitch variations.

In this paper, we propose an ALT method based on a cascading multi-task architecture that estimates the pitches and onset times of sung notes and then transcribes the lyrics at the character level in a pitch- and onset-conditioned manner (Fig. 1). More specifically, a convolutional recurrent neural network (CRNN) is used for jointly estimating the semitone-level pitches and onset times while eliminating the intra- and inter-note pitch variations (e.g., vibrato and glissando). Informed by this estimation, another CRNN is then used with CTC for learning audio-to-character alignment and mapping. These two CRNNs are trained jointly with backpropagation such that the sum of the pitch, onset, lyrics estimation losses is minimized. To mitigate the data insufficiency problem, we also use a domain adaptation method that fine-tunes a baseline ASR model pre-trained from huge speech data.

The main contribution of this paper lies in the first attempt for joint lyrics and pitch transcription towards comprehensive singing voice analysis. We experimentally show that the score information exclusive to music can be used effectively for finding audio-to-text alignment in end-to-end ALT with insufficient training data.

2. RELATED WORK

This section reviews related work on automatic lyrics transcription (ALT) and singing voice transcription (SVT).

2.1 Automatic Lyrics Transcription

ALT for music recordings under the presence of accompaniment sounds is still a difficult task due to the contamination of acoustic features [6–8]. A standard way of mitigating the adverse effect of accompaniment sounds is to take the two-step approach that performs singing voice separation [9] and lyrics transcription in this order. Although the remarkable improvement has been made in terms of the pure signal processing performance typically measured by the signal-to-distortion ratio (SDR) [10], the separated singing voice, however, is hard to transcribe, *i.e.*, the word error rate (WER) might be low, because an ALT model trained with clean isolated singing voice would suffer from the distortion of acoustic features and the mismatch between the training and test conditions. Although even ALT for clean singing voice [1] has much room for performance improvement due to the considerable variation of acoustic features caused by singing expressions, working directly on music recordings without singing voice separation could achieve better performance of ALT [7].

Inspired by the great success of the end-to-end approach to ASR, several attempts have been made for directly learning the mapping between non-aligned input and output sequences (audio and lyrics) of different lengths [6–8]. At the heart of the end-to-end learning is the audio-to-lyrics alignment based on the connectionist temporal classification (CTC) [2] and/or the attention mechanism [3]. The CTC-based approach estimates the posterior probabilities of labels (*e.g.*, words and characters) at the frame level and aims to maximize the total score obtained by efficiently accumulating the posterior probabilities of all possible *monotonic* alignment paths between the estimated and ground-truth label sequences. The attention-based approach is more powerful in that it can consider non-monotonic alignment and is thus useful for a wider variety of tasks (*e.g.*, machine translation). In general, however, the latter needs a larger amount of training data for finding the correct alignment and is thus hard to apply solely to ALT.

Some studies on audio-to-lyrics alignment have reported the effectiveness of using pitch information [11, 12]. Considering the semi-synchronous relationships between notes and characters, joint estimation of the pitches, onset times, and lyrics of singing voice would help an end-to-end model find the correct audio-to-lyrics alignment.

2.2 Singing Voice Transcription

The ultimate goal of SVT, a special case of automatic music transcription (AMT), is to estimate a human-readable vocal score underlying a given music recording. Towards this goal, a lot of efforts have been made for estimating the continuous fundamental frequencies (F0s) or discrete semitone-level pitches of singing voice at the frame or note level [13,14]. In particular, frame-level F0 estimation (*a.k.a.* melody extraction) has conventionally been considered as a subtask of SVT and intensively studied thanks to the great advance of supervised deep learning [15, 16]. There is, however, a big gap between melody extraction and genuine SVT because accurate scores are hard to obtain just by quantizing continuous F0 contours, typically at the tatum level, due to the large fluctuations and smooth transitions of F0s (*e.g.*, vibrato or glissando).

The latest study on SVT attempted to directly estimate a note sequence with discrete pitches and score positions [17]. This method is based on a hierarchical hidden semi-Markov model (HHSMM) that represents the generative process of observed singing spectra from a latent sequence of notes whose pitches and onset positions are assumed to follow a key-dependent Markov model and a metrical Markov model, respectively. The emission model was implemented with a CRNN that is pretrained to estimate the pitches of singing voice at the frame level from music spectra such that the intra- and inter-note F0 variations are eliminated. This technique forms the basis of the pitch and onset estimators used in our study.

3. PROPOSED METHOD

This section describes the proposed ALT method based on a cascading multi-task architecture.

3.1 Multi-task Learning Approach

Our method takes as input the mel-spectrogram of a music recording and that of the singing voice extracted from the recording with a DNN-based music separation method called Open-Unmix [18]. Since the singing voice separation is known to affect ALT, the original recording is thus used as well as the separated singing voice. Let $\mathbf{X} \in \mathbb{R}^{C \times F \times T}$ be the set of the input mel-spectrograms, where C is the number of channels ($C = 2$ in this paper), F is the number of frequency bins, and T is the number of frames.

We use as a basic building block a convolutional recurrent neural network (CRNN) consisting of a convolutional neural network (CNN) working as an encoder and a recurrent neural network (RNN) working as a decoder. The encoder extracts latent features from the input spectrograms and then the decoder estimates an output sequence while considering the sequential dependency of the latent features. The CRNN consists of residual CNN blocks with skip connections and RNN blocks (Fig. 2). Each CNN has a rectified linear unit (ReLU) and implemented with instance normalization. In contrast, each RNN block is a bidirectional LSTM (BiLSTM) [19] with layer normalization.

As shown in Fig. 1, our method uses two CRNNs. One CRNN is used for jointly estimating the posterior probabilities of the semitone-level pitches and those of the onset presence at the frame level. Given the estimated pitch and onset probabilities, the other CRNN is used for transcribing the lyrics from the spectrograms. Specifically, the estimated pitch and onset probabilities are fed together with the latent features extracted from the CNN into the RNN. Both CRNNs are trained jointly such the sum of the framewise pitch and onset estimation losses and the CTC-based lyrics transcription loss is minimized.

3.1.1 *Pitch and Onset Estimation*

The goal of pitch and onset estimation (supplementary task) is to estimate the framewise pitch and onset probabilities, denoted by $p(\text{pitch}|\mathbf{X}) \in \mathbb{R}^{K \times T}$ and $p(\text{onset}|\mathbf{X}) \in \mathbb{R}^{1 \times T}$, respectively, from the input data $\mathbf{X} \in \mathbb{R}^{C \times F \times T}$, where K is the number of the semitone-level pitches corresponding to MIDI note numbers plus no-pitch ($K = 128 + 1$).

Specifically, $\mathbf{X}$ is first fed to the series of multiple residual CNN blocks as follows:

$$\mathbf{Z}_{\text{PO}} = \text{CNN}(\mathbf{X}), \tag{1}$$

where $\mathbf{Z}_{\text{PO}} \in \mathbb{R}^{C' \times F \times T}$ is the output of the CNN and C' is the number of channels. Note that the zero padding is performed so that the output of each residual CNN block retains the shape of $\mathbf{X}$ except for the number of channels. Then $\mathbf{Z}_{\text{PO}}$ is reshaped into $\mathbf{Z}'_{\text{PO}} \in \mathbb{R}^{C'F \times T}$ and fed to the RNN as follows:

$$\mathbf{Y}_{\text{PO}} = \text{RNN}(\mathbf{Z}'_{\text{PO}}), \tag{2}$$

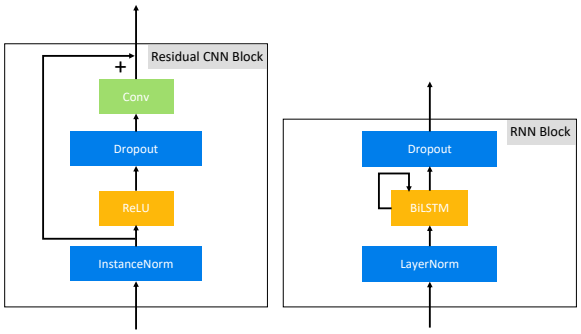


Figure 2. The residual CNN block and the RNN block.

where $\mathbf{Y}_{\text{PO}} \in \mathbb{R}^{C'' \times T}$ is the output of the RNN and C'' is the number of channels. Finally, $p(\text{pitch}|\mathbf{X})$ and $p(\text{onset}|\mathbf{X})$ are computed by feeding $\mathbf{Y}_{\text{PO}}$ to a fully-connected (FC) layer and the softmax and sigmoid functions, respectively, as follows:

$$[\mathbf{Y}_{\text{pitch}}, \mathbf{Y}_{\text{onset}}] = \text{FC}(\mathbf{Y}_{\text{PO}}), \tag{3}$$

$$p(\text{pitch}|\mathbf{X}) = \text{softmax}(\mathbf{Y}_{\text{pitch}}), \tag{4}$$

$$p(\text{onset}|\mathbf{X}) = \text{sigmoid}(\mathbf{Y}_{\text{onset}}), \tag{5}$$

where $\mathbf{Y}_{\text{pitch}} \in \mathbb{R}^{K \times T}$ and $\mathbf{Y}_{\text{onset}} \in \mathbb{R}^{1 \times T}$ are the intermediate outputs from the FC layer.

3.1.2 *Lyrics Transcription*

The goal of lyrics transcription is to estimate the framewise character probabilities, denoted by $p(\text{character}|\mathbf{X}) \in \mathbb{R}^{V \times L}$ from the input data $\mathbf{X} \in \mathbb{R}^{C \times F \times T}$, where V is the number of characters (dictionary size) including a special blank label used for CTC ($V = 33 + 1$ in this paper).

Specifically, in the same way as pitch and onset estimation, $\mathbf{X}$ is first fed to the series of multiple residual CNN blocks as follows:

$$\mathbf{Z}_{\text{L}} = \text{CNN}(\mathbf{X}), \tag{6}$$

where $\mathbf{Z}_{\text{L}} \in \mathbb{R}^{C' \times F \times T}$ is the output of the CNN. Then $\mathbf{Z}_{\text{L}}$ is reshaped into $\mathbf{Z}'_{\text{L}} \in \mathbb{R}^{C'F \times T}$, stacked with the estimated pitch probabilities $p(\text{pitch}|\mathbf{X}) \in \mathbb{R}^{K \times T}$ and onset probabilities $p(\text{onset}|\mathbf{X}) \in \mathbb{R}^{1 \times T}$, and fed to the RNN as follows:

$$\mathbf{Y}_{\text{L}} = \text{RNN}([\mathbf{Z}'_{\text{L}}, p(\text{pitch}|\mathbf{X}), p(\text{onset}|\mathbf{X})]), \tag{7}$$

where $\mathbf{Y}_{\text{L}} \in \mathbb{R}^{C'' \times L}$ is the output of the RNN and C'' is the number of channels. For computational simplicity, $\mathbf{Z}_{\text{L}}$ as well as $p(\text{pitch}|\mathbf{X})$ and $p(\text{onset}|\mathbf{X})$ are downsampled to $T/2$ frames with a 2-dimensional max-pooling layer. Finally, $p(\text{character}|\mathbf{X})$ is computed by feeding $\mathbf{Y}_{\text{L}}$ to a fully-connected (FC) layer and the softmax function as follows:

$$\mathbf{Y}_{\text{character}} = \text{FC}(\mathbf{Y}_{\text{L}}), \tag{8}$$

$$p(\text{character}|\mathbf{X}) = \text{softmax}(\mathbf{Y}_{\text{character}}), \tag{9}$$

where $\mathbf{Y}_{\text{character}} \in \mathbb{R}^{V \times T}$ is the intermediate output from the FC layer.

3.1.3 Joint Training with Domain Adaptation

The CRNN used for pitch and onset estimation and that for lyrics transcription are mutually dependent and thus trained jointly such that the sum of the framewise pitch and onset estimation losses (cross-entropies) and the lyrics transcription loss (CTC loss) is minimized. The CTC loss is computed from the framewise estimate $p(\text{character}|\mathbf{X})$ by efficiently accumulating the costs of all possible character sequences that can be reduced to the ground-truth character sequence by removing the blank labels. The pitch and onset probabilities $p(\text{pitch}|\mathbf{X})$ and $p(\text{onset}|\mathbf{X})$ and the underlying boundary information are considered to make $p(\text{character}|\mathbf{X})$ consistent with the ground-truth character sequence.

To mitigate the insufficiency of music data with lyrics annotations (DALI dataset [20]), we take the domain adaptation approach based on transfer learning [21]. Specifically, the CRNN used for lyrics transcription is trained using sufficient speech data (LibriSpeech corpus [22]) and then fine-tuned using both speech and music data.

3.2 Decoding

At run-time, we aim to estimate a series of note events with semitone-level pitches (MIDI note numbers) and onset times (frames) and transcribe the lyrics (Fig. 3). Specifically, the pitches with the maximum probabilities are taken from the estimated pitch probabilities $p(\text{pitch}|\mathbf{X})$. The onset times are determined with a peak picking strategy [23] with a window of 50 [ms] and a threshold of 0.4. A frame is counted as the onset time if the onset probability at this frame is maximal within 25 [ms] around this frame, where frames whose probabilities are less than 0.4 are excluded. The lyrics (best character sequence) are determined using a CTC decoder based on beam search [24] with a beam size of 25 frames and a 5-gram language model trained on the LibriSpeech corpus with a vocabulary of 200K words.

4. EVALUATION

This section reports a comparative experiment conducted for evaluating the effectiveness of the multi-task learning with pitch and onset estimation in ALT.

Details about data selection and training conditions can be found in the GitHub repository [1] of this paper.

4.1 Experimental Conditions

We explain the data used for evaluation, network configurations, compared methods, and evaluation measures.

4.1.1 Data

We used the DALI dataset [20] consisting of 5358 pieces of popular music in the English language along with fine-grained lyrics and pitch annotations. This dataset was made by collecting karaoke subtitle data and then searching for the corresponding audio data on the Internet, where an automatic alignment method was used for obtaining aligned

[1] https://github.com/TengyuDeng/lyrics-transcription-with-pitch-onset/

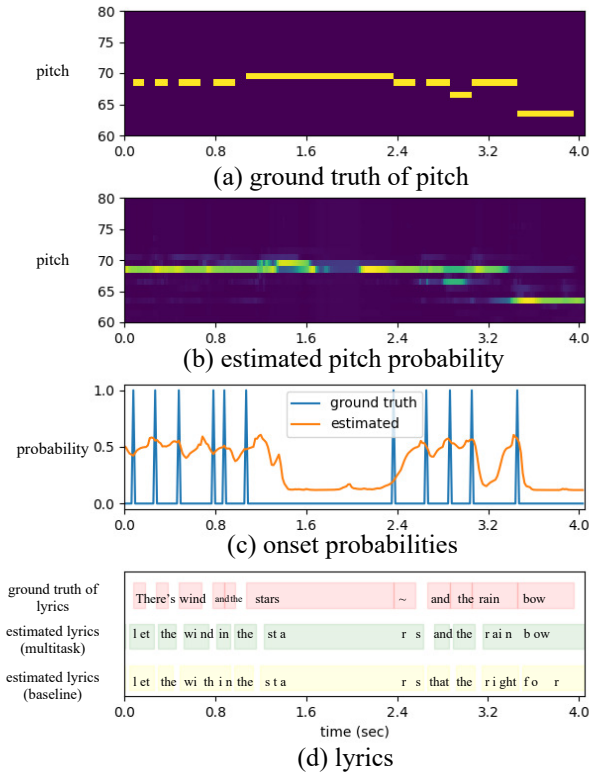


Figure 3. Example of estimations on a segment of a popular song from DALI. In (d), an estimated character is placed where the CTC decoder responds with the peak probability.

annotations. This dataset thus suffers from severe annotation errors [25]. First, the original karaoke data contained many problematic annotations such as global pitch shifts, wrong spellings, and onset time errors. Second, the automatic global alignment method often failed.

We thus filtered out music data with obviously wrong lyrics annotations and/or global pitch shift errors. The data selection procedure was based on the pitch estimation model and RWC Popular Music Database [26, 27]. This dataset contains 80 Japanese songs and 20 English songs, all of which are original popular songs and are provided with careful manual annotations. For simplicity, we used the framewise error rate given by

$$1 - \frac{\#\{\text{Frames Estimated Correctly}\}}{\#\{\text{Total Frames}\}}. \qquad (10)$$

We chose the 80 Japanese tracks of the RWC database and split them into a training set of 64 tracks and a test set of 16 tracks. The CRNN-based pitch estimation was trained on the training set, and attained 26.05% error rate on the test set, which was considered as a standard level.

The model trained on the training set from the RWC database was then exploited to test on the whole DALI dataset. The annotation of each song was shifted by -12, -11, ..., 0, ..., 11, and 12 semitones, and the predicted pitches of the pre-trained model were compared to all 25 pitch-shifted versions of annotations. The pitch shift value with the minimal framewise error rate was considered as the global pitch shift value of the annotation for a song, and

the minimal framewise error rate was recorded. Finally, the DALI dataset was filtered by the recorded framewise error rates. Specifically, songs with framewise error rates larger than a certain threshold were considered to be with problematic annotations.

Of all 5358 songs in the DALI dataset, 3272 songs could be accessed in our region and were in the English language. We selected 2515 songs with the data selection procedure, with a threshold of 50%. They were then split into a train set of 2263 songs, a validation set of 125 songs, and a test set of 126 songs. The total durations were 148.82 hours, 8.16 hours, and 8.47 hours, respectively.

For the transfer learning procedure, we used the LibriSpeech corpus [22] that contains 1000 hours of reading English speech sampled at 16 kHz. The CRNN described in Section 3.1.2 was trained on the train-clean-360 and train-other-500 sets of the LibriSpeech corpus, where all-zero matrices were used as $p(\text{pitch}|\mathbf{X})$ and $p(\text{onset}|\mathbf{X})$ in Eq. (7). This model was then fine-tuned with the training set of the DALI dataset.

4.1.2 Configurations

As described in Section 3.1, the audio signals, sampled at a rate of 48 kHz, were first separated into singing-voice-only signals with model umxhq from open-unmix [18]. Then for computational simplicity, the separated signals and the original mixed signals were resampled to 16 kHz, before they were converted to mel-spectrogram features, respectively. The audio signals were converted to mel-spectrogram with a window size of 32 ms and a hop length of 16 ms, and the resulted mel-spectrogram contained 80 mel-scaled features. We clipped the mel-spectrograms into pieces of 1000 frames, with a duration of about 16 s. Therefore, following the annotations in 3.1.1, we had $C = 2, F = 80, T = 1000$.

In the pitch and onset estimation network, 6 residual convolution blocks were stacked. The kernel sizes were (5,5), (5,5), (3,3), (3,3), (3,3), (1,1), and the numbers of output channels were 64, 32, 32, 32, 32, 1, respectively. After that, only 1 RNN block was applied to obtain the pitch and onsest estimation results. The dropout probability was set to 0 in each layer. In other words, we didn't adopt dropout in the pitch and onset estimation network.

In the lyrics transcription network, 6 residual convolution blocks were stacked. The kernel sizes were (5,5), (5,5), (3,3), (3,3), (3,3), (3,3), and the numbers of output channels were 64, 32, 32, 32, 32, 16, respectively. After that, 3 RNN blocks were stacked, and the number of hidden units in each LSTM was 512. The dropout probability was set to 0.2 in each layer.

When training the model, the Adam optimizer was used, and the parameters were $\beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The model was trained with a learning rate of 5×10^{-4}, and a warming up strategy was used. The learning rate linearly increased from 0 to 5×10^{-4} for the first 100 batches. We used an early stop strategy to manage the learning process. As mentioned in Section 4.1.1, the dataset was split into a training, a validation, and a test set. After training for an epoch, the model was tested on the validation set, and the learning process was terminated when the test statistics for the validation set stopped improving for 10 epochs.

Method	DALI-test	Jamendo
Ours (baseline)	69.22	77.3
Ours (oracle)	**64.41**	/
Ours (multi-task)	68.29	**76.2**
[6]	/	77.8
[7]	/	87.9

Table 1. Comparison of WER (%) with different methods.

4.1.3 Compared Methods

In order to test the performance of our system, we compared a couple of different settings. We also compared the proposed system with previous related works.

For lyrics transcription, in addition to the multi-task architecture, we also trained a model using zero dummy pitch and onset probabilities as a baseline model. Besides, a model was also trained using the ground truth pitches and onset times as oracle information. In order to compare with related works, the multi-task architecture was also tested on the jamendo dataset [6], and the results were compared with the end-to-end models in [6] and [7].

For pitch and onset estimation, we trained the model in Section 3.1.1 without the lyrics transcription part as the baseline model, and the results were compared with that in the multi-task scenario. We also evaluated the test data on VOCANO [28], a note-level vocal melody estimation toolkit available in public.

4.1.4 Evaluation Measures

The lyrics transcription was evaluated using word error rate (WER). The pitch and onset estimation was evaluated using the method first proposed in [29]. We considered the Correct Onset (COn) and the Correct Onset, Pitch (COnP) measures.

4.2 Experimental Results

Before fine-tuning on lyrics data, the lyrics transcription model was trained on LibriSpeech ASR corpus. The model reached a WER of 6.56% on the test-clean dataset and 20.86% on the test-other dataset.

WERs on the DALI-test dataset and the jamendo dataset are shown in Table. 1. On the DALI-test dataset, where the ground truth of pitch and onset information was available, the model reached the best performance when provided with this ground truth information. This shows that correct pitch and onset information can guide the system to find the correct alignment, so that the performance can be increased. In the multi-task architecture, the lyrics transcription model was trained with joint pitch and onset estimation. Although not as good as the oracle-given situation, the performance still gained some improvement. On the jamendo dataset, our multi-task architecture achieved

	COn			COnP		
	precision (%)	recall (%)	F value (%)	precision (%)	recall (%)	F value (%)
Ours(baseline)	53.21	**30.99**	**38.77**	36.92	**21.49**	**26.90**
Ours(multi-task)	**59.84**	28.69	38.41	**40.49**	19.57	26.14
VOCANO [28]	18.78	20.45	19.07	7.46	7.71	7.40

Table 2. Comparison of pitch and onset estimation results.

a similar improvement compared with our baseline model and beat main previous end-to-end ALT systems.

Table. 2 shows the pitch and onset evaluation results. Compared to the baseline model, the pitch and onset estimation jointly trained with the lyrics transcription model remained the same performance. However, for both the COn and COnP statistics, the multi-task scenario had a higher precision but a lower recall value than the baseline model. This shows that being jointly trained with the lyrics transcription model, especially the onset estimation was guided to be in favor of more confident onset positions. This lead to higher precision and lower recall values. It is notable that our models, both the baseline and the multi-task scenario, also gained better results than the results obtained when the VOCANO system was applied to the same test dataset.

5. CONCLUSION

This paper has presented a neural ALT method based on a multi-task learning architecture that estimates the pitch and onset information jointly and then transcribes the lyrics at the character level in a pitch- and onset-conditioned manner. The experiment using the DALI dataset showed that joint pitch and onset estimation can improve the performance of lyrics transcription. Although no significant overall improvement was attained in pitch and onset estimation, higher precision but lower recall rates were observed in the multi-task learning scenario. Our future work includes more comprehensive evaluation by gathering reliable data with accurate aligned lyrics and pitch annotations and using a data augmentation technique.

6. ACKNOWLEDGEMENT

This work is supported in part by JST PRESTO No. JPMJPR20CB and JSPS KAKENHI Nos. 19H04137, 20K21813, 21K02846, 21K12187, 22H03661.

7. REFERENCES

[1] E. Demirel, S. Ahlbäck, and S. Dixon, "Automatic lyrics transcription using dilated convolutional neural networks with self-attention," in *Proc. of International Joint Conference on Neural Networks*, 2020, pp. 1–8.

[2] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. of Advances in Neural Information Processing Systems*, 2017.

[4] S. Karita, N. Chen, T. Hayashi, T. Hori, H. Inaguma, Z. Jiang, M. Someki, N. E. Y. Soplin, R. Yamamoto, X. Wang, S. Watanabe, T. Yoshimura, and W. Zhang, "A comparative study on transformer vs rnn in speech applications," in *Proc. of IEEE Automatic Speech Recognition and Understanding Workshop*, 2019, pp. 449–456.

[5] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid CTC/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[6] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 181–185.

[7] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics alignment and transcription in polyphonic music: Does background music help?" in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 496–500.

[8] S. Basak, S. Agarwal, S. Ganapathy, and N. Takahashi, "End-to-end lyrics recognition with voice to singing style transfer," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021, pp. 266–270.

[9] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, D. FitzGerald, and B. Pardo, "An overview of lead and accompaniment separation in music," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1307–1335, 2018.

[10] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis, "mir_eval: A transparent implementation of common MIR metrics," in *Proc. of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2014, pp. 367–372.

[11] J. Pons, R. Gong, and X. Serra, "Score-informed syllable segmentation for a cappella singing voice with convolutional neural networks," in *Proc. of the 18th*

International Society for Music Information Retrieval Conference, Suzhou, China, 2017, pp. 383–389.

[12] J. Huang, E. Benetos, and S. Ewert, "Improving lyrics alignment through joint pitch detection," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2022.

[13] K. S. Rao, P. P. Das *et al.*, "Melody extraction from polyphonic music by deep learning approaches: A review," *arXiv preprint arXiv:2202.01078*, 2022.

[14] J. Salamon, E. Gomez, D. P. W. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.

[15] T.-H. Hsieh, L. Su, and Y.-H. Yang, "A streamlined encoder/decoder architecture for melody extraction," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 156–160.

[16] S. Yu, X. Sun, Y. Yu, and W. Li, "Frequency-temporal attention network for singing melody extraction," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021, pp. 251–255.

[17] R. Nishikimi, E. Nakamura, M. Goto, and K. Yoshii, "Audio-to-score singing transcription based on a crnn-hsmm hybrid model," *APSIPA Transactions on Signal and Information Processing*, vol. 10, 2021.

[18] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix – a reference implementation for music source separation," *Journal of Open Source Software*, 2019.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, November 1997.

[20] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Dali: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm," in *Proc. of the 19th International Society for Music Information Retrieval Conference*, Paris, France, 2018, pp. 431–437.

[21] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010.

[22] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 5206–5210.

[23] S. Böck, F. Krebs, and M. Schedl, "Evaluating the online capabilities of onset detection methods," in *Proc. of the 13th International Society for Music Information Retrieval Conference*, Porto, Portugal, 2012, pp. 49–54.

[24] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014.

[25] G. M. Brocal, R. Bittner, S. Durand, and B. Brost, "Data cleansing with contrastive learning for vocal note event annotations," in *Proc. of the 21st International Society for Music Information Retrieval Conference*, Montreal, Canada, 2020, pp. 255–262.

[26] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, "Rwc music database: Popular, classical and jazz music databases," in *Proc. of the 3rd International Conference on Music Information Retrieval*, Paris, France, 2002.

[27] M. Goto, "Aist annotation for the rwc music database," in *Proc. of the 7th International Conference on Music Information Retrieval*, Victoria, Canada, 2006, pp. 359–360.

[28] J.-Y. Hsu and L. Su, "Vocano: A note transcription framework for singing voice in polyphonic music," in *Proc. of the 22nd International Society for Music Information Retrieval Conference*, Online, 2021, pp. 293–300.

[29] E. Molina, A. M. Barbancho, L. J. Tardón, and I. Barbancho, "Evaluation framework for automatic singing transcription," in *Proc. of the 15th International Society for Music Information Retrieval Conference*, Taipei, Taiwan, 2014, pp. 567–572.

CONTRASTIVE AUDIO-LANGUAGE LEARNING FOR MUSIC

Ilaria Manco[1,2] **Emmanouil Benetos**[1] **Elio Quinton**[2] **György Fazekas**[1]
[1] School of EECS, Queen Mary University of London, London, U.K
[2] Music & Audio Machine Learning Lab, Universal Music Group, London, U.K.
{i.manco, emmanouil.benetos, george.fazekas}@qmul.ac.uk, elio.quinton@umusic.com

ABSTRACT

As one of the most intuitive interfaces known to humans, natural language has the potential to mediate many tasks that involve human-computer interaction, especially in application-focused fields like Music Information Retrieval. In this work, we explore cross-modal learning in an attempt to bridge audio and language in the music domain. To this end, we propose MusCALL, a framework for Music Contrastive Audio-Language Learning. Our approach consists of a dual-encoder architecture that learns the alignment between pairs of music audio and descriptive sentences, producing multimodal embeddings that can be used for text-to-audio and audio-to-text retrieval out-of-the-box. Thanks to this property, MusCALL can be transferred to virtually any task that can be cast as text-based retrieval. Our experiments show that our method performs significantly better than the baselines at retrieving audio that matches a textual description and, conversely, text that matches an audio query. We also demonstrate that the multimodal alignment capability of our model can be successfully extended to the zero-shot transfer scenario for genre classification and auto-tagging on two public datasets.

1. INTRODUCTION

Developing effective methods for finding music is at the core of Music Information Retrieval (MIR). Over the years, many approaches have been proposed to browse, search and discover music through a variety of interfaces. Beyond simple search by metadata, existing music retrieval systems allow to express queries via lyrics [1,2], audio examples [3], videos [4] and humming [5], among others [6–8]. Although each of these query types has its merits, none of these systems supports another popular way of searching for music today: through free-form text. For example, we commonly look for songs by typing text into a search engine [9] or by asking online song naming communities to identify a piece of music we do not have bibliographic information about [10]. It becomes evident then that enabling MIR systems to interpret natural language queries can have far-reaching benefits. This idea is not entirely new to MIR, with prior work [11–13] suggesting similar research directions in the past. So far, however, multimodal systems that integrate natural language have not been widely adopted within the MIR community, possibly due to a lack of suitable datasets or to the practical limitations of NLP methods predating modern language models.

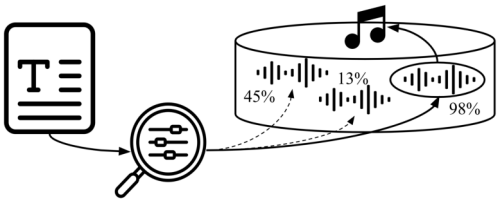


Figure 1: **Illustration of text-based music retrieval**, where audio items in a database are ranked according to their relevance to a free-form language query.

In light of recent breakthroughs in language modelling, we argue that audio-and-language learning has now the potential of closing the semantic gap in MIR [14], providing a bridge between computational representations of music signals and the high-level abstractions needed to use those representations in real-world scenarios. With this in mind, we propose MusCALL, a method for learning alignment between music-related audio and language data via multimodal contrastive learning. Our choice of a contrastive approach is inspired by the recent success of similar methods for joint visio-linguistic modelling (see Section 2.3). In designing MusCALL, we prioritise the ability to perform retrieval at scale and adopt a dual-encoder architecture, where modalities are processed independently. Compared to multimodal architectures with joint encoders and cross-modal attention mechanisms [15], this design allows to share embedding computations among pairs, resulting in a computationally more efficient model.

With this work, we aim to unify audio and language modelling, paving the way for MIR systems that can interpret language-based queries, as illustrated in Figure 1. Our primary contributions can be summarised as follows: (i) we explore multimodal contrastive learning in the context of music-related audio and language for the first time; (ii) we introduce a method for cross-modal retrieval of music, providing the first example of sentence-based music search; (iii) we perform an extensive set of experiments to systematically validate details of our approach and evaluate its performance on popular MIR tasks in a zero-shot setting. [1]

© I. Manco, E. Benetos, E. Quinton, G. Fazekas. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** I. Manco, E. Benetos, E. Quinton, G. Fazekas, "Contrastive Audio-Language Learning for Music", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] Code and supplementary material are available at https://github.com/ilaria-manco/muscall

2. RELATED WORK

2.1 Natural Language Processing in MIR

Prior works in the MIR literature have explored leveraging natural language in the music domain from different angles. Early efforts focused on text as a modality in isolation, adopting NLP techniques to construct knowledge bases from music-related text corpora [16], build semantic graphs for artist similarity from biographies [17], or perform genre classification based on album reviews [18]. Recent efforts, more closely related to the present work, have instead started favouring multimodal approaches. These have explored deep learning with multimodal input data, typically audio combined with text such as reviews or lyrics, for applications as varied as music classification and recommendation [19], mood detection [20], music emotion recognition [21] and music captioning [22–24].

2.2 Audio-Text Cross-modal Learning

Another related line of work is cross-modal learning that leverages audio and tags as text input. Works in this area have proposed contrastive learning-based approaches to enrich audio representations via cross-modal alignment, either for general-purpose audio classification [25, 26] or for music-focused tasks [27]. Differently from our work, these approaches require finetuning on the downstream tasks and none of them directly uses cross-modal representations. Others have explored pre-trained word embeddings in triplet networks to perform tag-based music retrieval via a multimodal embedding space [28, 29]. At a high level, these works share a similar approach to ours and all aim to learn multimodal audio representations by leveraging text. Unlike our work, however, none of them makes use of natural language, using tags as text input instead.

Finally, similarly to our work, [30] also addresses the problem of matching audio and long-form text for music retrieval, but offers a fundamentally different approach, which relies on bridging the audio and text modalities via a common emotion embedding space.

2.3 Learning from Language Supervision

In the previous sections, we have highlighted some research efforts towards bringing together audio and language, but, ultimately, multimodal learning still occupies a marginal role in MIR and has yet to fully enjoy the benefits of modern language models. In adjacent fields such as computer vision, jointly modelling vision and language has instead become a very active area of research, with several successful attempts at using multimodality to develop task-agnostic models that can easily adapt to novel tasks [31–34]. The key insight behind these models is that language captures many of the abstractions humans use to navigate the world and can therefore act as a rich supervisory signal for general-purpose learning, even in tasks that are not directly based on language [35]. Among these works, CLIP [33] is a particularly influential example, having demonstrated for the first time that supervision from natural language can induce highly generic visual representations at scale.

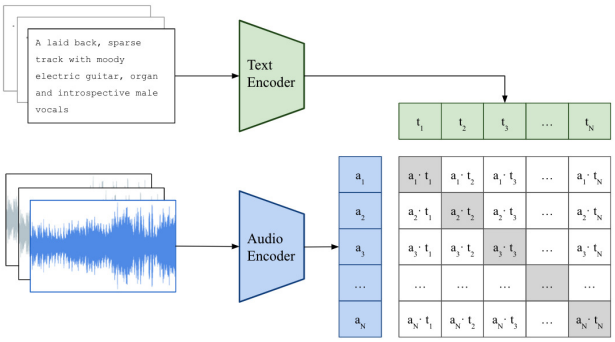


Figure 2: **Overview of MusCALL**. An audio encoder and a text encoder are trained via a contrastive loss to maximise the similarity between representations of N aligned (audio, text) pairs within a mini-batch. At test time, the similarity between embeddings in the learnt multimodal space is used to rank database items and perform cross-modal retrieval.

These breakthroughs suggest that natural language supervision has a large potential beyond the image domain. This has recently prompted the adoption of similar approaches in machine listening, where applications to both music [15] and non-music audio [36, 37] have started to emerge. A subset of works in this area have proposed to directly extend CLIP by incorporating audio as an additional modality [38–40]. Although similar in spirit to our work, these approaches exploit audio-visual correspondences in video, thus requiring large-scale visio-linguistic pre-training. Like these works, we also borrow from CLIP's contrastive dual-encoder approach, but we adapt this framework to the audio modality without any pre-training, while also addressing the specific set of challenges that arise from joint audio-linguistic modelling in the music domain.

3. MUSCALL

3.1 Multimodal Contrastive Learning (MCL)

Contrastive learning has emerged as a powerful paradigm in self-supervised representation learning for images [41] and audio [42–45]. At their core, all flavours of contrastive learning rely on constructing different views of the data and forcing representations of positive pairs of such views to have, on average, higher similarity than those of negative pairs. This encourages a model to learn representations that are *discriminative* with respect to changes to the content and *invariant* to nuisance transformations. In representation learning, this is a useful property, since it results in more robust and effective representations [46–48]. Most formulations of contrastive learning in unimodal scenarios use data augmentations to create multiple views of the inputs. When switching to the multimodal setting, however, different views naturally co-occur in the data, making this a natural testbed for contrastive learning [33, 49, 50].

3.2 Extending MCL to Music and Language

In this work, we explore multimodal contrastive learning (MCL) in the context of audio-linguistic data. More specifically, we consider pairs of music audio tracks and their

associated textual descriptions and aim to learn a model that is able to predict when items from a given pair match, i.e. when they are semantically aligned. An overview of our approach is shown in Figure 2. In practice, our goal is to learn two encoders, $f_a(\cdot)$ for the audio modality and $f_t(\cdot)$ for the text modality, such that for any given (audio, text) pair formed by the tuple (a_i, t_i), the resulting L2-normalised embeddings $\boldsymbol{z}_{a,i} = f_a(a_i)$ and $\boldsymbol{z}_{t,i} = f_t(t_i)$ lie closely in the joint embedding space, with respect to some distance metric, only if a_i and t_i represent similar content. Given our problem definition, we achieve this by optimising a type of N-pair contrastive loss, known as InfoNCE [51]. Based on this, our audio-to-text loss can be defined as follows:

$$\mathcal{L}_{a\to t} = -\frac{1}{N}\sum_{i}^{N}\log\frac{\exp\left(\boldsymbol{z}_{a,i}\cdot\boldsymbol{z}^{+}_{t,i}/\tau\right)}{\sum\limits_{\boldsymbol{z}\in\left\{\boldsymbol{z}^{+}_{t,i},\boldsymbol{z}^{-}_{t,i}\right\}}\exp\left(\boldsymbol{z}_{a,i}\cdot\boldsymbol{z}/\tau\right)}, \quad (1)$$

where N is the batch size, $\boldsymbol{z}^{+}_{t,i}$ and $\boldsymbol{z}^{-}_{t,i}$ are embeddings of positive and negative text samples for item a_i, and τ is a temperature parameter used to scale the similarity scores. By noting that $\mathcal{L}_{t\to a}$ can be defined symmetrically to Eq. (1), the total loss over all (audio, text) pairs in a mini-batch is simply obtained by summing the two losses together:

$$\mathcal{L}_{a\leftrightarrow t} = \mathcal{L}_{a\to t} + \mathcal{L}_{t\to a}. \quad (2)$$

Given a dataset of tuples $\mathcal{D} = \{(a_i, t_i)\}_{i=1:D}$, there are several plausible strategies for constructing positive and negative pairs. One such way, and arguably the most intuitive, is to follow the implicit alignment found in the data. In this case, for each sample i, the positive and negative sets within a mini-batch become: $\boldsymbol{z}^{+}_{x,i} = \{\boldsymbol{z}_{y,i}\}$ and $\boldsymbol{z}^{-}_{x,i} = \{z_{y,j} \mid \forall j \in \{1,\ldots,N\}, j \neq i\}$, where x and y denote the two different modalities. We refer to this sampling strategy as *instance discrimination* [52].

3.2.1 Content-Aware Loss Weighting

Constructing positive and negative samples by instance discrimination is a design choice that relies on two implicit assumptions: firstly, that all items in the dataset are correctly aligned, i.e. that each (audio, text) pair is formed by the most semantically close items amongst all possible pairings; secondly, that all non-aligned pairs represent sufficiently different content and are therefore equally valid as negatives. In a real-world dataset, this is unlikely to hold true in all cases. Intuitively, within a randomly sampled mini-batch, some tracks will share similarities with other tracks, and so will their respective captions.

The above observations suggest that a more informed sampling procedure or an alternative way to define our cross-modal loss may better suit the structural properties of our data. Due to its simplicity, we focus on the latter option and test this hypothesis by introducing some modifications to Eq. (1). By observing that similar captions are likely to correspond to similar audio, we can leverage this side information to estimate the *relevance* between non-paired (negative) items in a mini-batch. Since more relevant items should contribute more to the learning process, similarly to [53], we can introduce a relevance-based weight:

$$w_i = \exp\left(\frac{\frac{1}{N}\sum_{j}^{N}\mathrm{sim}(t_i, t_j)}{\kappa}\right), \quad (3)$$

where κ is a temperature hyperparameter and $\mathrm{sim}(t_i, t_j)$ a similarity score between text sample t_i and text sample t_j. We can then redefine our audio-to-text loss in Eq. (1) as follows:

$$\mathcal{L}^{*}_{a\to t} = -\frac{1}{N}\sum_{i}^{N} w_i \log\frac{\exp\left(\boldsymbol{z}_{a,i}\cdot\boldsymbol{z}^{+}_{t,i}/\tau\right)}{\sum\limits_{\boldsymbol{z}\in\left\{\boldsymbol{z}^{+}_{t,i},\boldsymbol{z}^{-}_{t,i}\right\}}\exp\left(\boldsymbol{z}_{a,i}\cdot\boldsymbol{z}/\tau\right)} \quad (4)$$

and obtain the total loss by summing this to its symmetrical counterpart $\mathcal{L}^{*}_{t\to a}$. We dub this procedure *content-aware loss weighting*, or *loss weighting* for short.

3.3 Combining MusCALL with Audio Self-Supervision

At its core, the design of our approach is centred around audio-text matching. However, we are also interested in learning representations that can be transferred to other tasks, especially in a zero-shot way. Prior work has demonstrated the benefit of using self-supervised learning (SSL) [54] to improve representation quality in a similar setting in the image domain [55]. We hypothesise that our model may also benefit from this and experiment with combining our multimodal contrastive objective with self-supervision on the audio modality. Similarly to the approach proposed in [55], we do this via multi-task learning, using an adaptation of SimCLR [48] to the audio modality as the self-supervised component. Two correlated views of the audio input are produced via a data augmentation pipeline and passed through a shared audio encoder. The SSL objective is then computed on the embeddings resulting from the two views and added to our cross-modal objective as follows:

$$\mathcal{L} = \lambda\mathcal{L}_{SSL} + (1-\lambda)\mathcal{L}_{a\leftrightarrow t}, \quad (5)$$

where $\mathcal{L}_{SSL}$ is the *NT-Xent* loss used in SimCLR [48] and $\lambda \in [0, 1]$ is a scalar weight.

We refer to the SSL-enhanced variant by $\text{MusCALL}_{\text{SSL}}$ to distinguish it from the base variant $\text{MusCALL}_{\text{BASE}}$.

3.4 Audio & Text Encoding

As our audio backbone, we choose ResNet-50 [57] operating on melspectrogram representations of the input and adopt the same architectural modifications introduced in CLIP [33]: 3 stem convolutions followed by average pooling instead of max pooling and anti-aliased blur pooling. We obtain fixed-length audio representations via an attention pooling mechanism: we append the average-pooled audio feature to the backbone output and compute multi-head self-attention, taking the output corresponding to the average-pooled feature as the global audio representation. Also analogously to CLIP, our text encoder is a Transformer [58, 59]. To avoid overfitting on our dataset, which is $\sim$ 2K times smaller than the dataset used to train CLIP,

Method	Text → Audio					Audio → Text				
	R@1	R@5	R@10	mAP10	MedR ↓	R@1	R@5	R@10	mAP10	MedR ↓
DCASE [56]	2.3	10.4	17.4	5.5	50	1.1	5.6	10.1	3.0	84
DCASE + CL	3.9	12.4	18.1	6.8	81.5	2.0	8.6	16.4	4.5	64
MusCALL (ours)	**25.9**	**51.9**	**63.3**	**36.0**	**5**	**25.8**	**53.0**	**63.0**	**35.9**	**5**

Table 1: **Cross-modal retrieval results**. MusCALL improves performance on all metrics by a large margin, compared to two variants of the baseline: one with the original triplet ranking loss (DCASE) and one with our loss (DCASE + CL).

we downsize the network and use 4 hidden layers, as we empirically find this to be the optimal depth (see Figure 3).

Two learned linear projections are applied to map the audio and text features produced by the audio and text backbones onto a 512-dimensional multimodal embedding space respectively. The resulting features are then L2-normalised before their dot product is calculated in the contrastive loss.

4. EXPERIMENTAL DESIGN

4.1 Dataset

We train and evaluate our model on a dataset of 250k (audio, text) pairs created from a production music library. This consists of full-length audio tracks, covering a broad range of genres, and a piece of text describing the overall musical content of each track, as shown in Table 3. For training, validation and testing, we use a random 80/10/10 split.

4.2 Implementation Details

For each audio track, we take a $20s$ random crop at training time, and the central $20s$ segment at testing time. Unless otherwise specified, we then apply a stochastic data augmentation pipeline, where each transformation is applied with an independent probability p. We adopt some of the same transformations, such as noise injection and pitch shift, as prior work on music audio representation learning [45, 60]. For each audio caption in the dataset, we take the text input in full and tokenise it following the same procedure as in CLIP, based on byte pair encoding [61] with a 49K token vocabulary and maximum sequence length of 77.

As an estimate for text-text similarity in the calculation of our loss scaling weight w_i in Eq. (4), we use the cosine distance between L2-normalised embeddings produced by a pre-trained Sentence-BERT [62]. We set the loss weighting temperature parameter κ to 0.005, following [53].

For the SimCLR module in $\text{MusCALL}_{\text{SSL}}$, following [45] we use a 256-dimensional non-linear projection layer and set the temperature parameter to 0.5. The loss scaling parameter λ in Eq. (5) is set to 0.3 as we find this to yield best results.

Together with the parameters for the audio and text encoders and multimodal projections, we also learn the temperature parameter τ in Eq. (1) to avoid tuning it as a hyperparameter. We train using the Adam optimizer, with weight decay 0.2, batch size of 256, initial learning rate of 5e-5, reduced throughout training following a cosine schedule. After training for a maximum of 150 epochs, we select the best model based on the R@10 score (see Section 5.1) computed on the validation set. To save GPU memory, we perform training with automatic mixed precision.

5. EXPERIMENTS & RESULTS

In this section we first describe our experimental setup and report our main results on cross-modal retrieval (Section 5.1), which constitute the focus of the paper. We then explore transferring our model to zero-shot music classification, highlighting key findings (Section 5.2).

5.1 Cross-Modal Retrieval

In cross-modal retrieval, given a query item of modality A, our goal is to identify the matching item of modality B. We can do this by casting the retrieval as a ranking problem: for a given query item of modality A, we rank all candidate items of modality B by the cosine similarity between the query embedding and each candidate embedding. The retrieval performance is then evaluated by computing the Recall at K (R@K) over the testing set as the percentage of correctly retrieved items within the top-K items for each query, with $\text{K} = \{1, 5, 10\}$. In line with previous work on cross-modal retrieval, we also report mean average Precision at 10 (mAP10) and Median Rank (MedR) in our main results. We construct our testing set by randomly sampling a subset of 1000 (audio, text) pairs from our testing split. This is chosen to be consistent in size with other datasets for sentence-based audio retrieval [63, 64].

Results Table 1 shows the performance of MusCALL on the cross-modal retrieval task. We compare this to the baseline system for the *Language-Based Audio Retrieval* subtask of Task 6 in the 2022 DCASE Challenge, [2] trained and evaluated on our dataset. This is a simplified version of the approach proposed in [56] and consists of a pre-trained *word2vec* model [65] as the text encoder and a convolutional recurrent neural network as the audio encoder. Average pooling is used to obtain global representations for each modality from the output of the encoders. These are then jointly trained via a triplet ranking loss [66]. We also consider a variant of this baseline where we use our loss instead of the triplet ranking loss, to provide a closer comparison to our approach.

Our results show that MusCALL significantly outperforms the baseline on both text-to-audio and audio-to-text retrieval. Enhancing the baseline with our loss slightly reduces this margin, bringing a 14.2% and 60.7% average

[2] `https://dcase.community/challenge2022`

Method	Prompt	Genre	Tagging	
		Acc.	ROC	PR
$\text{MusCALL}_{\text{BASE}}$	✗	55.5	**78.0**	28.3
$\text{MusCALL}_{\text{BASE}}$	✓	52.0	72.0	21.0
$\text{MusCALL}_{\text{SSL}}$	✗	58.2	**77.4**	**29.3**
$\text{MusCALL}_{\text{SSL}}$	✓	**62.0**	73.4	23.2

Table 2: **Zero-shot transfer results**. We report $\text{MusCALL}_{\text{BASE}}$ and $\text{MusCALL}_{\text{SSL}}$ accuracy on GTZAN (genre classification), and ROC-AUC and PR-AUC on MTAT (auto-tagging). *Prompt* indicates whether a template was used to wrap the class label.

improvement over the vanilla baseline for text-to-audio and audio-to-text respectively. However, the use of our contrastive loss alone does not account for the full difference, indicating that the design of our text and audio encoders and the use of linear projection layers, play a crucial role.

5.2 Zero-shot Transfer

By design, our cross-modal learning approach endows the model with the ability to interpret arbitrary text inputs. This powerful property can be exploited to perform new tasks based on textual descriptions via a transfer learning paradigm known in the literature as *zero-shot transfer* [33, 34, 67]. Similarly to zero-shot learning, zero-shot transfer aims to learn a model that can generalise to unseen classes or tasks without further training. But, in contrast to zero-shot learning, it somewhat relaxes the requirement that target classes must be completely unseen. Instead, the model is usually pre-trained in a task-agnostic fashion on large amounts of natural language text, and may be exposed to information relevant to the target tasks through this, although no supervised examples are provided.

We explore this paradigm in our work and investigate whether MusCALL exhibits zero-shot transfer capabilities on two tasks, genre classification and auto-tagging. We evaluate this on the most popular public datasets for these tasks, GTZAN [68] and MagnaTagATune (MTAT) [69]. Treating audio clips in each dataset as queries, we obtain classification predictions by considering the similarity scores between audio embeddings and text embeddings of the target labels, similarly to the procedure described for cross-modal retrieval in Section 5.1. Based on this, we calculate accuracy for GTZAN, and area under the receiver operating characteristic curve (ROC-AUC) and area under the precision-recall curve (PR-AUC) for the MTAT dataset.

In order to reduce the distribution shift between pre-training and downstream text input when doing zero-shot transfer, it is common practice to wrap the labels in templates. For example, the label "*rock*" may be wrapped in the sentence "*This is a rock song with electric guitars*", making it much closer to a typical caption encountered in training. Based on evidence that it can improve performance [33], we explore this in our evaluation by passing "*A* [LABEL] *track*" as the text input, but do not further tune the prompts to our model, leaving this for future work.

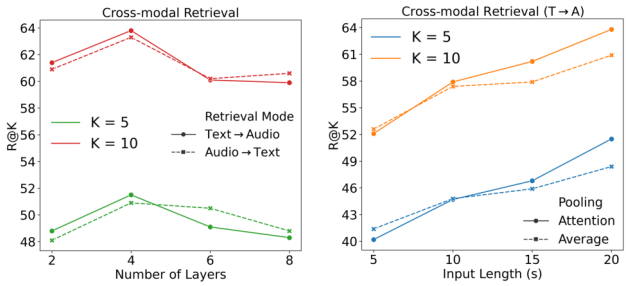


Figure 3: **The effect of varying** (left) the **depth** of the text encoder and (right) the **audio input length** on retrieval.

Results In Table 2 we compare zero-shot performance of two model variants, $\text{MusCALL}_{\text{BASE}}$ and $\text{MusCALL}_{\text{SSL}}$. We find that the self-supervised learning objective yields, on average, better performance, with a 5.4% improvement over $\text{MusCALL}_{\text{BASE}}$. This is not too surprising, as the SSL objective is designed to improve the representation quality of our audio branch and is therefore expected to have a positive effect on generalisation [70]. However, this improvement is not consistent across datasets and tasks, a result that may be attributed to different degrees of similarity between pre-training and downstream datasets, as found in prior work [71]. We also find that zero-shot performance is sensitive to the choice of text prompt. This confirms a well-known phenomenon reported in the literature [33, 72] and suggests that techniques such as prompt tuning and ensembling [33] may lead to further improvements.

Additionally, there are some important differences that should be considered when comparing $\text{MusCALL}_{\text{BASE}}$ and $\text{MusCALL}_{\text{SSL}}$, since the SSL module introduces more activations and overall learnable parameters. This has two implications: firstly, to offset the increased memory footprint while keeping the batch size unchanged and still satisfying our memory constraints, we reduce the size of the input audio to $10s$. We verify that this slightly degrades performance on cross-modal retrieval even in the absence of the SSL module, as shown in Figure 3. Assuming that this trend would be reflected in the zero-shot scenario, we conclude that $\text{MusCALL}_{\text{SSL}}$ would benefit from using longer audio clips. Secondly, the higher number of parameters makes $\text{MusCALL}_{\text{SSL}}$ prone to overfitting, a factor that we believe may also be limiting its performance.

6. ANALYSIS & DISCUSSION

We now discuss the qualitative characteristics of our model (Section 6.1) and examine the contributions of the main design choices through an ablation study (Section 6.2).

6.1 Qualitative Analysis

Similarity score distribution In Figure 4 (left) we show the kernel density estimation of the distributions of pairwise similarity scores in the joint embedding space for positive and negative pairs in our testing set. From this we can see that aligned pairs have distinctly higher scores compared to random ones, confirming that the model distinguishes positive and negative examples with a good level of confidence.

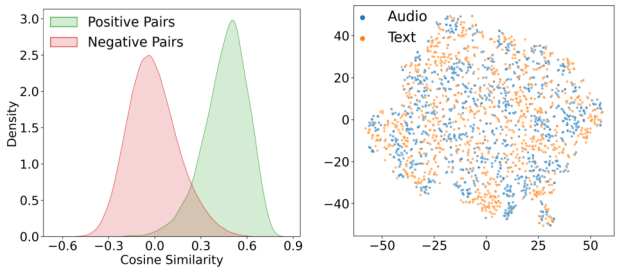


Figure 4: **Feature visualisation**. (Left) distributions of pair-wise similarity scores of aligned and non-aligned (audio, text) pairs in our testing set. (Right) t-SNE visualisation of audio and text features in the multimodal output space.

Feature visualisation Figure 4 (right) shows a t-SNE plot of audio and text embeddings in our testing set. We observe no separation between modalities in the output space, with representations of both audio and text inputs well mixed together, confirming that MusCALL learns to adequately map both input modalities to a common space.

Failure analysis In order to provide more context to our cross-modal retrieval results beyond metric-based evaluation, we also perform a qualitative error analysis by examining the relevance of the retrieved items in cases of incorrect retrieval. As highlighted in the examples in Table 3, in some cases, while the ground-truth item is not ranked first, MusCALL is still capable of retrieving audio tracks whose associated caption is semantically related to the query.

6.2 Ablation Study

The effect of data augmentations & random crop Table 4 shows that removing random cropping (RC) considerably degrades performance (-13.7% compared to $\text{MusCALL}_{\text{BASE}}$), while removing the audio augmentations (AA) only marginally alters results (-1.5%). When considering both together, we find that, in the absence of RC, the benefit of performing augmentations is amplified, and removing both has more drastic effect (-31.5%). This is expected, since both techniques effectively increase the variety of samples seen in training. To avoid costly hyperparameter tuning, we do not exhaustively explore audio transformations and their composition, and note that tailoring the AA pipeline, for example via additional frequency-domain transformations, may lead to better results.

The effect of loss weighting As also shown in Table 4, using our content-aware loss weighting (Section 3.2.1) results in comparable retrieval performance to our vanilla contrastive loss. To more closely observe the effect of using loss weighting, we compare the pairwise similarity distributions of positive and negative pairs produced by MusCALL with and without loss weighting (shown in the supplementary material). This reveals that similarity scores of positive pairs have lower variance and a higher mean when using loss weighting, suggesting that this technique nudges the learning process towards a better discrimination of positives and negatives, but not enough to produce significant improvements in the cross-modal retrieval task.

Query Text	Text of the Top-1 Audio
An atmospheric and introspective orchestral track featuring strings, piano, and synth.	*An inspirational and moody orchestral track featuring strings and choir.*
Deep chilled out space jazz with crisp beats and lush electronics.	*Jaunty swing featuring trumpet.*
Up tempo, pumping dance pop with female vocals.	*Quirky, fun, positive disco party music.*

Table 3: **Failure analysis** of text-to-audio retrieval.

LW	RC	AA	AP	Text → Audio		
				R@1	R@5	R@10
✗	✗	✗	✓	14.7	33.9	47.2
✗	✗	✓	✓	18.5	43.5	57.4
✗	✓	✓	✗	24.9	49.3	58.9
✗	✓	✗	✓	24.3	51.3	62.2
✗	✓	✓	✓	24.6	51.5	63.8
✓	✓	✓	✓	25.9	51.9	63.3

Table 4: **Ablations**: loss weighting (LW), random cropping (RC), audio augmentations (AA), attention pooling (AP).

The effect of attention pooling Our ablation study confirms that removing attention pooling hurts performance: on average, the Recall results drop by 4.9% compared to simple average pooling. We note that the advantage of using attention pooling becomes more pronounced as we increase the audio input length, as shown in Figure 3. This is particularly relevant in the music domain, since music signals exhibit structure over longer ranges compared to other types of audio signals like environmental sounds.

7. CONCLUSION

We presented MusCALL, a method for multimodal contrastive learning of audio-linguistic representations. By leveraging aligned (audio, text) pairs, MusCALL successfully learns to perform cross-modal retrieval, allowing to search for music via natural language queries and vice versa. Extending this multimodal alignment capability to the zero-shot setting, MusCALL can also be transferred to music classification tasks by simply providing target labels as text inputs. Through an extensive set of experiments, we validated the main design choices in our core approach and explored two variants. Through the first variant, we demonstrated the viability of using a text-based similarity metric to weigh the loss contribution of each negative sample, providing a starting point for improving multimodal contrastive learning on real-world music data. Through the second variant, we explored including a self-supervised objective to improve the audio representation quality. Both variants show promising results and provide opportunities for further research. Future work will focus on assessing their performance in more depth, particularly in the zero-shot scenario, including human evaluations alongside automatic metrics and considering a wider set of tasks.

8. ACKNOWLEDGEMENTS

This work was supported by UK Research and Innovation [grant number EP/S022694/1] and Universal Music Group. We would also like to thank The Alan Turing Institute for the support provided during the first author's time as an Enrichment Student.

9. REFERENCES

[1] M. Müller, F. Kurth, D. Damm, C. Fremerey, and M. Clausen, "Lyrics-Based Audio Retrieval and Multimodal Navigation in Music Collections," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4675 LNCS, pp. 112–123, 2007.

[2] K. Tsukuda, "Lyric Jumper: A Lyrics-Based Music Exploratory Web Service by Modeling Lyrics Generative Process." in *ISMIR*, 2017, pp. 544–551.

[3] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Disentangled Multidimensional Metric Learning for Music Similarity," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, 8 2020.

[4] B. Li and A. Kumar, "Query by Video: Cross-Modal Music Retrieval," in *20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 604–611.

[5] A. Ghias, J. Logan, D. Chamberlin, and B. C. Smith, "Query by humming," in *Proceedings of the third ACM international conference on Multimedia*. Association for Computing Machinery (ACM), 1995, pp. 231–236.

[6] M. Müller, A. Arzt, S. Balke, M. Dorfer, and G. Widmer, "Cross-Modal Music Retrieval and Applications: An Overview of Key Methodologies," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 52–62, 1 2019.

[7] P. Knees, M. Schedl, and M. Goto, "Intelligent User Interfaces for Music Discovery," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, pp. 165–179, 10 2020.

[8] K. Watanabe and M. Goto, "Query-by-Blending: a Music Exploration System Blending Latent Vector Representations of Lyric Word, Song Audio, and Artist," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*, 2019.

[9] C. Hosey, L. Vujović, B. St. Thomas, J. Garcia-Gathright, and J. Thom, "Just give me what I want: How people use and evaluate music search," in *Conference on Human Factors in Computing Systems - Proceedings*. Association for Computing Machinery, 5 2019.

[10] J. Ha, L. J. Stephen, D. Sally, and J. Cunningham, "Challenges in cross-cultural/multilingual music information seeking." in *ISMIR*, 2005.

[11] C. C. Liem, M. Müller, D. Eck, G. Tzanetakis, and A. Hanjalic, "The need for music information retrieval with user-centered and multimodal strategies," in *MM'11 - Proceedings of the 2011 ACM Multimedia Conference and Co-Located Workshops - MIRUM 2011 Workshop, MIRUM'11*, 2011, pp. 1–6.

[12] P. Knees, "Search & Select - Intuitively Retrieving Music from Large Collections," in *ISMIR*, 2007.

[13] B. Whitman and R. Rifkin, "Musical query-by-description as a multiclass learning problem," in *Proceedings of 2002 IEEE Workshop on Multimedia Signal Processing, MMSP 2002*. Institute of Electrical and Electronics Engineers Inc., 2002, pp. 153–156.

[14] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Advances in Neural Information Processing Systems*, 2013.

[15] I. Manco, E. Benetos, E. Quinton, and G. Fazekas, "Learning music audio representations via weak language supervision," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

[16] S. Oramas, L. Espinosa-Anke, M. Sordo, H. Saggion, and X. Serra, "Information extraction for knowledge base construction in the music domain," *Data and Knowledge Engineering*, 2016.

[17] S. Oramas, M. Sordo, L. Espinosa-Anke, and X. Serra, "A semantic-based approach for artist similarity," in *ISMIR*, 2015.

[18] S. Oramas, L. Espinosa-Anke, A. Lawlor, X. Serra, and H. Saggion, "Exploring Customer Reviews for Music Genre Classification and Evolutionary Studies," in *17th International Society for Music Information Retrieval Conference*, 2016.

[19] S. Oramas, F. Barbieri, O. Nieto, and X. Serra, "Multimodal Deep Learning for Music Genre Classification," *Transactions of the International Society for Music Information Retrieval*, vol. 1, no. 1, pp. 4–21, 9 2018.

[20] R. Delbouys, R. Hennequin, F. Piccoli, J. Royo-Letelier, and M. Moussallam, "Music mood detection based on audio and lyrics with deep neural net," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*, 2018.

[21] B. Jeon, C. Kim, A. Kim, D. Kim, J. Park, and J. W. Ha, "Music emotion recognition via end-To-end multimodal neural networks," in *CEUR Workshop Proceedings*, 2017.

[22] K. Choi, G. Fazekas, M. Sandler, B. Mcfee, and K. Cho, "Towards Music Captioning: Generating Music Playlist Descriptions," *arXiv preprint arXiv:1608.04868*, 2016.

[23] C. Tian, M. Michael, and H. Di, "Music autotagging as captioning," in *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*. Association for Computational Linguistics, 2020, pp. 67–72.

[24] I. Manco, E. Benetos, E. Quinton, and G. Fazekas, "MusCaps: Generating Captions for Music Audio," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[25] X. Favory, K. Drossos, V. Tuomas, and X. Serra, "COALA: Co-Aligned Autoencoders for Learning Semantically Enriched Audio Representations," in *International Conference on Machine Learning (ICML), Workshop on Self-supervised learning in Audio and Speech*, 2020.

[26] X. Favory, K. Drossos, T. Virtanen, and X. Serra, "Learning Contextual Tag Embeddings for Cross-Modal Alignment of Audio and Tags," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[27] A. Ferraro, X. Favory, K. Drossos, Y. Kim, and D. Bogdanov, "Enriched Music Representations with Multiple Cross-modal Contrastive Learning," *IEEE Signal Processing Letters*, vol. 28, pp. 733–737, 4 2021.

[28] J. Choi, J. Lee, J. Park, and J. Nam, "Zero-shot learning for audio-based music classification and tagging," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, 2019.

[29] M. Won, S. Oramas, O. Nieto, F. Gouyon, and X. Serra, "Multimodal Metric Learning for Tag-based Music Retrieval," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[30] M. Won, J. Salamon, N. J. Bryan, G. J. Mysore, and X. Serra, "Emotion Embedding Spaces for Matching Music to Stories," in *International Society for Music Information Retrieval Conference (ISMIR)*, 11 2021.

[31] J. Lu, D. Batra, D. Parikh, and S. Lee, "ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks," in *Advances in Neural Information Processing Systems*, 2019, pp. 13–23.

[32] Y. C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu, "UNITER: UNiversal Image-TExt Representation Learning," in *European Conference on Computer Vision*, 2020.

[33] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning Transferable Visual Models From Natural Language Supervision," in *International Conference on Machine Learning*. PMLR, 2021.

[34] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. V. Le, Y. Sung, Z. Li, and T. Duerig, "Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 4904–4916.

[35] J. Andreas, D. Klein, and S. Levine, "Learning with Latent Language," in *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1. Association for Computational Linguistics (ACL), 11 2018, pp. 2166–2179.

[36] X. Liu, X. Mei, Q. Huang, J. Sun, J. Zhao, H. Liu, M. D. Plumbley, V. Kılıç, and W. Wang, "Leveraging Pre-trained BERT for Audio Captioning," *arXiv preprint arXiv:2203.02838*, 2022.

[37] A.-M. Oncescu, A. S. Koepke, J. F. Henriques, Z. Akata, and S. Albanie, "Audio Retrieval with Natural Language Queries," in *Interspeech*, 5 2021.

[38] H.-H. Wu, P. Seetharaman, K. Kumar, and J. P. Bello, "Wav2CLIP: Learning Robust Audio Representations From CLIP," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 10 2022.

[39] Y. Zhao, J. Hessel, Y. Yu, X. Lu, R. Zellers, and Y. Choi, "Connecting the Dots between Audio and Text without Parallel Data through Visual Knowledge Transfer," *arXiv preprint arXiv:2112.08995*, 2021.

[40] A. Guzhov, F. Raue, J. Hees, and A. Dengel, "AudioCLIP: Extending CLIP to Image, Text and Audio," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

[41] C. Chen, D. Han, and J. Wang, "Multimodal Encoder-Decoder Attention Networks for Visual Question Answering," *IEEE Access*, pp. 1–1, 2 2020.

[42] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "BYOL for Audio: Self-Supervised Learning for General-Purpose Audio Representation," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 3 2021.

[43] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive Learning of General-Purpose Audio Representations," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 10 2021.

[44] H. Al-Tahan, Y. M. I. C. On, and U. 2021, "CLAR: Contrastive Learning of Auditory Representations," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2021.

[45] J. Spijkervet and J. A. Burgoyne, "Contrastive Learning of Musical Representations," in *ISMIR*, 2021.

[46] E. Fonseca, D. Ortego, K. McGuinness, N. E. O'Connor, and X. Serra, "Unsupervised Contrastive Learning of Sound Event Representations," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 11 2021.

[47] P. H. Le-Khac, G. Healy, and A. F. Smeaton, "Contrastive Representation Learning: A Framework and Review," *IEEE Access*, vol. 8, pp. 193 907–193 934, 10 2020.

[48] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," in *International conference on machine learning*. PMLR, 2020.

[49] J.-B. Alayrac, A. Recasens, R. Schneider, R. Arandjelović, J. Ramapuram, J. De Fauw, L. Smaira, S. Dieleman, and A. Zisserman, "Self-Supervised MultiModal Versatile Networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 25–37, 6 2020.

[50] M. Patrick, Y. M. Asano, P. Kuznetsova, R. Fong, J. F. Henriques, G. Zweig, and A. Vedaldi, "Multi-modal Self-Supervision from Generalized Data Transformations," *arXiv preprint arXiv:2003.04298v2*, 2020.

[51] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *arXiv preprint arXiv:1807.03748*, 2018.

[52] N. Saunshi, P. Orestis, S. Arora, K. Mikhail, and K. Hrishikesh, "A Theoretical Analysis of Contrastive Unsupervised Representation Learning," in *International Conference on Machine Learning*. PMLR, 2019.

[53] M. Zolfaghari, Y. Zhu, P. Gehler, and T. Brox, "CrossCLR: Cross-modal Contrastive Learning For Multimodal Video Representations," in *International Conference on Computer Vision (ICCV)*. Institute of Electrical and Electronics Engineers (IEEE), 9 2021, pp. 1430–1439.

[54] L. Ericsson, H. Gouk, C. C. Loy, and T. M. Hospedales, "Self-Supervised Representation Learning: Introduction, Advances and Challenges," *IEEE Signal Processing Magazine*, vol. 39, no. 3, pp. 42–62, 10 2021.

[55] N. Mu, A. Kirillov, D. Wagner, and S. Xie, "SLIP: Self-supervision meets Language-Image Pre-training," *arXiv preprint arXiv:2112.12750*, 2021.

[56] H. Xie, O. Räsänen, K. Drossos, and T. Virtanen, "Unsupervised Audio-Caption Aligning Learns Correspondences between Individual Sound Events and Textual Phrases," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

[57] K. He, Y. Wang, and J. Hopcroft, "A powerful generative model using random weights for the deep image representation," in *Advances in Neural Information Processing Systems*, 2016.

[58] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 2017-Decem. Neural information processing systems foundation, 6 2017, pp. 5999–6009.

[59] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners," 2019.

[60] M. Won, K. Choi, and X. Serra, "Semi-Supervised Music Tagging Transformer," in *International Society for Music Information Retrieval Conference (ISMIR)*, 11 2021.

[61] R. Sennrich, B. Haddow, and A. Birch, "Neural Machine Translation of Rare Words with Subword Units," in *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, vol. 3. Association for Computational Linguistics (ACL), 2016, pp. 1715–1725.

[62] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," in *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*. Association for Computational Linguistics, 8 2019, pp. 3982–3992.

[63] K. Drossos, S. Lipping, and T. Virtanen, "Clotho: An Audio Captioning Dataset," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 2020-May. IEEE, 5 2020, pp. 736–740.

[64] C. D. Kim, B. Kim, H. Lee, and G. Kim, "AudioCaps: Generating captions for audios in the wild," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics (ACL), 2019, p. 119–132.

[65] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. International Conference on Learning Representations, ICLR, 2013.

[66] J. Bromley, I. Guyon, Y. Lecun, E. Sickinger, R. Shah, A. Bell, and L. Holmdel, "Signature verification using a" siamese" time delay neural network," *Advances in neural information processing systems*, vol. 6, 1993.

[67] X. Zhai, X. Wang, B. Mustafa, A. Steiner, D. Keysers, A. Kolesnikov, and L. Beyer, "LiT: Zero-Shot Transfer with Locked-image Text Tuning," *arXiv preprint arXiv:2111.07991*, 2021.

[68] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 5, pp. 293–302, 7 2002.

[69] E. Law, K. West, M. Mandel, M. Bay, and J. Stephen Downie, "Evaluation of algorithms using games: The case of music tagging," in *Proceedings of the 10th ISMIR Conference*, 2009.

[70] L. Wang, P. Luc, Y. Wu, A. Recasens, L. Smaira, A. Brock, A. Jaegle, J.-B. Alayrac, S. Dieleman, J. Carreira, and A. van den Oord, "Towards Learning Universal Audio Representations," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5 2022, pp. 4593–4597.

[71] L. A. Hendricks, J. Mellor, R. Schneider, J.-B. Alayrac, and A. Nematzadeh, "Decoupling the Role of Data, Attention, and Losses in Multimodal Transformers," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 570–585, 1 2021.

[72] B. Lester, R. Al-Rfou, N. Constant, and G. Research, "The Power of Scale for Parameter-Efficient Prompt Tuning," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics (ACL), 12 2021, pp. 3045–3059.

MUSAV: A DATASET OF RELATIVE AROUSAL-VALENCE ANNOTATIONS FOR VALIDATION OF AUDIO MODELS

Dmitry Bogdanov Xavier Lizarraga-Seijas Pablo Alonso-Jiménez Xavier Serra
Music Technology Group, Universitat Pompeu Fabra, Spain
dmitry.bogdanov@upf.edu, xavier.lizarraga@upf.edu, pablo.alonso@upf.edu

ABSTRACT

We present MusAV, a new public benchmark dataset for comparative validation of arousal and valence (AV) regression models for audio-based music emotion recognition. To gather the ground truth, we rely on relative judgments instead of absolute values to simplify the manual annotation process and improve its consistency. We build MusAV by gathering comparative annotations of arousal and valence on pairs of tracks, using track audio previews and metadata from the Spotify API. The resulting dataset contains 2,092 track previews covering 1,404 genres, with pairwise relative AV judgments by 20 annotators and various subsets of the ground truth based on different levels of annotation agreement. We demonstrate the use of the dataset in an example study evaluating nine models for AV regression that we train based on state-of-the-art audio embeddings and three existing datasets of absolute AV annotations. The results on MusAV offer a view of the performance of the models complementary to the metrics obtained during training and provide insights into the impact of the considered datasets and embeddings on the generalization abilities of the models.

1. INTRODUCTION

Audio-based music emotion recognition is a popular task in music information retrieval (MIR) that has recently gained more presence in the context of industrial applications. It is relevant for building systems for navigation of music collections, music search, exploration, and recommendation, and diverse applications that can benefit from MIR, such as audio branding or music therapy.

There are two types of approaches to emotion recognition in MIR following research in music psychology and affective computing [1–3]. The categorical approach considers different discrete categories of emotions (or moods [1]) or their clusters [4, 5] separately. It relies on taxonomies of descriptive mood tags and is frequently addressed by research on music classification and autotagging [6–9]. In contrast, the dimensional approach proposes representations on a continuous scale for several dimensions [10], allowing for a direct comparison of different moods, which is convenient for many applications.

The dimensional approach is based on existing research in music psychology which proposes two-dimensional or three-dimensional representations, including *arousal* (energy and stimulation), *valence* (pleasantness and positivity), and *dominance* (potency and control) [11] or, alternatively, *depth* [12] or *tension* [13], with many representations inheriting from the circumplex model of emotion by Russell [14]. In general, the 2D arousal/valence (AV) representation is a common model widely adopted in affective computing in different domains including music.

Various MIR researchers have worked on building datasets of AV annotations of music and training machine learning models for their automatic regression from audio [10, 15–21]. These datasets have been created with different methodologies, music collections, and participating annotators. However, there is no common benchmark dataset that could be conveniently used to compare models proposed by researchers and trained on different datasets. Existing studies report model performances using dataset splits without validation of the trained models on external datasets, which has been found to be very informative in other music classification tasks [22–24], providing insights on the generalization abilities and preventing overoptimistic conclusions.

In this paper, we propose to establish a common dataset for complementary evaluation on external data. We describe our methodology for building such a dataset, taking into account music genre diversity, and using it for evaluation of AV regression models. In contrast to many previous studies, we use comparisons between pairs of songs as ground truth instead of absolute values (coordinates) within the 2D AV space to make our annotations easier to gather and potentially more reliable, as suggested by previous works on relative emotion annotation in music and other domains [25–28]. In addition, we apply loudness normalization to avoid bias in arousal annotations [29], which has not been considered in previous datasets. The proposed validation of AV emotion recognition models provides a complementary view on their performance giving an opportunity to estimate generalization capabilities

[1] Even though some researchers distinguish the terms "emotion" and "mood", with moods being longer-term perceptions of musical input, we will use both terms interchangeably in this paper for simplicity.

© D. Bogdanov, X. Lizarraga-Seijas, P. Alonso-Jiménez, and X. Serra. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** D. Bogdanov, X. Lizarraga-Seijas, P. Alonso-Jiménez, and X. Serra, "MusAV: A dataset of relative arousal-valence annotations for validation of audio models", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

of the models on a common ground truth.

Following this methodology, we build a dataset based on audio previews and metadata available via Spotify API and evaluate a selection of AV regression models based on state-of-the-art audio embeddings. We analyze annotation agreement, propose strategies for building refined subsets of the dataset with different levels of consistency of annotations, and discuss the performance of the AV models.

2. RELATED WORK

Music emotion recognition is challenging because of the biases in cultural background, generation, genre, and personality [2, 30–33]. Nevertheless, this task gathered research attention in MIR from early on [34,35] given potential applications. Table 1 summarizes public AV datasets previously used in research. They all contain music audio excerpts and crowdsourced explicit arousal/valence annotations (absolute values that characterize each track or comparisons of track pairs) except for the MuSe dataset.

2.1 AV datasets with absolute values

There is a considerable variety of approaches to AV regression [10, 36–38] based in different audio features, including MediaEval campaigns in 2013-2015 [39–42]. We highlight the three most commonly used datasets containing absolute value annotations:

- **EmoMusic** [17] has been presented for the MediaEval 2013 Emotion in Music Task [39]. It contains 744 full audio tracks as well as 45-second excerpts. All audio is sourced from Free Music Archive (FMA). The excerpts are manually annotated with AV values characterizing the overall feel of the tracks as well as dynamic AV values at different rates, additionally summarized over the segment length (mean and stdev).
- **DEAM** [16] contains 1,802 audio excerpts (58 full-length songs and 1,744 excerpts of 45 seconds). The audio comes from several sources including FMA, Jamendo, and the MedleyDB dataset. The dataset similarly contains the overall AV values and dynamic values at a one per second rate and their summary (mean and stdev). This dataset has been derived from EmoMusic and used for the MediaEval 2013-2015 Emotion in Music Task [42] and in more recent studies [43, 44].
- **MuSe** [20] provides track-level valence, arousal and dominance values derived from social tags associated with music tracks on Last.fm [2] by using a dictionary of emotional ratings of words [45]. The dataset includes annotations for 90,408 songs, however the audio is not directly available. Instead, the tracks are identified by metadata, including Spotify IDs for 61,630 tracks. Nevertheless, only 41,021 30-second audio previews are currently accessible via Spotify API. [3]

Importantly, the EmoMusic and DEAM datasets are limited in coverage and they do not represent a large variety of music available on commercial digital music platforms. The MuSe dataset has a significantly larger size and coverage, including 835 genres, achieved by sampling Last.fm using a diverse set of mood labels. Yet, its downside is that it is possibly noisy due to the tags-to-AV mapping. As a compromise, Panda et al. [37] propose to infer AV annotations from AllMusic [4] emotion tags, but they only use them to create a balanced annotation pool that is then manually validated. The resulting dataset (MER-TAFFC) contains 900 30-second track previews annotated by four AV emotion quadrants. In our work, we also follow an automated music preselection approach and prioritize large genre coverage while keeping the annotations manual. We can then compare AV regression models trained on EmoMusic, DEAM, and MuSe using our new dataset.

Dataset	# tracks	Type	Source
EmoMusic [17]	744 ft/exc	abs	MTurk
DEAM [16]	1,802 ft/exc	abs	MTurk
MuSe [20]	41,021 exc	abs	Last.fm tags
MER-TAFFC [37]	900 exc	quad	manual
CCMED-WCMED [46]	800 exc	rel	CrowdFlower
EMusic [26]	140 exc	rel	CrowdFlower
MusAV	2,092 exc	rel	manual

Table 1. Public music datasets for AV regression and the proposed MusAV dataset. ft: full tracks, exc: excerpts, abs: ranged absolute values, quad: quadrants, rel: relative annotations.

2.2 Relative annotations

Some researchers in affective computing highlighted the disadvantages of rating-based emotion annotation by absolute values and propose to use relative annotations [27,28]. In MIR this has been considered in few studies. Yang and Chen [25] propose to gather relative AV annotations and employ learning-to-rank algorithms to train models predicting absolute AV values. They discuss the limitations of absolute value rating-based annotations and show that relative annotations are significantly easier and have more within-subject and between-subject reliability. Their annotation experiment involved a corpus of 1,240 pop songs (30-second segments) and 99 annotators with an average of 4.3 annotators per song. However, they considered relative annotations only for valence and the audio for the dataset is not publicly available.

The idea of relative AV annotations has been further explored by Fan et al. for the case of experimental music [26] (the EMusic dataset) and classical music [46] (CCMED-WCMED). For the former dataset, they crowdsource pairwise track AV comparisons for 140 track segments from 9 genres by 823 annotators gathering up to three annotators per pair. The latter contains 800 track segments from Western and Chinese classical music with pairwise comparisons by 989 annotators. In addition, a similar study proposes relative ground truth for soundscape emotion recognition [47].

[2] https://www.last.fm/

[3] As of May 13, 2022.

[4] https://allmusic.com

3. ANNOTATION APPROACH

We follow a methodology that allows to avoid some of the limitations related to subjective annotation of arousal and valence with absolute values and instead consider comparative annotations on pairs of music tracks [25, 26]. Our main motivations are the following:

- Many practical applications are concerned not with the absolute AV values of songs, but with rankings songs according to these values. In such cases, relative annotations are convenient to validate ranking performance.
- For annotators, pairwise comparisons can be easier to understand and make decisions. They might require less effort than annotating with ranges of continuous values or Likert scales, which have more cognitive load [25,48].
- There is evidence that relative annotations have higher between-subject and within-subject agreement [25, 27].
- There are simple strategies to refine annotations according to the agreement between different annotators.
- Model evaluation can be a simple comparison of the ground-truth ordering of two tracks with the ordering according to the predicted AV values for all track pairs.
- Depending on how the models are trained, different models might predict AV values within different value ranges, which should be accounted for when comparing the performance metrics such as RMSE. Relative comparisons on pairs of tracks allows using simple common metrics that are compatible with all models.

To collect the dataset ground truth, we need an annotation tool able to reproduce pairs of music excerpts and collect the user's input. Such a tool should address potential sources of bias and simplify and speed up the annotation process. We define several requisites to accomplish this:

- We are interested in relative judgements about a pair of tracks (A and B) that can be formulated as the following question: "Which song has more *music property X*". The following choices are considered: *A*, *B*, or *same*. To minimize potential biases toward any of the choices, none of them should be selected by default.
- It may be difficult to maintain consistency when answering non-factual questions. Therefore, we consider that the interface should display multiple pairs in the same *page* to give the user an opportunity to improve coherence of their annotations before submitting the page.
- Loudness has a large impact in music perception. Higher loudness correlates to higher perceived arousal [29]. To minimize this effect, the annotation tool should include loudness normalization. Previous studies that proposed interfaces for AV annotations [16,17,25,26] ignored this issue and did not include any normalization.

4. THE MUSAV DATASET

We created our dataset following the described methodology, using Spotify API as a source for music audio previews and genre metadata. Using Spotify allows us to access a wide range of music, while the 30-second previews it provides are sufficiently long to capture an overall perceived emotion with AV annotations.

4.1 Preparing the annotation pool

We collected a list of 5,716 genres from *everynoise.com* [5] which corresponds to the genre taxonomy of the Spotify API [6] and contains broad genres as well as specific subgenres. We then used the API's *Search* method to select random tracks for each genre. We generated multiple queries for each genre (using the genre tag, a wildcard search string starting with a random character, and a random market) and picked a random track from the list of the returned results for each query. This method allowed us to diversify music coverage and avoid popularity bias, downloading 17,574 track previews for 4,386 genres (up to 15 tracks per genre). All audio previews are 30-second long MP3 files with a 96 kbit/s bitrate. Each preview has a corresponding metadata file obtained with the API's *Get Track* method, including artist and album metadata and various audio analysis features.

We then analyzed the loudness of the audio previews to discard tracks with atypical levels, computing the integrated loudness in LUFS [49] of each track with the Essentia audio analysis library [50]. Based on the distribution of the obtained values, we kept tracks within the range between -20 and -5 LUFS, which represents the range of healthy loudness levels for the majority of mastered music. As a result, we reduced our pool to 15,979 tracks by 3,630 genres.

We organized the tracks into triplets with three pairwise comparisons each, allowing for additional inconsistency checks according to gathered relations within each triplet. We randomly assigned the tracks to two types of triplets: *genre-triplets* with all tracks sharing the same genre (one triplet per genre) and *global-triplets* containing tracks from various genres (the remaining tracks) to account for a use-case of distinguishing emotions within the same genre. All resulting 5,326 triplets contain unique tracks. We randomly split all generated triplets into annotation chunks, each one containing 100 triplets with 80% being global-triplets and 20% genre-triplets.

4.2 Annotation tool and process

We implemented a custom tool according to the requirements in Section 3. For each pairwise comparison (a pair), we used the wavsurfer-js [7] player to display a navigable representation of the waveforms. To prevent loudness bias, we normalize the songs to a common level of -20 LUFS. We computed the normalization factors from the LUFS values precomputed in the dataset preparation step, and converted them to linear gain units as expected by wavsurfer-js.

Our interface formulates two questions: "Which song has more arousal?" and "Which song has more va-

[5] https://everynoise.com

[6] https://developer.spotify.com/documentation/web-api/reference

[7] https://wavesurfer-js.org/

lence?". For both questions, the choices are *A*, *B*, or *same* arousal/valence. The tool shows a configurable number of pairs on each screen page (6 by default) and a submit button that stores the answers from the current page and renders the next one. We present the annotator with multiple pairs on the same page, as this can facilitate double-checking decisions made across pairs to minimize inconsistencies. Additionally, it simplifies navigation of the annotation interface, reducing the amount of necessary mouse movements and clicks. Finally, the tool has a page counter and displays each pair's ID to facilitate reporting any possible issues. The annotator outputs one JSON file per pair containing the answers to the two questions.

Figure 1 depicts the annotation tool. The source code for the annotation tool is publicly available online. [8] The tool is distributed as a Docker web application.

Due to the limitations on effort and availability of our annotators, we have proceeded with 7 annotation chunks which account to 2,100 tracks assigned to 700 triplets with 2,100 pairwise track comparisons. Overall, we gathered annotations from 20 participants, including authors' colleagues and students, with a background in music and technology. Each chunk was presented to three different annotators. Every annotator was given a single chunk, with an exception of one annotator who worked with two chunks. All annotators were instructed about the meaning of arousal and valence beforehand following their common definition [16, 42] and were asked to focus on perceived emotion [51]. Participants were aware of the subjectivity of the task and we encouraged to provide their subjective opinion. In total, we gathered 6,255 comparative arousal and valence judgments on pairs of tracks after discarding 15 pairs that the annotators reported having non-music tracks (speech) and duplicated tracks. These annotations involve 2,092 track previews by 1,404 genres.

4.3 Annotation agreement and consistency

By having multiple people annotate the same chunks of audio, we can measure the agreement between annotators. Computing ordinal Krippendorff's alpha, we obtained values of 0.48 for arousal and 0.39 for valence, which indicates a fair to moderate level of agreement, which is consistent with previous studies [26, 42, 46].

For building our ground truth for arousal and valence, we defined two types of agreement for pairwise comparisons of tracks by three different annotators. If all three annotators agreed on a pair of tracks with the same answer (*A-A-A*, *B-B-B*, or *same-same-same*) the annotations for this pair were considered to be in *full agreement*. If only two annotators agreed, we checked whether the third annotator was in a soft (e.g., *A-A-same*, *same-same-A*) or hard (*A-A-B* or *B-B-A*) disagreement. In the case of the former, we considered the annotations for the pair to be in *majority agreement*. Table 2 presents the agreement statistics for all gathered annotations.

In addition, we checked whether pairwise comparisons contradict each other within triplets (that is, whether they are geometrically inconsistent). For example, for three tracks X, Y, and Z forming a triplet, if $X > Y$ and $Y > Z$, but $X \leq Z$, such triplet and all its pairs are considered inconsistent. We considered triplets as consistent only if all constituent pairs had full or majority agreement in the annotations and no contradictions have been found.

Agreement	Arousal		Valence	
	# pairs	%	# pairs	%
FA+MA	1,448	69.4	1,341	64.3
FA	975	46.8	810	38.8
FA+MA, CT	738	35.4	606	29.1
FA, CT	519	24.9	381	18.3

Table 2. Number and percentage of annotated track pairs with different levels of annotator agreement and consistency for arousal and valence. FA+MA: pairs with full or majority agreement. FA: pairs with full agreement. CT: only pairs belonging to consistent triplets.

As a result, we generated different subsets of the annotations, with 69.4% and 64.3% of track pairs having at least some level of agreement and 24.9% and 18.3% passing the most strict conditions (pairs with full agreement, belonging to consistent triplets) for arousal and valence, accordingly.

Finally, as we had two types of triplets, we checked the effect of genre on the agreement rate: 67% and 61% of pairs in global-triplets had either full or majority agreement compared to 76% and 75% in the case of genre-triplets for arousal and valence, accordingly. This observation revealed that it was slightly easier to reach agreement on pairs of tracks coming from the same genre than from different genres.

4.4 Dataset contents

We provide the following contents as part of the dataset, available online: [9]

- Metadata for the entire annotation pool. [10] Each triplet is identified by a triplet ID and contains track Spotify IDs, triplet type (global-triplet or genre-triplet) and genre information.
- Split of the annotation pool into annotation chunks. [11]
- Raw comparative arousal and valence annotations on track pairs by anonymized annotators.
- Processed ground-truth annotations with different levels of agreement and consistency (full and major agreement with/without triplet consistency).
- Track audio previews and metadata gathered from the Spotify API for the annotated chunks. [12]
- Dataset metadata statistics (e.g., genre distribution).
- Scripts to reproduce the creation of the dataset.

[8] https://github.com/MTG/musav-annotator

[9] https://mtg.github.io/musav-dataset

[10] All annotation metadata is licensed under CC BY-NC-SA 4.0.

[11] It is possible to expand the dataset by annotating more chunks.

[12] Available under request for non-commercial scientific research purposes only. Any publication of results based on this data must cite Spotify API as the source of the data.

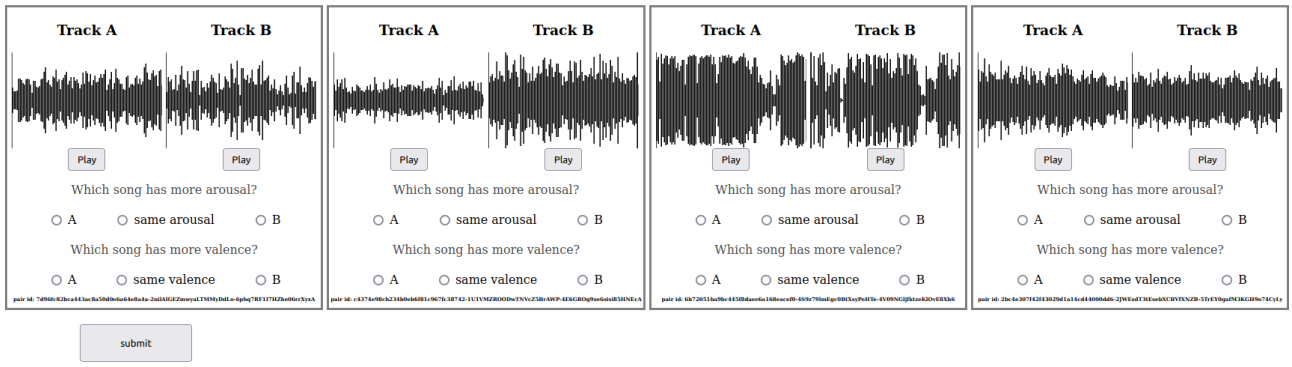


Figure 1. Screenshot of the annotation tool.

5. EXPERIMENTS

We demonstrate the dataset in use on the example of evaluating AV regression models based on audio embeddings.

5.1 Models

We created AV regression models based on three types of audio embeddings:

- **MusiCNN-MSD** (*musicnn*) is a music auto-tagging CNN with filter shapes motivated by music domain [52] trained on a subset of the Million Song Dataset. Its embedding layer has 200 units.
- **VGGish** (*vggish*) is a VGG architecture with an embedding layer of 128 units trained on audio from YouTube videos mapped to general purpose audio labels derived from their metadata [53]. The embeddings from this model were previously used for AV regression in combination with support vector regression [46].
- **EffNet-Discogs** (*effnet*) is an EfficientNet architecture trained to predict the music styles tags from Discogs. [13] The model produces 1280-dimensional embeddings and it is publicly available as part of Essentia models [54]. [14]

The embeddings are extracted on short one-, two-, and three-second audio excerpts for *vggish*, *effnet*, and *musicnn* models, accordingly. They are then used by the downstream regression models that we train. These models provide arousal-valence inference for each embedding vector, with a variable batch size with batch normalization. As Figure 2 depicts, we use a fully connected layer with a linear activation function, preceded by batch normalization and dropout. We also apply L1-L2 and L2 regularizers in the fully connected layer and dropout as regularization methods.

For training, we used three different datasets: DEAM, EmoMusic and MuSe, which we selected based on their music coverage as more appropriate for general use, with the goal to incorporate the resulting models as part of Essentia [24]. Each of them provides different audio excerpts (with 30-45 second duration) and arousal-valence values characterizing the overall emotion of each track. We extracted the embeddings with the pretrained models and used them as features. We followed a standard data splitting and loading strategy used in previous music classification publications [24]. To generate a train/test split stratified in terms of AV quadrants, we use Z-score normalization. However, we did not normalize the training data. The AV value range in all the datasets is from 1 to 9.

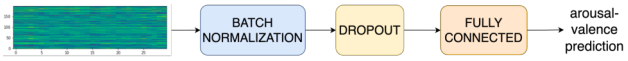


Figure 2. Arousal-valence backend model architecture.

Our models operate on short audio chunks and allow us to generate sequences of AV predictions over time with a new prediction every 1-3 seconds, depending on the receptive field of the embedding model used. Therefore, we compute the average of predictions on chunks to estimate the overall arousal/valence of a track.

In Table 3, we report Root Mean Square Error (RMSE) and Coefficient of Determination R^2 (R^2) commonly used for evaluation of regression models [17, 55, 56], obtained

	Arousal		Valence	
	R^2	RMSE	R^2	RMSE
deam-effnet	0.404	0.913	0.335	0.909
deam-musicnn	**0.417**	**0.894**	**0.400**	**0.818**
deam-vggish	0.396	0.963	0.344	0.963
emomusic-effnet	0.420	1.090	0.375	**0.948**
emomusic-musicnn	**0.451**	**1.030**	0.363	0.966
emomusic-vggish	0.429	1.037	**0.376**	0.973
muse-effnet	**0.143**	**1.148**	**0.089**	**1.581**
muse-musicnn	0.141	1.320	0.085	2.509
muse-vggish	**0.143**	**1.148**	0.085	1.584

Table 3. Evaluation metrics for the AV regression models on the held-out (testing) sets of the corresponding training datasets. The best values for each dataset are in bold.

[13] https://blog.discogs.com/en/genres-and-styles

[14] https://essentia.upf.edu/models.html

	Arousal				Valence			
# track pairs	FA+MA 1413	FA 950	FA+MA, CT 716	FA, CT 502	FA+MA 1310	FA 787	FA+MA, CT 588	FA, CT 368
deam-effnet	72.28	75.44	72.60	74.84	61.59	63.38	63.91	65.51
deam-musicnn	78.81	81.04	76.92	78.41	59.75	61.98	62.33	62.90
deam-vggish	78.40	82.14	79.33	81.55	62.32	64.86	66.47	67.83
emomusic-effnet	82.57	86.55	84.75	87.61	71.29	75.41	73.77	78.55
emomusic-musicnn	85.61	89.21	84.78	87.63	**74.80**	**78.76**	76.53	80.29
emomusic-vggish	**86.42**	**90.30**	**86.86**	**89.73**	70.81	77.03	74.51	**81.16**
muse-effnet	59.92	60.99	59.00	62.11	62.14	63.78	61.54	64.35
muse-musicnn	63.96	66.59	64.84	68.55	67.72	70.77	69.03	71.01
muse-vggish	66.34	69.03	64.63	68.00	62.27	66.35	62.50	68.22
Spotify API	83.31	86.67	83.17	85.95	73.44	74.59	**77.51**	77.68

Table 4. External validation results on the proposed MusAV dataset (the percentage of track pairs with a correctly predicted ordering). FA+MA: pairs with full or majority agreement. FA: pairs with full agreement. CT: only pairs belonging to consistent triplets. The highest values are marked in bold. The top three AV models are marked in gray.

on the datasets used for training the models.

5.2 External validation on MusAV

We used our new dataset to validate the performance of the models. To this end, we assessed whether the ground truth ordering of track pairs coincided with the ordering according to the AV values predicted by the models. In addition, we also evaluated arousal and valence estimations provided by Spotify API and computed from audio as an additional reference. [15] This reference possibly represents a common state of the art in industrial systems. We ensured that our external validation set is independent of the datasets used for training the models: EmoMusic and DEAM contain non-commercial music unavailable on Spotify, while the intersection with MuSe, for which we also used Spotify track previews, includes 24 tracks that we filtered out for our evaluation.

For simplicity, we discarded all ground-truth annotations marking two songs as equivalent (13% and 15% of the ground-truth pairs with full or majority agreement in the case of arousal and valence, respectively). Thus, we focused only on examples with clear difference in arousal or valence. Table 4 presents the accuracy of the models in terms of the percentage of track pairs with correct ordering. We report the results on different subsets of the AV ground truth to demonstrate various evaluation possibilities.

5.3 Discussion

Having a new common ground truth for all models, our external validation shows the impact of the training dataset and embeddings.

Remarkably, models trained on the EmoMusic dataset perform the best for both arousal and valence regression. This is surprising, given that this dataset is smaller and less diverse than DEAM, which was derived from EmoMusic. On the other side, models based on MuSe have the worst performance in the case of arousal. Even though MuSe is the largest dataset in terms of size and coverage of commercially-available music, it appears to be too noisy to be able to train efficient models for arousal. Notably, it is the only dataset out of three relying on user-generated tags instead of explicit AV annotations, and the employed process for mapping tags to AV values might be inherently noisier.

Second, given a dataset, the choice of embedding model also matters. For example, the *effnet* embeddings trained on a large music style dataset appear to be inefficient for emotion recognition. The models based on them are consistently worst in the case of arousal (with all three datasets used for training) and valence (with EmoMusic and MuSe used for training). In turn, our validation reveals high performance of the *vggish* embeddings in many cases, possibly due to their generalization ability which was previously evidenced in literature [24]. This observation contradicts the results obtained in the respective held-out sets, where it did not have a remarkable performance overall.

Finally, in our validation, some of the considered AV models have performance competitive with an industrial reference. Still, all of the considered models only achieve up to 90% accuracy for arousal and 81% for valence. This is in line with evidence that predicting valence (as well as its annotation) is generally considered more complex than arousal [17, 42].

6. CONCLUSIONS

We present a new public dataset of relative AV annotations for validation of audio-based AV models. To build it, we employ a methodology that maximizes coverage in terms of genres to gather our annotation pool and allows to assess consistency of annotations on triplets of songs. The dataset is based on audio previews from Spotify API which allows validating performance on diverse types of commercially-available music. As an example, we train and evaluate AV regression models based on three common AV datasets and three types of pretrained audio embeddings and show how such a benchmarking can provide valuable complementary information about model performances. The resulting pretrained models are publicly available as part of Essentia models.

[15] We consider the "energy" descriptor in the Spotify API as arousal.

7. ACKNOWLEDGEMENTS

This research was carried out under the project Musical AI - PID2019-111403GB-I00/AEI/10.13039/501100011033, funded by the Spanish Ministerio de Ciencia e Innovación and the Agencia Estatal de Investigación. We also thank Juan Sebastián Gómez Cañón for his suggestions and all participating annotators.

8. REFERENCES

[1] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, "Music emotion recognition: A state of the art review," in *International Society for Music Information Retrieval Conference (ISMIR 2010)*, 2010.

[2] Y.-H. Yang and H. H. Chen, *Music emotion recognition*. CRC Press, 2011.

[3] J. Grekow, *From content-based music emotion recognition to emotion maps of musical pieces*. Springer International Publishing, 2018.

[4] X. Hu, M. Bay, and J. S. Downie, "Creating a simplified music mood classification ground-truth set," in *International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[5] C. Laurier, M. Sordo, J. Serra, and P. Herrera, "Music mood representations from social tags." in *International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[6] X. Downie, C. Laurier, and M. Ehmann, "The 2007 MIREX audio mood classification task: Lessons learned," in *International Symposium on Music Information Retrieval (ISMIR 2008)*, 2008.

[7] C. Laurier and P. Herrera, "Audio music mood classification using support vector machine," in *International Conference on Music Information Retrieval (ISMIR 2007)*, 2007.

[8] K. Bischoff, C. S. Firan, R. Paiu, W. Nejdl, C. Laurier, and M. Sordo, "Music mood and theme classification-a hybrid approach." in *International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[9] D. Bogdanov, A. Porter, P. Tovstogan, and M. Won, "MediaEval 2019: Emotion and theme recognition in music using Jamendo," in *MediaEval 2019 Workshop*, 2019.

[10] Y.-H. Yang, Y.-C. Lin, Y.-F. Su, and H. H. Chen, "A regression approach to music emotion recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 448–457, 2008.

[11] J. A. Russell and A. Mehrabian, "Evidence for a three-factor theory of emotions," *Journal of Research in Personality*, vol. 11, no. 3, pp. 273–294, 1977.

[12] D. M. Greenberg, M. Kosinski, D. J. Stillwell, B. L. Monteiro, D. J. Levitin, and P. J. Rentfrow, "The song is you: Preferences for musical attribute dimensions reflect personality," *Social Psychological and Personality Science*, vol. 7, no. 6, pp. 597–605, 2016.

[13] T. Eerola and J. K. Vuoskoski, "A comparison of the discrete and dimensional models of emotion in music," *Psychology of Music*, vol. 39, no. 1, pp. 18–49, 2011.

[14] J. A. Russell, "A circumplex model of affect." *Journal of Personality and Social Psychology*, vol. 39, no. 6, pp. 1161–1178, 1980.

[15] T. Eerola, O. Lartillot, and P. Toiviainen, "Prediction of multidimensional emotional ratings in music from audio using multivariate regression models." in *International Society for Music Information Retrieval Conference (ISMIR 2009)*, 2009.

[16] M. Soleymani, A. Aljanaki, and Y. Yang, "DEAM: MediaEval database for emotional analysis in music," 2016. [Online]. Available: https://cvml.unige.ch/databases/DEAM/manual.pdf

[17] M. Soleymani, M. N. Caro, E. M. Schmidt, C.-Y. Sha, and Y.-H. Yang, "1000 songs for emotional analysis of music," in *ACM International Workshop on Crowdsourcing for Multimedia (CrowdMM 2013)*, 2013.

[18] J. Grekow, "Music emotion maps in arousal-valence space," in *IFIP International Conference on Computer Information Systems and Industrial Management (CISIM 2016)*, 2016.

[19] J. Bai, J. Peng, J. Shi, D. Tang, Y. Wu, J. Li, and K. Luo, "Dimensional music emotion recognition by valence-arousal regression," in *IEEE International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC 2016)*, 2016.

[20] C. Akiki and M. Burghardt, "MuSe: The musical sentiment dataset," *Journal of Open Humanities Data*, vol. 7, no. 10, 2021.

[21] K. W. Cheuk, Y.-J. Luo, B. Balamurali, G. Roig, and D. Herremans, "Regression-based music emotion prediction using triplet neural networks," in *International joint conference on neural networks (IJCNN 2020)*, 2020.

[22] A. Livshin, "The importance of cross database evaluation in musical instrument sound classification," in *International Conference on Music Information Retrieval (ISMIR 2004)*, 2003.

[23] D. Bogdanov, A. Porter, P. Herrera, and X. Serra, "Cross-collection evaluation for music classification tasks," in *International Society for Music Information Retrieval Conference (ISMIR 2016)*, 2016.

[24] P. Alonso-Jiménez, D. Bogdanov, J. Pons, and X. Serra, "TensorFlow audio models in Essentia," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, 2020.

[25] Yi-Hsuan Yang and H. H. Chen, "Ranking-based emotion recognition for music organization and retrieval," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 762–774, 2011.

[26] J. Fan, K. Tatar, M. Thorogood, and P. Pasquier, "Ranking-based emotion recognition for experimental music," in *International Society for Music Information Retrieval Conference (ISMIR 2017)*, 2017.

[27] A. Metallinou and S. Narayanan, "Annotation and processing of continuous emotional attributes: Challenges and opportunities," in *IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG 2013)*, 2013.

[28] G. N. Yannakakis and H. P. Martinez, "Grounding truth via ordinal annotation," in *IEEE International Conference on Affective Computing and Intelligent Interaction (ACII 2015)*, 2015.

[29] R. T. Dean, F. Bailes, and E. Schubert, "Acoustic intensity causes perceived changes in arousal levels in music: An experimental investigation," *PLOS One*, 2011.

[30] P. Kuppens, F. Tuerlinckx, J. A. Russell, and L. F. Barrett, "The relation between valence and arousal in subjective experience," *Psychological Bulletin*, vol. 139, no. 4, pp. 917–940, 2013.

[31] P. Kuppens, F. Tuerlinckx, M. Yik, P. Koval, J. Coosemans, K. J. Zeng, and J. A. Russell, "The relation between valence and arousal in subjective experience varies with personality and culture," *Journal of Personality*, vol. 85, no. 4, pp. 530–542, 2017.

[32] J. S. Gómez-Cañón, E. Cano, T. Eerola, P. Herrera, X. Hu, Y.-H. Yang, and E. Gómez, "Music emotion recognition: Toward new, robust standards in personalized and context-sensitive applications," *IEEE Signal Processing Magazine*, vol. 38, no. 6, pp. 106–114, 2021.

[33] J. Gómez-Cañón, E. Cano, Y. Yang, P. Herrera, and E. Gómez, "Let's agree to disagree: Consensus entropy active learning for personalized music emotion recognition," in *International Society for Music Information Retrieval Conference (ISMIR 2021)*, 2021.

[34] T. Li and M. Ogihara, "Detecting emotion in music," in *International Conference on Music Information Retrieval (ISMIR 2003)*, 2003.

[35] L. Lu, D. Liu, and H.-J. Zhang, "Automatic mood detection and tracking of music audio signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 1, pp. 5–18, 2005.

[36] F. Weninger, F. Eyben, and B. Schuller, "On-line continuous-time music mood regression with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)*, 2014.

[37] R. Panda, R. Malheiro, and R. P. Paiva, "Novel audio features for music emotion recognition," *IEEE Transactions on Affective Computing*, vol. 11, no. 4, pp. 614–626.

[38] B. Bhattarai and J. Lee, "Automatic music mood detection using transfer learning and multilayer perceptron," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 19, no. 2, pp. 88–96, 2019.

[39] M. Soleymani, M. N. Caro, E. M. Schmidt, and Y.-H. Yang, "The MediaEval 2013 brave new task: Emotion in music," in *MediaEval 2013 Workshop*, 2013.

[40] A. Aljanaki, Y.-H. Yang, and M. Soleymani, "Emotion in Music Task at MediaEval 2014," in *MediaEval 2014 Workshop*, 2014.

[41] ——, "Emotion in Music Task at MediaEval 2015," in *MediaEval 2015 Workshop*, 2015.

[42] ——, "Developing a benchmark for emotional analysis of music," *PLOS One*, vol. 12, no. 3, p. e0173392, 2017.

[43] S. Zhao, Y. Li, X. Yao, W. Nie, P. Xu, J. Yang, and K. Keutzer, "Emotion-based end-to-end matching between image and music in valence-arousal space," in *ACM International Conference on Multimedia (MM 2020)*, 2020.

[44] H. Liu, Y. Fang, and Q. Huang, "Music emotion recognition using a variant of recurrent neural network," in *International Conference on Mathematics, Modeling, Simulation and Statistics Application (MMSSA 2018)*, 2019.

[45] A. B. Warriner, V. Kuperman, and M. Brysbaert, "Norms of valence, arousal, and dominance for 13,915 English lemmas," *Behavior Research Methods*, vol. 45, no. 4, pp. 1191–1207, 2013.

[46] J. Fan, Y.-H. Yang, K. Dong, and P. Pasquier, "A comparative study of western and chinese classical music based on soundscape models," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, 2020.

[47] J. Fan, M. Thorogood, and P. Pasquier, "Emo-soundscapes: A dataset for soundscape emotion recognition," in *IEEE International Conference on Affective Computing and Intelligent Interaction (ACII 2017)*, 2017.

[48] D. Yang and W.-S. Lee, "Disambiguating music emotion using software agents." in *International Conference on Music Information Retrieval (ISMIR 2004)*, 2004.

[49] European Broadcasting Union (EBU), "EBU Tech 3341. Loudness metering: 'EBU mode' metering to supplement loudness normalisation in accordance with EBU R 128."

[50] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. Zapata, and X. Serra, "ESSENTIA: An audio analysis library for music information retrieval," in *International Society for Music Information Retrieval Conference (ISMIR 2013)*, 2013.

[51] N. F. Gutiérrez Páez, J. S. Gómez-Cañón, L. Porcaro, P. Santos, D. Hernández-Leo, and E. Gómez, "Emotion annotation of music: A citizen science approach," in *International Conference on Collaboration Technologies and Social Computing (CollabTech 2021)*, 2021.

[52] J. Pons and X. Serra, "musicnn: Pre-trained convolutional neural networks for music audio tagging," in *International Society for Music Information Retrieval Conference (ISMIR 2019) Late Breaking Demo*, 2019.

[53] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "CNN architectures for large-scale audio classification," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*, 2017.

[54] P. Alonso-Jiménez, X. Serra, and D. Bogdanov, "Music representation learning based on editorial metadata from Discogs," in *International Society for Music Information Retrieval (ISMIR 2022)*, 2022.

[55] C. J. Willmott, "On the validation of models," *Physical geography*, vol. 2, no. 2, pp. 184–194, 1981.

[56] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination r-squared is more informative than smape, mae, mape, mse and rmse in regression analysis evaluation," *PeerJ Computer Science*, vol. 7, p. e623, 2021.

WHAT IS MISSING IN DEEP MUSIC GENERATION? A STUDY OF REPETITION AND STRUCTURE IN POPULAR MUSIC

Shuqi Dai *
Carnegie Mellon University
shuqid@cs.cmu.edu

Huiran Yu *
Carnegie Mellon University
huiranyu@cs.cmu.edu

Roger B. Dannenberg
Carnegie Mellon University
rbd@cs.cmu.edu

ABSTRACT

Structure is one of the most essential aspects of music, and music structure is commonly indicated through repetition. However, the nature of repetition and structure in music is still not well understood, especially in the context of music generation, and much remains to be explored with Music Information Retrieval (MIR) techniques. Analyses of two popular music datasets (Chinese and American) illustrate important music construction principles: (1) structure exists at multiple hierarchical levels, (2) songs use repetition and limited vocabulary so that individual songs do not follow general statistics of song collections, (3) structure interacts with rhythm, melody, harmony and predictability, and (4) over the course of a song, repetition is not random, but follows a general trend as revealed by cross-entropy. These and other findings offer challenges as well as opportunities for deep-learning music generation and suggest new formal music criteria and evaluation methods. Music from recent music generation systems is analyzed and compared to human-composed music in our datasets, often revealing striking differences from a structural perspective.

1. INTRODUCTION

Structure is fundamental to music, as seen in the focus on form and analysis in music theory [1–3], music segmentation [4–6], structure analysis [7–9] and chorus detection [10] in MIR research, and recently in the attention given to long-term dependencies in music sequence generation using deep learning techniques [11, 12]. As a basic indicator of music structure, repetition contains important music information. Music relies heavily on repetition to create internal references, coherence and structure.

Unfortunately, the nature of repetition and structure in music is still not well understood, and much remains to be explored with music information retrieval techniques. For example, while music theory may suggest that songs have distinctive motives, our work quantifies and generalizes this notion. We will use "structure" to refer broadly to organizing principles in music, which are generally hierarchical and include sections, phrases and various kinds of patterns. A primary generator of structure is repetition, which includes not just music content within repeat signs but also approximate repetitions at different time scales.

In music generation, many researchers rely on deep learning models to capture music structure and organizing principles implicitly from data. However, repetition, especially long-term repetition structure, does not seem to emerge automatically in deep music generation. Deep learning is a promising direction, but such research should include evaluations where we can assess the successes and failures of new approaches. Moreover, some may argue that we do not need deep learning models to learn prevalent repetitions in music: we can produce repetition simply by generating phrases to the desired length and pasting them into a template. However, we will see that phrase structure, song structure, and other elements of music are intertwined, making this simple approach unable to reproduce characteristics of actual songs. Thus, we need a better understanding of repetition if we want machines to compose or even just listen to music in a more human way.

We aim to use formal models to explore music repetition and structure. By characterizing structural information in music, we can discover new principles of music organization and propose new challenges and evaluation strategies for music information retrieval and music generation.

An essential aspect of repetition structure is hierarchy. We use objective data analysis to support the existence and significance of multiple levels of hierarchy in popular music. We also present a number of results that show strong interactions between structure and pitch, rhythm, harmony, entropy and cross-entropy. Simply stated, structure can help predict pitch (or rhythm, harmony, etc.) and pitch (or rhythm, harmony, etc.) can help predict structure. These findings, in turn, challenge and inform research on deep learning to model hierarchical music structure.

Another important effect of repetition is that song-specific vocabulary of rhythm and pitch patterns is limited relative to what would be expected from the entire dataset. This vocabulary serves both to unify multiple phrases within a song and distinguish songs from others.

The main contribution of this work is a better understanding of the nature of repetition in popular music. We will see that repetition exists in many forms and at different levels of hierarchy. We offer ways to quantify music repetition structure, especially as it relates to pitch and

* The first two authors have equal contribution.

© S. Dai, H. Yu, and R. B. Dannenberg. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** S. Dai, H. Yu, and R. B. Dannenberg, "What is missing in deep music generation? A study of repetition and structure in popular music", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

rhythm, often by measuring entropy or cross-entropy. We also reveal striking differences from a structural perspective through case studies on recent deep music generation models. These and other findings offer challenges as well as opportunities for deep-learning music generation and suggest new formal music criteria and evaluation methods.

After describing related work in the next section, we discuss music repetition and structure in Section 3, how it relates to deep music generation in Section 4, and we present conclusions in Section 5.

2. RELATED WORK

Repetition is a key element of music structure. Repetition is one of the three commonly used principles for segmenting music, along with novelty at segment boundaries and homogeneity within segments [7]. People have developed a variety of segmentation and section detection methods based on repetition with acoustic features [4, 10]. Repetition becomes especially useful in segmenting symbolic music or lead sheet representations where timbre and texture may be lacking [13].

Repetition also plays an important role in music expectation and prediction [14, 15]. Studies of repetition and structure are common in Music Psychology [16]. For example, listening experiments with reordered Classical and Popular music have shown that listeners are rather insensitive to restructuring, but these results are subtle and somewhat ambiguous [17]. Music form and structure, including repetition, is also a major focus of Music Theory [1,18,19].

There are many deep learning models for music generation [11, 20–23], however, capturing repetition and long-term dependencies in music still remains a challenge. One mainstream approach is to model distribution of music via an intermediate representation (embedding), such as Variational Auto-Encoders (VAE) [20,24], Generative Adversarial Networks (GANs) [25] and Contrastive Learning [24, 26]. Due to their fixed-length representation and short-length output, it is difficult to exhibit long-term structure. Another popular trend is to use sequential models such as LSTMs and Transformers [11, 22, 27] to generate longer music sequences, but they still struggle to generate repetition and coherent structure on long-term time scales. Some recent work introduces explicit structure planning for music generation, which shows that using structure information leads to better musicality [12, 28, 29].

Current evaluation methods for music generation rarely consider repetition and structure. Deep music generation systems [11,30] use objective metrics such as negative log-likelihood, cross-entropy and prediction accuracy to compare generated music with ground-truth human-composed music. But these metrics do not precisely correspond to human perception and are not reliable for musicality. Another trend is using domain-knowledge [31] and musical features [32–35] such as pitch class, pitch intervals, and rhythm density to evaluate music statistically. However, most of them ignore even short patterns, and none evaluate music structure. In contrast, we offer quantitative and objective methods to evaluate music repetition and structure.

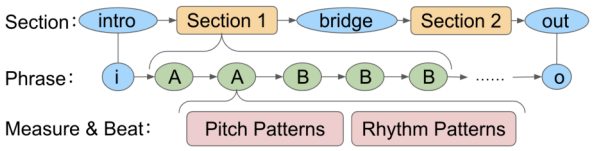


Figure 1: Structure hierarchy in pop music.

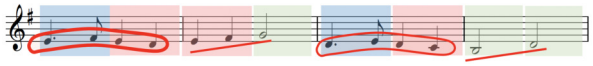


Figure 2: Repeated motives in a phrase in *Yankee Doodle*.

3. REPETITION AND STRUCTURE IN MUSIC

We are interested in three main problems concerning repetition and structure in music: (a) How are repetition and structure organized hierarchically? (b) How do different levels of structure interplay with other music elements? (c) How does repetition play out over time?

Unlike traditional music theory with case-by-case human analysis, we explore these problems in a data-driven approach. For training and testing, we use a Chinese pop song dataset POP909 [36], which has 909 pop song performances in MIDI, and an American pop song dataset PDSA [37], in MusicXML, which has 348 American pop songs originating from 1580 to 1924. We use only songs in 4/4 time to simplify analysis.

3.1 Repetition and Structure Hierarchy

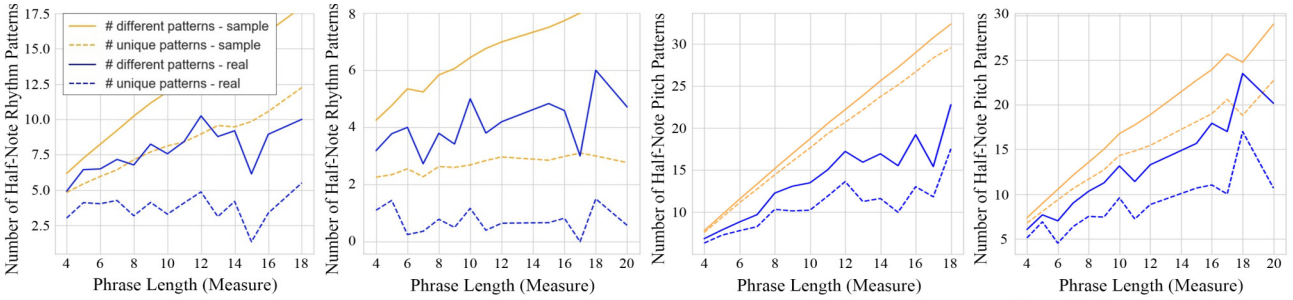
Music structure is hierarchical. It contains multiple levels of repetitions, ranging from low-level pitch and rhythm motives (patterns) to higher-level phrases (analogous to sentences) and sections (analogous to paragraphs). We describe these in more detail and use statistical and machine learning findings in the data to explore their significance.

3.1.1 Phrases and Sections

Researchers [13] found two levels of structure in POP909: *sections* and *phrases*. Phrases were identified by searching (automatically) for approximate repetitions of sequences of 4 or more measures. Non-unique phrases (those that match other phrases) often occur in sequences such as "AABBB" in Figure 1 called *sections*, which are by definition separated by non-repeated or non-melodic phrases. For example, Figure 1 has an intro, two sections connected by a bridge transition, and an outro, which is a typical pop song structure. Here "A" can be a verse, and "B" can be a chorus phrase. These phrase and section levels of structure were found to be predictive of aspects of pitch, rhythm and harmony, which shows that these two levels have significance for composition, probably for perceptual reasons.

3.1.2 Repetition Below the Phrase Level

There is at least another level of repetition below phrases (Figure 1). For example, in the first 8-measure phrase of the chorus in *Yankee Doodle* (Figure 2), the first and second half repeat elements of rhythm and interval. The colored boxes show repeated rhythm patterns, and the red lines point out repeated pitch contours. We want to assess whether this kind of repetition is common.



(a) POP909 phrases – rhythm pattern (b) PDSA phrases – rhythm pattern (c) POP909 phrases – pitch pattern (d) PDSA phrases – pitch pattern

Figure 3: Average number of different patterns (solid) and unique patterns (dashed) within a phrase vs. phrase length. (a)(b) are patterns of half-note melody note onset, (c)(d) are half-note pitch patterns. Blue lines are real phrases in the dataset. Orange lines are sampled phrases constructed by choosing each pattern at random from the entire dataset distribution.

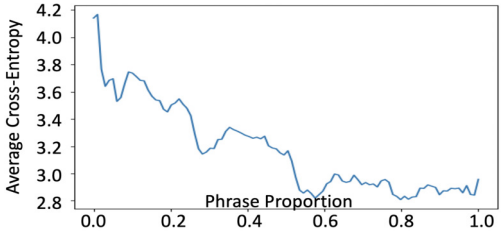


Figure 4: Average cross-entropy of diatonic pitch predictions over time within a phrase in POP909, using variable-order Markov models.

We extract the phrase-level structure in PDSA using the algorithm in [13], and use human-labeled structure in [13] for POP909. We study rhythm patterns by segmenting melody sequences into half-note onset patterns. PDSA has 54 different half-note patterns, while POP909 has all 128 possible patterns (onsets are quantized to 16th notes, and we assume an initial onset).

We claim that phrases have a limited vocabulary of onset patterns and therefore much repetition. This will of course be true necessarily if the entire dataset has a very limited vocabulary, so as a baseline for comparison, we construct random phrases by sampling half-note onset patterns from the entire dataset distribution. For POP909 songs, Figure 3(a) shows the average number of different onset patterns in phrases of different lengths (solid lines) and also the average number of patterns that occur only once in the phrase (dashed lines). This allows us to evaluate whether patterns within phrases (blue) have similar distributions to those of the entire dataset (orange). A similar graph for PDSA songs is shown in Figure 3(b). The analysis of pitch patterns is similar. Figure 3(c)(d) presents counts of melodic pitch patterns analogous to the onset pattern counts. Again, we see that real song phrases have a limited vocabulary compared to phrases assembled from random half-note units representing the entire dataset, and fewer real phrases go unrepeated.

If repetition of small patterns is frequent, variable-order Markov models [38, 39] can make good song-specific online predictions by updating the model at each element of the observed sequence. Conceptually, a variable-order Markov model estimates separate Markov chains of order 0 to N based on observed state (e.g., pitch) transitions. When data is too sparse to use a higher order, the model falls back (iteratively) to the next lower order. Repetitions of different lengths are readily learned, thus prediction accuracy indicates repetition significance. Figure 4 shows the average cross-entropy for pitch prediction over the length of all phrases, with phrase length normalized to 1. The clear downward trend shows the extent to which internal repetition enables better prediction.

In summary, we find abundant evidence for repetition within phrases:

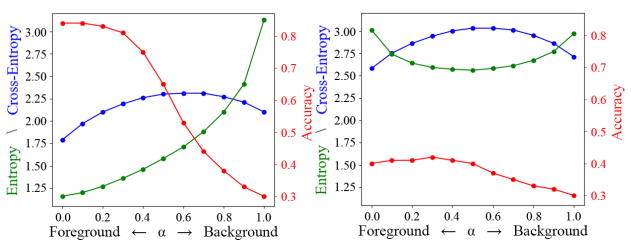
- Compared to sampling onset patterns at random from the POP909 distribution, real phrases have fewer distinct onset patterns, and more onset patterns are repeated in the phrase (Fig 3(a)). Same trend occurs in PDSA (Fig 3(b)), whose phrases contain even fewer distinct onset patterns.
- Rhythm patterns show a clear repetition structure within phrases. Considering each measure as a rhythm pattern, in PDSA, 7% of phrases have the same rhythm pattern in every measure, 28% have a repetition structure in one of the forms: "abab," "aabbaabb," "aba," or "abababa."
- The vocabulary of pitch patterns within a song or phrase is also very limited compared to the whole dataset, implying pitch sequence repetitions within the phrase level.
- The cross-entropy of predicting pitches decreases over time in a phrase, suggesting within-phrase repetition.

3.1.3 Song-Specific Vocabulary and Common Patterns

Motives and patterns below the phrase level create a song-specific vocabulary and have a long-term dependency throughout the whole piece. For example, in Beethoven's Symphony No.5, the first movement begins with a distinctive four-note "short-short-short-long" (the famous "Fate Motive"). It repeats everywhere throughout the piece, and Beethoven arranged it logically to unify the entire work.

We chose variable-order Markov models to further explore patterns because (a) it is easy to tune the weights between models trained on different data, (b) models can learn different lengths of pattern repetition. The following results show 1) song-specific vocabulary is critical in prediction, 2) there is long-term dependency between phrases.

To see the effects of song-specific vocabulary and context, we compare the results of variable-order Markov models predicting pitch trained on many other songs (background model), versus training on the same song after removing a short segment to be predicted (foreground model). Predictions are a linear combination of the back-



(a) Including duplicated phrases in foreground training

(b) Hold out repeated phrases in foreground training

Figure 5: Average entropy, cross-entropy and accuracy of predicting the first eight diatonic pitches in each phrase, tuning between foreground and background in POP909.

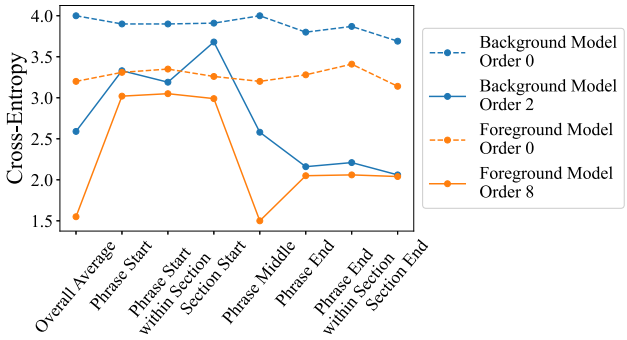


Figure 6: Average cross-entropy on diatonic pitches at different structure level positions in POP909 dataset predicted by background and foreground variable Markov models.

ground and foreground models. Figure 5(a) shows that the pure foreground model always obtains the best results. One possible reason is that phrase-level repetition in the foreground allows memorization. Thus, we removed all the duplicated or similar phrases in the foreground training and tested only on the first eight notes in each phrase (without updating the model) to eliminate both the effects of phrase-level repetitions and within-phrase motive repetitions. With almost no information from the repeated phrase, we might expect little information within a song to help predict pitches. However, Figure 5(b) indicates that we obtain the best prediction accuracy when incorporating 70% foreground and 30% background information. Since duplicate phrases have been removed and the first eight notes involve little within-phrase repetition, the results show that there must be a song-specific vocabulary or context that repeats across different phrases throughout the song, but not across the database (otherwise background would work better).

3.2 Structural Influence

Can repetition structure be considered orthogonal to melody, harmony and rhythm generation? If so, we can simply generate music note-by-note, ignoring structure, and then impose structure by repeating generated sequences. However, if there is significant interaction between structure and other facets of music, then structure is an integral part of music analysis, music modeling, and music generation. Our findings show the latter is the case.

Previous work has found systematic interactions between repetition structure and three facets of music: melody, rhythm and harmony, with interactions at both the phrase level and the section level [13]. For example, chords at the ends of phrases differ from chords in the middle of phrases. Furthermore, final chords in sections differ significantly from final chords in phrases that are not at the ends of sections. This does not mean that "good music" must reflect a structural hierarchy, but at least this finding offers insight into how music generation might be improved, and it raises questions for further study.

In this work, we have also explored confidence and surprise in music construction to better understand the role of different levels of structure in music. In Figure 6, we trained on the entire dataset (background model) of melody pitch sequences, holding out test songs; and also trained on single songs (foreground model), holding out all phrase repetitions to eliminate prediction by memorizing phrases, and holding out eight notes at a time for testing. Only order 0 (histogram) and the best-performing order maxima (2 and 8, respectively) are shown. As seen in the previous section, the foreground model predictions are better by 1 bit on average, indicating more repetition within songs (foreground) than across songs (background).

Even though the Markov models have no positional encoding or conditions, we can still see a dramatic difference in predictability at different structure positions (Figure 6). For example, using the background model, cross-entropy at the start of sections is over 3.5 (bits), while other phrase starts are about 3.2, and for most notes (phrase middle), cross-entropy is only 2.6. Using the foreground model, we see a different pattern, but predictability still varies substantially according to position in the structure.

3.3 Repetition and Structure Over Time

Music is not repeated randomly. After seeing different levels of hierarchy in Section 3.1, we ask: Is there a schema for repetition? How does repetition play out over time?

Huron suggests that if music is to manipulate prediction through repetition, it makes sense to repeat some of the music early on [14]. This affords immediate pleasure from successful prediction rather than delaying until all novel material is exhausted. POP909 supports this hypothesis: More than 50% of the phrases repeat immediately, and almost all phrases repeat within a quarter of the song.

In Section 3.1, we have seen that most rhythmic and melodic patterns in a phrase are repetitions. At the song level as well, using the phrase repetition labels in POP909, we found that for 79% of songs, 15% to 35% of their duration consists of new material and the rest is repetition.

Returning to our consideration of structure over time: How does surprise vary with structure? We might expect less surprise at the ends of sections to give a sense of completion or resolution. Figure 7 Left shows a histogram percentage of phrase-level repetition over the course of a song. We see a relatively low repetition rate in the first 1/10 of the song. The repetition rate sharply increases as we progress to the first 1/5 of the song because, after introducing the initial materials, most songs repeat them. There is a notice-

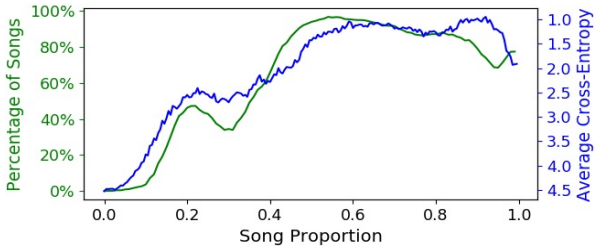


Figure 7: **Left** (green): Percentage of POP909 songs that have phrase repetitions at different song locations. **Right** (blue): Average prediction cross-entropy using variable-order Markov models on diatonic pitches in POP909 songs over time, and note the y-axis is inverted.

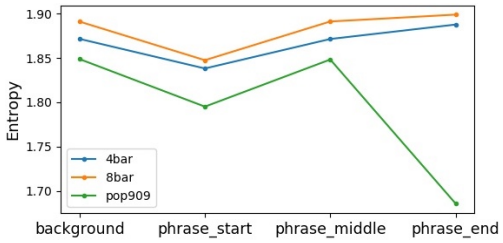


Figure 8: Entropy of pitch scale degree distribution at different phrase positions, comparing generated 4-bar and 8-bar phrases from VAE with real phrases in POP909.

able drop around the quarter-way point where many songs introduce new material. In the second half, almost everything is a repetition or variation of what came in the first half. Finally, the graph shows novel material is often introduced near the end. In Figure 7 Right, we use the variable-order Markov model to calculate the average cross-entropy over time on melody pitches. To show the correspondence, we flip the vertical cross-entropy axis. Note the similarity between the trends in repetition and cross-entropy.

From these results, it is clear that repetition is not random but follows a plan in which novelty is revealed, presumably to enhance the enjoyment or effect of the music. This is perhaps surprising because other research shows that music can be substantially rearranged without destroying positive impressions [40]. Whether this organization matters to listeners or simply reflects composers' intentions requires further research.

4. IMPLICATIONS FOR DEEP MUSIC GENERATION

One application of our studies is to gain insight into deep learning models for music generation. We can apply the analyses from Section 3 to melodies generated from deep learning models. Through these case studies, we can characterize repetition and structure from deep music generation and compare to human-composed songs. To be clear, we are not claiming that any particular structure is *necessary* or even *good*. Our goal is to illuminate possibilities and better understand both real and generated music. We then discuss our results and point out new directions and ideas for future work in deep music generation.

4.1 Lack of Repetition and Structure

We studied repetition and structure in deep learning generated melodies to answer three questions: 1) do melodies have multiple levels of repetition and structure? 2) do they have song-specific vocabulary and common patterns? 3) how does cross-entropy vary over time? We used two deep music generation models: One is a VAE model using representation learning [24], chosen because it uses contrastive learning to generate longer sequences (8 bars) than other VAE models. The other model is Music Transformer [11].

4.1.1 Do deep melodies have multiple levels of structure?

We want to know if deep-learning-generated music has multiple levels of structure and whether any structure is reflected in pitch and rhythm. Unfortunately, most deep learning models can generate only a limited length: VAEs and GANs usually have a fixed length from 2 to 8 bars, about the length of just one phrase. Sequential models like Transformer and LSTM can generate longer sequences, but we cannot find any salient repetition comparable to repeated phrases by listening or by automated analysis. These differences from popular songs are striking.

Thus, the only thing we can do is to test on short generated sequences, consider them as a phrase, and see if they have phrase structure. We tested the 4 bar and 8 bar generated phrases from a VAE model [24] and used 1000 examples of each duration. Figure 8 shows that there is no significant differences between the entropy of pitch scale degree distributions at the start of phrases, middle of phrases and end of phrases, but we see significant differences in the real POP909 phrases. Also, the probabilities of different pitches at different phrase-level positions only change slightly in the analyzed outputs. For example, the probability of seeing the tonic at phrase end is 35% in real POP909 phrases, while at the other positions it is around 20%. The probability of seeing the tonic in VAE output is in the range of 20% to 21% for all positions.

4.1.2 Do deep melodies have song-specific vocabulary and common patterns?

Following the logic in Figure 3, we count the rhythm (melody note onset) patterns for whole songs from Music Transformer and from POP909 itself. We trained the Music Transformer on 80% of songs in POP909, using 10% for validation and 10% for testing. We initialized the generation with the first 2 to 6 bars of the test songs, but did not count the initialization bars as part of the generated melody. To calculate the song length, we use the total length of melodic phrases, omitting measures that are empty or have long rests.

In Figure 9, we see that transformer-generated results have a much higher rhythm vocabulary compared to real songs. Many people have observed that deep-learning-generated music has many outlier notes and patterns that appear once and sound like mistakes. Here we see that transformer-generated melodies have a higher number of unique rhythm patterns compared to real songs, which might explain the sense of too many outliers.

We also analyzed half-note pitch patterns in 4-bar and 8-bar phrases from the PDSA, from random pitch patterns drawn from the entire PDSA distribution, and for the VAE

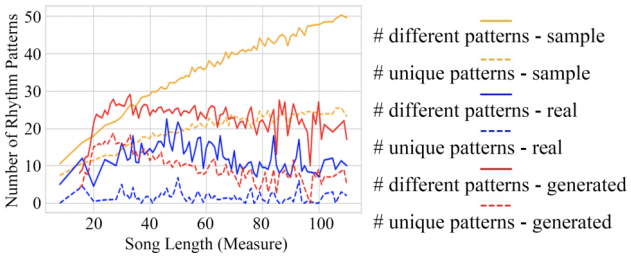


Figure 9: Analysis of note onset pattern vocabulary from randomly sampled POP909 patterns (yellow), Music Transformer songs (red), and POP909 songs (blue) as a function of song length. Solid lines are the total number of different patterns, and dashed lines are the number of unique (not repeated) patterns.

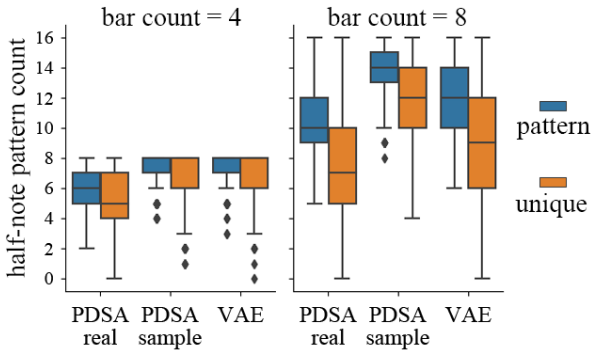


Figure 10: Comparison of half-note pitch pattern counts in PDSA phrases, randomly sampled PDSA pattern phrases, and phrases from VAE. The y-axis counts the number of different patterns (blue) or number of unique (not repeated) patterns (orange) found in 4 or 8 bars. The VAE generated patterns are significantly more than the patterns in the real PDSA dataset, with p-value less than 10^{-5}.

model [24] (see Figure 10). Vocabulary is more limited in PDSA phrases compared to the random patterns and VAE patterns. The VAE pattern counts are closer to those of synthetic phrases and thus may lack some of the coherence, redundancy, and predictability of PDSA.

4.1.3 Entropy over time

We also compared trends in cross-entropy of the pitch sequence over the course of songs from POP909 (Figure 7 blue) and Music Transformer (Figure 11). Although both exhibit an overall cross-entropy decrease over the course of the phrase, there are important differences. POP909 cross-entropy increases at around 20% of the song length, probably due to the introduction of novel and contrasting patterns, and also around the end. POP909's average cross-entropy is greater than one bit per note except for a small region around 90% of song length, while Music Transformer is below one on average for the last 30% of the song, suggesting overly predictable sequences.

4.2 Discussion and New Directions

Rather than learning and reproducing general statistics of datasets, we need to learn how songs strategically diverge from background or stylistic norms to create interest, surprise, and individuality. It is particularly interesting that

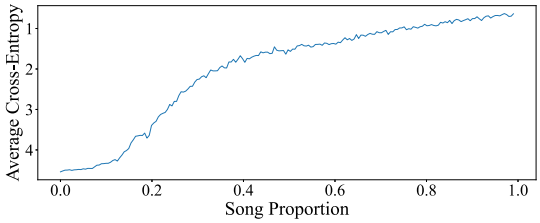


Figure 11: Average prediction cross-entropy using Markov models on diatonic pitches over time in 910 generated music pieces from music transformer, y-axis inverted.

phrases can be better predicted by relatively short phrases within the same song than by large amounts of training data from other songs. It seems plausible that songs of the same artist or same sub-genre may be more predictive than songs in general. Our findings also reinforce previous findings that using structure in MusicFrameworks [12] results in better human evaluations.

Examples in Section 4.1 suggest that we can compare generated music to real music using measures of structure, repetition and entropy. Matching these measures is not guaranteed to make music "better," but we should not simply ignore clear objective differences. We would at least expect differences to be small when the task is to imitate a style or genre. We can also speculate that these measures are relevant to listener preferences even if they do not tell a complete story.

Although we have focused on popular music, repetition and hierarchical structure seem to be ubiquitous in music. Pop music, with its nearly exact repetitions, seems easier to study than Classical music where we might expect more variation, development and modulation, which make repetition less obvious. We are encouraged by our results using variable-order Markov models on pitch sequences. These studies reveal much more information than we expected and do not rely on finding wholly duplicated measures, which we used to extract phrases and sections. Perhaps this approach will be of use in Classical and other music.

5. CONCLUSION

It should be no surprise that structure, repetition, pitch, rhythm, harmony and entropy are all strongly connected and interdependent. We have offered new ways to explore these connections objectively, using a data-driven approach without relying on subjective human analyses.

Among our findings are that within-song and within-phrase vocabulary and repetition are not a reflection of more general background statistics from a collection of songs. Instead, songs and phrases gain "individuality" through more repetition and smaller vocabulary. This has important implications for machine learning and music generation systems.

There are clear differences between measurements of real songs and those of many music generation systems, suggesting that there are important gaps to fill through new research. We hope that this work will inspire further research in the roles played by repetition and structure in music as well as methods to learn repetition and structure.

6. ACKNOWLEDGEMENT

We would like to thank Ziyue Piao and Ying Yee Wong for their assistance in the early stage.

7. REFERENCES

[1] W. E. Caplan, *Classical Form: A Theory of Formal Functions for the Instrumental Music of Haydn, Mozart, and Beethoven, Revised Edition*. Oxford University Press, 2000.

[2] P. Goetschius, *Lessons in music form: A manual of analysis of all the structural factors and designs employed in musical composition*. Library of Alexandria, 1904, vol. 1.

[3] D. M. Green, *Form in tonal music*. Holt, Rinehart and Winston, 1979.

[4] R. B. Dannenberg and M. Goto, "Music structure analysis from acoustic signals," *Handbook of Signal Processing in Acoustics*, vol. 1, pp. 305–331, 2009.

[5] P. Allegraud, L. Bigo, L. Feisthauer, M. Giraud, R. Groult, E. Leguy, and F. Levé, "Learning sonata form structure on mozartś string quartets," *Transactions of the Int. Society for Music Information Retrieval Conf.*, vol. 2, Dec. 2019.

[6] J. Salamon, "Deep embeddings and section fusion improve music segmentation," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[7] O. Nieto, G. J. Mysore, C. C. Wang, J. B. L. Smith, J. Schlüter, T. Grill, and B. McFee, "Audio-based music structure analysis: Current trends, open challenges, and applications," *Transactions of the Int. Society for Music Information Retrieval Conf.*, vol. 3, no. 1, p. 246–263, 2020.

[8] M. Hamanaka, K. Hirata, and S. Tojo, "Musical structural analysis database based on GTTM," in *Proc. of the 15th Int. Society for Music Information Retrieval Conf.*, 2014.

[9] E. N. G. Shibata, R. Nishikimi and K. Yoshii, "Statistical music structure analysis based on a homogeneity-, repetitiveness-, and regularity-aware hierarchical hidden semi-markov model," in *Proc. of the 20th Int. Society for Music Information Retrieval Conf.*, 2019.

[10] J. Paulus, M. Muller, and A. Klapuri, "Audio-based music structure analysis," in *Proc. of the 11th Int. Society for Music Information Retrieval Conf.*, 2010, pp. 625–636.

[11] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[12] S. Dai, Z. Jin, C. Gomes, and R. B. Dannenberg, "Controllable deep melody generation via hierarchical music structure representation," in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, 2021.

[13] S. Dai, H. Zhang, and R. B. Dannenberg, "Automatic analysis and influence of hierarchical structure on melody, rhythm and harmony in popular music," in *in Proc. of the 2020 Joint Conference on AI Music Creativity (CSMC-MuMe 2020)*, 2020.

[14] D. Huron, *Sweet Anticipation: Music and the Psychology of Expectation*. Cambridge, MA: MIT Press, 2006.

[15] E. Narmour *et al.*, *The analysis and cognition of melodic complexity: The implication-realization model*. University of Chicago Press, 1992.

[16] E. H. Margulis, *On Repeat: How Music Plays the Mind*. Oxford University Press, 2013.

[17] J. J. Rolison and J. Edworthy, "The role of formal structure in liking for popular music," *Music Perception: An Interdisciplinary Journal*, vol. 29, no. 3, pp. 269–284, 2012.

[18] O. Julian and C. Levaux, Eds., *Over and Over: Exploring Repetition in Popular Music*. Bloomsbury Academic, 2018.

[19] J. Summach, "The structure, function, and genesis of the prechorus," *Music Theory Online*, vol. 17, no. 3, October 2011.

[20] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proc. of the International conference on machine learning*. PMLR, 2018, pp. 4364–4373.

[21] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[22] C. Payne, "Musenet," *OpenAI, openai.com/blog/musenet*, 2019.

[23] J.-P. Briot, G. Hadjeres, and F. Pachet, "Deep learning techniques for music generation," *Springer*, vol. 10, 2019.

[24] S. Wei and G. Xia, "Learning long-term music representations via hierarchical contextual constraints," in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, 2021.

[25] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[26] G. Hadjeres and L. Crestel, "Vector quantized contrastive predictive coding for template-based music generation," *arXiv preprint arXiv:2004.10120*, 2020.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[28] G. Medeot, S. Cherla, K. Kosta, M. McVicar, S. Abdallah, M. Selvi, E. Newton-Rex, and K. Webster, "Structurenet: Inducing structure in generated melodies." in *Proc. of 19st Int. Conference on Music Information Retrieval Conf., ISMIR*, 2018, pp. 725–731.

[29] T. Collins and R. Laney, "Computer-generated stylistic compositions with long-term repetitive and phrasal structure," *Journal of Creative Music Systems*, vol. 1, no. 2, 2017.

[30] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *Proc. of the 18th Int. Society for Music Information Retrieval Conf.*, Suzhou, China, 2017.

[31] C.-H. Chuan and D. Herremans, "Modeling temporal tonal relations in polyphonic music through deep networks with a novel image-based representation," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[32] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4773–4784, 2020.

[33] A. Elowsson and A. Friberg, "Algorithmic composition of popular music," in *Proc. of the 12th Int. Conference on Music Perception and Cognition and the 8th Triennial Conf. of the European Society for the Cognitive Sciences of Music*, 2012, pp. 276–285.

[34] S. Dai, X. Ma, Y. Wang, and R. B. Dannenberg, "Personalized popular music generation using imitation and structure," *arXiv preprint arXiv:2105.04709*, 2021.

[35] B. L. Sturm and O. Ben-Tal, "Taking the models back to music practice: Evaluating generative transcription models built using deep learning," *Journal of Creative Music Systems*, vol. 2, pp. 32–60, 2017.

[36] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proc. of 21st Int. Conference on Music Information Retrieval Conf.*, 2020.

[37] D. Berger and C. Israels, *The Public Domain Song Anthology*. Charlottesville: Aperio, Mar 2020.

[38] R. Begleiter, R. El-Yaniv, and G. Yona, "On prediction using variable order markov models," *Journal of Artificial Intelligence Research*, vol. 22, pp. 385–421, 2004.

[39] J. Cleary and I. Witten, "Data compression using adaptive coding and partial string matching," *IEEE transactions on Communications*, vol. 32, no. 4, pp. 396–402, 1984.

[40] Z. Eitan and R. Y. Granot, "Growing oranges on mozart's apple tree: "inner form" and aesthetic judgment," *Music Perception: An Interdisciplinary Journal*, vol. 25, no. 5, pp. 397–417, 2008.

HETEROGENEOUS GRAPH NEURAL NETWORK FOR MUSIC EMOTION RECOGNITION

Angelo Cesar Mendes da Silva
Universidade de São Paulo, Brazil
angelo.mendes@usp.br

Diego Furtado Silva
Universidade Federal de São Carlos, Brazil
diegofs@ufscar.br

Ricardo Marcondes Marcacini
Universidade de São Paulo, Brazil
ricardo.marcacini@usp.br

ABSTRACT

Music emotion recognition has been a growing field of research motivated by the wealth of information that these labels express. Recognition of emotions highlights music's social and psychological functions, extending traditional applications such as style recognition or content similarity. Once musical data are intrinsically multi-modal, exploring this characteristic is usually beneficial. However, building a structure that incorporates different modalities in a unique space to represent the songs is challenging. Integrating information from related instances by learning heterogeneous graph-based representations has achieved state-of-the-art results in multiple tasks. This paper proposes structuring musical features over a heterogeneous network and learning a multi-modal representation using Graph Convolutional Networks with features extracted from audio and lyrics as inputs to handle the music emotion recognition tasks. We show that the proposed learning approach resulted in a representation with greater power to discriminate emotion labels. Moreover, our heterogeneous graph neural network classifier outperforms related works for music emotion recognition.

1. INTRODUCTION

Music is intrinsically connected to human emotions. At the same time that a songwriter expresses emotions in the composed songs, we can use music's emotional information to characterize the listener's moments and feelings. Music emotion recognition (MER) is a task that highlights social and psychological functions comprised by recordings, extending traditional applications in the area of music information retrieval [1]. According to the literature, mapping the spectrum of emotions can be done by analyzing the values in arousal and valence domains [2]. Besides, the emotions can be represented by labels that indicate negative, positive, and neutral values of arousal and valence and labels that point to emotional expressions, such as happiness, anger, or satisfaction [3]. Each label may be defined according to the significance of valence and arousal simultaneously, as illustrated in Figure 1. The MER task aims to use a musical representation to estimate these significances or predict discretized labels.

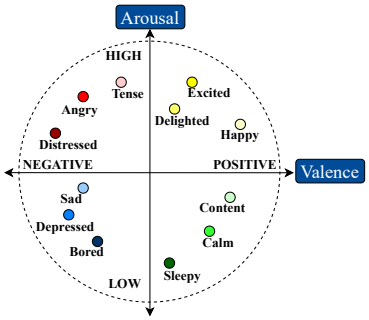


Figure 1: Emotions mapped in the arousal and valence domains.

Textual information is the most used data type in tasks involving emotion labels [4]. When emotion recognition is performed on music data, lyrics are commonly used as input [5, 6]. However, some studies emphasize the importance of acoustic features for the same task [7]. The fact that both modalities complement musical perception is well-known [8–10]. However, there is no consensual approach for incorporating multiple modalities into a unified representation.

Defining the structure to deal with different data types, such as text and audio, is a preliminary step to generate a unified multi-modal representation of musical data. Building this structure is an open problem in the literature. Techniques based on feature fusion are the most explored approach in the literature [11, 12]. Fusion of features can be performed by simply concatenating text and audio features or even learning embeddings via multimodal deep learning. However, these existing methods have limitations regarding incomplete modalities, deal with modalities of different dimensionality, and the interpretability of the generated representations [13].

Heterogeneous networks are a well-known representation for manipulating multiple modalities of unstructured data [14, 15]. This resource has been explored due to the ability to map data on graphs, representing connections between objects (nodes) through edges. Therefore, this process produces a graph that can be used as input in Graph

© A. C. M. da Silva, D. F. Silva, and R. M. Marcacini. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** A. C. M. da Silva, D. F. Silva, and R. M. Marcacini, "Heterogeneous Graph Neural Network for Music Emotion Recognition", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.*, Bengaluru, India, 2022.

Neural Network models [16]. Structuring data on heterogeneous networks has been widely studied in varied data such as text and images [17,18], but there are still few studies for musical data.

This work introduces a novel model for music emotion recognition that uses a heterogeneous network to structure audio and lyrics features and build a new multi-modal graph-based representation of musical data using the graph convolutional network (GCN). Our proposed model, described in detail in Sec. 3, uses content-based features (audio descriptors) together with features extracted from lyrics (embeddings from the language model). The music graph topology is based on relations defined by cluster information from audio e textual metadata. Some music nodes have labels with emotional information. Therefore, the proposed structure is composed of a heterogeneous network, and GCN can predict the emotion of unlabeled music nodes.

We apply a network regularization framework to propagate and refine information between neighboring nodes from different modalities. The matrix resulting from this regularization, along with the graph connections, is used as input for training our GCN. In this embedding space, music with similar emotions are close to each other, while dissimilar ones are further apart.

To evaluate our approach, we used the publicly available dataset PMEmo [19], which has 794 songs represented by audio descriptors and lyrics with annotations for arousal and valence domains. We measured the relevance of the learning process of a multi-modal graph-based representation for MER. Our proposal was compared with two different methods and other works from the literature that reproduced a similar experiment.

Our method was able to obtain a meaningful embedding representation that unifies different audio and text music features across the heterogeneous network. In addition to the heterogeneous network enabling interpretability of the relationships between text and audio features, our method outperforms other existing methods for music emotion recognition tasks.

2. RELATED WORK

The construction of methods to learn representations for unstructured data aims to reduce the work for features extraction and selection to describe the data and make learning algorithms less dependent on these processes [20]. Learning a representation for musical data involves manipulating features obtained by multiple modalities and projected in different spaces [21, 22].

Learning to represent music to recognize the comprised emotion involves exploring different features [1, 23]. We can observe most applications using the lyrics to represent the music [5, 6, 24] justified by the direct association we can make with the meaning of words and emotion labels. However, we have evidence that acoustic features also contain information that characterizes emotions [11,12,25]. In summary, there is no consensus on representing music using multiple modalities for the MER task, although we note that audio and lyrics features are relevant.

Recent work about music representation learning has explored strategies based on deep neural networks, such as early and late fusion [26, 27] and evaluating combinations between different input features [13]. This strategy advances and presents successful results that justify the representation learning process. However, there is a demand for approaches that explain the semantic complementarity existing between the features that justify the obtained results [28]. We explore the heterogeneous network resources to embed multiples music features and explicit the latent relationships among nodes, similar to [29, 30]. Moreover, we learn a graph-based representation that concentrates information from related music.

The complementarity between distinct features in musical perception justifies the motivation to build a multimodal representation [31]. For the emotion recognition problem, [7] describes what semantic information exists in various acoustic features and which emotions each feature can emphasize. We note that lyrics are also associated with emotions and must be used for the recognition task [32]. Therefore, we aim to build a multi-modal representation that integrates features obtained through the audio and lyrics and compares its performance against unimodal scenarios.

The construction of multi-modal embeddings requires that distinct features be mapped in a unified space. Graph-based methods have been applied in recent years when nodes and edges have different types [33]. We can exploit the relationships between neighboring nodes to build an embedding space that aggregates information from the node features even in different spaces. Significant advances are observed in multi-modal and unstructured data applications [34].

For musical data, [35] explores the task of similarity between artists. Authors use data previously structured to create the graph and apply it in GCN to build music embeddings with acoustic characteristics and tags. Finally, the embeddings are used to predict links to new artists. In contrast, [36] introduces a proposal of graph-based representation learning with the graph being constructed from a co-occurrence matrix obtained by the presence of pairs of songs in playlists.

Our proposal aims to create a heterogeneous network with multiple musical features and connect nodes that share cluster information. This network produces the features input for GCN to build a multi-modal representation and handle the music emotion recognition task. Our learned representation does not depend on previous information to structure the graph. Although we do not extend the approach to the multi-task scenario, it can be easily adapted for different tasks, maintaining the same heterogeneous network.

3. MODELLING

Our goal is to recognize emotions present in songs, such as a classification problem $Y = f(X)$, where Y represents the set of emotion labels, X represents the songs, and $f(.)$

denotes the function responsible for predicting emotions from a multi-modal musical representation given as input.

The proposed method for handling the MER is divided into three steps. The first step is to construct a heterogeneous network to structure songs in nodes and organize the acoustic and lyrics features available in the PMEmo dataset into different layers and build the relationships. Then, in sequence, we regulate the network to propagate information between nodes of different layers and explore the semantic complementarity between features of both modalities. Finally, we use the regularized network as input our multi-modal network embedding method using Graph Convolutional Network (GCN). The generated embeddings are also used by GCN to classify the emotion of the songs.

3.1 Heterogeneous network to structure music data

Musical data are intrinsically multi-modal, formed by features semantically complementary. Therefore, building a representation that incorporates multiple features demands manipulating modalities organized in different spaces. In this sense, heterogeneous networks offer resources for structuring data as nodes, with each modality projected on a layer and forming relationships between nodes for information sharing.

Mapping unstructured data on graphs has been gaining attention in the literature, motivated by the success of GNN-based learning models. For musical data, [37, 38] proposes to build a graph relating nodes from a distance function where nearby nodes are connected, while [39] assumes that all nodes are connected by adding weights in the edges, and as the works aforementioned [35, 36] that employing existing information to create a graph structure. We propose constructing the graph topology by using clusters for each modality as network nodes. Thus, we model the relationships between music, audio and texts from the cluster labels for each modality. Figure 2 illustrates this process, where songs are initially clustered according to their audio features. In this work, we use the k-means algorithm; however, the modeling is not dependent on a specific clustering algorithm. Then both clusters and songs are considered network nodes. The links indicate association between songs and clusters formed with audio features. This process is repeated for textual features, thereby allowing to obtain a heterogeneous network with three layers as shown in Figure 3.

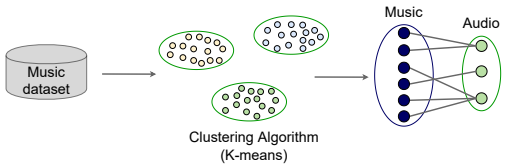


Figure 2: Illustration of mapping songs and respective audio features in a bipartite graph using clustering. This process is repeated for text features to generate the proposed heterogeneous network.

The heterogeneous network is formally defined as $N = (O, R, W)$, where O represents the set of objects, R represents the relationships and W represents the weights. Let $o_i \in O$ as the notation for an object, $r\_o\_i, o\_j = (o_i, o_j) \in R$ indicates whether there is a connection between the objects o_i and o_j, where the weight of r_{o_i,o_j} is given by w_{o_i,o_j} with $w \in W$. Currently, our propose use binaries relationships, $w_{o_i,o_j} \in \{0, 1\}$, for example, if a music node o_i is associated to a cluster with textual information, or audio information, o_j.

The benchmark dataset used has textual and acoustic features so that the set of objects $O = \{O_M \cup O_A \cup O_T\}$ is organizing each feature modality in the heterogeneous network proposed for musical data. O_M are objects representing each music in the dataset, O_A are objects representing acoustic features formed by audio descriptors, and O_T are objects representing textual features extracted from the lyrics.

3.2 Network regularization

In the proposed heterogeneous network, the objects $o \in O$ have one or more initial features vectors. For example, an object $o_m \in O_M$ that represents a music might contain an acoustic feature vector $x_{o_m}^{(A)} \in \mathbb{R}^{d_A}$, as well as a textual feature vector $x_{o_m}^{(T)} \in \mathbb{R}^{d_T}$. The vectors are in their respective spaces $\mathbb{R}^{d_A}$ (e.g. melspectrogram or chromagram) and $\mathbb{R}^{d_T}$ (e.g. word2vec or BERT). However, objects in sets O_A and O_T exclusively have the initial features of their modalities. Still, we highlight that objects in O_M may contain absent modalities, such as missing lyrics for some musics. Thus, inspired by the concept of embedding propagation from networks [40], we integrate a regularization framework capable of propagating information between objects of different modalities according to the network topology.

The regularization process exploits the network topology to propagate information between objects to complement missing information and adjust existing features according to the object label. The process is represented in Figure 3. The music is the input data, where each feature modality composes a layer in the heterogeneous network. Objects in the central layer are in O_M and are connected to objects of other types, having their feature vectors. The objects in O_A contain audio descriptors, and objects in O_T contain lyrics features. When finish regularization process, all objects will have feature vectors from objects of different modalities.

The proposed method to perform the network regularization is an instance of label propagation methods [41]. This step aims to propagate the object's features, assuming two constraints: neighboring objects must have similar features; the final feature vector must be similar to the initial features for objects. Formally, network regularization can be associated with a representation learning problem. We define as learning a mapping function $f : o_i \rightarrow \mathbf{z}_{o_i} \in \mathbb{R}^d$, where $\mathbf{z}_{o_i}$ is the learned vector of object $o_i \in O$ in network $N(O, R, W)$. The equation 1 defines the function to be minimized to learn the new space $\mathbf{Z} \in \mathbb{R}^d$, in which all

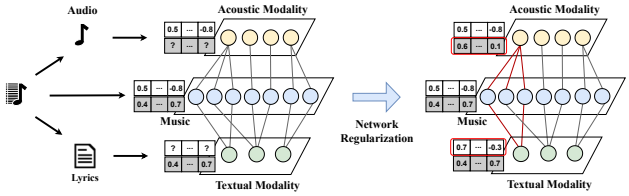


Figure 3: The heterogeneous network regularization aims to fill in missing modalities in objects. Objects on the central layer have relationships with objects on other layers. After regularization, all nodes receive information from neighbors nodes (highlights in red) with different modalities from the network topology.

heterogeneous network objects are mapped.

$$Q(\mathbf{Z}) = \frac{1}{2} \sum_{o_m \in O_M} \sum_{o_a \in O_A} w_{o_m,o_a} (\mathbf{z}_{o_m} - \mathbf{z}_{o_a})^2 + \frac{1}{2} \sum_{o_m \in O_M} \sum_{o_t \in O_T} w_{o_m,o_t} (\mathbf{z}_{o_m} - \mathbf{z}_{o_t})^2 + \mu \sum_{o_i \in O_X} (\mathbf{z}_{o_i} - \mathbf{x}_{o_i})^2 \quad (1)$$

The two initials terms of the regularization function are responsible for computing the proximity between the feature vectors for each pair of related objects in the network, indicating the weight of the relationship between the objects. The last term is responsible for computing the distance between the features of objects that had initial features, O_X, and the features learned in the space $\mathbf{Z}$. The parameter μ determines the level of preservation in the initial features. The high value indicates the more preservation of the features, while low values allow adjusting the features according to the network topology. The Equation (1) is applied for each modality, textual and acoustic. Therefore, at the end of the regularization process, we obtained Z_{audio} and Z_{text} for all nodes in the network. Finally, we concatenate both spaces, $Z_{\oplus} = Z_{audio} \oplus Z_{text}$.

3.3 Graph Convolutional Networks

The general idea of graph learning methods is to iteratively update object representations, combining them with representations of their neighbors. In this context, initially, the objects are represented by $Z_{\oplus}$ resulting from the regularization. The neighboring nodes are obtained from the heterogeneous network N, indicated by an adjacency matrix A. In particular, we used a GCN to learn the final representation and handle the MER task. The representation obtained by the GCN is represented by the matrix $F \in R^{T \times D}$, where T is the number of nodes and D is the dimension of the new unified space learned. In general, a GCN can be described according to Equation 2,

$$H^{(l+1)} = f(H^{(l)}, A) = \alpha(AH^{(l)}W^{(l)}) \quad (2)$$

where $H^{(0)} = X$, l, represents the current layer, A represents the adjacency matrix, $W^{(l)}$ represents the weight in l-th layer in a neural network, and $\alpha(.,.)$ defines the activation function. The layer $H^{(l+1)} = H^{(L)} = U$ contains the new learned space by GCN.

We pre-process the adjacency matrix A in the GCN. First, we need to add self-loops on each node to consider its features in the learning process. Then, we add A to identity matrix I, which result in $\hat{A}$. In addition, we normalize A to maintain the representation vectors scale in each layer. We did the matrix normalization by multiplying A with D^{-1}, where D indicates a diagonal matrix formed by the degrees of each node in the network, or A by $D^{-\frac{1}{2}}$, for symmetric normalization. Thus, the adjacency matrix used in the GCN is defined by Equation 3,

$$\hat{A} = D^{-\frac{1}{2}} S D^{-\frac{1}{2}} \quad (3)$$

where $S = A + I$, I represents the identity matrix, and $D_{ii} = \sum_j S_{ij}$ indicates the node degree.

The proposed GCN architecture comprises three graph convolutional layers combined with a hyperbolic tangent activation function. The dimensionality of vector input for each layer is indicated by the previous layer's output. The first layer receives the learned matrix $Z_{\oplus}$ to produce vectors with 512, 256, and 128 dimensions in next layers. Furthermore, we computed a softmax cross-entropy as the last layer to indicate the probability that each music is associated with emotion labels. Finally, we train the networks using the ADAM optimizer with a learning rate of 0.0001 and 3000 epochs.

4. EXPERIMENTS

Our experiments aim to evaluate the construction of a heterogeneous network to structure musical data and learn and evaluate a multi-modal graph-based representation in the music emotion recognition task. The k parameter defines the number of clusters used to connect objects in the heterogeneous network. This parameter is the main target of the analysis during network construction. The number of clusters allows us to create new connections between objects that initially have different labels, exploit regularization to propagate information and refine their representations.

We evaluated the representation learned with the GCN by comparing using two classifiers that utilize as input the representations formed by features provided in the dataset, including combinations between them. Finally, our results are relationships with similar works of literature. In all evaluation scenarios, our proposal presented the best results.

4.1 Dataset and features

The PMEmo is a popular music dataset with emotional annotations as a benchmark in music emotion retrieval and recognition [19]. PMEmo dataset containing emotion annotations of 794 songs in arousal and valence domains with audio and lyrics features. For acoustic features, PMEmo has a pre-computed vector composed of audio descriptors, which represent the music chorus information. In addition,

there are the lyrics and a source code for textual features to pre-process it and build the feature vector based on Bag-of-Words (BoW).

In addition to the representations of each modality, the authors of the PMEmo dataset [19] also presented competitive approaches that explore emotion classification based on fusion features, which were explored for comparison with our proposed method.

Besides these two representations, we build another textual feature on the lyrics using a pre-trained language model based on BERT models [1] fine-tuned to sentence similarity tasks, to have a more robust representation than the BoW. The model uses a triplet network structure to produce lyrics embeddings. Thus, we explored five strategies to represent the music that varies between isolated features and concatenation between them, as indicated in the PMEmo baseline scenario: `Audio`, `Bert`, `BoW`, `Audio+BoW`, `Audio+Bert`. Therefore, we have resources to build uni-modal and multi-modal scenarios to evaluate the emotion recognition task.

We have discretized the annotations in arousal and valence domains to transform the values in the labels, according to the works [42, 43]. First, we remove instances with missing information, and normalize the values of both domains with a range between 1 and 9. So, we assign the labels according to the values. An instance is negative if the normalized value is < 4.5; neutral if value $>= 4.5$ and < 5.5; or positive if value $>= 5.5$. The table 1 summarize the number of instances distributed for each label, as well as the dimensionality for all feature vectors.

Feature size	
Audio	6373
Bert	512
Bert	7670

Instances in each label		
	Arousal	Valence
Negative	151	145
Neutral	100	96
Positive	354	364
Total	605	

Table 1: Summary of dataset properties

4.2 Experimental setup

The representation learning with a GCN is done through a transductive process, where we need to know the complete dataset. As our goal is to recognize the emotions present in the music, we divided the dataset into stratified ten folds and masked the labels in test folds. As comparison approaches, we reproduce another transductive classification method based on label propagation (LPA). The instance labels in the test set are changed during the learning process and must converge to the original labels. We also reproduce an inductive scenario using a multilayer perceptron classifier (MLP). In this scenario, the test set is unknown, and the labels are predicted according to patterns learned during training [2]. In both scenarios, the features were normalized for each split according to train and test folds.

We evaluated a variation in the number of clusters that structure the objects represented by acoustic and textual features such that $k \in \{3, 7, 13, 20, 30\}$. The smallest value represents the original number of labels, while the largest value is defined by $\sqrt{N}$, where N indicates the samples in the dataset. We assume the hypothesis that by increasing the number of clusters, we can relate objects in the network with similar content but do not share the same label or objects that share a label and can be associated with objects with more similar content.

4.3 Results and discussions

We reported the mean and standard deviation for the F1-score and Accuracy metrics for each musical data representation strategy for the three evaluated methods. We refer to our representation as MRLGCN, an acronym for Music Representation Learning using GCN. Emotion labels are commonly explored in conjunction with text-based representations, so acoustic features are proposed to complement the representation. We can notice in Figure 4 that representations that concatenate both features do not lead to performance improvement. Observing the performance obtained by our approach, we can highlight the relevance of a learning process to incorporate the information of the different features.

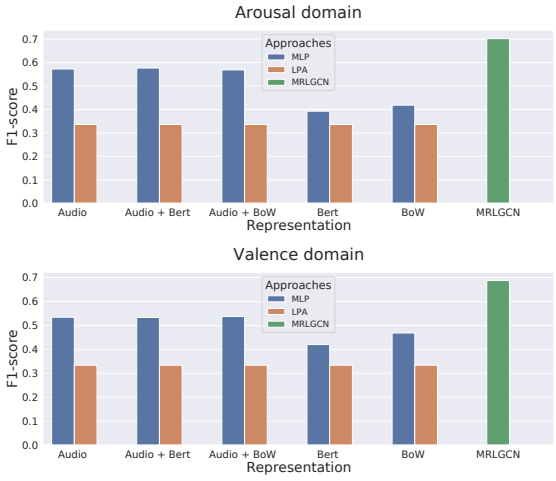


Figure 4: In the uni-modal scenario, textual features are more associated with emotion labels than acoustic features. In the multi-modal scenario, concatenating acoustic and lyrics features did not result in a more efficient representation, showing that exploring the semantic complementarity demands more robust strategies. Our representation presented the best result in both domains, evidencing the presence of relevant information in all the features used.

Figure 5 shows the results obtained by our method when the number k of clusters varies. We can notice that the performance of MRLGCN increases when k increases. It validates the hypothesis that we have objects with similar content that must be related, even those that do not share labels. Although it shows an upward trend, it is expected

[1] https://huggingface.co/sentence-transformers/distiluse-base-multilingual-cased-v1

[2] The LPA and MLP algorithms used are implementations available in the sklearn library using default parameters.

that the performance stabilizes and decreases after a certain k. This behavior is explained by the appearance of disconnected graphs that influence the regularization of the network, affecting the propagation of information between objects.

Clusters	F1-score		Accuracy	
	μ	σ	μ	σ
3	0.4530	0.096	0.6421	0.050
7	0.5231	0.087	0.6701	0.051
13	0.6010	0.107	0.7224	0.059
20	0.6612	0.120	0.7589	0.073
30	0.7018	0.131	0.7833	0.084

(a) Arousal domain

Clusters	F1-score		Accuracy	
	μ	σ	μ	σ
3	0.4820	0.089	0.6659	0.055
7	0.5300	0.099	0.6945	0.062
13	0.6095	0.127	0.7386	0.079
20	0.6399	0.137	0.7564	0.078
30	0.6873	0.151	0.7842	0.095

(b) Valence domain

Table 2: Results obtained with average (μ) and standard deviation (σ) for F1-score and accuracy metrics for each k value.

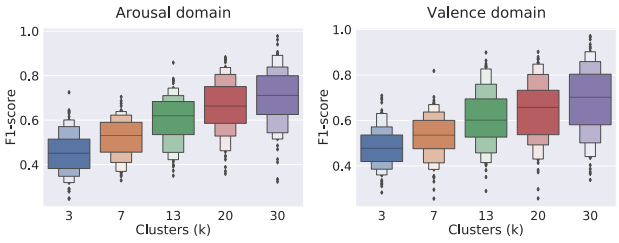


Figure 5: Performance in arousal and valence domains according to increase k value. The variation in the number of clusters highlights that the contents of the objects have similar relationships that may not be presented only by the label information.

Due to the unbalanced data distribution, we report a confusion matrix for each k to validate that the values presented for both metrics result from a learning process without bias towards the majority label. We can see that the increase in the metrics shown in Table 2 is accompanied by a proportional increase in the percentage of correct predictions for three labels, as seen in Figure 6.

5. CONCLUDING REMARKS

This work presents a new proposal to structure musical data using heterogeneous networks and build a graph-based multi-modal representation using Graph Convolutional Network to handle music emotion recognition tasks. We map acoustic and textual features as objects in different layers on a heterogeneous network, using cluster information to build relationships among objects, and insert a network regularization framework to propagate information between objects of other modalities. The embeddings

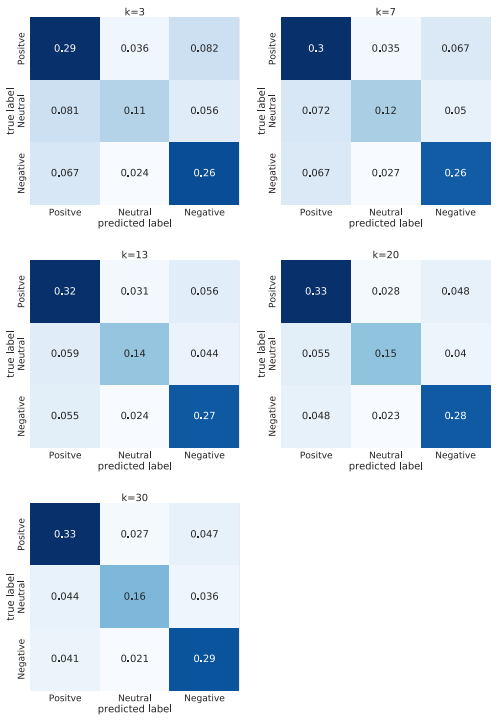


Figure 6: We highlighted the behavior of balanced growth of correct predictions across all labels, without bias for the majority label. This matrix evidences the learning process during the increase in the number of clusters.

resulting are used as input for GCN to learn a new music representation, with application in music emotion recognition task. Our proposal presented superior results in all evaluation scenarios and an significant improvement ratio concerning works in the literature. The results reinforce the applicability of graph-based representations in unstructured and multi-modal data.

Our proposal can be seen as a strategy that exploits a welcome trade-off for musical data representation. First, heterogeneous networks show explicit relationships between audio and text features and facilitate interpretability. Second, a GCN-based deep neural network learns embeddings that map the heterogeneous network into a promising representation for music emotion recognition.

In future work, we want to add other features as layers in the heterogeneous network and propose the inclusion of an importance matrix during representation learning. We will define criteria for determining optimal parameters for building the heterogeneous network and GCN architecture. In addition, we want to identify semantic relevance for each feature, measure the discriminative capacity's impact on learned representation, and evaluate the effect of the regularization step, like an ablation study. Finally, we can integrate the regularization process and GCN in an end-to-end framework for emotion music classification. The source code, datasets, and heterogeneous networks used in this paper are publicly available at `https://github.com/AngeloMendes/Heterogeneous-Graph-Neural-Network-for-Music-Emotion-Recognition`.

6. ACKNOWLEDGEMENTS

The authors of this work would like to thank the Center for Artificial Intelligence (C4AI-USP) and the support from the São Paulo Research Foundation (FAPESP) grant #2019/07665-4) and from the IBM Corporation, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brazil (CAPES) – grant #PROEX-12049601/D, Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPQ) grant #426663/2018-7.

7. REFERENCES

[1] G. Rotter and I. Vatolkin, "Emotions," in *MUSIC DATA ANALYSIS*, C. Weihs, D. Jannach, I. Vatolkin, and G. Rudolph, Eds. CRC Press, 2017, ch. 21, pp. 511–535.

[2] P. B. Posner J, Russell JA, "The circumplex model of affect: an integrative approach to affective neuroscience, cognitive development, and psychopathology." *Dev Psychopathol*, vol. 17(3), pp. 715–734, 2005.

[3] M. Ogihara and Y. Kim, "Mood and emotional classification," in *Music data mining*, T. Li, M. Ogihara, and G. Tzanetakis, Eds. CRC Press, 2012, ch. 5, pp. 135–160.

[4] S. Zad, M. Heidari, H. James Jr, and O. Uzuner, "Emotion detection of textual data: An interdisciplinary survey," in *2021 IEEE World AI IoT Congress (AIIoT)*. IEEE, 2021, pp. 0255–0261.

[5] J. Choi, J. Song, and Y. Kim, "An analysis of music lyrics by measuring the distance of emotion and sentiment," in *2018 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, 2018, pp. 176–181.

[6] J. Abdillah, I. Asror, and Y. F. A. Wibowo, "Emotion classification of song lyrics using bidirectional lstm method with glove word representation weighting," *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, vol. 4, no. 4, pp. 723 – 729, Aug. 2020.

[7] R. Panda, R. M. Malheiro, and R. P. Paiva, "Audio features for music emotion recognition: a survey," *IEEE Transactions on Affective Computing*, pp. 1–1, 2020.

[8] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, "Music emotion recognition: A state of the art review," in *International Society for Music Information Retrieval Conference*, vol. 86, 2010, pp. 937–952.

[9] H. Xue, L. Xue, and F. Su, "Multimodal music mood classification by fusion of audio and lyrics," in *MultiMedia Modeling*, X. He, S. Luo, D. Tao, C. Xu, J. Yang, and M. A. Hasan, Eds. Cham: Springer International Publishing, 2015, pp. 26–37.

[10] R. Rajan, J. Antony, R. A. Joseph, J. M. Thomas *et al.*, "Audio-mood classification using acoustic-textual feature fusion," in *2021 Fourth International Conference on Microelectronics, Signals & Systems (ICMSS)*. IEEE, 2021, pp. 1–6.

[11] Y. Pandeya and J. Lee, "Deep learning-based late fusion of multimodal information for emotion classification of music video," *Multimed Tools Appl*, 2020.

[12] C. Chen and Q. Li, "A multimodal music emotion classification method based on multifeature combined network classifier," *Mathematical Problems in Engineering*, 2020.

[13] H. Wu, C. Kao, Q. Tang, M. Sun, B. McFee, J. Bello, and C. Wang, "Multi-task self-supervised pre-training for music classification," *Proceedings - ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 2021-June, pp. 556–560, 2021.

[14] C. Shi and P. Yu, *Heterogeneous Information Network Analysis and Applications*, 01 2017.

[15] C. Yang, Y. Xiao, Y. Zhang, Y. Sun, and J. Han, "Heterogeneous network representation learning: A unified framework with survey and benchmark," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–1, 12 2020.

[16] C. Shi, *Heterogeneous Graph Neural Networks.* Singapore: Springer Singapore, 2022, ch. 16, pp. 351–370.

[17] H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge & Data Engineering*, vol. 30, no. 09, pp. 1616–1637, sep 2018.

[18] F. Chen, Y.-C. Wang, B. Wang, and C.-C. J. Kuo, "Graph representation learning: a survey," *APSIPA Transactions on Signal and Information Processing*, vol. 9, p. e15, 2020.

[19] K. Zhang, H. Zhang, S. Li, C. Yang, and L. Sun, "The pmemo dataset for music emotion recognition," in *International Conference on Multimedia Retrieval*. New York, NY, USA: Association for Computing Machinery, 2018, p. 135–142. [Online]. Available: https://doi.org/10.1145/3206025.3206037

[20] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, Aug 2013.

[21] W. W. Y. Ng, W. Zeng, and T. Wang, "Multi-level local feature coding fusion for music genre recognition," *IEEE Access*, vol. 8, pp. 152 713–152 727, 2020.

[22] H. Phan, H. L. Nguyen, O. Y. Chén, L. Pham, P. Koch, I. McLoughlin, and A. Mertins, "Multi-view audio and music classification," 2021.

[23] J. S. Gómez-Cañón, E. Cano, T. Eerola, P. Herrera, X. Hu, Y.-H. Yang, and E. Gómez, "Music emotion recognition: Toward new, robust standards in personalized and context-sensitive applications," *IEEE Signal Processing Magazine*, vol. 38, no. 6, pp. 106–114, 2021.

[24] Y. Agrawal, R. Shanker, and V. Alluri, "Transformer-based approach towards music emotion recognition from lyrics." in *European Conference on Information Retrieval*, 2021, pp. 167–175.

[25] F. H. Rachman, R. Sarno, and C. Fatichah, "Music emotion detection using weighted of audio and lyric features," in *Information Technology International Seminar*, 2020, pp. 229–233.

[26] H. C. Wang, S. W. Syu, and P. Wongchaisuwat, "A method of music autotagging based on audio and lyrics." *Multimedia Tools and Applications*, vol. 80, p. 15511–15539, 2021.

[27] S. Grollmisch, E. Cano, C. Kehling, and M. Taenzer, "Analyzing the potential of pre-trained embeddings for audio classification tasks," in *European Signal Processing Conference (EUSIPCO)*, 2021, pp. 790–794.

[28] J. Kim, J. Urbano, C. C. S. Liem, and A. Hanjalic, "One deep music representation to rule them all? a comparative analysis of different representation learning strategies," *Neural Computing and Applications*, Mar 2019. [Online]. Available: https://doi.org/10.1007/s00521-019-04076-1

[29] A. C. M. da Silva, P. R. V. do Carmo, R. M. Marcacini, and D. F. Silva, "Instance selection for music genre classification using heterogeneous networks," *Simpósio Brasileiro de Computação Musical*, vol. 18, pp. 11–18, 2021.

[30] A. C. M. da Silva, D. F. Silva, and R. M. Marcacini, "Multimodal representation learning over heterogeneous networks for tag-based music retrieval," *Expert Systems with Applications*, vol. 207, pp. 1–9, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422012003

[31] P. Knees and M. Schedl, *Music similarity and retrieval: an introduction to audio-and web-based strategies*. Springer, 2016, vol. 36.

[32] D. Edmonds and J. Sedoc, "Multi-emotion classification for song lyrics," in *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2021, pp. 221–235.

[33] F. Xia, K. Sun, S. Yu, A. Aziz, L. Wan, S. Pan, and H. Liu, "Graph learning: A survey," *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 02, pp. 109–127, apr 2021.

[34] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *IEEE Data Eng. Bull.*, vol. 40, no. 3, pp. 52–74, 2017. [Online]. Available: http://sites.computer.org/debull/A17sept/p52.pdf

[35] F. Korzeniowski, S. Oramas, and F. Gouyon, "Artist similarity with graph neural networks." in *International Conference on Music Information Retrieval*, 2021, pp. 350–357.

[36] A. Saravanou, F. Tomasi, R. Mehrotra, and M. Lalmas, "Multi-task learning of graph-based inductive representations of music content," in *International Conference on Music Information Retrieval*, 2021, pp. 602–609.

[37] F. Simonetta, F. Carnovalini, N. Orio, and A. Roda, "Symbolic music similarity through a graph-based representation," in *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, 09 2018, pp. 1–7.

[38] D. d. F. P. Melo, I. d. S. Fadigas, and H. B. d. B. Pereira, "Graph-based feature extraction: A new proposal to study the classification of music signals outside the time-frequency domain," *PLOS ONE*, vol. 15, no. 11, pp. 1–26, 11 2020. [Online]. Available: https://doi.org/10.1371/journal.pone.0240915

[39] S. Dokania and V. Singh, "Graph representation learning for audio & music genre classification," 2019.

[40] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, pp. 1–20, 03 2020.

[41] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in neural information processing systems*, 2004, pp. 321–328.

[42] Z. Yin, Y. Wang, L. Liu, W. Zhang, and J. Zhang, "Cross-subject eeg feature selection for emotion recognition using transfer recursive feature elimination," *Front Neurorobot*, pp. 11–19, 2017.

[43] D. Garg and G. K. Verma, "Emotion recognition in valence-arousal space from multi-channel eeg data and wavelet based deep learning framework," *Procedia Computer Science*, vol. 171, pp. 857–867, 2020, third International Conference on Computing and Network Communications (CoCoNet'19). [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050920310644

Papers - Session VI

AND WHAT IF TWO MUSICAL VERSIONS DON'T SHARE MELODY, HARMONY, RHYTHM, OR LYRICS ?

Mathilde Abrassart
Ircam Amplify
abrassart@ircam.fr

Guillaume Doras
Ircam, Sorbonne Université, CNRS, STMS Lab
doras@ircam.fr

ABSTRACT

Version identification (VI) has seen substantial progress over the past few years. On the one hand, the introduction of the metric learning paradigm has favored the emergence of scalable yet accurate VI systems. On the other hand, using features focusing on specific aspects of musical pieces, such as melody, harmony, or lyrics, yielded interpretable and promising performances. In this work, we build upon these recent advances and propose a metric learning-based system systematically leveraging four dimensions commonly admitted to convey musical similarity between versions: melodic line, harmonic structure, rhythmic patterns, and lyrics. We describe our deliberately simple model architecture, and we show in particular that an approximated representation of the lyrics is an efficient proxy to discriminate between versions and non-versions. We then describe how these features complement each other and yield new state-of-the-art performances on two publicly available datasets. We finally suggest that a VI system using a combination of melodic, harmonic, rhythmic and lyrics features could theoretically reach the optimal performances obtainable on these datasets.

1. INTRODUCTION

The version identification (VI) problem has received much attention over the last two decades. Pioneering works showed promising accuracy on small audio datasets, but remained difficult to scale to larger modern audio corpora. The recent introduction of data-driven approaches based on neural networks led to significant progress towards accurate yet scalable VI systems [1].

Different paradigms are currently active: one approach considers VI as a classification task, and intends to classify versions into the same class [2], while another approach formulates VI as a metric learning problem, and intends to minimize (resp. maximize) a distance between versions (resp. non-versions) [3, 4]. Recent works have also proposed a combination of both [5]. Metric learning or classification approaches seem to yield similar performances, as it has been observed for other MIR applications [6]. These systems also differ according to a perhaps more important aspect: their input feature. Some use a generic audio representation, such as the Constant-Q transform [7], and rely on the expressivity of the network to disentangle relevant musical features. Others use specialized features, such as the melodic line, the harmonic structure and/or the lyrics, and rely on input data to discriminate between versions and non-versions [3, 4, 8, 9].

In this work, we pursue in the direction of a metric-based approach using specialized features. We build upon the system described in [8], conserving its principle and architecture, and explore the use of new input features. The reason is motivated by three practical considerations: firstly, the metric learning approach yields a very compact representation of audio (its embedding), which can be conveniently stored, indexed and queried in very large databases. Secondly, the embedding space and the musical similarity measure that is obtained for each different specialized features is meaningful from a musical perspective, and can be reused for other purposes, e.g. playlist generation. Thirdly, the use of specialized features requires smaller models that are faster to train and less energy consuming than larger architectures [10].

It can reasonably be assumed that different versions of the same musical work share at least one of these four features: the melodic line, the harmonic structure, the rhythmic patterns and sometimes the lyrics. The role of melody and harmony in version similarity has been thoroughly investigated [8]. The role of the lyrics has also been studied recently, albeit not from a metric learning perspective [9].

In this work, we present a systematic study of the contribution of these four features to version similarity, and describe a metric learning-based system combining all of them. We show that this combination provides new state-of-the-art performances on two publicly available datasets. We also show that an oracle using these feature embeddings nearly achieves the maximum theoretical performances on these datasets, suggesting that design of future VI systems reaching these performances may be possible.

The rest of this paper is organized as follows: we briefly review the previous studies inspiring our current work (Section 2). We then describe how we extracted rhythmic and lyrics features, and our metric learning-based architecture (Section 3). We present our experiments, discuss our results (Section 4), and illustrate them with some examples (Section 5). We conclude this paper with our future work.

© M. Abrassart, G. Doras. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. Abrassart, G. Doras, "And what if two musical versions don't share melody, harmony, rhythm, or lyrics ?", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

2. RELATED WORK

In this section, we present a brief overview of the main concepts that inspired our present work.

2.1 Metric learning

Learning a similarity metric that generalizes to unseen examples is a common objective in machine learning. The goal is to learn how to map the data of interest into a compact representation (its embedding), and to minimize (resp. maximize) the distance between the embeddings of similar (resp. dissimilar) examples. Various MIR applications rely on a concept of musical similarity, e.g. music classification [11], music recommendation [12], or VI systems [1], among many others. Musical similarity between two tracks is typically evaluated first deriving an intermediate feature representation from the audio waveform and then computing a distance between feature pairs.

In the past few years, metric learning has proven its efficiency to build scalable yet accurate VI systems. These modern architectures typically rely on a CNN-based model trained with a triplet loss [13] to embed the musical information contained in the input feature into a single vector embedding that can be rapidly compared via Euclidean distance computation. However, these different systems have made different choices regarding their input features: for instance, Doras et al. [3] used a melodic line representation, Yesiler et al. [4] used a harmonic structure representation, and Du et al. [5] used a more generic CQT. The choice between specialized or generic features seems to have a non-negligible impact on the required size of the models (the former uses a 5-layers CNN, while the latter uses a ResNet50).

In this work, we choose the first alternative, and we propose to explore other specialized features beside melody and harmony, in particular the rhythmic and lyrics features.

2.2 Rhythm patterns detection

Musical similarity based on rhythmic patterns has long been investigated in MIR research, e.g. for audio retrieval [14] or music classification [15]. With the purpose of analysis of musical style and recognition of musical genres, Pampalk et al. introduced the fluctuation patterns (FP), representing rhythmic patterns in different frequency bands, and their evolution over time [16]. The basic assumption is that similar songs exhibit similar characteristic rhythmic patterns, and that comparing FP between tracks shall give enough information about their similarity or dissimilarity from a genre or mood perspective. We propose here to extend this idea to VI context, and to use the fluctuation patterns to discriminate versions from non-versions.

In practice, the FP is obtained computing the audio Mel spectrogram, summing up the high Mel bands to highlight the low frequencies and performing a second STFT in each mel band along the time axis. This results in a 3-dimensional matrix with axes corresponding to the the Mel bands, the frequency modulation and the time. The frequency modulation axis therefore represents the periodicity of the loudness in the corresponding Mel band: for example, a drum kick playing at 120 bpm will be represented here with a frequency modulation at 2 Hz in the low frequency bands. Finally, a perceptual filter is applied on each frequency modulation band. This filter is supposed to highlight the frequency modulations most perceived by the human ear ; for example, a frequency modulation at 4 Hz gives a more intense feeling of fluctuation strength. The 3-dimensional matrix is then averaged along the frequency axis, resulting in a representation of the variations of the frequency modulation over time.

One of the FP limitation is the use of a linear scale to represent periodicities. This was addressed by Pohle et al., who used a log scale to represent frequency modulations [17]. The advantage is that the same onset structure played at different tempi will have all its activations shifted by the same amount along the frequency modulation axis. Another way to achieve this tempo invariance is to compute periodicities with a Constant-Q transform (CQT) instead of a STFT [18].

2.3 Lyrics recognition

In automatic speech recognition (ASR), the traditional approach relies on a language model and an acoustic model, typically implemented as a Hidden Markov Model, possibly coupled with a neural network [19]. An alternative approach consists in implementing both language and acoustic models as a single neural network, trained in an end-to-end fashion with a Connectionist Temporal Classification (CTC) loss [20]. CTC-based models outputs the probability distribution of symbols at each time frame, which can be decoded into the most likely sequence of symbols via classical beam search. This approach has become very popular since fully convolutional end-to-end architectures have achieved performances comparable to those of the hybrid architectures [21].

Although singing voice has many obvious differences with speech, automatic lyrics recognition (ALR) or alignment (ALA) systems are usually directly inspired by ASR applications. Moreover, the recent introduction of large lyrics annotated audio datasets, such as Dali [22], has fostered the development of new ALR systems, whether they are based on the traditional [23] or end-to-end [24] architectures. While the former seems to yield better results [25], the latter has the advantage of its simplicity, both at training and inference time.

However, very few attempts have been made to use lyrics to assess audio track similarity. It was proposed to improve a query-by-humming system [26], but to the best of our knowlege, only Vaglio et al. proposed the use of lyrics for version identification [9]. They used an existing ALR system to extract lyrics from the audio, and estimate track similarity via a string matching algorithm. However, the comparison cost of such algorithm quickly becomes prohibitive when querying large modern corpora, and limits the scalability of this approach.

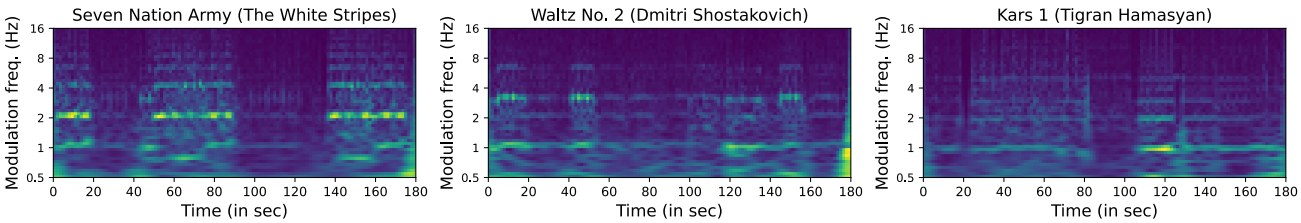


Figure 1: Examples of the Constant Q Fluctuation Patterns (CQFP).

3. PROPOSED METHOD

In this section, we describe and motivate our design choices. We first present how we extracted our rhythmic and lyrics features, and publicly release our datasets [1] . We then present our metric learning-based VI model.

3.1 Rhythmic features

Our assumption is that the FP representation described in Section 2.2 displays both the local rhythmic patterns along the frequency modulation axis, as well the global rhythmic evolution of the piece over time. We therefore propose to use this representation as our rhythmic feature for version identification. However, we introduce a CQT to compute the periodicities, and to achieve tempo invariance along the frequency modulation dimension. We choose as minimum frequency for the CQT 0.5 Hz (i.e. 30 bpm which we assume would be the tempo of the slowest tracks), and to cover up to 5 octaves i.e. 16Hz (960 bpm) with 10 bins per octave to get all the rhythmic subtleties. We kept the same other parameters as in the original implementation [16].

As an illustration, Figure 1 represents the CQ-FP obtained for different tracks with characteristic time signature. The first example has a 4/4 time signature, and clearly displays a rhythmic pattern present around 2 Hz (~120 bpm), as well as another periodicity around 4 Hz, which corresponds to a clear binary rhythm. The second example has a 3/4 time signature, and the rhythm of a waltz appears clearly: one beat at 1 Hz (60 bpm) and another at about 3 Hz. Finally, the third example has a 5/4 time signature (irregular), and we find this characteristic by observing bands at 1, 2 and 3 Hz. It can also be observed on each example that our rhythmic feature also represents the global structure of the piece over time, which is probably another relevant aspect in a VI context. Finally, as the modulation frequency dimension has a constant Q-factor, a change in tempo would not change the spacing between activations.

3.2 Lyrics features

We argue that accurate lyrics recognition is not required for version identification, and that identifying only a few common words, or even a few common character sequences, between tracks shall be sufficient to determine whether they are versions or not. We thus implemented a deliberately simple fully convolutional ALR system inspired by recent ASR system [21, 27].

[1] https://ircam-anasynth.github.io/papers/2022/abrassart

Model It is an 8-layers 2D-CNN with 3x3 kernels. Max-pooling 2x2 is applied on the first 2 layers to decrease time and frequency dimensions, and max-pooling 1x2 is applied on the next 6 layers only for frequency dimension. The first layer has 64 filters, doubled at each layer up to 512. A dropout with rate 0.3 is applied.

Inputs/outputs We use no pre-processing on the audio (no data augmentation, no voice separation). As classically done in ASR, we use 40 band Mel-spectrogram with 10ms timeframes as input feature. The model outputs a posteriorgram corresponding to the log-probabilities of the 'a...z' letters, the space symbol and the CTC blank symbol, i.e. 28 bins in total. An output example is shown Figure 2.

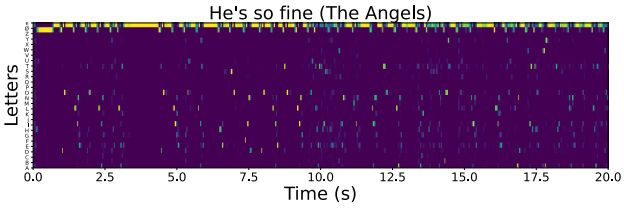


Figure 2: A posteriorgram obtained for 20 sec. of audio.

Training and inference The model is trained on the Dali dataset, which provide 12k+ polyphonic audio with lyrics annotations at the word level. We used audio chunks of 10 seconds, and used a CTC loss with Adam optimizer to train the model to align between audio and text. We evaluated its performances on a distinct Dali test subset using a greedy beam search decoding with no language model, and achieved a modest Character Error Rate (CER) of 0.496.

3.3 Convolutional architecture

The same simple architecture (with different configurations) has shown its ability to capture relevant melodic or harmonic similarity between versions [3,4]. It consists in a plain 5-layers Convolutional Neural Network (CNN), encoding each input feature using time and frequency max-pooling, while increasing the number of filters at each layer. The CNN output feeds a gated temporal attention mechanism [28], which was found to help the model to focus on relevant portions of the input features. We refer the reader to our previous work [8] for implementation details.

In this work, we keep the exact same generic architecture, and propose two new configurations to process also the rhythmic and lyrics features. We summarize the configuration used for each of the four features in Table 1.

Features	Melodic [3]						Harmonic [4]						Rhythmic						Lyrics					
Layers	n	k	ks	p	ps	d	n	k	ks	p	ps	d	n	k	ks	p	ps	d	n	k	ks	p	ps	d
1	64	3x3	1x1	2x2	2x2	0.0	256	180x12	1x1	1x12	1x1	0.0	64	3x20	1x1	1x2	1x2	0.4	64	10	1	5	2	0.3
2	128	3x3	1x1	2x2	2x2	0.1	256	5x1	1x1	1x1	1x1	0.0	128	3x3	1x1	1x2	1x2	0.3	128	10	1	5	2	0.2
3	256	3x3	1x1	2x2	2x2	0.1	256	5x1	1x1	1x1	1x1*	0.0	256	3x3	1x1	1x2	1x2	0.2	256	10	1	5	2	0.1
4	512	3x3	1x1	2x2	2x2	0.2	512	5x1	1x1	1x1	1x1	0.0	512	3x3	1x1	1x2	1x2	0.1	512	10	1	5	2	0.1
5	1024	3x3	1x1	2x2	2x2	0.3	512	5x1	1x1	1x1	1x1*	0.0	1024	3x3	1x1	1x2	1x2	0.0	1024	10	1	5	2	0.0

Table 1: Configuration of the 5-layers CNN used for the features. n : number of filters, k : kernel size, ks : kernel stride, p : pool size, ps : pool stride, d : dropout. 1st dimension is time, 2d dimension is frequency. Convolutions are "same" for Me, Rh and Ly, and "valid" for Ha. *Ha uses dilation rate of 20 and 13 along time dimension on 3^{rd} and 5^{th} layers respectively.

Rhythmic features Given the tempo invariance on the periodicity dimension explained in Section 3.1, the first layer has a 20 bins kernel on this axis to capture all patterns within 2 octaves (for instance quarter, eight and sixteenth notes). All other layers have 3x3 kernel with max-pooling of size 2 on the periodicity axis.

Lyrics features We conducted various experiments to find the best kernel size to apply to the lyrics. It appeared that a rather short receptive field of 10 bins yield the best results. We kept it for all layers, with a mean-pooling of size 5.

Finally, a dense layer applied after the temporal attention block outputs a 512 bins embeddings, L2-normalized so that each track becomes a point on the surface of the unit hypersphere, bounding the distance between 2 points within the [0,2] interval.

3.4 Embedding concatenation

In this work, we investigated only a late embedding fusion scheme, which simply consists in concatenating each feature embedding, and to L2-normalize the concatenated result. It is straightforward to show that the distance between a pair of normalized concatenation of n feature embeddings is the quadratic mean of the n distances between each feature embedding pair. All feature combination scores in Section 4 have been obtained using this method.

The practical advantage of this simple concatenated embedding is twofold: it remains easy to store and to query in an large index, and the lookup can be done for some specific feature combinations only (zero masking the unwanted feature embeddings).

4. EXPERIMENT AND RESULTS

In this section, we present our experimental setup and results. For brevity, we will denote melodic, harmonic, rhythmic and lyrics features by their abbreviations Me, Ha, Rh, and Ly, respectively.

4.1 Experimental setup

We use the exact same protocol as in our previous work [8].

Training We trained our four models on the publicly available dataset SHS_{5+} [1], which contains ~62k covers of ~7.5k works. We used the provided features for Me and Ha, and extracted Rh and Ly from the audio, as described in sections 3.1 and 3.2. We used a semi-hard triplet loss, using an Adam optimizer, an initial learning rate of $1e^{-4}$ and a batch size of 64. All other details are replicated from [8].

Test We tested our four models on SHS_{4-} [1], containing ~50k covers of ~20k works. We also retrained models on $SHS_{5+/4-}$ and tested on Da-Tacos [2], containing 13k covers of 1k works and 2k confusing tracks [29]. As some samples overlap between Da-Tacos, SHS_{4-} and Dali, we made sure that none of these samples were used for scoring the models, as done in [5].

For each feature, we used the corresponding trained model to compute each track embedding, and computed their pairwise distance matrix. For feature combinations, distance matrix is computed using the quadratic mean of each feature distance matrix, as described in Section 3.4.

4.2 Results

We summarize in Table 2 the performances obtained on SHS_{4-} and Da-Tacos by our models, reporting the metrics classically used for VI: Mean Average Precison (MAP), mean number of correct answers in the first 10 (MT@10) and mean rank of first correct answer (MR1).

Train set	SHS_{5+}			Pruned $SHS_{5+/4-}$		
Test set	Pruned SHS_{4-}			Pruned Da-Tacos		
Input feature	MAP	MT@10	MR1	MAP	MT@10	MR1
Me	0.427	0.822	1131	0.363	4.064	97
Ha	0.538	1.003	982	**0.488**	**5.256**	**63**
Rh	0.099	0.231	2921	0.055	0.689	244
Ly	**0.672**	**1.190**	**968**	0.393	4.596	199
Me+Ha	0.693	1.256	453	**0.626**	**6.668**	**32**
Me+Ha+Ly	**0.800**	**1.396**	291	0.602	6.480	33
Me+Ha+Rh	0.688	1.250	413	0.557	5.994	33
Me+Ha+Rh+Ly	0.785	1.378	**286**	0.560	6.054	33
Me+Ha (O)	0.879	1.521	97	0.837	8.709	4
Me+Ha+Rh (O)	0.939	1.607	21	0.905	9.303	**1**
Me+Ha+Ly (O)	**0.963**	**1.637**	**14**	**0.918**	**9.398**	**1**
Me+Ha+Rh+Ly (O)	**0.978**	**1.658**	**4**	**0.951**	**9.657**	**1**

Table 2: Performance metrics obtained on SHS_{4-} and Da-Tacos for input features and their combinations. O=oracle

We first examine the performances of each single feature. Me and Ha results are in line with our previous work [8].

Rhythmic features The performances obtained using Rh are clearly lower, which suggests that our rhythm feature is not providing relevant VI information, or that the rhythm patterns themselves are not specific enough to discriminate between versions and non-versions. As a consequence, adding Rh degrades our Me+Ha baseline. This is not entirely surprising, as many non-versions might exhibit similar rhythmic patterns (we will however see in Section 4.3 and Section 5 that Rh can be relevant in some cases).

[2] https://github.com/MTG/da-tacos

Lyrics features In contrast, the use of lyrics clearly outperforms the other features on SHS_{4-}. This confirms that versions often share the same lyrics, and that even a very inaccurate ALR system can be beneficial to VI. The combination Me+Ha+Ly improves by more than 10% the Me+Ha performances, which will be explained in Section 5.1.

On Da-Tacos, we observe much less clear-cut results. As already noticed by Vaglio et al. [30], Da-Tacos has about 20% of instrumental songs. A closer look to our results shows that these songs typically produce false positives. Following Vaglio et al., we pruned the instrumental songs from Da-Tacos, and recomputed the results for Ly, obtaining MAP=0.674, MT@10=5.931 and MR1=59, which is consistent with the values obtained for SHS_{4-}.

4.3 Oracle results

We also present in Table 2 the results obtained by an oracle. This oracle only considers the best performing feature to compute each pairwise distance: for each pair of tracks, it only uses the feature embeddings yielding the lowest (resp. highest) distance for versions (resp. non-versions).

Interestingly, it appears that the performances of an oracle using three or four features approaches the theoretical optimal values on our two datasets. This is particularly clear with the Me+Ha+Ly and Me+Ha+Rh+Ly combinations: the MAP tends to 1.0, i.e. most versions have been ranked in the first answers for each query. The MR1 also tends to 1, i.e. first answer is correct for most queries. The MT@10 also tends to the theoretical optimal values (in SHS_{4-}, each track has 2, 3 or 4 versions, and its optimal MT@10=1.695, while in Da-Tacos, each track has 0 or 12 versions, and its optimal MT@10=10).

The Figure 3 displays the contribution of each feature to the oracle score, i.e. which feature is the most relevant to determine if two tracks are versions ("Positive pairs") or non-versions ("Negative pairs").

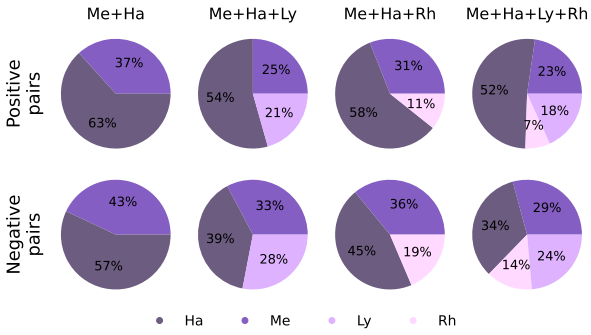


Figure 3: Most relevant feature proportions to identify positive and negative pairs on SHS_{4-}.

It appears that Ha is usually the most efficient feature to discriminate between versions or non-versions. However, both Me and Ly contribute importantly to identification (e.g. resp. 25% and 21% of the positive pairs, 33% and 28% of the negative pairs in the Me+Ha+Ly combination). Finally, and even though Rh alone yields poor results, its contribution is not negligible when combined with other features (e.g. 14% of the negative pairs in the Me+Ha+Rh+Ly combination).

4.4 Comparison with state-of-the-art

Performances obtained by recent VI systems are summarized in Table 3 (second column indicates the embedding size used by each system).

Our system using Me+Ha+Ly improves the state-of-the-art on SHS_{4-} and on Da-Tacos (without instrumentals).

Test set		SHS_{4-}			Da-Tacos		
Model	Emb.	MAP	MT@10	MR1	MAP	MT@10	MR1
Doras et al. [8]	512	0.660	1.080	657	0.635	6.744	30
Vaglio et al. [30]	n/a	n/a	n/a	n/a	0.804*	n/a	n/a
Du et al. [5]	1536	n/a	n/a	n/a	**0.791**	n/a	**19.2**
Me+Ha+Ly (ours)	1536	**0.800**	**1.396**	**291**	**0.818***	**7.205***	**16***
					0.602	6.480	33

Table 3: Sota comparison on SHS_{4-} and Da-Tacos. *results obtained on Da-Tacos-Vocals (w/o instrumental tracks).

5. QUALITATIVE ANALYSIS

In this section, we illustrate qualitatively the previous quantitative results. We encourage readers to listen to the audio samples available on the paper companion website [1] as part of reading the paper.

5.1 Ly vs. Me+Ha examples

We intend here to illustrate how Ly improves the Me+Ha system. Figure 4 plots the distance obtained for Me+Ha and Ly between randomly sampled positive (green) and negative (red) pairs.

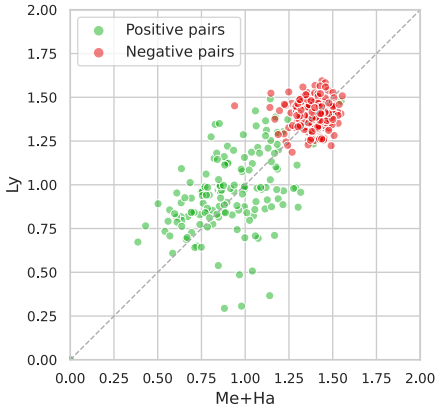
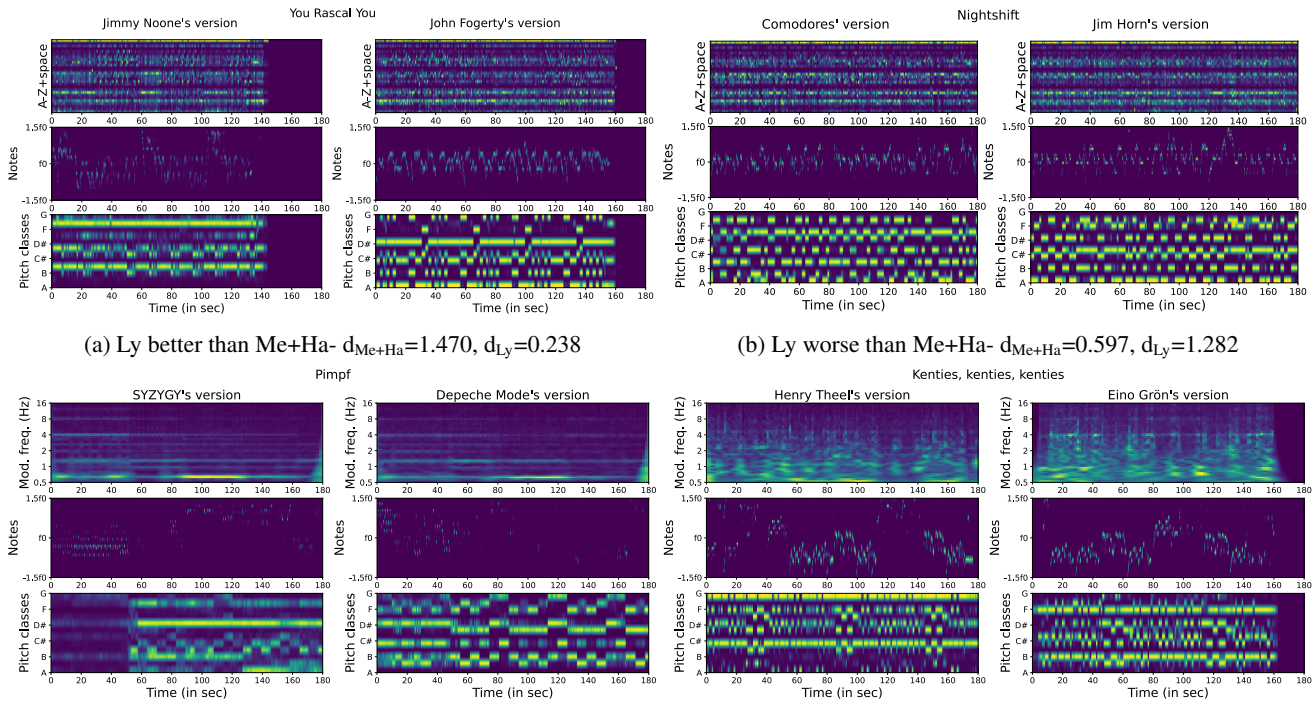


Figure 4: Pairwise distances for Me+Ha vs. Ly (500 pairs from SHS_{4-}).

This plot confirms that Me+Ha and Ly are complementary, as already seen in Section 4.3. The dots situated away from the diagonal correspond to track pairs scored correctly by Me+Ha and incorrectly by Ly (or vice-versa). As certain features yield better results for certain songs, their combination will statistically improve the results, as we will now illustrate with some contrasted examples.

Ly > Me+Ha There are many versions whose musical style, melody and harmony differ greatly from the original, and where only the lyrics can help to identify them. This is illustrated on Figure 5(a), which shows that the Jimmy Noone's and John Fogerty's versions of "You Rascal You" are very different musically while the lyrics exhibit enough similarity to be correctly identified.



(a) Ly better than Me+Ha- d_{Me+Ha}=1.470, d_{Ly}=0.238

(b) Ly worse than Me+Ha- d_{Me+Ha}=0.597, d_{Ly}=1.282

(c) Rh better than Me+Ha - d_{Me+Ha}=1.084, d_{Rh}=0.459

(d) Rh worse than Me+Ha - d_{Me+Ha}=0.498, d_{Rh}=1.380

Figure 5: Examples for rhythm and lyrics : Rh > Me+Ha 5(c), Rh < Me+Ha 5(d), Ly > Me+Ha 5(a) and Ly < Me+Ha 5(b)

It also appears that our approximated ALR system is efficient for different languages. For instance, the versions of Asta Kask and of The Hep Stars of the song "I natt jag drömde", are very different in melody and harmony (d_{Me+Ha}=1.448) while the lyrics remain similar (d_{Ly}=0.342), despite the fact that the lyrics are in Swedish.

Ly < Me+Ha As already mentioned, using Ly will yield wrong results in presence of instrumental version (i.e. no lyrics). In the example shown in Figure 5(b), the version of "Nightshift" by the Commodores has lyrics while the one of Jim Horn's does not. We noticed that the system sometimes considers lead instruments as voices. However, this Ly false negative is correctly caught by the Me+Ha.

5.2 Rh vs. Me+Ha examples

Although less obvious than for Ly, combining Rh and Me+Ha also appears to be complementary in some cases.

Rh > Me+Ha Even though Rh yields poor performances in general, there are cases where it is the only feature available to identify versions. This is illustrated on Figure 5(c), which shows the Rh, Me and Ha features for two versions of "Pimpf". In this song, the melody is almost non-existent, and the harmony is very different between both versions. Only a few bass notes in the middle are salient enough to identify the song, and this short bassline appears similarly on the two FP features.

Rh might also be a good discriminating feature in other cases, e.g. for live concert versions. One version of "Mama's Little Baby" is recorded in studio while the other is a concert filmed from the audience. The Me+Ha distance between these versions is high (d_{Me+Ha}=1.329) because of the bad live recording quality. But, the drums are distinguishable enough to find similarity (d_{Rh}=0.714).

Rh < Me+Ha But Rh often yields wrong results. This is illustrated on Figure 5(d), which shows the features of two versions of "Kenties kenties kenties". Although melody and harmony are similar, the rhythm is very different, and the use of Rh produces a false negative.

6. CONCLUSION

It was shown previously that VI systems combining melody and harmony yields promising performances. In this paper, we proposed to consider also rhythmic and lyrics features to improve these results further. We showed that an existing rhythmic feature commonly used for genre classification is only helpful in a few cases, such as live version identification. But we also showed that an approximate lyrics representation can improve the performances of existing melody and harmony-based systems. We explained these results by the fact that detecting correctly only a few character sequences appears to be enough to distinguish versions and non-versions. We showed that our system combining these features establishes new state-of-the-art on two public datasets. More importantly, we indicated that these feature combinations provide enough information to approach the theoretical optimal performances obtainable on these datasets.

In our future work, we will investigate a more elaborated fusion scheme in order to train our model to behave as an oracle: our objective is to teach the system how to choose between available features to pick only the most relevant one for each pair of tracks. This might answer the question of whether the concept of musical version can be reduced to its melodic, harmonic, rhythmic, and lyrics dimensions.

7. ACKNOWLEGMENTS

Authors would like to thank the anonymous reviewers for their helpful comments, and for suggesting the Figure 3.

8. REFERENCES

[1] F. Yesiler, G. Doras, R. M. Bittner, C. J. Tralie, and J. Serrà, "Audio-based musical version identification: Elements and challenges," *IEEE Signal Processing Magazine*, vol. 38, no. 6, 2021.

[2] Z. Yu, X. Xu, X. Chen, and D. Yang, "Learning a representation for cover song identification using convolutional neural network," *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, 2020.

[3] G. Doras and G. Peeters, "Cover detection using dominant melody embeddings," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2019.

[4] F. Yesiler, J. Serrà, and E. Gómez, "Accurate and scalable version identification using musically-motivated embeddings," *Proceedings of ICASSP (International Conference on Acoustics, Speech and Signal Processing)*, 2020.

[5] X. Du, K. Chen, Z. Wang, B. Zhu, and Z. Ma, "Bytecover2: Towards dimensionality reduction of latent embedding for efficient cover song identification," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 616–620.

[6] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Disentangled multidimensional metric learning for music similarity," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[7] Z. Yu, X. Xu, X. Chen, and D. Yang, "Temporal pyramid pooling convolutional neural network for cover song identification," *Proceedings of the International Joint Conference on Artificial Intelligence*, 2019.

[8] G. Doras, F. Yesiler, J. Serrà, E. Gomez, and G. Peeters, "Combining musical features for cover detection," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2020.

[9] A. Vaglio, R. Hennequin, M. Moussallam, and G. Richard, "The words remain the same - cover detection with lyrics transcription," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2021.

[10] C. Douwes, P. Esling, and J.-P. Briot, "Energy consumption of deep generative audio models," *arXiv preprint arXiv:2107.02621*, 2021.

[11] M. Slaney, K. Weinberger, and W. White, "Learning a metric for music similarity," in *International Symposium on Music Information Retrieval (ISMIR)*, vol. 148, 2008.

[12] B. McFee, L. Barrington, and G. Lanckriet, "Learning content similarity for music recommendation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 8, pp. 2207–2218, 2012.

[13] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of IEEE CVPR (Conference on Computer Vision and Pattern Recognition)*, 2015.

[14] J. Foote, M. Cooper, and U. Nam, "Audio retrieval by rhythmic similarity." in *ISMIR*. Citeseer, 2002.

[15] S. Dixon, F. Gouyon, G. Widmer *et al.*, "Towards characterisation of music via rhythmic patterns." in *ISMIR*, 2004.

[16] E. Pampalk, "Computational models of music similarity and their application in music information retrieval," Ph.D. dissertation, 2006.

[17] T. Pohle, D. Schnitzer, M. Schedl, P. Knees, and G. Widmer, "On rhythm and general music similarity." in *ISMIR*, 2009, pp. 525–530.

[18] H. Foroughmand and G. Peeters, "Deep-rhythm for tempo estimation and rhythm pattern recognition," in *International Society for Music Information Retrieval (ISMIR)*, 2019.

[19] E. Trentin and M. Gori, "A survey of hybrid ann/hmm models for automatic speech recognition," *Neurocomputing*, vol. 37, no. 1-4, pp. 91–126, 2001.

[20] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[21] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[22] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Dali: a large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm." in *19th International Society for Music Information Retrieval Conference*, ISMIR, Ed., 2018.

[23] C. Gupta, E. Yılmaz, and H. Li, "Acoustic modeling for automatic lyrics-to-audio alignment," *arXiv preprint arXiv:1906.10369*, 2019.

[24] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," *arXiv preprint arXiv:1902.06797*, 2019.

[25] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics alignment and transcription in polyphonic music: Does background music help?" in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 496–500.

[26] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 627–636.

[27] N. Zeghidour, Q. Xu, V. Liptchinsky, N. Usunier, G. Synnaeve, and R. Collobert, "Fully convolutional speech recognition," *arXiv preprint arXiv:1812.06864*, 2018.

[28] J. Serrà, S. Pascual, and A. Karatzoglou, "Towards a universal neural network encoder for time series." in *CCIA*, 2018.

[29] F. Yesiler, C. Tralie, A. A. Correya, D. F. Silva, P. Tovstogan, E. Gómez, and X. Serra, "Da-tacos: A dataset for cover song identification and understanding," in *Proceedings of ISMIR (International Society for Music Information Retrieval)*, 2019.

[30] A. Vaglio, R. Hennequin, M. Moussallam, G. Richard, and F. d'Alché Buc, "Audio-based detection of explicit content in music," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 526–530.

A DIFFUSION-INSPIRED TRAINING STRATEGY FOR SINGING VOICE EXTRACTION IN THE WAVEFORM DOMAIN

Genís Plaja-Roglans **Marius Miron** **Xavier Serra**
Music Technology Group, Universitat Pompeu Fabra, Barcelona, Spain
{genis.plaja, marius.miron, xavier.serra}@upf.edu

ABSTRACT

Notable progress in music source separation has been achieved using multi-branch networks that operate on both temporal and spectral domains. However, such networks tend to be complex and heavy-weighted. In this work, we tackle the task of singing voice extraction from polyphonic music signals in an end-to-end manner using an approach inspired by the training and sampling process of denoising diffusion models. We perform unconditional signal modelling to gradually convert an input mixture signal to the corresponding singing voice or accompaniment. We use fewer parameters than the state-of-the-art models while operating on the waveform domain, bypassing the phase estimation problem. More concisely, we train a non-causal WaveNet using a diffusion-inspired strategy while improving the said network for singing voice extraction and obtaining performance comparable to the end-to-end state-of-the-art on MUSDB18. We further report results on a non-MUSDB-overlapping version of MedleyDB and the multi-track audio of Saraga Carnatic showing good generalization, and run perceptual tests of our approach. Code, models, and audio examples are made available. [1]

1. INTRODUCTION

Singing voice extraction, which involves separating the vocal source from music recording mixtures, has received a lot of attention from the Audio Signal Processing (ASP) and Music Information Retrieval (MIR) communities in the recent years. The problem can be modelled in the waveform domain [1–4], the frequency domain [5–7], or a combination of both [8–10]. In general, spectrogram-based approaches have been more popular despite having to deal with the problem of the complex phase, usually leading to artifacts or unnaturalness of the separated sources. Within the Music Demixing Challenge (MDX) framed in ISMIR 2021 [11], diverse submissions achieved state-of-the-art source separation performance, in the majority of cases being multi-branch networks combining features from both time and frequency domains [8–10], proposing therefore solutions to account for the problem with the phase. Nonetheless, these models tend to be heavy-weighted and include engineered strategies to improve the predicted outputs.

While the problem of source separation has been shown to be challenging on the waveform domain, promising results have been reported [2, 3, 12], opening the door for the development of models that bypass the problem with the complex phase. However, as the performance improves, the model size and complexity accordingly grow.

In this work we propose a training and sampling strategy inspired on the recently emerged denoising diffusion models [13] to perform end-to-end singing voice extraction. Denoising diffusion models are a novel class of generative models theoretically grounded in the non-equilibrium statistical physics that can gradually convert one distribution into another using a Markov chain [14], while learning to perform the reverse process. More concisely, numerous works use diffusion models to convert a signal from a particular data distribution to a simple one (e.g. isotropic Gaussian noise) by gradually adding samples of the said simple distribution. Subsequently, the model is trained to reverse the perturbation process and generate data samples of the original distribution using the easily tractable noise as input [15–17]. Diffusion models have recently emerged as a versatile and high-performance method for data generation, outperforming classical generative approaches for the task of image generation [13]. In the fields of ASP and MIR, diffusion models have also shown promising performance for speech synthesis [16, 18], speech restoration and enhancement [13, 15, 19–21], audio super-resolution [22], singing voice synthesis [23], and symbolic music generation [24]. Despite that, the literature does not include many attempts to use diffusion models for source separation, being [25], to the best of our knowledge, the only attempt.

Despite the use of deterministic signals as diffusion perturbation in place of Gaussian noise has shown promising results in reverting different arbitrary types of image noise and performing image morphing [26], to the best of our knowledge, no exploration of this idea in the audio domain has been reported to date. In this work, we build on top of DiffWave, a versatile diffusion model for speech synthesis [16] that is based on a non-causal WaveNet architecture that has been previously used for music source separation [12]. We introduce an end-to-end approach to model

[1] github.com/genisplaja/diffusion-vocal-sep

© G. Plaja-Roglans, M. Miron, and X. Serra. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** G. Plaja-Roglans, M. Miron, and X. Serra, "A diffusion-inspired training strategy for singing voice extraction in the waveform domain", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

the task of singing voice extraction as a diffusion-alike process, by converting between two audio data distributions that share similar content, in this case the singing voice and the corresponding mixture. We propose to use a deterministic diffusion perturbation, a music mixture, to gradually transform its corresponding isolated singing voice into the mixture while learning to conduct the reverse process at inference. Given the formulation of the diffusion process, we train a model that learns to estimate the perturbation at different ratios. Subsequently, we leverage from the parametrization of the reverse process of diffusion models to chain these estimations and sample, given an input mixture, improved vocal source separation in terms of artifacts and interferences compared to the vanilla trained network.

2. METHOD

2.1 Diffusion process

We assume that the waveform-domain signal corresponding to the mixture m is the sum of the singing voice v and the accompaniment a, such as: $m = v + a$. Our goal is to estimate v given m. In the following sections we formalize our method by relating it with the diffusion theory in the literature. In Figure 1 we display the two main steps of our method: the diffusion and the reverse process.

Diffusion. The diffusion process assumes perturbing the training data with different scales of noise iteratively following a Markov chain [13]:

$$q(x_1, ..., x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{1}$$

The input signal x_0 is gradually perturbed by incrementally adding a particular signal in small T diffusion steps. This process results into a latent variable x_T of same distribution of the perturbation. The standard diffusion schema, introduced in [13] and subsequently used in most of the diffusion research, perturbs x_0 with random Gaussian noise, therefore x_T is an isotropic Gaussian noise distribution. In our case, the input signal x_0 is initialized with the isolated singing voice v, and perturbed by incrementally adding the mixture m. This results in x_T being a mixture-alike signal, containing both voice and accompaniment. Therefore, $q(x_t|x_{t-1})$ in Eq. 1 is an operation to add a small amount of perturbation m to the given signal x_{t-1}, moving to the next diffusion step x_t. We use m as perturbation to account for the formal diffusion design in [13], in which the latent variable x_T is expected to belong to the same data distribution as the perturbing noise. The level of perturbation at each diffusion step is controlled by β_t, which is a small positive coefficient within a fixed noise schedule denoted β. That said, using the parametrization proposed in [13], we can compute any given diffusion step using:

$$q(x_t|x_0) = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}m \tag{2}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Note also that the most common inference input of a singing voice extraction model – which in our case is x_T – is a mixture. Given Eq. 2, the perturbation is a mixture m to ensure $x_T \approx m$, otherwise the said condition may not be given.

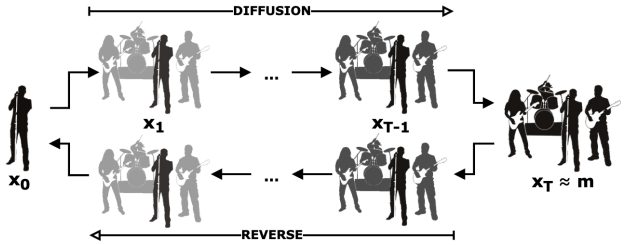


Figure 1. Overview of our diffusion-inspired training approach for the case of singing voice extraction.

Modelling musical signals using a mixture of Gaussian functions has been previously explored in [27]. Note also that architectures similar to DiffWave have been already used to model mixture, accompaniment, and vocal signals [12, 28] (for further detail see Section 2.2).

Reverse process. The reverse process aims at iteratively reverting the perturbation added by the diffusion:

$$p_\theta(x_0, ..., x_{T-1}|m) = \prod_{t=1}^{T} p_\theta(x_{t-1}|x_t) \tag{3}$$

We propose to parameterize the variable $p_\theta(x_{t-1}|x_t)$ as $\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon(x_t, t)\right)\frac{1}{\sqrt{\alpha_t}}$, being $\epsilon(x_t, t)$ the perturbation estimated by the model at a given diffusion step t. This parametrization is leveraged from the diffusion sampling process in [13]. However, we remove the deviation parameter from the original parametrization, which in our deterministic noise case yields worse predictions. This process can be seen as an iterative refinement of the latent variable x_T to convert it to x_0, in our case to iteratively transform an input mixture to its corresponding singing voice source.

Training. The training objective of the original diffusion process is to maximize the log likelihood of: $p_\theta(x_0) = \int p_\theta(x_0, ..., x_{T-1}|x_T)p_{latent}(x_T)dx_{1:T}$ [13], considering stochastic noise as perturbation. Since in this context it is not possible to calculate the said integral, in the literature this problem is approached by maximizing its variational lower bound (ELBO) [13]. The reader is referred to [16], for a detailed development of this maximization. In our case, relying on the ELBO is not required given the deterministic perturbation. Therefore, by using pairs of (x_t, x_0) [13], and referring to Eq. 2, we are able to effectively optimize the model with parameters θ using the following objective [13]:

$$L(\theta) = \left\|\epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1 - \bar{\alpha}_t}m, t))\right\|^2 \tag{4}$$

where ϵ is the target or true perturbation, whereas ϵ_θ is the perturbation the model estimates based on $\bar{\alpha}, t$, the mixture m and the input x_0. In the case of extracting the singing voice, we use the accompaniment a instead of mixture m as the target corresponding to ϵ. Therefore, we do not include the voice into the target of the training process, thus we alleviate the loss of vocal quality that may occur after several reverse steps.

Noise schedule. Choosing the noise schedule β has been found to be crucial for the performance of diffusion models [13] [2] and in fact, efforts have been done to learn said variable instead of defining it manually [29]. In our scenario, we require a schedule β that accounts for the proposed diffusion-inspired approach, in which x_0 is expected to be predominant in the completely perturbed signal x_T. Moreover, our perturbation m is the mixture corresponding to x_0, thus x_0 is contained in the perturbation, which may lead to an abnormally over-loud source in the completely diffused x_T. Ultimately, the inference input is a mixture, therefore we propose to use a schedule that produces a latent variable x_T as close as possible to an actual mixture.

Our noise schedule is defined by $\beta_0 = 1^{-4}$, $\beta_T = 0.2$, and $T = 20$, being 20 a reliable option for audio as found in [16]. This noise schedule produces $q(x_T|x_0) \approx m$, and we denote it β_{20}. Note that the closer β_T to 1, we should expect a more aggressive transformation, leading to less interference at the expense of losing quality of the estimated source. Recent diffusion-based works in the audio domain successfully model their task using 4–8 steps [21]. To study the effect of the number of diffusion steps and explore the feasibility of modelling the task with less computational expense, we experiment with a new schedule β_8, which is defined by $\beta_0 = 1^{-4}$, $\beta_T = 0.5$, and $T = 8$.

Accompaniment estimation. To perform accompaniment estimation we initialize x_0 to be the musical accompaniment a, while the singing voice v is the target ϵ for training in Eq. 4, and we adjust β. Since usually the accompaniment is a mixture of multiple sources, and the singing voice – now the perturbation – is normally predominant, we may increase the number of diffusion steps to 100 in the schedule and experiment with a more granular reverse process. The other parameters remain unchanged.

2.2 Network details

We propose to use the unconditional vanilla DiffWave to learn the reverse process [16]. Although recent works in music separation propose various improvements with regards to the architectures used, in this work we focus on exploring a novel diffusion-inspired process for source separation. At the same time, we aim at improving a smaller model that has been previously applied for source separation using our diffusion-inspired training and sampling method, while leaving the improvements on the network, or using a different architecture, as future work. Nonetheless, we consider the versatility and light weight of the entire method an advantage. Note that within the scope of this paper, we perform monaural source separation.

Architecture. The DiffWave architecture is based on a modified WaveNet [30] extended with bidirectional dilated convolution modules (Bi-DilConv), aiming at removing the autoregressive generation constraint so that the model is non-causal and the reverse process is done in T steps. The said Bi-DilConv modules have been previously applied for the problem of music source separation [12, 28]. The used non-causal WaveNet consists of a stack of L residual layers, which are equally grouped into N blocks. Therefore, each block includes L/N layers with skip-connections as the original WaveNet. A Bi-DilConv module with kernel-size 3 is included in each layer. The size of the dilation, initialized at 1, is doubled at each layer of a block: $[1, 2, 4, 8, ..., 2^{L/N} - 1]$. Before going through the stacked blocks, the input is projected using a 1D convolutional layer of C channels of features. Similarly to the original WaveNet, the output is obtained by summing the skip connections of all the residual blocks. For our experiments we configure the WaveNet architecture with $L = 30$, $N = 10$, and $C = 64$, which in [16] is found to effectively work while preserving the light weight of the architecture.

Diffusion step embedding. Since the training process is based on optimizing Eq. 4 given a pair (x_t, x_0), we need to input the diffusion step t to the model. We use a 128-dimension encoding vector for each t [16], defined as [31]:

$$t_{embed} = \Big[sin\left(10^{\frac{0*F}{S-1}}t\right),\ ...,\ sin\left(10^{\frac{(S-1)*f}{S-1}}t\right),\\ cos\left(10^{\frac{0*F}{S-1}}t\right),\ ...,\ cos\left(10^{\frac{(S-1)*F}{S-1}}t\right)\Big] \quad (5)$$

where F is the embedding factor and S is half the embedding size. Note that F and S are pre-defined and fixed hyperparameters, which in our experimentation are set to $F = 4$ and $S = 64$. Next, the embedded diffusion step is passed through three dense layers, the first two having size $S * 2$, while the latter maps the latent embedding into the C channels the input is projected to, therefore we can add the embedding to the input of each residual layer.

Conditioning. Using a vocoder paradigm, diffusion-based approaches for audio modelling usually use conditioning to guide the signal generation, providing clues to obtain a particular desired output. Audio-related diffusion works in the literature propose to guide the signal generation using, for instance, mel-spectrogram [16, 32] or linguistic features [33]. We do not condition our network for three reasons: **(1)** Conditioning our vocal extraction model, for instance on the mel-spectrogram of the target vocal signal, would probably improve the output quality, however the task of source separation assumes that data of this kind is not available at inference, **(2)** Our latent variable x_T is not an isotropic Gaussian but a known mixture recording containing the target signal, therefore it serves as a conditioner to guide the transformation towards the isolated singing voice, **(3)** We reduce the model size.

2.3 Post-processing

To account for a possible over-increase of the signal amplitude during the reverse process, the output of said process is clamped to the amplitude levels of the input mixture.

During the iterative signal transformation at inference we may generate audio content and artifacts not contained in the original vocal signal. Although not perceptually significant, these artifacts are heavily penalized by objective evaluation metrics [34]. Moreover, the iterative nature of the algorithm may accumulate several of the said artifacts

[2] Despite being aware that our perturbation is a deterministic signal instead of stochastic noise, we still use the term noise schedule in this work for easier understanding in relation with cited works.

in the final prediction. As an optional step, we use the multichannel Wiener filter to improve our separation, a well-known process used in numerous source separation works [35]. We use the Python version in `norbert` [36].

Since our model operates in an end-to-end manner, we use the following procedure to apply the Wiener filtering. Let $\hat{v}$ be an estimated vocal source and $\hat{a}$ the corresponding estimated musical accompaniment. The inputs of the Wiener process are the Short-Time Fourier Transform (STFT) of the input mixture m, and the magnitude STFT of both $\hat{v}$ and $\hat{a}$ estimates. The outputs of the Wiener process are the filtered complex spectrograms for $\hat{v}$ and $\hat{a}$. We take the magnitude of said spectrograms and combine it with the corresponding phases $\phi(\hat{v})$ and $\phi(\hat{a})$ that our proposed model estimates. In that sense, we do not use the Wiener filtering to estimate the phase as typically done for spectrogram-based approaches that cannot estimate such complex target, but refine our estimation using the Expectation-Maximization algorithm [37] to make sure that the predicted signal is contained in the input mixture.

3. RELATION WITH PREVIOUS WORK

The literature on waveform-based singing voice extraction is mainly based on encoder/decoder architectures, with the non-causal adaptation of WaveNet [12, 30] as the only exception. Wave-U-Net [3] is an autoencoder inspired by its spectrogram-based counterpart U-Net [7], while ConvTas-Net [2, 4] estimates prediction masks in the mid-point of the network. The leading model is Demucs, now available in two versions v1 [1] and v2 [2], which is also a convolutional autoencoder with a bidirectional LSTM in the bottleneck. There is a growing tendency on the size of the above-mentioned models, while all use similar, standard training procedures. In contrast, we focus on the training strategy and propose a diffusion-inspired approach for a light-weight model. Building on top of DiffWave, we train a non-causal WaveNet very similar to the one used for music source separation [30], differing only on the number of residual layers, the output projection (since WaveNet in [30] estimates multiple targets while in DiffWave the target is only the added perturbation by the diffusion process), and the additional diffusion step embedding.

Our work has common aspects with [39], in which a novel training strategy is built on top of Demucs (v2) by modelling and learning the dependencies between target sources, and adding an iterative refinement at the output using a Gibbs sampling process. Contrastingly, we use a diffusion-inspired algorithm to train a smaller baseline model for source separation, not to directly estimate the target sources but to gradually transform a mixture signal into its corresponding singing voice.

The literature of diffusion-related approaches for music source separation is scarce. In [25], an improved reverse process based on Langevin dynamics is proposed and applied to several autoregressive models, including WaveNet, which shows competitive performance on separating the vocals from a piano accompaniment. However no experiments on MUSDB18 [40] are reported.

4. EXPERIMENTS

4.1 Experimental setup

We include the following models in our experiments: **(1)** Singing voice extraction model with different noise schedules: β_{20} and β_8 and β_1, **(2)** Singing voice extraction model with β_{20} and Wiener filtering, using the accompaniment obtained by subtracting the estimated vocals $\hat{v}$ from the input mixture m, **(3)** Accompaniment extraction model with β_{100}, **(4)** Combination of singing voice extraction model with β_{20} and accompaniment extraction model with β_{100} using Wiener filtering. For the experiments we use ADAM optimizer with learning rate of 2^{-4} and batch size of 8. The models are first trained for 200k steps and next, we keep training while evaluating the performance using BSS Eval [41] repeatedly when ≈ 500 training steps are completed, storing the model that performs the best on the validation set, until 500k steps.

We use the MUSDB18 dataset [40] for training. The accompaniment is computed as the linear sum of the *bass*, *drums*, and *other* sources as represented in the dataset. To train the models we first split the tracks in MUSDB18 in chunks of 4 seconds to optimize the training process and obtain more variate data batches along the training steps. We do not disregard the unvoiced samples, aiming at improving the estimation on vocal silences [28].

For a comparison with the state-of-the-art, our models are evaluated on the test set of MUSDB18, using the standardized metrics for source separation (Signal-to-Distortion Ratio or SDR, Signal-to-Interference Ratio or SIR, and Signal-to-Artifact Ratio or SAR) [42]. We use the BSS Eval implementation and the evaluation setup from the SiSEC challenge [41], using window and hop sizes of 1 second, and subsequently computing the median over all the 1-second estimations of each song. We finally report the median over the entire MUSDB18 testing tracks. We compare our approach with the waveform-based state-of-the-art: WaveNet, Wave-U-Net, ConvTas-Net and Demucs v1 and v2. We report the metrics that the best versions of these methods obtain on the testing set of MUSDB18 [41]. For WaveNet we consider the best performing configuration in [28], and for ConvTas-Net the music source separation version proposed in [2].

Being aware of the possible biases that might occur if training and evaluating on data from the same distribution, even if the splits are properly differentiated, we consider two additional testing sets: (1) A non-MUSDB-overlapping version of MedleyDB [43], in which we remove the 46 overlapping tracks between MedleyDB and MUSDB18 [44] disregarding also the tracks from shared artists between the two even if the track is not present in both, and (2) A manually-cleaned subset of the multi-track audio of the Saraga Carnatic dataset [45] (ground-truth accompaniment is not available). The track list of both datasets is made available in the accompanying repository.

The objective metrics in [41] are not always correlated with the scores from perceptual evaluations of music source separation [46]. However, perceptual tests are

			Singing Voice			Accompaniment		
Model	**Params**	**Diff. steps**	**SDR**	**SIR**	**SAR**	**SDR**	**SIR**	**SAR**
WaveNet [12] (w/ add. loss [38])	≈ 3.3M	-	4.49	13.52	6.17	11.39	16.37	13.49
Wave-U-Net [3]	≈ 10.2M	-	4.97	13.98	4.41	11.11	15.30	11.44
ConvTas-Net [2]	≈ 8.75M	-	6.43	-	-	-	-	-
Demucs (v1) [1]	-	-	5.44	-	-	-	-	-
Demucs (v2) [2]	≈ 450M	-	6.84	-	-	-	-	-
Ours (vocal)	≈ 750K	1	4.81	9.21	8.09	-	-	-
Ours (vocal)	≈ 750K	8	5.63	10.55	8.86	-	-	-
Ours (vocal)	≈ 750K	20	5.59	10.78	8.89	-	-	-
Ours (vocal) + Wiener	≈ 750K	20	5.66	11.60	8.49	-	-	-
Ours (accomp)	≈ 750K	100	-	-	-	11.12	13.11	16.44
Ours (vocal & accomp) + Wiener	≈ 750K + 750K	20 + 100	6.07	12.77	8.61	11.72	14.44	16.81

Table 1. Performance comparison between our model and the waveform-based state-of-the-art. Metrics in dB.

time-consuming and expensive to conduct. We run a perceptual evaluation of the vocal separation for four models: Wavenet (again the best model in [28]), Wave-U-Net (the best model in [3] for monaural separation), Demucs (the v2 model for stereo mixture and 4 sources), and our best model (combining both vocal and accompaniment extraction models using Wiener). We reiterate the experiment in [28] with the same 5 songs (10 second excerpts) and including now our model and Demucs v2. [3]

In this perceptual experiment we follow a double-blind multi-stimulus experimental design with a hidden reference. Similarly to [28], participants are asked to assess the global quality of vocal separation taking into account the suppression of other sources and the lack of distortion, rating the stimuli on scale from 1 to 5, with 1 being *very intrusive interferences from other sources and degraded audio*, and 5 being *unnoticeable interferences from other sources and not degraded audio*. In contrast to [28], the order of the songs is randomized, so that the final rating does not depend on a predefined ordering. In addition, we include the ground-truth vocal stem as a hidden reference along the other stimuli corresponding to the vocal separation of the four models being tested. This hidden reference is used as a control stimuli to filter-out participants that have not performed the training stage, have not understood the task, or do not have sufficient expertise. The participants are asked to calibrate the volume using a tone burst. Then, they perform a training stage where detailed instructions and three audio examples from the same song are presented: the reference mixture, the ground-truth vocal stem and a poor quality separation using a model not included in our test. We use the webMUSHRA framework [47] to implement the experimental design in an online test.

4.2 Objective evaluation

In Table 1 we compare the performance of our approach with the waveform-domain state-of-the-art models. No metrics on accompaniment separation and SIR/SAR for vocals are reported for Demucs and ConvTas-Net since these target to *vocals, bass, drums and other*. We observe that our vocal extraction model outperforms WaveNet (note that DiffWave is based on WaveNet), Wave-U-Net, and Demucs v1 in terms of SDR, the latter by a slight difference. Our model provides notable improvement on SAR, which is translated into an output with less artifacts. When combining our vocal and accompaniment extraction models through Wiener filter we obtain closer performance to ConvTas-Net, while Demucs v2 is still leading on SDR $\approx$ $0.8dB$ above. While iteratively transforming an input mixture, for instance, to the corresponding singing voice, incorrectly estimated accompaniment that is not recognised in the subsequent reverse steps may accumulate in the final prediction. Although the said interferences might not be audible at naked ear, these are penalized by the metrics. The Wiener filtering provides notable improvement on that issue, as especially noted in the SIR and also in the perceptual evaluation in Section 4.3, albeit the source quality is slightly compromised. Finally, note that we use fewer parameters, enhancing the portability and reproducibility of our approach.

The β_1 singing extraction model, which directly estimates the perturbation by transforming the input mixture in a single run, scores similar than the baseline WaveNet, however we observe a notable decrease of SIR, while SAR improves. This may be given the transformation nature of our approach, which removes the perturbation from the target source instead of directly estimating the source. The β_8 model scores similar than β_{20}, however the SIR decreases while SAR is maintained. While adding more steps provides improved interference removal, no notable negative effect is observed in the quality of the estimated source. Nonetheless, if the computational expense is prioritized, the β_8 model may be used for an optimized inference since the measured performance drop in our experiments is not dramatic. In fact, both models predict faster than real-time on a TITAN Xp GPU. For accompaniment separation, using β_{100} provides better predictions. However, as observed for vocals, we may be also capable of modelling this task using less steps, with no significant performance drop.

[3] jordipons.me/apps/end-to-end-music-source-separation

		MUSDB18			MedleyDB			Saraga Carnatic
Model for singing voice	**Model weight**	**SDR**	**SIR**	**SAR**	**SDR**	**SIR**	**SAR**	**SDR**
Ours (vocal, β_{20}, no Wiener)	$\approx$ 26MB	5.59	10.78	8.89	4.86	8.87	9.06	4.11
Wave-U-Net [3]	$\approx$ 117MB	4.97	13.98	4.41	1.61	7.47	4.50	2.13
Demucs (v2) [2]	$\approx$ 1GB	6.84	-	-	6.01	-	-	6.12

Table 2. Performance comparison of our baseline model and state-of-the-art on additional test datasets. Metrics in dB.

In Table 2 we evaluate our baseline β_{20} singing voice extraction model (with no post-processing) on the two additional testing datasets presented in Section 4.1. We perform the same evaluation procedure for Wave-U-Net and Demucs v2, comparing how the three models generalize to the testing datasets, using the performance on MUSDB18 – which is also the training dataset for the three – as reference. Our model and Demucs v2 show good generalization to MedleyDB, both getting a similar and small performance drop. Contrastingly, the Wave-U-Net performance is negatively affected in terms of both SDR and SIR. A similar scenario is observed in the Carnatic Music experiment. While Wave-U-Net generalization is again compromised, our model and Demucs v2 are decently able to maintain the performance, the latter being less affected by the change of domain. Similarly to what observed in the MUSDB18 experiment, Demucs v2 predictions include less artifacts, especially in the high-frequency range, being reflected in the metrics as such. Audio examples of this experiment are available in the accompanying repository.

We analyse the behaviour of the β_{20} singing voice model along the steps in the reverse process. We observe that the SIR (interf.) notably increases along the steps, at a compromise of a much less steeply SAR (artifacts) decrease. Namely, as we iteratively transform the signal from mixture to singing voice, we remove the accompaniment while trying to maintain the quality of the singing voice, relying on the model trained with our diffusion-inspired strategy to estimate the perturbation at each step while alleviating the additional interferences incorrectly generated during the reverse process. We note that given the parametrization of the reverse process, stronger transformation is performed in the first steps (1 to 5 for β_{20}), while the rest of the steps refine the final estimation. For that reason, fair or good performance – relatively to the overall track difficulty – on the first step normally leads to enhanced final output, while bad initial performance may even be further degraded through the reverse process.

4.3 Perceptual evaluation

In total 40 people participated in our experiment, 4 of them being excluded because they scored the ground-truth stimuli lower than a separation stimuli. We compute Mean Opinion Score (MOS) by averaging the ratings for all songs and all participants. The results in terms of MOS are presented in Figure 2. Note that Ground-truth is not reaching 5, meaning that the test includes difficult cases with distorted vocals or large unvoiced segments. We observe that the 95% Confidence Interval for our model is very similar to Demucs and notably higher than both Wave-U-Net and especially WaveNet, a very similar architecture to the instance we have trained using our diffusion-inspired strategy. Such test suggests that the predictions made by our approach may include artifacts or interferences that affect negatively the standardized metrics but are not perceivable at naked ear. This test may be extended in future to separately study the perceivable distortion and interference.

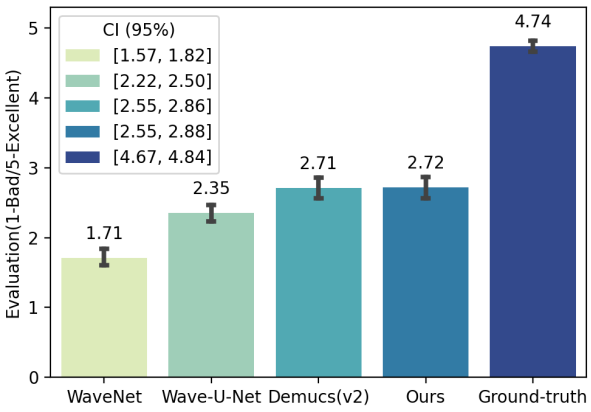


Figure 2. Perceptual evaluation report for the waveform-based state-of-the-art models and ours

5. CONCLUSIONS

In this paper we leverage from the denoising diffusion algorithm to propose a training and sampling strategy for singing voice extraction. The model trained using our approach learns to gradually transform a mixture into its corresponding vocal source or accompaniment, achieving comparable performance to the waveform-based state-of-the-art on MUSDB18. In addition, we evaluate how our approach generalizes to other testing sets, showing decent generalization to these out-of-domain data. We also run a perceptual test in which our approach scores similar than Demucs v2 and outperforms the others. Our approach operates on an architecture similar to WaveNet and obtains better objective and perceptual evaluation. This work has a broad future outlook. For the next steps, we look at separating other sources and supporting stereo. We also look at extending the approach, for instance, by using a different or improved network to learn the reverse process, focusing on U-Nets which are the leading architecture music source separation. We may also consider conditioning, the key element of diffusion-based approaches in the literature. Finally, our diffusion-inspired reverse parametrization may be further improved to better refine the predictions.

6. ACKNOWLEDGEMENTS

This work was carried out under the projects Musical AI - PID2019-111403GB-I00/AEI/10.13039/501100011033 and NextCore - RTC2019-007248-7 funded by the Spanish Ministerio de Ciencia, Innovación y Universidades (MCIU) and the Agencia Estatal de Investigación (AEI). We would like to thank the 40 participants that took the perceptual test for this work.

7. REFERENCES

[1] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Demucs: Deep Extractor for Music Sources with extra unlabeled data remixed," 2019. [Online]. Available: http://arxiv.org/abs/1909.01174

[2] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music Source Separation in the Waveform Domain," 2019. [Online]. Available: http://arxiv.org/abs/1911.13254

[3] D. Stoller, S. Ewert, and S. Dixon, "Wave-U-Net: A multi-scale neural network for end-to-end audio source separation," *In Proc. of the 19th Int. Society for Music Information Retrieval Conf. (ISMIR),* Paris, France, pp. 334–340, 2018.

[4] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing Ideal Time-Frequency Magnitude Masking for Speech Separation," *IEEE/ACM Transactions on Audio Speech and Language Processing*, vol. 27, no. 8, pp. 1256–1266, 2019.

[5] N. Takahashi and Y. Mitsufuji, "D3Net: Densely connected multidilated DenseNet for music source separation," 2020. [Online]. Available: http://arxiv.org/abs/2010.01733

[6] T. Li, J. Chen, H. Hou, and M. Li, "Sams-Net: A Sliced Attention-based Neural Network for Music Source Separation," *In Proc. on the 12th Int. Symposium on Chinese Spoken Language Processing (ISCSLP)*, 2021.

[7] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep U-Net convolutional networks," *In Proc. of the 18th Int. Society for Music Information Retrieval Conf. (ISMIR),* Suzhou, China, pp. 745–751, 2017.

[8] A. Défossez, "Hybrid spectrogram and waveform source separation," 2021. [Online]. Available: http://arxiv.org/abs/2111.03600

[9] M. Kim, W. Choi, J. Chung, D. Lee, and S. Jung, "KUIELab-MDX-Net: A Two-Stream Neural Network for Music Demixing," 2021. [Online]. Available: http://arxiv.org/abs/2111.12203

[10] C.-Y. Yu and K.-W. Cheuk, "Danna-Sep: Unite to separate them all," 2021. [Online]. Available: http://arxiv.org/abs/2112.03752

[11] Y. Mitsufuji, G. Fabbro, S. Uhlich, F.-R. Stöter, A. Défossez, M. Kim, W. Choi, C.-Y. Yu, and K.-W. Cheuk, "Music Demixing Challenge 2021," *Frontiers in Signal Processing*, no. January, 2022.

[12] D. Rethage, J. Pons, and X. Serra, "A wavenet for speech denoising," *In Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Paris, France, vol. 2018-April, pp. 5069–5073, 2018.

[13] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *In Proc. of the 33th Advances in Neural Information Processing Systems (NeurIPS),* Online, pp. 6840–6851, 2020.

[14] C. Jarzynski, "Equilibrium free-energy differences from nonequilibrium measurements: A master-equation approach," *Physical Review*, 1997.

[15] J. Zhang, S. Jayasuriya, and V. Berisha, "Restoring degraded speech via a modified diffusion model," *In Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH),* Online, vol. 4, pp. 2753–2757, 2021.

[16] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "DiffWave: A Versatile Diffusion Model for Audio Synthesis," *In Proc. of the 9th Int. Conf. on Learning Representations (ICLR),* Vienna, Austria, 2021.

[17] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," *In Proc. of the 32nd Int. Conf. on Machine Learning (ICML),* Lille, France, vol. 3, pp. 2246–2255, 2015.

[18] N. Chen, Y. Zhang, H. Zen, R. J. Weiss, M. Norouzi, and W. Chan, "WaveGrad: Estimating Gradients for Waveform Generation," *In Proc. of the 9th Int. Conf. on Learning Representations (ICLR),* Vienna, Austria, 2021.

[19] Y.-J. Lu, Z.-Q. Wang, S. Watanabe, A. Richard, C. Yu, and Y. Tsao, "Conditional Diffusion Probabilistic Model for Speech Enhancement," *In Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Singapore, 2022.

[20] Y. J. Lu, Y. Tsao, and S. Watanabe, "A Study on Speech Enhancement Based on Diffusion Probabilistic Model," *In Proc. of the Asia-Pacific Signal and Information Processing Association Annual Summit and Conf. (APSIPA ASC),* Online, pp. 659–666, 2021.

[21] J. Serrà, S. Pascual, J. Pons, R. O. Araz, and D. Scaini, "Universal speech enhancement with score-based diffusion," 6 2022. [Online]. Available: http://arxiv.org/abs/2206.03065

[22] J. Lee and S. Han, "NU-wave: A diffusion probabilistic model for neural audio upsampling," *In Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH),* Brno, Czech Republic, vol. 4, no. 3, pp. 2698–2702, 2021.

[23] J. Liu, C. Li, Y. Ren, F. Chen, and Z. Zhao, "DiffSinger: Singing Voice Synthesis via Shallow Diffusion Mechanism," *In Proc. of the 36th Conf. on Artificial Intelligence (AAAI),* Online, 2022.

[24] G. Mittal, J. Engel, C. Hawthorne, and I. Simon, "Symbolic Music Generation with Diffusion Models," *In Proc. of the 22th Int. Society for Music Information Retrieval Conf. (ISMIR),* Online, pp. 468–475, 2021.

[25] V. Jayaram and J. Thickstun, "Parallel and Flexible Sampling from Autoregressive Models via Langevin Dynamics," *In Proc. of the Int. Conf. on Learning Representations (ICLR),* Online, 2021.

[26] A. Bansal, E. Borgnia, H.-M. Chu, J. S. Li, H. Kazemi, F. Huang, M. Goldblum, J. Geiping, and T. Goldstein, "Cold diffusion: Inverting arbitrary image transforms without noise," 8 2022. [Online]. Available: http://arxiv.org/abs/2208.09392

[27] J. L. Durrieu, G. Richard, B. David, and C. Févotte, "Source/filter model for unsupervised main melody extraction from polyphonic audio signals," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, 2010.

[28] F. Lluís, J. Pons, and X. Serra, "End-to-end music source separation: Is it possible in the waveform domain?" *In Proc. of the Annual Conf. of the Int. Speech Communication Association (INTERSPEECH),* Graz, Austria, vol. 2019-September, pp. 4619–4623, 2019.

[29] D. P. Kingma, T. Salimans, B. Poole, and J. Ho, "Variational Diffusion Models," *In Proc. of the 35th Conf. on Neural Information Processing Systems (NeurIPS 2021),* Online, pp. 21 696–21 707, 2021.

[30] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," 2016. [Online]. Available: http://arxiv.org/abs/1609.03499

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems (NeurIPS),* Long Beach, USA, vol. 2017-December, pp. 5999–6009, 2017.

[32] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller, "Deep Voice 3: Scaling text-to-speech with convolutional sequence learning," *In Proc. of the Int. Conf. on Learning Representations (ICLR),* Vancouver, Canada, 2018.

[33] S. O. Arik, G. Diamos, A. Gibiansky, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep Voice 2: Multi-speaker neural text-to-speech," *In Proc. of the Conf. on Neural Information Processing Systems (NeurIPS),* Long Beach, USA, 2017.

[34] P. Chandna, M. Blaauw, J. Bonada, and E. Gomez, "A Vocoder Based Method for Singing Voice Extraction," *In Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Brighton and Hove, UK, vol. 2019-May, pp. 990–994, 2019.

[35] A. Nugraha, A. Liutkus, and E. Vincent, "Multichannel audio source separation with deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 9, pp. 1652–1664, 2016.

[36] A. Liutkus and F. R. Stöter, "sigsep/norbert," 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3269749

[37] N. Duong, E. Vincent, and R. Gribonval, "Underdetermined reverberant audio source separation using a full-rank spatial covariance model," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1830–1840, 2010.

[38] P. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Singing-voice separation from monaural recordings using deep recurrent neural networks," *In Proc. of the 15th Int. Society for Music Information Retrieval Conf. (ISMIR),* Taipei, Taiwan, pp. 477–482, 2014.

[39] E. Manilow, C. Hawthorne, C.-Z. Huang, B. Pardo, and J. Engel, "Improving Source Separation by Explicitly Modeling Dependencies Between Sources," *In Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP),* Singapore, 2022.

[40] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "MUSDB18 - a corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[41] F. R. Stöter, A. Liutkus, and N. Ito, "The 2018 Signal Separation Evaluation Campaign," *Lecture Notes in Computer Science*, vol. 10891, pp. 293–305, 2018.

[42] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 4, no. 14, pp. 1462–1469, 2006.

[43] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. Bello, "MedleyDB: A multitrack dataset for annotation-intensive MIR research," *In Proc. of the 15th Int. Society for Music Information Retrieval Conf. (ISMIR),* Taipei, Taiwan, pp. 155–160, 2014.

[44] "SigSep MUSDB Website," https://sigsep.github.io/datasets/musdb.html, accessed: 01-05-2022.

[45] A. Srinivasamurthy, S. Gulati, R. C. Repetto, and X. Serra, "Saraga: Open Datasets for Research on Indian Art Music," *Empirical Musicology Review*, 2020.

[46] E. Cano, D. Fitzgerald, and K. Brandenburg, "Evaluation of quality of sound source separation algorithms: Human perception vs quantitative metrics," *In Proc. of the 24th European Signal Processing Conf. (EUSIPCO),* Budapest, Hungary, pp. 1758–1762, 2016.

[47] M. Schoeffler, S. Bartoschek, F.-R. Stöter, M. Roess, S. Westphal, B. Edler, and J. Herre, "webmushra—a comprehensive framework for web-based listening tests," *Journal of Open Research Software*, vol. 6, no. 1, 2018.

A MODEL YOU CAN HEAR: AUDIO IDENTIFICATION WITH PLAYABLE PROTOTYPES

Romain Loiseau[1,2] **Baptiste Bouvier**[3] **Yann Teytaut**[3]
Elliot Vincent[1,4] **Mathieu Aubry**[1] **Loic Landrieu**[2]

[1] LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS, France
[2] LASTIG, Univ Gustave Eiffel, IGN, ENSG
[3] STMS Lab, UMR 9912 (IRCAM, CNRS, Sorbonne University), Paris, France
[4] INRIA and DIENS (ENS-PSL, CNRS, INRIA)

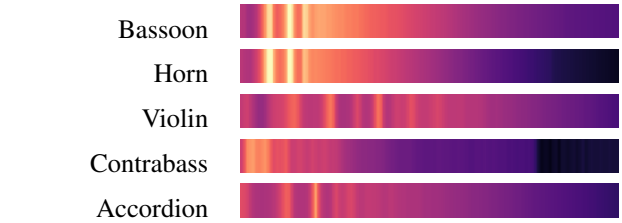
romain.loiseau@enpc.fr

ABSTRACT

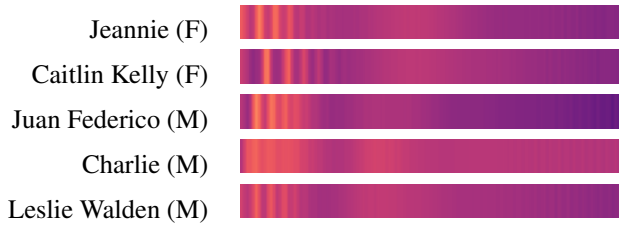
Machine learning techniques have proved useful for classifying and analyzing audio content. However, recent methods typically rely on abstract and high-dimensional representations that are difficult to interpret. Inspired by transformation-invariant approaches developed for image and 3D data, we propose an audio identification model based on learnable spectral prototypes. Equipped with dedicated transformation networks, these prototypes can be used to cluster and classify input audio samples from large collections of sounds. Our model can be trained with or without supervision and reaches state-of-the-art results for speaker and instrument identification, while remaining easily interpretable. The code is available at: https://github.com/romainloiseau/a-model-you-can-hear

1. INTRODUCTION

The emergence of deep learning approaches dedicated to audio analysis has led to significant performance improvements [1, 2]. Although these methods take sound excerpts as input, they typically rely on high-dimensional latent spaces, making the interpretation of their decisions difficult and limiting the insights gained on the considered data. Furthermore, such methods typically require large corpora of annotated sounds for supervision. We take a different approach, inspired by the recent Deep Transformation-Invariant (DTI) clustering [3] framework which learns prototypes in input space to analyze images or 3D point clouds [4]. Each prototype is associated with a set of transformations (*e.g.* rotations or morphological transformations) learned in the manner of Spatial Transformation Networks [5]. The DTI clustering model optimizes a reconstruction loss and associates each input with the prototype that provides the best reconstruction. We propose adapting this approach to the audio domain. The prototypes become log-scaled, Mel-frequency spectrograms, and their associated learnable transformations correspond to plausible sound alterations. We also introduce an additional loss based on cross-entropy in the supervised setting, which we demonstrate can boost the classification accuracy while preserving some interpretability.



(a) Instrument dataset SOL [6, 7]



(b) Speech dataset Librispeech [8]

Figure 1: **Playable Prototypes.** Our model rely on a small set of spectral prototypes that can be used for clustering and classification. We show examples of such prototypes learned from two datasets.

2. RELATED WORK

Audio Classification. Musical instrument and speaker identification are two of the standard tasks used to benchmark audio classification models. Although early musical instrument identification methods focused on distinguish-

© R. Loiseau, B. Bouvier, Y. Teytaut, E. Vincent, M. Aubry, and L. Landrieu. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** R. Loiseau, B. Bouvier, Y. Teytaut, E. Vincent, M. Aubry, and L. Landrieu, "A Model You Can Hear: Audio Identification with Playable Prototypes", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.*, Bengaluru, India, 2022.

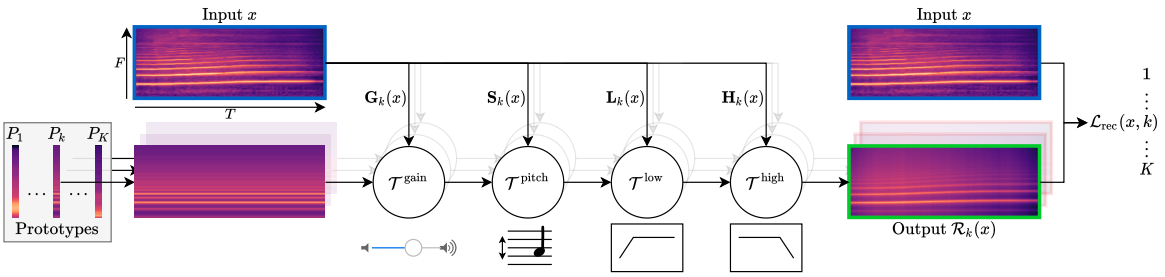


Figure 2: **Method overview.** Given an **input sound**, we predict for each prototype a *gain*, a *pitch* shift, as well as *low* and *high frequency filters* at each timestamp to generate the **output**. Prototypes and transformations are learned jointly using a reconstruction loss in either a supervised or unsupervised setting.

ing individual tones achieve appreciable precision, support vector machines operating on spectro-temporal sound representations can reach almost perfect accuracy. Similarly to supervised classification of instruments that play individual notes that could be considered solved [9, 10], speaker identification is mostly handled by convolutional and/or recurrent neural networks [11], which achieve almost perfect performances [12]. However, these models rely on complex and abstract latent representations that are not easily interpretable and cannot be visualized as spectrograms or heard in the audio domain.

Prototype-Based Methods. Auto-Encoders [13, 14] learn a compact latent representation of an input sample and are trained using a reconstruction loss. Using regularizations and constraints, the latent space can be adjusted for various tasks, such as classification [15]. However, the learned features remain largely non-interpretable and abstract. Inspired by the literature on images [16], the authors of [17] propose to generate prototypes in the latent space and analyze their similarity to the encoded input samples using a frequency-dependent measure. Although the latent prototypes can be decoded into input space, their role in class prediction remains obfuscated, as all similarities are mixed with a learned linear layer.

Transformation-Invariant Modeling. Deep transformation-invariant clustering [3] learns explicit prototypes in input space. Each prototype is equipped with dedicated transformation networks, allowing a small set of prototypes to faithfully represent a collection of samples. The resulting models can be used for downstream tasks such as classification [3], few-shot segmentation [4], and even multi-object instance discovery [18]. Jaderberg *et al*. [5] also proposes learning differentiable transformations in the input space, and feed the transformed data to a classification network. We propose to extend these ideas to the audio domain by learning prototypical log-Mel spectrograms along with adapted transformations.

3. METHOD

We consider a set of N audio clips $x_1, \cdots, x_N \in \mathbb{R}^{F \times T}$, each characterized by their log-Mel spectrogram with T time steps and a spectral resolution of F bins. The samples are annotated with class labels $y_1, \cdots, y_N \in \mathcal{Y}$ with $\mathcal{Y}$ a given class set $\{1, \cdots, K\}$. Our goal is to learn an interpretable model that can perform classification and clustering for the audio sample collection $x_1, \cdots, x_N$. Our model is based on prototypical spectrograms equipped with spectral transformation networks, which we present in Sec. 3.1. We propose an unsupervised training scheme in Sec. 3.2, and a supervised setting in Sec. 3.3. Finally, we provide details on parameterization and training of the model in Sec. 3.4.

3.1 Sound Reconstruction Model

Our model consists of K prototypes $P_k \in \mathbb{R}^F$ directly interpretable in the spectral domain, each equipped with a set of dedicated transformation networks (See Figure 2). Both prototypes and networks are jointly trained to reconstruct input samples. We can then use the reconstruction error as a measure of the compatibility between a sample and a given prototype.

Spectral Transformations Model. Even a single instrument or speaker cannot be meaningfully characterized by a single sound. In fact, they are typically capable of producing a rich and varied set of sounds. We propose equipping each prototype with a set of transformation networks that learn to change specific characteristics of the prototype, such as its amplitude and pitch, so that it can faithfully approximate a wide variety of samples. This also allows the prototypes to learn more subtle audio attributes, such as timbre or intonations.

Transformation networks associated to each prototype P_k take as input an audio sample $x \in \mathbb{R}^{F \times T}$ as input and predicts a spectral transformation for each time step t in $[1, T]$. For each prototype k, we propose a set of transformations specifically designed for the audio domain, illustrated in Figure 3:

• *Amplification.* Given an audio sample x, we predict a gain $\mathbf{G}_k(x) \in \mathbb{R}^T$ for all time step t. The transformation $\mathcal{T}_g^{\text{gain}}(m)$ adds the gain g to all frequencies of a log-Mel spectrogram $m \in \mathbb{R}^F$.

• *Pitch Shifting.* We map an audio sample x to a pitch shift $\mathbf{S}_k(x) \in \mathbb{R}^T$ for all time step t. The transformation $\mathcal{T}_s^{\text{pitch}}(m)$ shifts the log-Mel spectrogram $m \in \mathbb{R}^F$ by s.

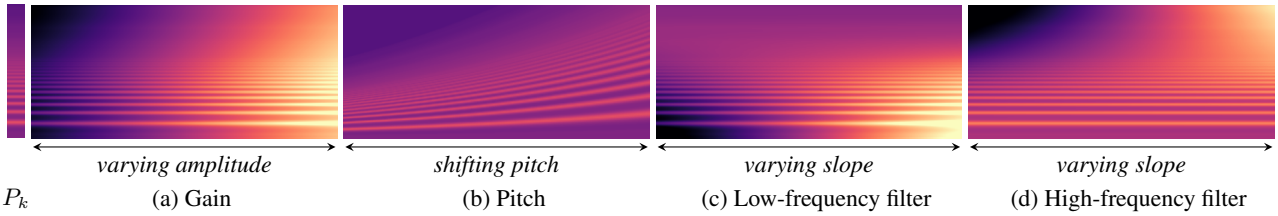


Figure 3: **Spectral Transformations.** Our proposed transformations learn to alter the gain (a), pitch (b) and frequency support (c,d) of a given prototype spectrogram P_k.

We use linear interpolation for handling non-integer shifts.

• *High- and Low-Frequency Filter.* Given an audio sample x, we define for each time t an affine low-frequency filter $\mathbf{L}_k(x)[t] \in \mathbf{R}^F$ by predicting its slope and cutoff frequency. Similarly, we predict a high-frequency filter $\mathbf{H}_k(x)[t] \in \mathbb{R}^F$. The transformations $\mathcal{T}_L^{\text{low}}$ and $\mathcal{T}_H^{\text{high}}$ simply add the corresponding filters L and H to a log-Mel spectrogram.

These transformations can be written as follows for a log-Mel spectrogram $m \in \mathbb{R}^F$ and a mel-frequency $f \in [1, F]$:

$$\mathcal{T}_g^{\text{gain}}(m)[f] = m[f] + g \tag{1}$$
$$\mathcal{T}_s^{\text{pitch}}(m)[f] = m[\Phi(f, s)] \tag{2}$$
$$\mathcal{T}_L^{\text{low}}(m)[f] = m[f] + L[f] \tag{3}$$
$$\mathcal{T}_H^{\text{high}}(m)[f] = m[f] + H[f], \tag{4}$$

with $\Phi(f, s) = A \log_{10}\left(1 + s\left(10^{f/A} - 1\right)\right)$ the mel-aware pitch shifting, with $A = 2595$ [19]. Our transformations have links with moment matching methods [20, 21]. In fact, modulating the amplitude, pitch, and spectral support of spectrograms is similar to matching the moment of zeroth order (amplitude gain), first order (pitch shift), and second order (frequency filters).

Reconstruction Model. For each prototype k and each time step t, we successively apply the gain, pitch shift, low-pass, and high-pass transformations in this specific order. The resulting reconstruction $\mathcal{R}_k(x)$ of an input sound x by a prototype P_k is defined as follows:

$$\mathcal{R}_k(x)[t] = \mathcal{T}^{\text{high}}_{\mathbf{H}_k(x)[t]} \circ \mathcal{T}^{\text{low}}_{\mathbf{L}_k(x)[t]} \circ \mathcal{T}^{\text{pitch}}_{\mathbf{S}_k(x)[t]} \circ \mathcal{T}^{\text{gain}}_{\mathbf{G}_k(x)[t]}(P_k), \tag{5}$$

where $\mathcal{R}_k(x)[t]$ denotes the t-th timestamp of the reconstruction, and $\circ$ denotes the composition of transformations. Note that the chosen transformations are constrained and limited on purpose, as we want each prototype to represent a restricted and consistent set of audio samples.

We measure the quality of the reconstruction $\mathcal{R}_k(x)$ of an input sample x by the k-th prototype as the spectro-temporal average of their ℓ_2 distance:

$$\mathcal{L}_{\text{rec}}(x, k) = \frac{1}{T} \sum_{t=1}^{T} \|x[t] - \mathcal{R}_k(x)[t]\|^2 . \tag{6}$$

Note that since this function is differentiable and continuous, it can be used as a loss function to train prototypes and transformation networks as defined in the next sections.

3.2 Unsupervised Training

We propose an unsupervised learning scheme in which we do not have access to labels $y_1, \cdots, y_N$ during training. Our goal is to cluster the samples $x_1, \cdots, x_N$ into meaningful groups of sounds that can be well approximated by the same prototype spectrogram and its associated transformations. We can train our model by minimizing the following loss:

$$\mathcal{L}_{\text{clustering}} = \sum_{n=1}^{N} \min_{k=1}^{K} \mathcal{L}_{\text{rec}}(x_n, k) . \tag{7}$$

This loss is the sum of the reconstruction errors with optimal cluster assignment and is a classical clustering objective used by K-means-based methods.

3.3 Supervised Training

In the supervised setting, each prototype and associated transformation networks are dedicated to a class and trained to reconstruct the samples from this class only. Once trained, the model predicts for an input x the class of the prototype with the best reconstruction.

To promote better class discrimination, we propose a dual reconstruction-prediction objective. We associate each input x with a probabilistic class prediction defined as the softmax of the negative reconstruction error between prototypes. For an input x of label y, we define the prediction loss $\mathcal{L}_{\text{ce}}$:

$$\mathcal{L}_{\text{ce}}(x, y) = -\log\left(\frac{\exp\left(-\beta \mathcal{L}_{\text{rec}}(x, y)\right)}{\sum_{k=1}^{K} \exp\left(-\beta \mathcal{L}_{\text{rec}}(x, k)\right)}\right), \tag{8}$$

where β is a learnable parameter. This loss encourages each prototype to specialize in reconstructing the samples of its associated class. To train our network, we use a weighted sum of both losses:

$$\mathcal{L}_{\text{classif}} = \sum_{n=1}^{N} \mathcal{L}_{\text{rec}}(x_n, y_n) + \lambda_{\text{ce}} \mathcal{L}_{\text{ce}}(x_n, y_n) , \tag{9}$$

with λ_{ce} an hyperparameter set to 0.01.

3.4 Parameterization and Training Details

Architecture. We implement the functions $\mathbf{G}_k$, $\mathbf{S}_k$, $\mathbf{L}_k$ and $\mathbf{H}_k$ as U-Net style neural networks [23] operating on

	SOL [6, 7]						LibriSpeech [8]					
	w/o supervision			w supervision			w/o supervision			w supervision		
	OA	AA	$\mathcal{L}_{\text{rec}}$	OA	AA	$\mathcal{L}_{\text{rec}}$	OA	AA	$\mathcal{L}_{\text{rec}}$	OA	AA	$\mathcal{L}_{\text{rec}}$
Raw prototypes	29.3	13.2	12.4	28.4	39.5	12.9	13.3	13.2	8.6	60.5	59.5	8.6
⌞ *gain* transformation	32.2	15.8	3.4	37.4	40.1	3.8	44.4	43.3	3.4	78.7	79.1	3.4
⌞ *pitch-shifting*	**37.6**	**19.1**	2.6	51.6	46.1	2.9	46.8	46.9	2.6	77.2	77.8	2.6
⌞ *frequency-filters*	33.4	14.6	**1.5**	55.1	47.7	**2.1**	**48.8**	**49.5**	**1.6**	88.8	89.4	**1.6**
⌞ cross-entropy loss	—	—	—	**98.9**	**94.5**	2.5	—	—	—	**99.9**	**99.9**	1.9

Table 1: **Ablation Study.** We show the impact of the increasing complexity of our modeling on datasets' validation sets.

	OA	AA
SOL [6, 7]		
Autoencoder + K-means [22]	28.7	12.3
† APNet [17] + K-means [22]	**37.3**	**18.2**
Ours w/o supervision	34.5	15.4
LibriSpeech [8]		
Autoencoder + K-means [22]	11.0	11.1
† APNet [17] + K-means [22]	36.3	36.4
Ours w/o supervision	**48.6**	**49.5**

Table 2: **Clustering Results.** Clustering performances on the test sets. †: Note that APNet requires labels at training time.

	OA	AA	$\mathcal{L}_{\text{rec}}$
SOL [6, 7]			
Direct Classification	97.8	94.8	—
APNet [17]	95.3	91.3	**0.1**
Ours w supervision	**99.3**	**95.8**	2.6
LibriSpeech [8]			
Direct Classification	99.4	99.5	—
APNet [17]	97.8	97.8	**0.2**
Ours w supervision	**99.9**	**99.9**	2.6

Table 3: **Classification Results.** Accuracy and reconstruction error computed on the test sets.

the log-Mel spectrograms $x \in \mathbb{R}^{T \times F}$. To save parameters, all networks share the same encoder, defined as a sequence of 2-dimensional spectro-temporal convolutions and pooling operations. This encoder produces a sequence of feature maps $z_0, \cdots, z_R$ with decreasing temporal and spectral resolutions $T/2^r \times F/2^r$. The spectral dimension of these maps is then collapsed using learned pooling operations: $\tilde{z}_r$ is of resolution $T/2^r$. Each decoder is then defined as a sequence of 1-dimensional temporal convolutions and unpooling operations. The spectral maps $\tilde{z}_0, \cdots, \tilde{z}_R$ are used through skip-connections in the decoders. The prototypes $P_1, \cdots, P_K$ and the inverse temperature β are directly trainable parameters of the model.

Curriculum Learning. We ensure the stability of our model by gradually increasing its complexity along the training procedure. First, we learn the raw prototypes without any transformation. Once the training loss does not decrease for 10 straight epochs, we add the gain transformation. We then add the pitch shift and spectral filters successively following the same procedure.

Implementation details. We use the ADAM [24] optimizer with a learning rate of 10^{-4}, and a weight decay of 10^{-6} for the transformation networks and 0 for β and the prototypes. Our model has 545k parameters with $K = 128$ prototypes. For comparison, the reconstruction model proposed by Zinemanas *et al*. [17] has 1.8M parameters.

4. EXPERIMENTS

We evaluate our approach in both supervised and unsupervised settings and on two datasets.

Datasets. We consider the following datasets:

• **SOL [6,7].** This dataset contains 24 450 samples of individual notes played with various playing techniques by 33 different instruments and sampled at 44.1kHz. We evaluate instrument classification.

• **Librispeech [8].** This 1000-hour corpus contains English speech sampled at 16kHz. We selected the 128 predominant speakers from the `train-clean-360` set. We evaluate the classification of speaker.

For both datasets, we randomly selected 70% of the clips for training, 10% for validation, and 20% for testing. We always use as many prototypes as the number of classes in the dataset.

Metrics. To assess the quality of classification models, we report the following metrics:

• *Overall Accuracy (OA).* Percentage of input samples correctly classified by our model, *i.e*., best reconstructed by the prototype assigned to their true class.

• *Average Accuracy (AA).* Average of the classwise accuracy, computed across classes without weights.

• *Reconstruction error ($\mathcal{L}_{rec}$).* To assess the quality of the reconstruction, we also report the reconstruction error $\mathcal{L}_{\text{rec}}$.

Methods trained in the unsupervised setting cluster the

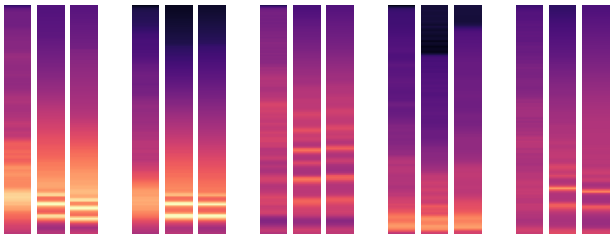


Figure 4: **Learned Meaningful Prototypes.** Comparison between the average spectrogram from one class (left), and the prototypes learned in the supervised setting without (middle) and with (right) cross-entropy loss. Our model can learn spectral representations that are crisper and more discriminative than a simple average.

audio samples instead of classifying them. We associate each cluster with the majority class of its assigned training samples. We can then predict for each test sample the class of its best-fitting cluster, allowing us to compute all the classification metrics.

Baselines. To put the performance of our method in context, we trained different baselines:

• *Classification.* We trained a temporal convolutional network to classify log-Mel spectrograms. We supervise this network with the cross-entropy on the labels. This network, which does not provide a reconstruction error, is called *Direct Classification*. We also evaluated the APNet [17] approach on our datasets. However, this method is limited in its interpretability, as it relies on latent prototypes and uses a fully connected layer to classify samples based on their similarity to the prototypes.

• *Clustering.* We trained a two-dimensional convolutional autoencoder without skip-connections to reconstruct log-Mel spectrograms. We supervised this network with the ℓ_2 norm between the input and the reconstruction. We then use K-means [22] on the innermost feature maps to cluster the samples. We also use K-means directly on the predicted similarities between feature maps and APNet prototypes [17]. Note this last approach requires to first train APNet in a supervised setting.

4.1 Quantitative Results

Reconstruction. As can be seen in Table 1, each successive addition of a transformation decreases the reconstruction error $\mathcal{L}_{\text{rec}}$ and improves the accuracy of classification. This shows that the curriculum training scheme allows our model to learn insightful and complementary spectral transformations. Supervised with the cross-entropy loss, our model reaches high accuracy, at the cost of a slightly higher reconstruction error. In fact, the additional objective not only encourages the reconstruction models $\mathcal{R}_k$ to provide a good reconstruction for the samples of their assigned class, but also a worse reconstruction for other classes.

Finally, we see that while adding frequency-filters yields better reconstruction, they hurt classification performance on SOL [6, 7] in the unsupervised setting. We hypothesize that the filters makes the transformation too expressive, reducing the specificities of the prototypes.

Clustering. As shown in Table 2, our method performs well compared to the baselines. For SOL, our method is affected by the class imbalance between samples, but still produces competitive results. In particular, note that our approach does not need any labels during training, in contrast to the APNet-based approach [17]. For LibriSpeech, our method produces convincing clustering results for speaker identification.

Classification. As shown in Table 3, our method reaches almost perfect precision for both datasets. While APNet [17] reaches better reconstruction scores, this method learns powerful transformations in an abstract latent space. This leads to a model that is more expressive but less interpretable. In contrast, we restrict the scope of the spectral transformations so that a prototype relates in a meaningful way to the samples that it reconstructs well. Yet, our model is still capable of producing faithful reconstructions, which would be suitable for audio generation tasks.

4.2 Interpretability

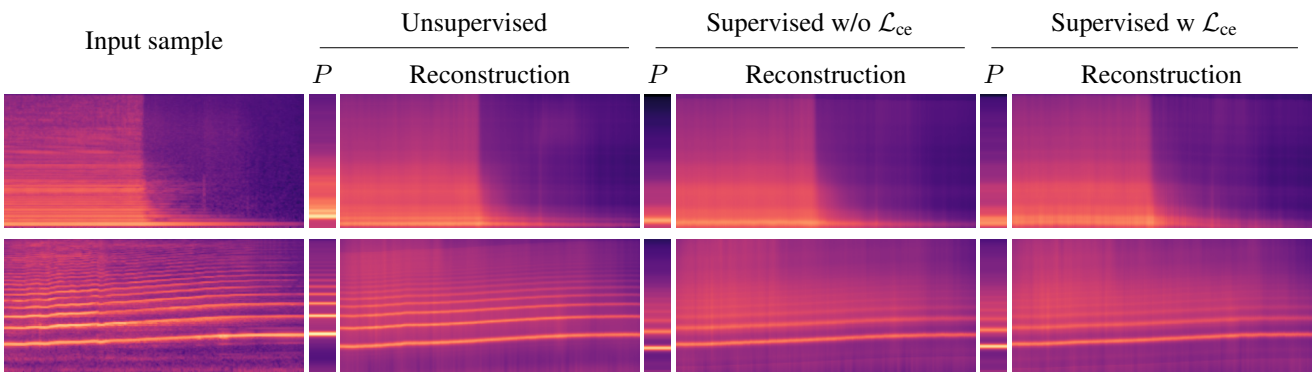
As shown in Figure 5, our model learns to reconstruct complex input samples (*e.g.* different notes, techniques, or voices) by automatically adjusting the amplitude, pitch, and spectral support of spectral prototypes. As seen on Figure 4 and Figure 1, these prototypes capture rich spectral characteristics such as timbre, and can be interpreted as an "acoustical fingerprint". Furthermore, as the prototypes are given in the spectral domain, they can be easily played and analyzed. Audio examples featuring the learned prototypes are provided in the supplementary materials.

In Figure 6, we represent the reconstruction of sounds played by various instrument with different prototypes. While the reconstructions are clearly more faithful when using the instruments' dedicated prototype, this opens the perspective for further MIR application such as interpretable timbre transfer.

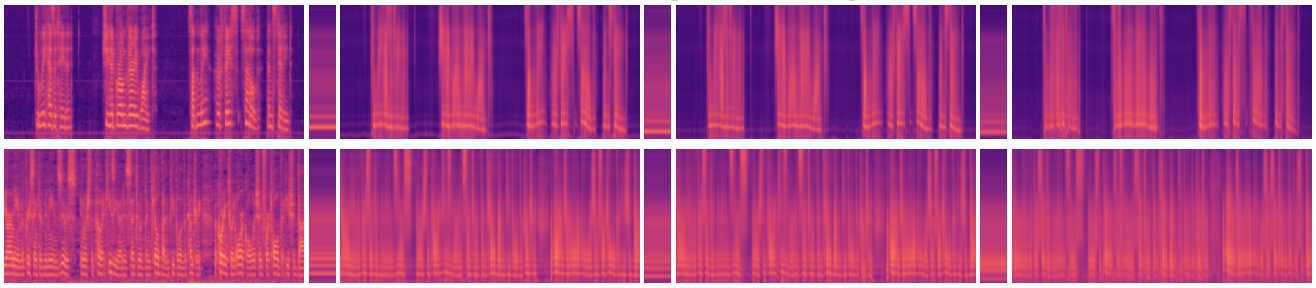
5. CONCLUSION

We presented a novel approach to audio understanding by representing large collections of audio clips with few spectral prototypes equipped with learned transformation networks. Our model produces concise, expressive, and interpretable representations of raw audio excerpts that can be used to obtain state-of-the-art results for audio classification and clustering tasks.

Acknowledgements. This work was supported in part by ANR project READY3D ANR-19-CE23-0007 and was granted access to the HPC resources of IDRIS under the allocation 2022-AD011012096R1 made by GENCI. We thank Theo Deprelle, Nicolas Gonthier, Tom Monnier and Yannis Siglidis for inspiring discussions and feedbacks.



(a) **SOL [6, 7].** Reconstructions of an audio clip of contrabass (top) and clarinet (bottom).



(b) **LibriSpeech [8].** Reconstructions of an audio clip of C. Nendza (F) (top) and C. Berriux (M) (bottom) speaking.

Figure 5: **Reconstruction of Input Sounds.** For each input sample, we show the reconstruction provided by different settings of our model as well as the corresponding prototype P used for reconstruction. Note how the timbre of the speaker or instrument is well represented while the inputs fluctuates in pitch and intensity. This leads to an insightful characterization of each instrument or speaker.

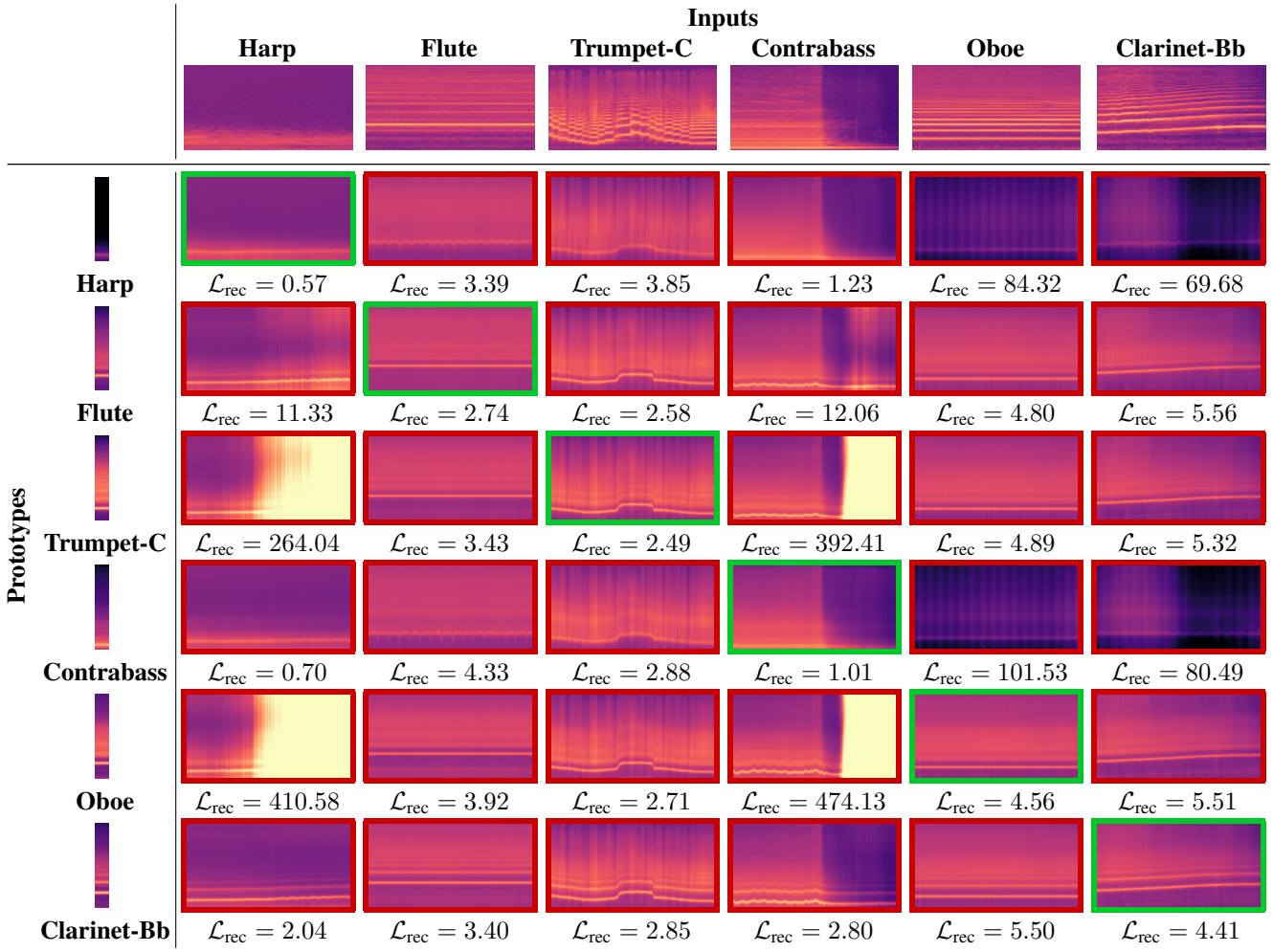


Figure 6: **Reconstructions from Different Prototypes.** We show the reconstructions of input samples (columns) from SOL [6, 7] by learned prototypes (lines) in the supervised setting without cross-entropy loss. The prototype of the corresponding instruments provides the lowest reconstruction error (green frame).

6. REFERENCES

[1] J. Abeßer, "A review of deep learning based methods for acoustic scene classification," *Applied Sciences*, 2020.

[2] N. Ndou, R. Ajoodha, and A. Jadhav, "Music genre classification: A review of deep-learning and traditional machine-learning approaches," in *IEMTRONICS*, 2021.

[3] T. Monnier, T. Groueix, and M. Aubry, "Deep Transformation-Invariant Clustering," in *NeurIPS*, 2020.

[4] R. Loiseau, T. Monnier, M. Aubry, and L. Landrieu, "Representing Shape Collections with Alignment-Aware Linear Models," in *3DV*, 2021.

[5] M. Jaderberg, K. Simonyan, and A. Zisserman, "Spatial transformer networks," *NeurIPS*, 2015.

[6] G. Ballet, R. Borghesi, P. Hoffmann, and F. Lévy, "Studio online 3.0: An internet "killer application" for remote access to IRCAM sounds and processing tools," in *Journées d'Informatique Musicale*, 1999.

[7] C. E. Cella, D. Ghisi, V. Lostanlen, F. Lévy, J. Fineberg, and Y. Maresz, "OrchideaSOL: a dataset of extended instrumental techniques for computer-aided orchestration," *ICMC*, 2020.

[8] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based on public domain audio books," in *ICASSP*, 2015.

[9] D. Bhalke, C. Rao, and D. S. Bormane, "Automatic musical instrument classification using fractional fourier transform based-MFCC features and counter propagation neural network," *Journal of Intelligent Information Systems*, 2016.

[10] V. Lostanlen, J. Andén, and M. Lagrange, "Extended playing techniques: the next milestone in musical instrument recognition," in *International Conference on Digital Libraries for Musicology*, 2018.

[11] Z. Bai and X.-L. Zhang, "Speaker recognition based on deep learning: An overview," *Neural Networks*, 2021.

[12] F. Ye and J. Yang, "A deep neural network model for speaker identification," *Applied Sciences*, 2021.

[13] N. Tits, F. Wang, K. E. Haddad, V. Pagel, and T. Dutoit, "Visualization and interpretation of latent spaces for controlling expressive speech synthesis through audio analysis," *Conference of the International Speech Communication Association*, 2019.

[14] F. Roche, T. Hueber, S. Limier, and L. Girin, "Autoencoders for music sound modeling: a comparison of linear, shallow, deep, recurrent and variational models," *Sound and Music Computing*, 2018.

[15] J. Naranjo-Alcazar, S. Perez-Castanos, P. Zuccarello, F. Antonacci, and M. Cobos, "Open set audio classification using autoencoders trained on few data," *Sensors*, 2020.

[16] O. Li, H. Liu, C. Chen, and C. Rudin, "Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions," in *AAAI*, 2018.

[17] P. Zinemanas, M. Rocamora, M. Miron, F. Font, and X. Serra, "An interpretable deep learning model for automatic sound classification," *Electronics*, 2021.

[18] T. Monnier, E. Vincent, J. Ponce, and M. Aubry, "Unsupervised Layered Image Decomposition into Object Prototypes," in *ICCV*, 2021.

[19] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, 1937.

[20] H. Tamaru, Y. Saito, S. Takamichi, T. Koriyama, and H. Saruwatari, "Generative moment matching network-based random modulation post-filter for dnn-based singing voice synthesis and neural double-tracking," in *ICASSP*, 2019.

[21] M. Andreux and S. Mallat, "Music generation and transformation with moment matching-scattering inverse networks," in *ISMIR*, 2018.

[22] L. Bottou and Y. Bengio, "Convergence properties of the k-means algorithms," in *NeurIPS*, 1995.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *MICCAI*, 2015.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

AN EXPLORATION OF GENERATING SHEET MUSIC IMAGES

Marcos Acosta
Harvey Mudd College
mdacosta@g.hmc.edu

Irmak Bukey
Pomona College
ibab2018@mymail.pomona.edu

TJ Tsai
Harvey Mudd College
ttsai@g.hmc.edu

ABSTRACT

Many previous works in recent years have explored various forms of music generation. These works have focused on generating either raw audio waveforms or symbolic music. In this work, we explore the feasibility of generating sheet music images, which is often the primary form in which musical compositions are notated for other musicians. Using the PrIMuS dataset as a testbed, we explore five different sequence-based approaches for generating lines of sheet music: generating sequences of (a) pixel columns, (b) image patches, (c) visual word tokens, (d) semantic tokens, and (e) XML-based tags. We show sample generated images, discuss the practical challenges and problems with each approach, and give our recommendation on the most promising paths to explore in the future.

1. INTRODUCTION

This work explores the feasibility of generating music in the form of sheet music images. Below, we provide a brief overview of recent work in the area of music generation, motivate the problem of generating sheet music images, and describe five sequence-based approaches that we will explore in this study. Our main goal is to identify the practical challenges and problems with each of the five approaches and to provide a recommendation on the most promising paths to explore in the future.

Music generation has been an active research topic of interest to the MIR community in recent years. Many recent works in this area fall into one or more of the following three research thrusts. The first research thrust is controllable music generation, in which a user can control certain aspects of the music generation process. Some examples of this include providing a template from which variations can be generated [1] [2], providing a video to which a suitable background music track is generated [3], specifying lyrics for the generated song [4], or disentangling different characteristics of music like style & melody [5], pitch & rhythm [6] [7], or structure & content [8]. A second research thrust is generating music with realistic short-term and long-term structure. Most recent works achieve this by representing music as a sequence of discrete tokens and using Transformer-based architectures to learn long-term dependencies. Some prominent examples of this include OpenAI's Jukebox model [4] and Google Magenta's Music Transformer model [9] [10]. Other works focus on encoding music in a way that allows the generated music to exhibit appropriate metrical structure at the beat, bar, and phrase levels [11]. A third research thrust is to generate music in different contexts and musical representations. Some examples include multi-part music [12], multi-track music [13] [14] [15] [16], guitar tabs [17], and jazz lead sheets [18]. Much of this work focuses on how to encode different music representations or different aspects of music (like pitch, duration, velocity, and timing [11]) in a form that is suitable for training with a Transformer model.

This work falls into the last group and explores music generation in a different musical representation: sheet music images. To the best of our knowledge, this is the first systematic study of generating sheet music images.

Why generate music in the form of sheet music images? We present four reasons. First, sheet music is the predominant medium by which compositions are shared in many genres of music. If a piece of music is intended to be a composition (as opposed to a demo for an academic research paper), the expectation is that the composition will be in the form of sheet music. Second, sheet music contains important musical information that is not contained in MIDI files (which is the most common symbolic music representation). For example, musical symbols in the sheet music like bar lines, rests, note ties, phrasing markings, sharps and flats, and key changes may not appear in a generated MIDI file in any explicit way. Third, generating sheet music cleanly decouples the musical composition and expressive performance aspects of music, allowing us to focus exclusively on the composition problem. Fourth, there is an abundance of sheet music data (>650k scores) available to the research community through the International Music Score Library Project (IMSLP) [19]. For the above reasons, we believe that studying the generation of sheet music images is an interesting and relevant problem.

Our goal is to explore the feasibility of generating sheet music. As an initial step, we focus on the specific task of generating monophonic incipit images, which are short snippets of sheet music that are used to identify a musical work. Using the PrIMuS dataset [20] as a testbed, we explore five different sequence-based approaches for generating incipits: (a) generating sequences of pixel columns, (b) generating sequences of image patches, (c) generating sequences of visual word tokens, (d) generating sequences

© M. Acosta, I. Bukey, and T. Tsai. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. Acosta, I. Bukey, and T. Tsai, "An Exploration of Generating Sheet Music Images", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

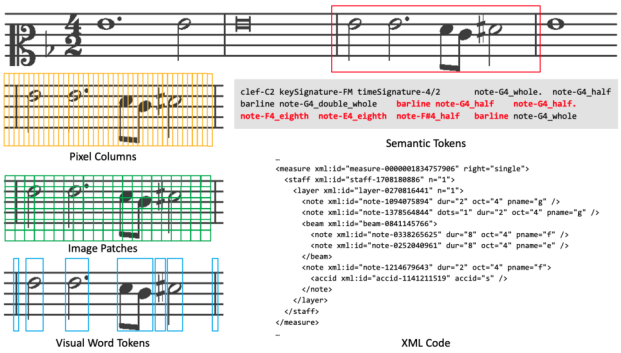


Figure 1. An example incipit from the PrIMuS dataset (top). We explore five different ways to generate sheet music: generating pixel columns, image patches, visual word tokens, semantic tokens, and XML code.

of semantic tokens, and (e) generating MEI code, an XML-based sheet music representation. For each of these five approaches, we train GPT-2 [21] and AWD-LSTM [22] language models, generate example incipits, and characterize the main challenges and problems with each approach.

2. SYSTEM DESCRIPTIONS

In this section we describe five different sequence-based approaches for generating a monophonic incipit image. All five approaches use language models to generate sequences of tokens, but they differ in the way that the tokens are constructed or defined.

2.1 Generating Pixel Columns

The first approach is to generate sequences of pixel columns. This is done in five steps, which are described in the following five paragraphs.

The first step is to standardize the incipits. The raw incipits in the PrIMuS dataset have variable height and width, so it is necessary to standardize the height of the incipit. This is done by selecting a fixed height for all incipits, ensuring that the five staff lines are vertically centered in the resulting image. This ensures that the staff lines appear in the same vertical pixel positions in each image. After the standardization, the images have a fixed height but variable width.

The second step is to convert pixel columns to words. Here, we interpret each standardized image as a sequence of words, where each word corresponds to a single pixel column. We convert pixel columns to words by simply binarizing the (grayscale) pixel values and interpreting each column's binary representation as a word.

The third step is to train a language model on the word sequences. This can be done in a self-supervised manner by predicting the next word in a sequence. We ran experiments with two different language models: (a) the AWD-LSTM model [22], which is a 3-layer LSTM model that incorporates multiple forms of dropout and (b) the GPT-2 small model [21], which is a 6-layer Transformer decoder model. We use the implementation of AWD-LSTM from the fastai [23] library and the implementation of GPT-2 from the huggingface [24] library. We use the default recommended vocabulary size in each respective library (GPT-2 small 30k, AWD-LSTM 60k), and map infrequently occurring words to a special unknown word token <unk>.

The fourth step is to generate new word sequences given a starting seed sequence. We do this by generating a new token at each time step and using the generated token as input to the next time step. For AWD-LSTM, we sample from the softmax distribution with a temperature of 0.8 (default in fastai). For GPT-2, we use top $K = 50$ sampling (default in huggingface).

The fifth step is to render the generated word sequence as an image. Since each word is a binary string of 0's and 1's corresponding to a single pixel column, we simply concatenate the generated words in a matrix and directly render it as a black-and-white image.

2.2 Generating Image Patches

The second approach is to generate sequences of image patches. This is done in five steps, which are described in the following five paragraphs.

The first step is to standardize the incipits. This is done in the same manner as described in Section 2.1 above.

The second step is to convert image patches to words. Figure 1 shows a graphical illustration of this process for a single measure (green squares at left). Instead of using pixel columns as words, we can instead use 15×15 square image patches as words. This allows individual words to retain context in both the vertical and horizontal dimensions, rather than only along the vertical dimension. This approach has been shown to work effectively for image classification in the Vision Transformer model [25]. After breaking the incipit image into square patches, we form a sequence by considering patches from top to bottom and from left to right. Each image patch is binarized and interpreted as a single word.

The third step is to train a language model on the word sequences. This is done in the same manner as before. We tried training both with and without special <col> tokens indicating the end of each column of patches but found that including the <col> tokens did not help.

The fourth step is to generate new word sequences given a starting seed sequence. We generate a sequence of words autoregressively as described previously.

The fifth step is to render the generated word sequence as an image. Each word is a binary string of length $15 \times 15 = 225$ and can be directly reshaped into a binary image patch. The image patches are assembled in the same rasterized order to generate the incipit image.

2.3 Generating Visual Word Tokens

The third approach is to generate sequences of visual word tokens. This is done in five steps, which are described in the following five paragraphs.

The first step is to standardize the incipits. This is done in the same manner as described in Section 2.1.

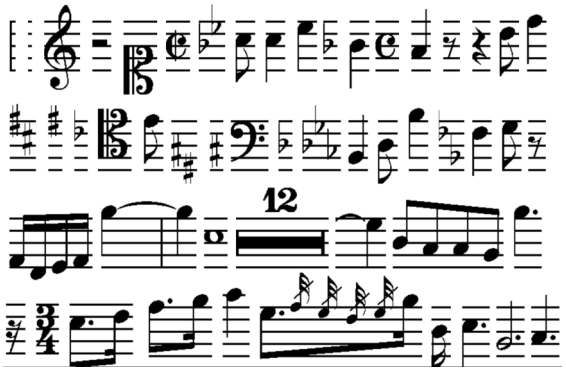


Figure 2. Visual word tokens. The top two rows show some of the most common visual word tokens, and the bottom two rows show a random selection of words in the vocabulary.

The second step is to tokenize each standardized incipit into a sequence of visual word tokens. This can be done by identifying pixel columns that only contain empty staff lines, and then interpreting these pixel columns as a separator (similar to tokenizing text based on whitespace). Figure 1 shows an example of this process for a single measure (bottom left). With the PrIMuS dataset, there are only a few unique "empty" pixel columns, but with other sheet music data one could train a simple model to identify empty pixel columns. Note that this process yields visual word tokens that span the entire height of the incipit and have variable width. The top two rows of Figure 2 show some of the most common visual word tokens, and the bottom two rows show a random selection of other visual word tokens in the vocabulary. One benefit of this approach is that entire symbols (e.g. clef signs) are interpreted as a single word, which allows the model to focus on modeling sequences of symbols rather than on rendering common symbols correctly. We found it helpful to impose a minimum separator width to avoid splitting certain symbols (like single whole notes) into two parts.

The third step is to train a language model on the visual word token sequences. This is done in the same manner as described in Section 2.1. The separator pixel columns are removed from the data and do not appear in training.

The fourth step is to generate new word sequences given a starting seed sequence. We generate a sequence of visual word tokens autoregressively as described in Section 2.1.

The fifth step is to render the generated visual word token sequence as an image. For simplicity, we simply concatenate the visual word tokens side by side and insert a fixed-width separator between each visual word token.

2.4 Generating Semantic Tokens

The fourth approach is to generate a sequence of semantic tokens. This is done in four steps, which are described in the next four paragraphs.

The first step is to represent each incipit as a sequence of semantic tokens. In the original PrIMuS dataset [20], the creators introduce a language for encoding musical symbols in a way that captures semantically meaningful information. Figure 1 shows an example incipit (top) and its corresponding sequence of semantic tokens (gray colored box on right). We tokenize by symbol and by symbol characteristics, so that a string "note-E4_eighth rest-eighth" would be converted into the sequence "note E4 eighth rest eighth". After tokenizing in this manner, there are 279 unique words across the PrIMuS dataset.

The second step is to train a language model on the semantic token sequences. This is done in the same manner as described in Section 2.1.

The third step is to generate new word sequences given a starting seed sequence. We generate semantic word tokens autoregressively as described in Section 2.1.

The fourth step is to render the generated semantic token sequence as an image. We first convert the semantic token sequence into Plaine and Easie [26] code using a relatively simple set of rules and logic. We then render the Plaine and Easie code as an image using Verovio [27].

2.5 Generating XML Code

The fifth approach is to generate MEI [28] code, which is a way of encoding music notation in XML form. This is done in four steps, which are described in the following four paragraphs.

The first step is to preprocess the MEI representation into a sequence of words. Figure 1 shows a portion of the MEI representation (bottom right) that encodes the measure highlighted in red (top right). To simplify the data, we filter out information that is not needed for the engraving process, such as headers, footers, and XML identifiers. We also insert spaces to keep conceptually distinct tokens separate. For example, a tag "<staff n='1'>" would be tokenized into the sequence "<staff", "n=", "'1'", and ">". Tokenizing the MEI representations in this manner, we get a vocabulary of 769 words across the PrIMuS dataset.

The second step is to train a language model on the MEI token sequences. This is done in the same manner as described in Section 2.1.

The third step is to generate new word sequences given a starting seed sequence. We generate a sequence of MEI tokens autoregressively until a closing </music> tag is generated. Note that, unlike the previous four methods, we cannot directly control the width of the generated incipit since the nested XML tags need to be properly closed.

The fourth step is to render the generated MEI representation as an image using Verovio. Note that the generated MEI code may not be a properly formatted XML document, so this rendering process may or may not be successful.

3. EXPERIMENTAL RESULTS

In this section we present sample incipits generated by using all five approaches described in Section 2. We focus on qualitative analysis of the generated images in this section, and in Section 4 we perform several quantitative analyses on the two most promising approaches.

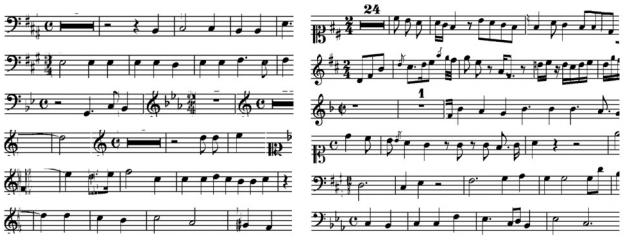


Figure 3. Sample images from generating pixel columns with AWD-LSTM (left) and GPT-2 (right).

All systems described in Section 2 were trained on the PrIMuS dataset [20]. It contains 87,678 real-music incipits, which are sequences of notes (often the first ones) used to identify a melody or musical work. Each incipit in the dataset is available in five different formats: (1) Plaine and Easie code, (2) a rendered image, (3) an MEI representation, (4) a sequence of semantic tokens (as described in Section 2.4), and (5) a sequence of agnostic tokens, which encodes the graphical symbols in the music score with their position in the staff but without any musical meaning. This dataset is ideal for our study because the incipits are short, there are a large number of incipits, and there are multiple representations of the data. We use the rendered image for the first three approaches (pixel columns, image patches, visual word tokens), the semantic tokens for the fourth approach (semantic tokens), and the MEI representation for the fifth approach (XML code).

During language model training, we use a 90-10 train-validation split for AWD-LSTM and a 95-5 train-validation split for GPT-2. Once the language model training has converged, we use the model to generate sample incipits as described in Section 2. Below, we present sample images generated from each of the five approaches and comment on the problems with each approach.

Figure 3 shows sample images from generating pixel columns. The left half of the figure shows images generated from the AWD-LSTM model, and the right half of the figure shows images generated from the GPT-2 model. We can immediately see several issues with the generated images: there are visual artifacts that appear as vertical lines (particularly with the AWD-LSTM model), the measures do not have a consistent number of beats, there are invalid key signatures (top left incipit), and the AWD-LSTM model seems to reset the clef and key/time signatures frequently. On the positive side, this approach is able to model symbols like bar lines, grace notes, clefs, key signatures, time signatures, and multiple bar rests in a single unified framework.

Figure 4 shows sample images from generating image patches. The left half of the figure shows images generated from the AWD-LSTM model, and the right half of the figure shows images generated from the GPT-2 model. We see some visual artifacts that come from incoherently generated symbols (e.g. the time signature in the fourth example on the right, the sixteenth note beams in the last example on the right), the measures do not have a consistent number of beats, and occasionally the model seems to

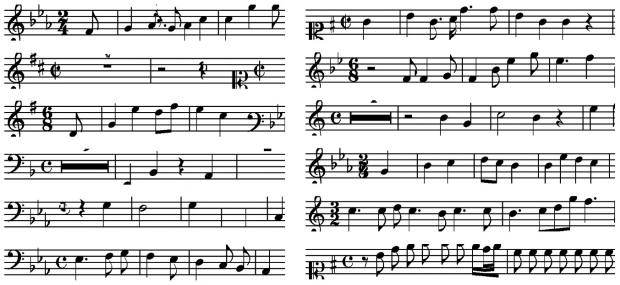


Figure 4. Sample images from generating image patches with AWD-LSTM (left) and GPT-2 (right).

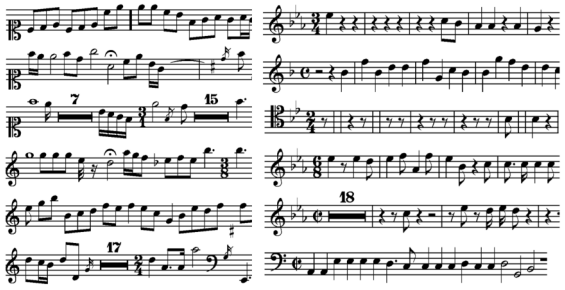


Figure 5. Sample images from generating visual word tokens with AWD-LSTM (left) and GPT-2 (right).

be confused about where each image patch is located relative to the global image (e.g. the incorrectly placed bar lines in the fifth example on the left). Again, the AWD-LSTM model seems to have a problem with resetting the clef and key signature too frequently (e.g. the second and third examples on the left). Compared to the pixel column approach, this approach seems to have somewhat less variety in terms of rhythms and durations (e.g. sixteenth notes, rests).

Figure 5 shows sample images from generating visual word tokens. The left half of the figure shows images generated from the AWD-LSTM model, and the right half of the figure shows images generated from the GPT-2 model. We don't see any visual artifacts in reconstructing symbols since each token is itself an entire symbol (or collection of symbols), but we see a lot of semantically incoherent patterns. For example, there are accidentals that appear without a corresponding note (fifth example on left), ties to notes that don't appear (second example on left), and bar lines sometimes appear too frequently (third example on right) and at other times too infrequently (last three examples on left). Measures still do not have a consistent number of beats, but we do notice that the model is often metrically coherent and correct immediately after a time signature symbol is generated. There are stylistic issues as well: some measures are technically coherent but do not follow typical conventions of sheet music, such as generating three consecutive quarter note rests in a measure rather than a whole measure rest (top example on right).

Figure 6 shows sample images from generating semantic tokens. The left half of the figure shows images generated from the AWD-LSTM model, and the right half of

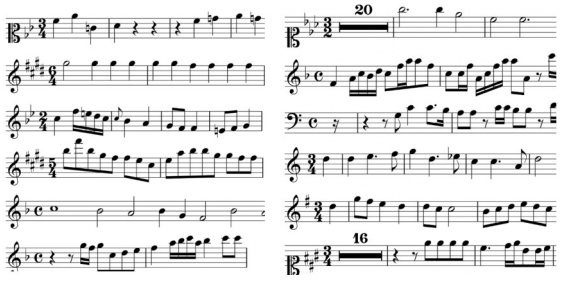


Figure 6. Sample images from generating semantic tokens with AWD-LSTM (left) and GPT-2 (right).

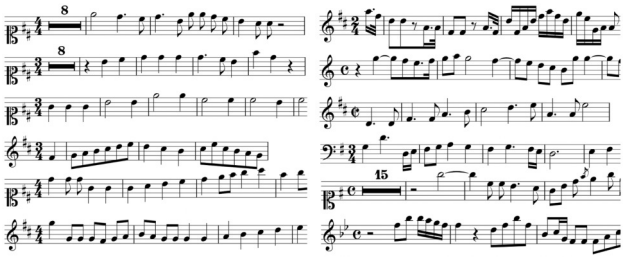


Figure 7. Sample images from generating XML code with AWD-LSTM (left) and GPT-2 (right).

the figure shows images generated from the GPT-2 model. There is a noticeable improvement in semantic coherence compared to the previous three approaches. For example, most generated measures have the correct number of beats, even across different time signatures (e.g. 3/4 in the first example on left, 6/4 in the second example on left, 2/4 in the third example on left, 4/4 in last example on left). However, we still occasionally see measures with an incorrect number of beats (e.g. fifth example on left), and the model particularly seems to struggle with unusual time signatures (e.g. 5/4 in the fourth example on left). The models seem to display a good amount of diversity in rhythms and durations, both with notes and rests. Note that, because the incipit is translated from semantic tokens to Plaine and Easie code and then to a rendered image, accidentals are guaranteed to be rendered correctly with a given key signature (e.g. first and third examples on left).

Figure 7 shows sample images from generating MEI code. The left half of the figure shows images generated from the AWD-LSTM model, and the right half of the figure shows images generated from the GPT-2 model. Note that some generated MEI code was ill-formed and could not be rendered as an image, so the examples below are self-selected from the examples that were valid MEI files. We can see that most measures have the correct number of beats, and this holds true across a range of time signatures (e.g. 4/4, 3/4, 2/4, 2/2 in the examples in Figure 7). Because the MEI representation explicitly encodes symbols like note beams and ties, the generated incipits also seem to obey most stylistic conventions. The AWD-LSTM model has noticeably less diversity than the GPT-2 model, but the latter demonstrates a reasonably good diversity of rhythms and durations in notes and rests.

Summary. The three image-based approaches (pixel columns, image patches, visual word tokens) are not recommended because the generated incipits are not semantically coherent, even when they do not have any obvious visual artifacts. The primary issue with generating semantic tokens and MEI code is metrical coherence – ensuring that the number of beats in a measure is consistent with the time signature. Generating MEI code also has the issue of malformed XML code that cannot be rendered properly. [1]

Our experiments lead us to two main conclusions. First, the two approaches that we believe are most promising to explore in future work are generating sequences of semantic tokens and generating XML code. Second, the main technical issue that must be resolved to generate meaningful sheet music is metrical coherence – exploring ways to utilize powerful and flexible language models while also conforming to the strict metrical rules of music. This emerges as a main challenge for future work.

4. ANALYSIS

In this section we perform additional quantitative analyses on the two recommended approaches: generating semantic tokens and generating MEI code.

We can quantify how metrically coherent the generated semantic token sequences are in the following manner. First, we generate 16,000 incipits in the form of semantic token sequences. When generating these sequences, we specify the clef, key signature, and time signature as part of the seed sequence, and we divide the 16,000 incipits evenly across eight different time signatures. Second, we segment each incipit into distinct measures by detecting bar lines in the generated sequence. If the generated sequence contains a change of time signature, we only consider the portion of the incipit before the time signature change and ignore the rest. This policy allows us to measure and compare metrical coherence across different time signatures. Third, we calculate the fraction of measures that are metrically coherent, which we define as a measure that contains the correct number of beats specified by the time signature. We can determine this based on the time signature and the time duration of each semantic token. Note that some semantic tokens like slurs, bar lines, grace notes, and clef symbols have no duration, while all notes and rests have a non-zero duration.

Table 1 shows the results of this analysis on the generated semantic token sequences. The leftmost column shows the different time signatures that we used as starting seeds. The next two columns show the total number of measures generated by the AWD-LSTM model and the percentage of those generated measures that are metrically coherent. Note that each incipit will have a variable number of measures, so we can only indirectly control the total number of generated measures. The last two columns show the same information for the GPT-2 model. The bottom row shows a total across all time signatures.

[1] One caveat is that our recommendations assume that the amount of training data is the same across different encodings.

Time	AWD-LSTM		GPT-2	
Signature	# meas	% coh	# meas	% coh
2/4	5820	75.2	8794	80.9
3/4	6229	79.5	8191	79.4
4/4	4109	77.6	5936	77.9
5/4	3762	0.3	5276	35.1
6/4	3640	70.4	5553	77.0
7/4	3781	0.05	5947	0.4
3/8	7091	77.9	10069	77.0
6/8	4334	75.8	7303	75.7
Total	45982	61.7	57069	66.0

Table 1. Measuring metrical coherence of generated semantic token sequences. Columns indicate the number of generated measures and the percentage of measures that have the correct number of beats.

Time	LSTM		GPT-2	
Signature	# meas	% coh	# meas	% coh
2/4	4754	35.2	6650	85.4
3/4	4998	69.4	6098	86.9
4/4	4418	74.1	4810	77.8
5/4	5400	3.2	4391	9.0
6/4	4298	13.4	3519	59.5
7/4	5586	2.4	3667	8.3
3/8	4120	22.5	7060	91.5
6/8	4354	81.1	5048	83.0
Total	37928	36.3	41243	68.3

Table 2. Measuring metrical coherence of generated MEI code.

There are two things to notice about Table 1. First, both models have very low metrical coherence for uncommon time signatures like 5/4 and 7/4. For example, less than 1% of measures in 7/4 were metrically coherent in both models, which is likely a reflection of the fact that this time signature has little or no representation in the PrIMuS dataset. Clearly, these language models are not learning a notion of metrical structure and are unable to generalize to unseen or uncommon time signatures. Second, the metrical coherence measures for GPT-2 and AWD-LSTM are very similar for common time signatures like 3/4, 4/4, 3/8, and 6/8, with numbers falling somewhere between 75-80%. For less common time signatures, however, GPT-2 often significantly outperforms AWD-LSTM, suggesting that it is able to generalize slightly better.

We measure the quality of the generated MEI code in two different ways. The first measurement is simply the percentage of generated incipits that are valid XML documents. Note that every MEI document begins with a <music> tag and ends with a corresponding </music> tag. Therefore, when generating MEI code, we autoregressively generate tokens until a closing </music> tag is encountered. The resulting MEI document is then rendered with Verovio into an SVG file. If the rendering process generates an error or warning, we consider the MEI code to be malformed. We generated 16,000 incipits with each model (2,000 incipits for each of the 8 time signatures shown in Table 1) and found that 93.1% of the AWD-LSTM incipits and 95.9% of the GPT-2 incipits were valid.

The second way to measure the quality of generated MEI code is to quantify metrical coherence. This is done in a similar manner as described above: we generate 2,000 incipits for each of the same eight time signatures, segment each incipit into measures by detecting bar lines, and then calculate the percentage of measures that are metrically coherent. Note that some of the incipits will not be valid XML documents, so in our analysis we only consider measures from generated documents that are valid.

Table 2 shows the results of this analysis on the generated MEI incipits. There are two things to notice. First, we see that GPT-2 outperforms AWD-LSTM on metrical coherence for every time signature, sometimes by very large margins. For instance, on incipits with a 3/8 time signature, 91.5% of the measures generated by GPT-2 are metrically coherent, compared to only 22.5% for AWD-LSTM. Second, comparing metrical coherence between semantic tokens (Table 1) and MEI (Table 2), we see that AWD-LSTM performs substantially worse with MEI representations across all time signatures (61.7% vs 36.3% total). GPT-2 seems to perform better with MEI on common time signatures, but has extremely variable performance with uncommon time signatures.

Our quantitative analyses have a clear takeaway: metrical coherence is the primary issue with both approaches. The percentage of metrically coherent measures is sufficiently low as to be prohibitive, particularly with less common time signatures. It is clear that simply having more data will not solve this problem, since having more data will still have an extreme imbalance across different time signatures. Some possible solutions include using rejection sampling or incorporating timing and position information either into the Transformer model as additional embeddings or into the data itself (e.g. REMI [11]). We identify this as a main challenge for future work.

5. CONCLUSION

We explore the feasibility of generating monophonic sheet music images using sequence-based models. We consider five different approaches: generating sequences of pixel columns, image patches, visual word tokens, semantic tokens, and XML-like tags. We train language models for all five approaches on the PrIMuS dataset and generate sample incipits to better understand the pros and cons of each approach. Our main findings are that: (a) the image-based approaches can yield realistic looking sheet music but are often semantically incoherent and are therefore not recommended, (b) generating semantic tokens and MEI code yield the most semantically coherent sheet music, and (c) metrical coherence is the biggest issue that needs to be addressed in future work.

6. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 2144050.

7. REFERENCES

[1] D. von Rütte, L. Biggio, Y. Kilcher, and T. Hoffman, "FIGARO: Generating symbolic music with fine-grained artistic control," *arXiv preprint arXiv:2201.10936*, 2022.

[2] G. Hadjeres and L. Crestel, "Vector quantized contrastive predictive coding for template-based music generation," *arXiv preprint arXiv:2004.10120*, 2020.

[3] S. Di, Z. Jiang, S. Liu, Z. Wang, L. Zhu, Z. He, H. Liu, and S. Yan, "Video background music generation with controllable music transformer," in *Proceedings of the ACM International Conference on Multimedia*, 2021, pp. 2037–2045.

[4] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[5] K. Choi, C. Hawthorne, I. Simon, M. Dinculescu, and J. Engel, "Encoding musical style with transformer autoencoders," in *International Conference on Machine Learning*, 2020, pp. 1899–1908.

[6] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2019, pp. 596–603.

[7] J. Jiang, G. G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa, "Transformer VAE: A hierarchical model for structure-aware and interpretable music representation learning," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 516–520.

[8] J. Jiang, G. Xia, and R. Dannenberg, "Representing music structure by variational attention," in *ML4MD Workshop at the International Conference on Machine Learning*, 2019.

[9] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *International Conference on Learning Representations*, 2018.

[10] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *International Conference on Learning Representations*, 2019.

[11] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[12] F. T. Liang, M. Gotham, M. Johnson, and J. Shotton, "Automatic stylistic composition of Bach chorales with deep LSTM," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2017, pp. 449–456.

[13] J. Ens and P. Pasquier, "MMM: Exploring conditional multi-track music generation with the transformer," *arXiv preprint arXiv:2008.06048*, 2020.

[14] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2019, pp. 685–692.

[15] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang, "MuseGAN: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[16] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "PopMAG: Pop music accompaniment generation," in *Proceedings of the ACM International Conference on Multimedia*, 2020, pp. 1198–1206.

[17] Y.-H. Chen, Y.-H. Huang, W.-Y. Hsiao, and Y.-H. Yang, "Automatic composition of guitar tabs by transformers and groove modeling," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2020, pp. 756–763.

[18] S.-L. Wu and Y.-H. Yang, "The jazz transformer on the front line: Exploring the shortcomings of AI-composed music through quantitative measures," in *Proceedings of the International Society for Music Information Retrieval Conference*, 2020, pp. 142–149.

[19] "IMSLP Petrucci Music Library," https://imslp.org, accessed: 2022-05-20.

[20] J. Calvo-Zaragoza and D. Rizo, "End-to-end neural optical music recognition of monophonic scores," *Applied Sciences*, vol. 8, no. 4, p. 606, 2018.

[21] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[22] S. Merity, N. S. Keskar, and R. Socher, "Regularizing and optimizing LSTM language models," *arXiv preprint arXiv:1708.02182*, 2017.

[23] J. Howard and S. Gugger, "Fastai: a layered API for deep learning," *Information*, vol. 11, no. 2, p. 108, 2020.

[24] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38–45.

[25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[26] "Plaine & Easie Code," https://www.iaml.info/plaine-easie-code, accessed: 2022-05-20.

[27] "Verovio: A Music Notation Engraving Library," https://www.verovio.org, accessed: 2022-05-20.

[28] "Music Encoding Initiative," https://music-encoding.org, accessed: 2022-05-20.

HPPNET: MODELING THE HARMONIC STRUCTURE AND PITCH INVARIANCE IN PIANO TRANSCRIPTION

Weixing Wei[1] **Peilin Li**[1] **Yi Yu**[2] **Wei Li**[13]
[1] School of Computer Science and Technology, Fudan University , China
[2] Digital Content and Media Sciences Research Division, National Institute of Informatics (NII), Japan
[3] Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, China
wxwei20@fudan.edu.cn, plli21@m.fudan.edu.cn, yiyu@nii.ac.jp, weili-fudan@fudan.edu.cn

ABSTRACT

While neural network models are making significant progress in piano transcription, they are becoming more resource-consuming due to requiring larger model size and more computing power. In this paper, we attempt to apply more prior about piano to reduce model size and improve the transcription performance. The sound of a piano note contains various overtones, and the pitch of a key does not change over time. To make full use of such latent information, we propose HPPNet that using the Harmonic Dilated Convolution to capture the harmonic structures and the Frequency Grouped Recurrent Neural Network to model the pitch-invariance over time. Experimental results on the MAESTRO dataset show that our piano transcription system achieves state-of-the-art performance both in frame and note scores (frame F1 93.15%, note F1 97.18%). Moreover, the model size is much smaller than the previous state-of-the-art deep learning models.

1. INTRODUCTION

Automatic music transcription (AMT) is a crucial task in Music Information Retrieval (MIR). This task converts the music in audio format to musical notation formats such as MIDI and sheet music. Transcribing from wave format is a process of message compression that reduces the message from universe form (wave) to abstract form (sheet music) that will help with music understanding.

Piano transcription is a popular subtask of AMT. Predicting a set of concurrent pitches present in the same frame is a challenging problem. Over the past decades, plenty of methods have been applied to the piano transcription task, e.g., using Factorization-based models (Smaragdis et al. [1]), using sparsity coding and unsupervised analysis (Abdallah et al. [2]), adaptive estimation of harmonic spectra (Vincent et al. [3]), and using SVM-HMM structure (Nam et al. [4]). Some methods are motivated by the piano's acoustics features, such as the attack/decay [5] model.

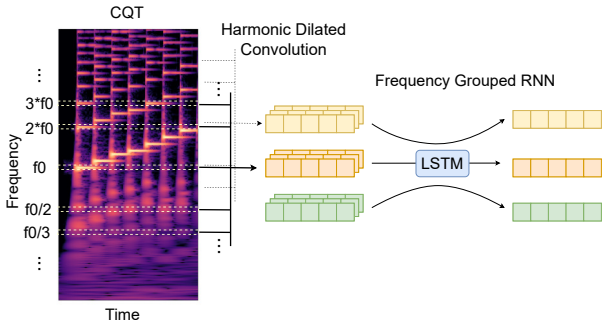


Figure 1. Harmonic Dilated Convolution and Freqency Grouped RNN. The former captures possible harmonic series information for all frequency groups by dilated convolution. The latter feeds each single frequency group to the same LSTM layer to detect whether there is a harmonic series in the frequency group.

In recent years, with the development of deep learning and the existence of large scale labeled datasets, neural networks have become a popular method, such as using RNNs [6] and using methods based on CNNs [7]. With the release of the MAESTRO [8] dataset in the field of piano transcription, the Onsets & Frames transcription system [9] made significant progress. Hawthorne et al. focus on onsets and offsets together to predict frame labels. Transformer is a revolutionary architecture in other fields, and Hawthorne et al. [10] explore Transformers potential on piano transcription. Generative Adversarial Networks [11] also show its potential in improving performance. Kong et al. [12] proposed a high-resolution AMT system trained by regressing precise onset and offset times of piano notes, which achieved state-of-the-art performance in note prediction.

However, previous SOTA models are becoming more and more resource-consuming. The spectrograms of the solo piano are highly structured. Each tone has a harmonic series, and the pitch of a key does not change over time. In view of this prior, we attempt to model such harmonic structure and pitch-invariance over time to improve model interpretability and algorithm efficiency.

© Weixing Wei, Peilin Li, Yi Yu, and Wei Li. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Weixing Wei, Peilin Li, Yi Yu, and Wei Li, "HPPNet: Modeling the Harmonic Structure and Pitch Invariance in Piano Transcription", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

We propose the Harmonic Dilated Convolution (HD-Conv) to capture harmonic series information and the Frequency Grouped Long Short-Term Memory (FG-LSTM) to detect if there is an active pitch accounting for high energy in a specific frequency band. We apply HD-Conv and FG-LSTM to model the harmonic structure and pitch-invariance prior over time in a model called HPPNet. The model achieves state-of-the-art piano transcription performance. Remarkably, it is an approach that uses much fewer parameters of only 1.2 million while other models like the Transformer [10] use 54 million ones. The reduction of parameters is attributed to the use of FG-LSTM, which applies LSTM in a single frequency group, and the shared acoustic model. The primary contribution of this work is an tiny model that achieves state-of-the-art performance using much fewer parameters.

The rest of this paper is organised as follows: Section 2 describes the proposed model component's details: harmonic dilated convolution and the frequency grouped recurrent neural networks. Section 3 illustrates the experimental setup, with results analysed in Section 4. Conclusions are discussed in Section 5.

2. ARCHITECTURE

The two critical components of our model are the harmonic dilated convolution and the frequency grouped recurrent neural networks. The former is used to model harmonics structure, and the latter is designed based on the invariance of piano pitches over time as demonstrated in Figure 1.

2.1 CQT input

We use the Constant-Q Transform (CQT) [13] as our input feature. Unlike the log-mel spectrogram which is partially log-scaled, all the frequency bins of the CQT are geometrically spaced. As shown in Eq. (1), the spacing between fundamental frequency and overtones does not vary with the change of the fundamental frequency. Such property makes it suitable for dilated convolution, as described in [14].

$$\begin{aligned} d_k &= log_{2^{1/Q}}(k \cdot f_0) - log_{2^{1/Q}}(f_0) \\ &= Q \cdot log_2(k), \end{aligned} \tag{1}$$

where k is the serial number of the harmonic series, d_k denotes the distance between fundamental frequency and the k-th overtone on a log frequency scale, Q indicates the number of frequency bins per octave, and f_0 is the fundamental frequency.

2.2 Harmonic Dilated Convolution

In recent years, some methods are proposed to modeling harmonic structure in neural networks. The Harmonic constant-Q transform (HCQT) [15] captures the harmonic relationships by a 3-dimensional CQT array for pitch tracking in polyphonic music. Harmonic convolution is applied in [16] to denoise speech audios. Dilated convolution [17] and sparse convolution [18] are used in Wang's works but they are still not perfect ways to capture harmonic structure. Our model captures harmonic information in a simple but effective way. As shown in our acoustic model (Figure 2), we feed the CQT into multiple dilated convolution [19] layers with different dilation rates and sum the outputs for the following layers. The dilation rates are the distance between fundamental frequency and overtones on a log frequency scale as in Eq. (1). We call such dilated convolutions applied on the log-frequency dimension and with dilation rates of spaces between harmonic series the Harmonic Dilated Convolution (HD-Conv).

Model	Onsets & Frames		HPPNet	
	Acoustic	LSTM	Acoustic	FG-LSTM
inputs	T×229	T×768	T×352	T×128
outputs	T×768	T×88	T×88×128	T×1
units	-	256	-	128
parameters	4.3M	3.5M	421K	99K

Table 1. The parameters of different layers in Onsets & Frames and HPPNet. Acoustic denotes the acoustic models, T denotes the number of frames in a training sample.

2.3 Frequency Grouped LSTM

In the feature map output by the harmonic acoustic model, each frequency unit with multiple channels contains corresponding possible harmonic information to detect if there is an active pitch accounting for high energy in a specific frequency band. However, detecting multiple pitches in polyphonic music is challenging since harmonic series interfere with each other. Time-domain correlation is required to obtain smooth pitch contours. Previous works [9] [12] applied bidirectional Recurrent Neural Network (biRNN) [20] to model the temporal relationship of frames. They flatten the channel dimension and frequency dimension of the feature map output by the acoustic model to a long hidden dimension as the input of RNN. There are some problems with such processing. One is that the long hidden dimension led to an unnecessarily large amount of model parameters. This leads to large amount of parameters both in acoustic model and Long Short-Term Memory (LSTM) [21], as described in Table 1.

The sub-band and multi-band techniques are widely applied in speech and music processing [22–24]. The common practice is splitting the full-band spectral representation into multiple sub-bands and processing them separately, then concatenating the outputs of each sub-band for subsequent processing. Specifically, some methods based on splitting sub-bands (frequency-LSTM [25]) or generating sub-bands (multi-band MelGAN [26]) show impressive performance with lightweight models. The splitting of sub-bands reduces the size of network input, and different sub-bands sharing the same sub-network further reduces the model size.

Similar to the multi-bands techniques, we segment the output of harmonic dilated convolution to 88 frequency

groups. Since the piano is a fixed-pitch keyboard instrument, the fundamental frequency of each key does not vary as time changes. This enables the model process each frequency group independently. Based on this assumption, we feed each frequency group to the same LSTM layer individually to model the temporal relationship. We term this the Frequency Grouped LSTM (FG-LSTM) illustrated in Figure 1. Feeding a single frequency group to the LSTM makes inputs and units size of FG-LSTM smaller. This method significantly reduces the amount of parameters in our model.

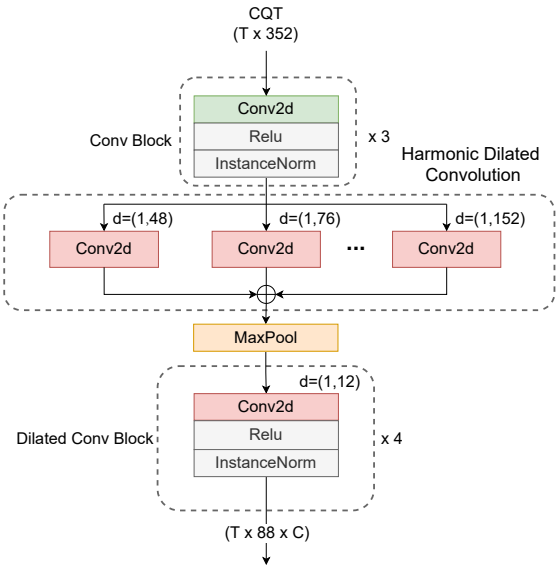


Figure 2. Acoustic model of HPPNet. It is composed of three identical regular convolution layers, a harmonic dilated convolution layer, a max pooling layer, and four identical dilated Conv blocks. Different dilated convolution layers have different dilation rates denote by d. The output is a feature map with size $T \times 88 \times C$, where T is the frame numbers, 88 denotes the numbers of piano keys, C is the channel size.

2.4 Model Details

Raw audios are clipped to 20-second pieces and resampled to 16 kHz. Then, the CQT is computed by nnAudio [27] with hop length 320 (20 millisecond), an FFT window of 2048, bins per octave of 48, fmin of 27.5 Hz, frequency bins number of 352, and log amplitude. The model takes the CQT feature with size $T \times 352$ as input, where T is the frame number of each audio clip. The time resolution and frequency resolution of inputs are limited by the computational resource, higher input resolution may have better performance.

The model is composed of a convolutional acoustic model and multiple FG-LSTM heads. The former structure captures the harmonic representations of a tone, and the latter establishes connections over temporal series. In the acoustic model shown in Figure 2, the first three convolution layers extract local information with kernel size 7×7, a ReLu activation, and instance normalization.

Then followed by eight dilated convolution layers parallel with different dilations of 48, 76, 96, 111, 124, 135, 144, and 152 in the frequency dimension. These dilation rates are calculated by Eq. (1) with $Q = 48$ and rounded to integers. All the outputs of these layers integrate the harmonic information for each frequency unit.

The max-pooling layer downsamples the frequency bins from 352 to 88 with pooling size of 4, four bins per semitone to one bin per semitone. Then four identical dilated convolution blocks make the network deeper to achieve higher learning capacity. Each block contains one convolution layer with kernel size of 5×3, and dilation of 1×12.

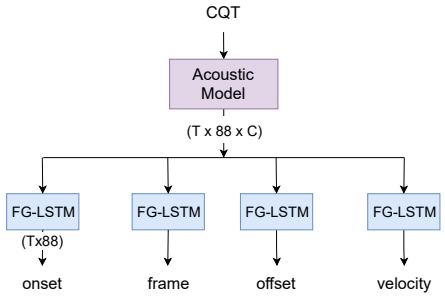


Figure 3. HPPNet-base. Onset, frame, offset, and velocity tasks share the same acoustic model.

Finally, four FG-LSTM heads are used to predict frames, onsets, offsets, and velocities separately as shown in Figure 3. These outputs are then decoding to note events with the same way as the Onsets & Frames [8] model. Note that the offset head is just used for training and not directly used during decoding. Each head uses a linear layer with a sigmoid activation function to output the prediction. All the outputs are matrices of size $T \times 88$, where T denotes frame number and 88 represents the 88 piano keys. The LSTM is bidirectional, and both forward and backward directions have 64 units. The hyperparameters of the network are summarized in Table 2. We denote this model the HPPNet-base.

Layer	repeat	kernel	filters	dilation
Conv	3	7×7	16	-
HD-Conv	1	1×3	128	48, 76, ..., 152
Dilated Conv	4	5×3	128	1×12

Table 2. Hyperparameters of convolution layers.

All the FG-LSTM heads in HPPNet-base share the same acoustic model, making it simple and efficient. But in our experiments, we try multiple acoustic models and find that if a stand-alone acoustic model is used to predict the onset, the model will perform better in onset detection. Since the onset F-measure is the metric that correlates best with human judgment [28], we use an individual acoustic model for onset. Frame, offset, and velocity share another acoustic model. The HPPNet-sp is shown in Figure 4. The output representations of the onset acoustic model are concatenated with outputs of the other acoustic model to pre-

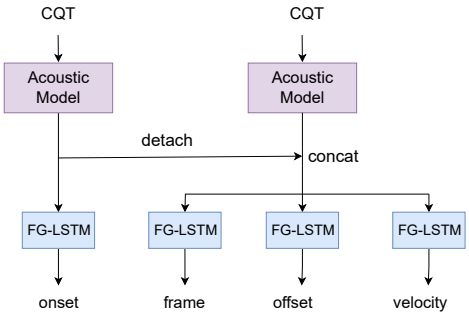


Figure 4. HPPNet-sp. Onset and other tasks use separate acoustic models.

dict frame, offset, and velocity (detach to avoid backward propagation).

3. EXPERIMENTS

In this section we first introduce the datasets we used. And then, we describe the details of the experiment and the training loss for our model.

3.1 Datasets

MAESTRO [8] (MIDI and Audio Edited for Synchronous Tracks and Organization): It includes about 200 hours of paired audio and MIDI recordings from ten years of International Piano-e-Competition. Audio and MIDI files are aligned with 3 ms accuracy and sliced to individual musical pieces annotated with composer, title, and year of performance. The MIDI have been collected by Yamaha Disklaviers which have a high-precision MIDI capture system.

MAPS [29] (MIDI Aligned Piano Sounds): It includes about 31 GB of CD-quality recordings and corresponding annotations of isolated notes, chords, and complete piano pieces. The records contain both synthesized audio and real audio by Virtual Piano software and a Yamaha Disklavier respectively.

3.2 Experimental setup

We use PyTorch [1] to implement our model. The training is performed by Adam [30] optimizer with mini-batch size 4, and a learning rate of 0.0006 for 200k to 500k steps with early stopping. The training time on an NVIDIA GeForce 3060 GPU with 12 GB VRAM is about 48 hours. The note and frame scores are calculated by the mir_eval library [31]. The evaluation uses a 50 ms tolerance for note onset and an offset ratio of 0.2 as the default configuration in mir_eval. The onset and frame thresholds are both set to 0.4 on the sigmoid output in our model. All the hyperparameters are selected by the performance on the validation split of MAESTRO.

The losses we use for training are similar to the Onsets & Frames model. The losses of onset, frame, and offset are binary cross-entropy losses. The weighted frame loss is not used in our training. The loss of velocity is the mean squared error loss. All losses are summed together as the final loss. All the losses are as follows:

$$l_{bce}\left(y\ ,\ \hat{y}\right) = -wy\log(\hat{y}) - (1-y)\log(1-\hat{y}), \quad (2)$$

$$L_{onset} = \sum_{p=1}^{88}\sum_{t=1}^{T} l_{bce}\left(n_{p,t}\ ,\ \hat{n}_{p,t}\right), \quad (3)$$

$$L_{frame} = \sum_{p=1}^{88}\sum_{t=1}^{T} l_{bce}\left(f_{p,t}\ ,\ \hat{f}_{p,t}\right), \quad (4)$$

$$L_{offset} = \sum_{p=1}^{88}\sum_{t=1}^{T} l_{bce}\left(o_{p,t}\ ,\ \hat{o}_{p,t}\right), \quad (5)$$

$$L_{velocity} = \sum_{p=1}^{88}\sum_{t=1}^{T} n_{p,t}\left(v_{p,t} - \hat{v}_{p,t}\right)^2, \quad (6)$$

$$L = L_{onset} + L_{frame} + L_{offset} + L_{velocity}. \quad (7)$$

where l_{bce} denotes the binary cross entropy loss, p is the piano note index range from 1 to 88, T is the frame number in a sample, w denotes the positive weight ($w = 2$ for onset, $w = 1$ for frame and offset), y denotes ground true and $\hat{y}$ denotes predicted values range from 0 to 1, $n_{p,t}$, $f_{p,t}$, $o_{p,t}$, and $v_{p,t}$ are the ground true of onset, frame, offset, and velocity seperately.

3.3 Baselines

Our model is compared with five typical models, including High-Resolution piano transcription [12], Onsets & Frames [9], Adversarial onsets & frames [11], Semi-CRFs [32] and the sequence-to-sequence Transformer [10]. The results are shown in Table 3. High-Resolution and Onsets & Frames are convolution-based models composed of convolution layers and LSMT layers. Especially, the High-Resolution is the best model in note F1 score which is 96.72%. Adversarial onsets & frames is an adversarial training scheme that operates on the Mel-spectrogram. Semi-CRFs is a Semi-Markov Conditional Random Fields (semi-CRF) based model, which treats note intervals as holistic events. Sequence-to-sequence Transformer uses a generic Transformer architecture with standard decoding methods.

4. RESULTS

We compare our model with some existing state-of-the-art methods using the MAESTRO dataset and the MAPS dataset. Then, ablation studies are done to demonstrate the affects of FG-LSTM, and HD-Conv. We also examine the model performance on small datasets.

[1] www.pytorch.org

Model	Params	FRAME			NOTE			NOTE W/OFFSET			NOTE W/OFFSET & VEL.		
		P (%)	R (%)	F1(%)	P (%)	R (%)	F1(%)	P (%)	R (%)	F1(%)	P (%)	R (%)	F1(%)
		MAESTRO v1											
Onsets & Frames [8]	26M	92.11	88.41	90.15	98.27	92.61	95.32	82.95	78.24	80.50	79.89	75.37	77.54
Adversarial onsets & frames [11]	26M	93.1	89.8	91.4	98.1	93.2	95.6	83.5	79.3	81.3	82.3	78.2	80.2
		MAESTRO v2											
High-Resolution [12]	20M	88.71	90.73	89.62	98.17	95.35	96.72	83.68	81.32	82.47	82.10	79.80	80.92
Semi-CRFs [32]	9M	93.85	88.72	91.11	98.66	94.50	96.51	90.68	86.89	**88.72**	89.68	85.96	**87.75**
HPPNet-sp	1.2M	92.36	93.46	**92.86**	98.31	96.18	**97.21**	85.36	83.54	84.41	83.85	82.08	82.93
		MAESTRO v3											
Onsets & Frames [reproduced]	26M	94.43	85.50	89.68	98.67	92.08	95.22	82.25	76.88	79.44	80.80	75.55	78.05
Transformer [10]	54M	-	-	-	-	-	96.13	-	-	83.94	-	-	82.75
Semi-CRFs [32]	9M	93.79	88.36	90.75	98.69	93.96	96.11	90.79	86.46	**88.42**	89.78	85.51	**87.44**
HPPNet-tiny	151K	92.26	91.98	92.06	96.75	94.94	95.82	83.77	82.26	83.00	82.77	81.29	82.01
HPPNet-base	820K	92.35	92.75	92.51	97.60	94.90	96.25	84.03	83.48	83.57	82.94	81.23	82.12
HPPNet-sp	1.2M	92.79	93.59	**93.15**	98.45	95.95	**97.18**	84.88	82.76	83.80	83.29	81.24	82.24

Table 3. Transcription result evaluated on the MAESTRO dataset. Train split of MAESTRO v3 is used for training and test splits of different versions are used for evaluation.

Model	Params	FRAME			NOTE			NOTE W/OFFSET			NOTE W/OFFSET & VEL.		
		P (%)	R (%)	F1(%)	P (%)	R (%)	F1(%)	P (%)	R (%)	F1(%)	P (%)	R (%)	F1(%)
Onsets & Frames	26M	-	-	82.02	-	-	83.04	-	-	61.84	-	-	48.07
Onsets & Frames*	26M	92.86	78.46	84.91	87.46	85.58	86.44	68.22	66.75	67.43	52.41	51.22	51.77
HPPNet-sp	1.2M	88.42	86.81	87.56	91.61	82.38	86.63	65.01	63.84	64.39	60.35	59.26	59.77

Table 4. The transcription evaluation result on MAPS test split, which training was done on the MAESTRO v3 train split. Onsets & Frames* means Onsets & Frames with audio augmentation. The training of HPPNet is without audio augmentation.

4.1 Comparison with baselines

Along with the HPPNet-base and HPPNet-sp, we also evaluate the HPPNet-tiny, which has a similar structure as HPPNet-base but reduces the maximum convolution channel size and units in LSTM from 128 to 48.

Evaluation on the MAESTRO dataset is shown in Table 3. The HPPNet-sp achieves state-of-the-art in both frame F1 score and note F1 score, which improved 3.24 percentage points and 0.49 percentage points, reaching 92.86% and 97.21% in MAESTRO v2 respectively. In the last two columns, under the condition of note with offset, it still outperforms other models except the event-based Semi-CRFs method. Even slight structure of HPPNet-base still achieves 92.51% and 96.25% on frame F1 score and note F1 score, respectively.

Moreover, another significant advantage is that our model has much fewer parameters, around 820 thousand HPPNet-base and 1.2 million HPPNet-sp. The lightweight structure HPPNet-tiny with 151 thousand parameters still outperforms the baseline Onsets & Frames, that the frame F1 score and the note F1 score reaching 92.06% and 95.82%. This proves the effectiveness of HPPNet in both improving performance and reducing model size.

Empirical results in Table 3 show that among all compared frame-level models, our model performs best in all scores. It is interesting to note that HPPNet achieves improved recall while not or only slightly loosing precision. But the event-based Semi-CRFs has better prediction on offsets, for it predicts note intervals directly rather than individual frames. Maybe future works that combining the frame-level and event-level can take advantage of the both sides.

To further explore the generalization of the HPPNet in different datasets, we train the model on the MAESTRO dataset and evaluate it on the MAPS dataset. The result is shown in Table 4. we can find that the HPPNet-sp's note F1 still reaches 87.56%. This result is better than Onsets & Frames of 83.04%, even outperforming Onsets & Frames with audio augmentation of 86.44%. The result confirms that the HPPNet has better generalization ability on piano transcription task.

4.2 Ablation study

We further analyze the role of the HD-Conv layer and FG-LSTM in the HPPNet. In addition to the HPPNet, we evaluate three types of models, the F1 in frame-wise

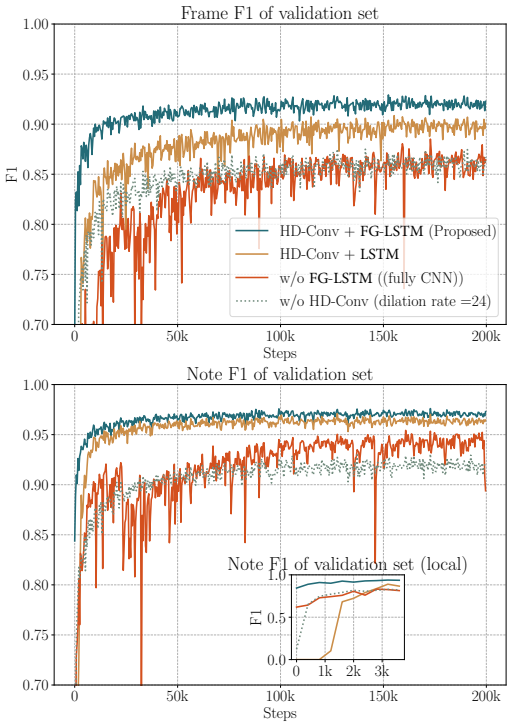


Figure 5. F1 in frame-wise and note-wise of MAESTRO v3 validation set.

and note-wise during training per epoch shown in Figure 5:

- To verify the effect of FG-LSTM: change FG-LSTM to normal LSTM with frequency flattening.
- To verify the effect of LSTM: remove time-series layers, which change them to linear layers.
- To verify the effect of harmonic dilated convolution: replace the harmonic dilated convolution with fixed dilated convolution (dilation rate of 24).

These results suggest that the the HPPNet outperforms the other three ablated models. The F1 of normal LSTM decreases about three percentage points and one percentage point on frame-wise and note-wise, respectively. Usual LSTM starts later than the HPPNet in note-wise F1 at the beginning of training, and it starts to increase after about 1k steps. The HD-Conv + vanilla LSTM is better than changing time-series layers to linear layers. Model without RNN converges slowly and fluctuates drastically. A large receptive field helps model get global information. Using the fixed dilation rate has a similar receptive field to HD-Conv. But the result shows it does not capture useful information for the detection of frames and onsets, leading to poor performance.

4.3 Performance on small datasets

The HPPNet trained on big dataset such as compete MAESTRO shows promising results, and it also shows better generalization when inference on MAPS. To evaluate the performance of the HPPNet on smaller dataset, we also train the model with 30% and 10% of MAESTRO training split. All the samples are randomly selected and without data augmentations. The results are displayed in Table 5. When the training set size decreases to 10%, the HPPNet-sp drops less than Onsets & Frames on frame F1 and note F1 scores with 88.31% and 93.52% respectively, while the Onsets & Frames approach only yields 80.36% and 90.62%. This demonstrates that the HPPNet relies less on data thanks to the small amount of parameters and priors contained in HD-Conv and FG-LSTM.

Model	Data	FRAME			NOTE		
		P (%)	R (%)	F1(%)	P (%)	R (%)	F1(%)
Onsets& Frames	100%	94.43	85.50	89.68	98.67	92.08	95.22
	30%	91.14	80.00	85.08	98.13	89.57	93.60
	10%	90.66	72.70	80.36	96.64	85.46	90.62
HPPNet-sp	100%	92.79	93.59	**93.15**	98.45	95.95	**97.18**
	30%	90.72	92.11	91.35	96.46	94.46	95.43
	10%	92.10	84.96	88.31	96.59	90.94	93.52

Table 5. The transcription evaluation result on MAESTRO v3 test split, which training was done on the 100%, 30%, and 10% of MAESTRO v3 training split respectively. Considering the annotation accuracy, we use subsets of MAESTRO rather than the MAPS for evaluation.

5. CONCLUSION

In this paper, we design a model that carries piano inductive biases to capture piano priors. The HPPNet is proposed to model the harmonic structure and the pitch-invariance over time in piano transcription. Experimental results show that the model achieves state-of-the-art performance on the MAESTRO dataset, with frame F1 of 93.15% and note F1 of 97.18%. The success stems from two aspects: (i) the harmonic dilated convolution exploited to capture harmonics structure; and (ii) the frequency grouped LSTM designed based on the pitch-invariance of piano key over time. Furthermore, the model parameters are much fewer than the previous state-of-the-art models.

The results of the model on piano transcription are encouraging. And it may also be suitable for other instruments with similar pitch characteristics. But some challenges remain, such as improving performance in offset detection and modifying the model to other polyphonic transcriptions involving vocals and instruments with a less stable pitch. The harmonic dilated convolution implemented by multiple parallel dilated convolutions is computational consuming and needs to be optimized in the future.

6. ACKNOLEDGEMENT

This work was supported by National Key R&D Program of China(2019YFC1711800), NSFC(62171138). Wei Li and Yi Yu are corresponding authors of this paper.

7. REFERENCES

[1] A. Khlif and V. Sethu, "An iterative multi range non-negative matrix factorization algorithm for polyphonic music transcription," in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR*, 2015, pp. 330–335.

[2] S. A. Abdallah and M. D. Plumbley, "Unsupervised analysis of polyphonic music by sparse coding," *IEEE Transactions on Neural Networks*, vol. 17, no. 1, pp. 179–196, 2006.

[3] E. Vincent, N. Bertin, and R. Badeau, "Adaptive harmonic spectral decomposition for multiple pitch estimation," *IEEE Transactions on Speech Audio Processing*, vol. 18, no. 3, pp. 528–537, 2010.

[4] J. Nam, J. Ngiam, H. Lee, and M. Slaney, "A classification-based polyphonic piano transcription approach using learned feature representations," in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR*, 2011, pp. 175–180.

[5] T. Cheng, M. Mauch, E. Benetos, and S. Dixon, "An attack/decay model for piano transcription," in *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR*, 2016, pp. 584–590.

[6] S. Böck and M. Schedl, "Polyphonic piano note transcription with recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 2012, pp. 121–124.

[7] S. Sigtia, E. Benetos, and S. Dixon, "An end-to-end neural network for polyphonic piano music transcription," *IEEE ACM Transactions on Audio Speech and Language Processing. TASLP*, vol. 24, no. 5, pp. 927–939, 2016.

[8] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the MAESTRO dataset," in *International Conference on Learning Representations, ICLR*, 2019, pp. 1–6.

[9] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. H. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, 2018, pp. 50–57.

[10] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. H. Engel, "Sequence-to-sequence piano transcription with transformers," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 246–253.

[11] J. W. Kim and J. P. Bello, "Adversarial learning for improved onsets and frames music transcription," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 670–677.

[12] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution piano transcription with pedals by regressing onset and offset times," *IEEE ACM Transactions on Audio Speech and Language Processing. TASLP*, vol. 29, pp. 3707–3717, 2021.

[13] J. C. Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America, JASA*, vol. 89, no. 1, pp. 425–434, 1991.

[14] W. Wei, P. Li, Y. Yu, and W. Li, "Harmof0: Logarithmic scale dilated convolution for pitch estimation," in *2022 IEEE International Conference on Multimedia and Expo, ICME*, 2022, pp. 1–6.

[15] R. M. Bittner, B. McFee, J. Salamon, P. Li, and J. P. Bello, "Deep salience representations for F0 estimation in polyphonic music," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR*, 2017, pp. 63–70.

[16] Z. Zhang, Y. Wang, C. Gan, J. Wu, J. B. Tenenbaum, A. Torralba, and W. T. Freeman, "Deep audio priors emerge from harmonic convolutional networks," in *8th International Conference on Learning Representations, ICLR*, 2020, pp. 1–8.

[17] X. Wang, L. Liu, and Q. Shi, "Enhancing piano transcription by dilated convolution," in *19th IEEE International Conference on Machine Learning and Applications, ICMLA*, 2020, pp. 1446–1453.

[18] X. Wang, L. Liu, and Q. Shi, "Harmonic structure-based neural network model for music pitch detection," in *19th IEEE International Conference on Machine Learning and Applications, ICMLA*, 2020, pp. 87–92.

[19] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proceedings of International Conference on Learning Representations, ICLR*, 2016, pp. 1–4.

[20] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, pp. 2673–2681, 1997.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] N. Takahashi and Y. Mitsufuji, "Multi-scale multi-band densenets for audio source separation," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, 2017, pp. 21–25.

[23] X. Li and R. Horaud, "Multichannel speech enhancement based on time-frequency masking using subband long short-term memory," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA*, 2019, pp. 298–302.

[24] B. Wang and Y.-H. Yang, "Performancenet: Score-to-audio music generation with multi-band convolutional residual network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 1174–1181.

[25] Y. Luo, C. Han, and N. Mesgarani, "Ultra-lightweight speech separation via group communication," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, 2021, pp. 16–20.

[26] G. Yang, S. Yang, K. Liu, P. Fang, W. Chen, and L. Xie, "Multi-band melgan: Faster waveform generation for high-quality text-to-speech," in *IEEE Spoken Language Technology Workshop (SLT)*, 2021, pp. 492–498.

[27] K. W. Cheuk, H. Anderson, K. Agres, and D. Herremans, "Nnaudio: An on-the-fly GPU audio to spectrogram conversion toolbox using 1d convolutional neural networks," *IEEE Access*, vol. 8, pp. 161 981–162 003, 2020.

[28] A. Ycart, L. Liu, E. Benetos, and M. T. Pearce, "Investigating the perceptual validity of evaluation metrics for automatic piano music transcription," *Transactions of the International Society for Music Information Retrieval Conference TISMIR*, pp. 68–81, 2020.

[29] V. Emiya, R. Badeau, and B. David, "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle," *IEEE ACM Transactions on Audio Speech and Language Processing. TASLP*, vol. 18, no. 6, pp. 1643–1654, 2009.

[30] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of International Conference on Learning Representations, ICLR*, 2015, pp. 1–9.

[31] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "Mir_eval: A transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, 2014, pp. 1–6.

[32] Y. Yan, F. Cwitkowitz, and Z. Duan, "Skipping the frame-level: Event-based piano transcription with neural semi-crfs," in *Advances in Neural Information Processing Systems, NeurIPS*, vol. 34, 2021, pp. 20 583–20 595.

GENERATING MUSIC WITH SENTIMENT USING TRANSFORMER-GANS

Pedro L. T. Neves
State University of Campinas
p185770@dac.unicamp.br

Jose Fornari
State University of Campinas
fornari@unicamp.br

João B. Florindo
State University of Campinas
florindo@unicamp.br

ABSTRACT

The field of Automatic Music Generation has seen significant progress thanks to the advent of Deep Learning. However, most of these results have been produced by unconditional models, which lack the ability to interact with their users, not allowing them to guide the generative process in meaningful and practical ways. Moreover, synthesizing music that remains coherent across longer timescales while still capturing the local aspects that make it sound "realistic" or "human-like" is still challenging. This is due to the large computational requirements needed to work with long sequences of data, and also to limitations imposed by the training schemes that are often employed. In this paper, we propose a generative model of symbolic music conditioned by data retrieved from human sentiment. The model is a Transformer-GAN trained with labels that correspond to different configurations of the valence and arousal dimensions that quantitatively represent human affective states. We try to tackle both of the problems above by employing an efficient linear version of Attention and using a Discriminator both as a tool to improve the overall quality of the generated music and its ability to follow the conditioning signals.

1. INTRODUCTION

One of the driving factors behind the human experience of music is the emotional content that it conveys. Several works in the area of Musical Information Retrieval (MIR) have focused on Music emotion recognition, that is, automatic recognition of the perceived emotion of music based solely on the musical information itself [1]. In this context, emotion is often represented according to the valence and arousal dimensions originated from to the Russell circumplex model [2]. While valence expresses how pleasurable or displeasurable an emotion is, arousal represents the level of alertness associated with that emotion, going from relaxed to excited. However, within the realm of *Deep Learning* comparatively little research has focused on reverting this process, that is, instead of predicting the affective content of a musical passage, being able to generate musical pieces that project a desired emotional state.

Deep Neural Networks are now capable of generating songs that display coherent chord progressions, melodies and even lyrics [3], despite the fact that these characteristics often do not persist across longer time scales. While there are several reasons behind this fact, two of them stand out. Firstly, there are the inherent computational and memory costs involved in modeling longer sequences of data. Secondly, the most common technique used to train these models, teacher forcing or Maximum Likelihood Estimation (MLE) [4], makes them work with data distributions that are different during training and inference time (real vs synthetic). This is known as exposure bias [5]. One of the ways to alleviate this problem is by delegating the task of judging which samples are good and which are not to a different neural network that is trained in conjunction with the generative model. This is the motivation behind the use of Generative Adversarial Networks (GANs) [6] within the realm of sequence generation [7].

In this work, we present a generative model of music capable of synthesizing songs conditioned by values of valence and arousal that correspond to perceived sentiment [2]. The ability of automatically generating music that follows a specific pattern of emotions can be interesting in various contexts, e.g. producing soundtracks to accompany story-driven forms of media such as Video-Games and Movies, which often use music as a means of guiding the audience towards a specific emotional state that suits the narrative. The goal here is to provide users who may not have the musical background necessary for composition a way of translating their perception into songs that can suit their artistic aspirations.

In an effort to mitigate the shortcomings of teacher forcing-style training, we complement it with an additional adversarial signal provided by a Discriminator network. This addition has a positive effect on the generated samples, and we demonstrate, through evaluations of our model both via automatic metrics and human feedback, that the proposed Transformer GAN obtains a performance that competes with a current state-of-the-art model, even while having a smaller set of parameters and using a simpler representation of music. To summarise, our contributions are as following:

- We present a neural network that, up to our knowledge, is the first generative model based on GANs to produce symbolic music conditioned by sentiment.

© Pedro L. T. Neves, Jose Fornari, and João Batista Florindo. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Pedro L. T. Neves, Jose Fornari, and João Batista Florindo, "Generating music with sentiment using Transformer-GANs", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

- We show, both through automatic and human evaluation, that our model obtains a competitive performance with that of a current state-of-the-art model on the task of music generation conditioned by sentiment.
- We show that promising results come from using Generative Adversarial Networks within the context of music generation conditioned by sentiment, as our model obtains a good performance despite having less parameters, using a simpler symbolic representation, and, being, up to our knowledge, the first on this task to be trained via an adversarial scheme.

2. RELATED WORKS

2.1 Generative Adversarial Networks

Generative Adversarial Networks, or simply GANs, consist of a theoretical framework in which two Neural Networks, the Generator and the Discriminator, through competition, optimize a model that implicitly approximates a data distribution by generating samples that try to mimic the features that it observes on a given set of samples originating from that distribution. Each of the networks is trained to optimize an objective function. The objective of the Discriminator D is to separate the real samples from those that are created by the Generator G, whose job is to produce samples that are so similar to those of the real distribution that the Discriminator is unable to determine which of them are real and which are fake. This whole process is equivalent to a min-max game that can be formalized by Equation 1.

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \quad (1)$$

In the equation above, $p_{\text{data}}(\boldsymbol{x})$ is the real data distribution, and $p_z(\boldsymbol{z})$ is a prior on input noise variables that come from a normal distribution, and which are then mapped to data space by G, so that the synthetic distribution, p_g, can be learned.

Due to the inherent instability of the adversarial process, several works have focused on improving the convergence and the quality of the samples generated by GANs via new objective functions, regularization and normalization techniques, and model architectures. Of particular interest to this work is RSGAN [8], which substitutes the standard GAN loss for the non-saturating Relativistic Standard Loss. Here, as the authors put it, the Discriminator estimates the probability that the given real data is more realistic than a randomly sampled fake data. Equations 2 and 3 correspond to the objectives for the Discriminator and the Generator, respectively.

$$\mathcal{L}_{RSGAN,D} = - \mathbb{E}_{(x_r,x_f)\sim(\mathbb{P},\mathbb{Q})}[\log(\text{sigmoid}(D(x_r) - D(x_f)))] \quad (2)$$

$$\mathcal{L}_{RSGAN,G} = - \mathbb{E}_{(x_r,x_f)\sim(\mathbb{P},\mathbb{Q})}[\log(\text{sigmoid}(D(x_f) - D(x_r)))], \quad (3)$$

in which $\mathbb{P}$ and $\mathbb{Q}$ are, in this order, the real and fake disributions.

In order to provide more stability to the training process, WGAN-GP [9] introduces a Gradient Penalty in the form of an additional loss that enforces a Lipschitz constraint on the Discriminator. This loss is expressed in Equation 4:

$$\mathcal{L}_{GP} = \mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(\|\nabla_{\hat{x}} D_\phi(\hat{x})\|_2 - 1)^2] \quad (4)$$

where $\hat{x}$ is sampled along straight lines between pairs of points taken from the real and fake data distributions, and ϕ are the Discriminator parameters.

One possible strategy that can be used to combat the effect known as exposure bias [5], characterized by the disparity between the data available to the network during training and inference time when the model is trained via standard Maximum Likelihood Estimation (MLE), is the insertion of a Discriminator into the training process as a tool to guide the Generator. Nevertheless, generating discrete sequences with GANs is notoriously hard. This is mostly due to the fact that the outputs of some sequence models are discretized in a way that prohibits the gradient of the Discriminator loss to propagate through the Generator. Several strategies have been proposed to circumvent this issue [10–12]. Here, we employ the Gumbel-Softmax technique presented in [11] and applied to adversarial training in [12]. Mathematically, if we have a categorical distribution with class probabilities $\pi_i, i \in i, ..., d$, we can draw samples y from this distribution using:

$$y = \text{one\_hot}\left(\arg\max_i [g_i + \log \pi_i]\right) \quad (5)$$

where each $g_i, i \in i, ..., d$ is taken from a Gumbel Distribution with location 0 and scale 1. From this expression, one can obtain a continuously differentiable approximation of the categorical distribution parameterized in terms of the softmax function:

$$y = \text{softmax}((1/\tau)(\pi + g)), \quad (6)$$

where τ is a temperature parameter that regulates how close to the categorical (lower τ) *versus* to the uniform distribution (higher τ) is y. This parameter is annealed from large values to close to zero during training. Here, we use the same schedule as in [13], that is, $1/\tau = (1/\tau_{\min})^{n/N}$, where n is the index of the current global optimization, N is the total number of steps and $\tau_{\min}$ is a hyperparameter which we chose to be 10^{-2}.

2.2 Transformers

Shortly after its presentation in 2017, the Transformer [14] became the most popular architecture in the field of Natural Language Processing (NLP), and it has also been successfully applied to other areas, such as image recognition [15] and audio [16]. The main reason behind its success is the

Self-Attention mechanism, which allows it to model relationships between all elements of a sequence. The mechanism takes in matrices Q, K, and V, corresponding, respectively, to Queries, Keys and Values, and calculates a weighted sum of the values where the coefficients are given by a similarity function between the queries and the keys.

One of the models developed with the intent of decreasing the computational costs associated with the calculation of Attention matrices [17–20] is the Linear Transformer [18], which, as the name suggests, employs a version of the attention mechanism with computational and memory requirements that grow linearly with respect to sequence length (standard Attention has quadratic requirements), and is also faster than regular attention. This is achieved via the substitution of the standard softmax similarity score by one that allows the matrix multiplications necessary for the calculation of the attention matrix to be factorized in a more efficient way. If that kernel has a feature representation $\phi(x)$, one can write the Attention matrix as:

$$A(Q,K,V)_i = \frac{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j) V_j}{\sum_{j=1}^{N} \phi(Q_i)^T \phi(K_j)}. \qquad (7)$$

With this expression, it is possible to calculate the factor inside the sum only once, and to reuse it to find all the queries. Specifically, the authors use the kernel with a feature map that results in a positive similarity function: $\phi(x) = \text{elu}(x)+1$, where elu is the exponential linear function [21]. In this same work, the authors also discover an analogy between the Transformer and the RNN. We refer to [18] for further details on the Linear Transformer.

2.3 Generative Models of Music

As of the writing of this paper, most of the state-of-the art generative models of symbolic music are Transformers or Transformer-based. Music Transformer [22] uses relative self-attention [23] to process longer sequences of data and generate music that exhibits long-term structure. MuseNet [24] is able to produce multi-track compositions spanning several musical genres and artists by employing time, note and structural embeddings that give the model more context.

Midinet [25] and MuseGAN [26] are GAN-based generative models of music that use Convolutional Neural Networks (CNNs) as both the Generator and Discriminator. In those works, music is represented in the form of piano-rolls that work analogously to images. Most similar to our work are [27], where the Discriminator is trained to judge the content both locally and globally, and [13], which uses a Transformers-XL [28] as Generator and a pre-trained BERT [29] as Discriminator, and also employs the Gumbel-Softmax trick discussed above.

Several works have also explored conditioning musical generation with emotional content. In [30], human emotions are captured from images of human faces and then categorized into 7 categories. Then, the researchers try to generate music based on these emotions. In [31], Biaxial LSTM networks are used to produce polyphonic music, and the generation can be conditioned by emotion via 4 parameters originating from the valence and arousal dimensions. In [32], the authors translate between the visual and musical domains via emotions. Specifically, they use neural networks to extract emotional content from images and music, and subsequently feed the content originated from both pieces of data into a network to condition music generation. Music Fadernets [33] constitute a framework that can infer high-level feature representations by first modelling their equivalent low-level attributes, which are easier to quantify. These low-level features are then used for style transfer across arousal states. In [34], a model dubbed Bardo Composer is presented as a system to generate backgorund music for tabletop role-playing games. This is done through Stochastic Bi-Objective Beam Search (SBBS), a search algorithm that samples from a distribution of sequences and selects for one that maximizes for realism and emotion. Furthermore, [35] uses a genetic algorithm to influence specific LSTM units that learn to encode sentiment in a pre-training language modeling stage, allowing it to steer the passages that it generates towards a desired affective state. In [36], the authors use pre-defined mood-tags associated with each chord in a progression to guide the generative process step-by-step. The EMOPIA dataset [37] contains musical passages separated into four quadrants that each corresponds to a combination of positive or negative arousal and valence. The excerpts on the dataset are from piano transcriptions of pop songs that were labeled by its authors. In this same work, the authors also use a Transformer model that employs the Compound Word representation [38] to generate songs conditioned by affective states.

One of the factors that influence the final quality of the samples generated by models of symbolic music is the representation used to train them. For contemporary styles, like Pop or Hip-Hop, where a rigid metrical grid is often followed, it is desirable to incorporate data about the rhythmic structure of the songs into the representation. REMI [39], which stands for revamped MIDI-derived events, is a beat-based approach to modeling music that encodes this information through tokens that signal the beginning of a bar and the passage of each beat, while still maintaining some flexibility by allowing local tempo changes. More specifically, there NOTE_ON, NOTE_DURATION, VELOCITY, TEMPO, BAR and BEAT. A NOTE_ON event indicates the start of a note, NOTE_DURATION corresponds to the duration of that note, and VELOCITY is a parameter that indicates the intensity with which the keys on a piano are played, and correlates to the volume of the note produced. Finally, TEMPO dictates the tempo of the musical excerpt from the moment the token is produced forward, and BAR events indicate the start of a new bar. The Compound-Word Transformer [38] uses a separate embedding for each type of musical element (pitch, chord, tempo value, etc.).

3. METHODS

3.1 Architecture and Loss functions

Both the Generator and Discriminator are Transformers with linear versions of the Attention Mechanism [18]. Each of the models is composed by 6 Attention blocks.

The Generator has two roles. In the first place, it has to predict each item of each sequence from the real dataset based on the previous elements, that is, it has to complete pieces of already existing sequences. In standard Transformer fashion, a lower triangular or look-ahead mask is applied to the original sequence in order to prevent the model from seeing its future elements. The second objective of the network is to generate sequences that are similar to those of the real set from scratch, that is, without context from the real dataset, such that these sequences can fool the Discriminator. These sequences are generated step-by-step in an autoregressive manner, which is done via the Transformer-RNN analogy made in [18]. ¨

The Generator is conditioned via special scale and bias parameters that influence the Layer Normalization [40] layers existing within the Attention mechanism. There is one of these couples for each class on the dataset and one for the unlabeled sequences. Formally, if i, k and c stand, respectively, for position in the sequence, feature channel and class label, we have:

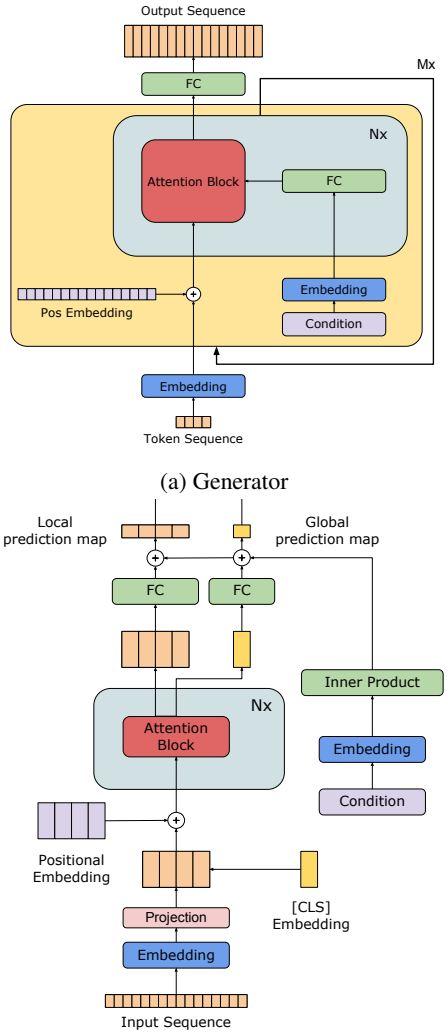
$$s'_{i,k} = \gamma^c_k \cdot s_{i,k} + \beta^c_k \tag{8}$$

where s and s' are input and output sequences, and γ and β are scale and bias.

The Discriminator takes each sequence as a whole and tries to determine if it is real or fake. To design this network, we took inspiration from the Visual Transformer [15], separating the sequences into patches of a certain length and transforming each patch into a single feature vector.

Our Discriminator has two outputs. First, there is a single feature unit that indicates if the passage originates from the real or fake datasets and if it exhibits the desired characteristics provided by the conditional signal. To produce this output, a [CLS] token is concatenated to the input sequence, similarly to BERT [29]. Then, the conditional information is incorporated via an inner product between an embedding of this information and the [CLS] representation. This essentially means that the model works as a Projection Discriminator [41]. The second output is a prediction map where each unit corresponds to a single patch in the sequence, that is, for every collection of musical symbols with length equal to the patch size, there is a value predicting whether that patch is real or fake. This technique, often used in image generating-GANs [42], ensures that the model prioritizes local structure.

Each of these two outputs serves the purpose of incorporating priors about musical structure into our model. A common way to frame the task of musical generation is as a language modelling one: each individual symbol is treated as a word, and these words compose phrases, periods, and so on. Using this analogy, the goal behind the proposed local loss is to inform if each particular sentence in a text is realistic or not, or, translating that to music, if every short musical idea in the form of a phrase or part of a phrase is realistic or not. The global prediction unit, on the other hand, acts as a signal that encapsulates the overall quality of the sequence and its ability to convey the desired emotional state, complementing the local prediction map. The networks are illustrated in Figure 1.



(a) Generator

(b) Discriminator

Figure 1: Generator and Discriminator. FC stands for Fully-Connected Layer. N is the number of self-attention blocks from which the models are made. M represents the number of autoregressive steps the Generator executes, that is, the sequence length.

3.2 Datasets

We used two datasets to train our models, mainly as a means to allow the networks to use a larger training corpus. Both consist only of songs performed on piano. The AILABS17k dataset [38] contains over 108 hours of piano covers of pop songs automatically transcribed by a state-of the art piano transcription model [43] and converted into MIDI files. The EMOPIA dataset [37] was constructed in

a similar fashion to the one above, but its songs were afterwards labeled by the authors of the dataset according to perceived sentiment [2]. The clips on this dataset amount to approximately 11 hours. We used AILABS 17K to allow the networks to learn a general representation of music in a larger corpus, while also being trained on a smaller collection of songs that contains annotated data.

3.3 Training

We use the data representation proposed in [39], which consists of the NOTE_ON, NOTE_DURATION, VELOCITY, TEMPO, BAR and BEAT events described previously. Before the introduction of the Discriminator, the Generator was trained until convergence through the teacher forcing method (26000 steps). This pre-training stage guaranteed the stability of the adversarial stage that was to follow [13, 27, 44]. In this stage, the Generator was trained simultaneously on both datasets, and taking into consideration the difference in size between these datasets, to balance the training process, we alternated between optimization steps on randomly sampled batches from each set. The network worked with sequences of length 2048, corresponding, on average, to 1 minute of content.

For the adversarial stage, we used sequences of size 128. In order to reduce the effects of gradient variance that are inherent to the sampling process, sequences with length 16 originated from the real set were given to the Generator such that it could have a starting point to perform generation [27]. The Discriminator worked with subsequences of length 16, and the training was done exclusively on the EMOPIA dataset [37].

The networks were trained via a combination of the teacher forcing objective plus the RSGAN objective [8] with gradient penalty [9]. We performed 1 optimization step of the Discriminator per Generator step, using a learning rate of $1 \cdot 10^{-4}$. In total, 26000 global optimization steps were performed. The overall objective for the generator in this training stage is:

$$\mathcal{L}_G = \mathcal{L}_{\text{MLE}} + \alpha\mathcal{L}_{\text{RSGAN}_G\text{-global}} + \beta\mathcal{L}_{\text{RSGAN}_G\text{-local}}, \quad (9)$$

$$\text{where } \mathcal{L}_{mle} = -\mathbb{E}_{x\sim P}[\log G_\theta(x)] \quad (10)$$

where the factors $\mathcal{L}_{\text{RSGAN}_G\text{-global}}$ and $\mathcal{L}_{\text{RSGAN}_G\text{-local}}$ are respectively the global and local GAN losses detailed above, α and β, which we empirically chose to be equal to 1, are hyperparameters controlling the relative intensity of each loss factor, and $\mathcal{L}_{\text{MLE}}$ is the Maximum Likelihood. The Discriminator was simply trained with the local and global RSGAN objectives given previously in Equation 2 plus the global and local gradient penalties based on Equation 4 and regulated by a hyperparameter λ (which as per [9], we chose to be 10) . This loss is expressed as Equation 11. An algorithm outlining the training scheme is available in the supplementary material.

$$\mathcal{L}_D = \mathcal{L}_{\text{RSGAN}_D\text{-global}} + \beta\mathcal{L}_{\text{RSGAN}_D\text{-local}} + \lambda(\mathcal{L}_{\text{GP-global}} + \mathcal{L}_{\text{GP-local}}). \quad (11)$$

4. EXPERIMENTS

We evaluated our models both with respect to the overall quality of the samples they produce and their ability to generate songs that convey the conditioning emotional signals. To achieve this purpose, we used both the automatic evaluation metrics proposed in [26, 45] and a set of human evaluation metrics to compare our GAN with the system that, as far as we know, corresponds to a state-of-the-art generative model of symbolic music conditioned by sentiment currently available in the literature. Specifically, our model was compared with the Compound-Word Transformer [38] variation used by the authors of the EMOPIA article [37] to generate music conditioned by emotional class. We also compared our adversarially trained model with a Vanilla Transformer model that was not trained with the adversarial scheme. This model corresponds to the version of the generator network obtained after the pretraining was completed. Code for this work, along with audio samples, are available at [1] .

For the automatic metrics, we chose Pitch Range (distance between highest and lowest pitch), Number of Pitch Classes, and Polyphony (the average number of simultaneous notes). These metrics were calculated using the Muspy library [46]. For each model, we generated 400 samples (100 for each class) and evaluated these samples with respect to the characteristics above, then averaged the results to produce the overall model score. The results are presented in Table 1.

	PR	NPC	POLY
Real Data (EMOPIA) [37]	50.94	8.50	5.60
Baseline [37]	49.76	**8.52**	4.36
Transformer	48.79	8.65	4.37
Transformer GAN	**50.73**	9.45	**4.43**

Table 1: Comparison between the samples generated by ours and a state-of-art model. PR stands for Pitch Range, NPC is Number of Pitch Classes and POLY is Polyphony. The best results are highlighted in bold.

As it can be seen, our model obtains a superior performance than that of the baseline and the pre-trained model on two of the three metrics. Furthermore, it should be mentioned that the Generator is significantly smaller than the baseline, having only 6 Attention layers, while the latter has 12. To be more precise, the baseline has $\sim 40M$ parameters, while our generator has $\sim 25M$ and our Discriminator has $\sim 27M$. Also relevant is the fact that the representation used to train the baseline, that is, the Compound-Word representation [38], is more complex than the REMI representation that we use [39], and has also been proved to be better than REMI. Since the focus of our work was to implement the GAN framework within the context of symbolic music generation conditioned by sentiment, and given the complexity of this framework, especially when it is applied to the discrete domain, we chose to use a simpler representation in order to maintain the focus of our work on the implementation of the GAN. But as

[1] http://github.com/pneves1051/transformers_sentiment

our results show, adapting it to work within the adversarial context, which we leave to future work, could bring about even better results.

Finally, we performed a survey in which participants were asked to judge the musical excerpts generated by the models both in terms of their overall quality and their ability to convey the desired sentiments. Specifically, participants rated the samples on a 5-point Likert scale that ranged from very low to very high with respect to the following characteristics: Human-likeness, Originality, Structure, Overall Quality, Valence, and Arousal. Details from each characteristic and the corresponding scale are given in the supplementary material. The participants were recruited from the researchers' online circles. Each of the participants had to listen to 12 musical excerpts in total, 4 from each model, and within those 4 item groups, one from each of the 4 emotional classes. Before the test, some text explaining the basic concepts behind the research was shown to the participants. In total, 18 individuals participated in the experiment.

We present the average participants' scores given to each model for Human-likeness, Originality, Structure and Overall Quality in Table 2. These results suggest that both the proposed Transformer, and the Transformer GAN, are competitive with a state-of-the-art model with respect to the four qualitative metrics.

On the next step of the evaluations, we took the participants' answers to the questions related to Valence and Arousal and compared them to the real emotional labels provided to the model during the generation process. Figure 2 illustrates the results of this experiment.

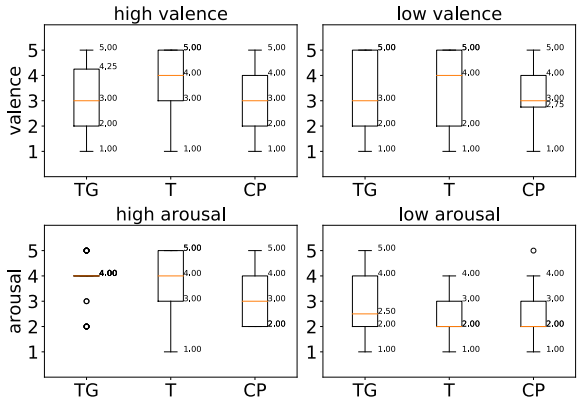


Figure 2: Results of the experiment where participants rated musical samples according to their perceptions about valence and arousal. The acronyms TG, T and CP correspond, respectively, to the Transformer GAN, Transformer and Compound-Word Transformer Baseline models.

Once again, we find that the models we developed obtain a performance that competes with that of the current state-of-the-art. By comparing the medians and quartiles of these boxplots, we can draw several conclusions. Firstly, all models seem to have more difficulty capturing valence than they do arousal. This may be an indication that musical aspects which have a great impact over arousal, such as velocity and tempo, are more easily understood by neural networks, while factors which have an impact over valence, such as modality or frequency of harmonic change [47], are more difficult to model for these systems.

Furthermore, we see that, in general, our models do not stand behind when compared to the baseline. Between the three, by observing the boxplots, it seems that while the Transformer and the Compound-Word baseline sometimes produce samples that situate themselves more strongly to the side to which they theoretically pertain (e.g., for the high valence and low arousal categories), the Transformer GAN surpasses the simple Transformer due to the fact that it never situates more than 50% of the excerpts on the incorrect side of the middle line, which in this case is represented by the number 3. While the excerpts generated by the Compound Word Transformer also show this same characteristic, we see that there is not a strong reason for us to believe that one stands out in comparison to the other.

Overall, given the superior ratings of the Transformer GAN respective to the automatic metrics and its competitiveness with a state-of-the art model with respect to the human evaluations, and given the considerations above about model size and representation, the Transformer GAN seems to be a promising model for music generation conditioned by sentiment.

4.1 Conclusion and future work

We introduced a model capable of generating musical excerpts conditioned by labels that represent perceived emotion. Through the use of MLE pre-training and adversarial training, we guided our model towards understanding some aspects of the relationship between musical structure and affect. Our experiments show that both in terms of quality and the ability to communicate emotion, the samples generated by the proposed model achieve competitive results. Furthermore, our work points to several possible avenues for future research, such as the use of other symbolic representations of music, the development of generative models conditioned by emotion that work directly on audio, and further exploration of the use of affective conditioning signals, e.g., using them to guide automatic composition second-by-second or to automatically generate royalty free music notation based on specific sentiments.

	H	O	S	OQ
Baseline [37]	3.32 ± 1.29	2.93 ± 1.13	3.18 ± 1.30	3.49 ± 1.04
Transformer	3.75 ± 1.24	3.22 ± 1.19	3.76 ± 1.14	3.89 ± 1.14
Transformer-GAN	3.56 ± 1.34	3.06 ± 1.21	3.38 ± 1.09	3.44 ± 1.15

Table 2: Results of the Survey where participants were asked to rate the samples generated by several models. The columns are, respectively, Human-Likeness, Originality, Structure and Overall Quality.

5. REFERENCES

[1] Y. Kim, E. Schmidt, R. Migneco, B. Morton, P. Richardson, J. Scott, J. Speck, and D. Turnbull, "Music emotion recognition: A state of the art review," *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010*, 01 2010.

[2] J. A. Russell, "A circumplex model of affect." *Journal of personality and social psychology*, vol. 39, no. 6, p. 1161, 1980.

[3] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," 2020.

[4] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[5] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, p. 1171–1179.

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, 06 2014.

[7] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[8] A. Jolicoeur-Martineau, "The relativistic discriminator: a key element missing from standard GAN," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=S1erHoR5t7

[9] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/892c3b1c6dccd52936e27cbd0ff683d6-Paper.pdf

[10] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[11] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. [Online]. Available: https://openreview.net/forum?id=rkE3y85ee

[12] M. J. Kusner and J. M. Hernández-Lobato, "Gans for sequences of discrete elements with the gumbel-softmax distribution," 2016.

[13] A. Muhamed, L. Li, X. Shi, S. Yaddanapudi, W. Chi, D. Jackson, R. Suresh, Z. C. Lipton, and A. J. Smola, "Symbolic music generation with transformer-gans," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 408–417.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[15] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[16] H. Akbari, L. Yuan, R. Qian, W.-H. Chuang, S.-F. Chang, Y. Cui, and B. Gong, "Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text," 2021.

[17] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[18] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5156–5165.

[19] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rkgNKkHtvB

[20] K. M. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller, "Rethinking attention with performers," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=Ua6zuk0WRH

[21] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[22] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=rJe4ShAcF7

[23] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 464–468. [Online]. Available: https://aclanthology.org/N18-2074

[24] C. Payne, ""musenet."," 2019.

[25] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "Midinet: A convolutional generative adversarial network for symbolic-domain music generation," in *ISMIR*, 2017.

[26] H.-W. Dong and Y.-H. Yang, "Convolutional generative adversarial networks with binary neurons for polyphonic music generation," in *ISMIR*, 2018.

[27] N. Zhang, "Learning adversarial transformer for symbolic music generation," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–10, 2020.

[28] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 2978–2988. [Online]. Available: https://aclanthology.org/P19-1285

[29] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL*, 2019.

[30] R. Madhok, S. Goel, and S. Garg, "Sentimozart: Music generation based on emotions," in *ICAART*, 2018.

[31] K. Zhao, S. Li, J. Cai, H. Wang, and J. Wang, "An emotional symbolic music generation system based on lstm networks," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. IEEE, 2019, pp. 2039–2043.

[32] X. Tan, M. Antony, and H. Kong, "Automated music generation for visual art through emotion." in *ICCC*, 2020, pp. 247–250.

[33] H. H. Tan and D. Herremans, "Music fadernets: Controllable music generation based on high-level features via low-level feature modelling," in *Proc. of the International Society for Music Information Retrieval Conference*, 2020.

[34] L. Ferreira, L. Lelis, and J. Whitehead, "Computer-generated music for tabletop role-playing games," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 16, no. 1, 2020, pp. 59–65.

[35] L. N. Ferreira and J. Whitehead, "Learning to generate music with sentiment," *Proceedings of the Conference of the International Society for Music Information Retrieval*, 2019.

[36] D. Makris, K. R. Agres, and D. Herremans, "Generating lead sheets with affect: A novel conditional seq2seq framework," *arXiv preprint arXiv:2104.13056*, 2021.

[37] H.-T. Hung, J. Ching, S. Doh, N. Kim, J. Nam, and Y.-H. Yang, "EMOPIA: A multi-modal pop piano dataset for emotion recognition and emotion-based music generation," in *Proc. Int. Society for Music Information Retrieval Conf.*, 2021.

[38] W. Hsiao, J. Liu, Y. Yeh, and Y. Yang, "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs," *CoRR*, vol. abs/2101.02402, 2021. [Online]. Available: https://arxiv.org/abs/2101.02402

[39] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 1180–1188. [Online]. Available: https://doi.org/10.1145/3394171.3413671

[40] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[41] T. Miyato and M. Koyama, "cGANs with projection discriminator," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ByS1VpgRZ

[42] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[43] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, 2018*, 2018. [Online]. Available: https://arxiv.org/abs/1710.11153

[44] W. Nie, N. Narodytska, and A. Patel, "Relgan: Relational generative adversarial networks for text generation," in *International conference on learning representations*, 2018.

[45] L.-C. Yang and A. Lerch, "On the evaluation of generative models in music," *Neural Computing and Applications*, vol. 32, no. 9, pp. 4773–4784, 2020.

[46] H.-W. Dong, K. Chen, J. McAuley, and T. Berg-Kirkpatrick, "Muspy: A toolkit for symbolic music generation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020.

[47] P. N. Juslin, J. A. Sloboda *et al.*, "Music and emotion," *Theory and research*, 2001.

IMPROVING CHORAL MUSIC SEPARATION THROUGH EXPRESSIVE SYNTHESIZED DATA FROM SAMPLED INSTRUMENTS

Ke Chen[1] **Hao-Wen Dong**[1] **Yi Luo**[2] **Julian McAuley**[1]
Taylor Berg-Kirkpatrick[1] **Miller Puckette**[1] **Shlomo Dubnov**[1]
[1] UC San Diego, USA [2] Tencent AI Lab, China
[1]{knutchen, hwdong, jmcauley, tberg, msp, sdubnov}@ucsd.edu [2]oulyluo@tencent.com

ABSTRACT

Choral music separation refers to the task of extracting tracks of voice parts (e.g., soprano, alto, tenor, and bass) from mixed audio. The lack of datasets has impeded research on this topic as previous work has only been able to train and evaluate models on a few minutes of choral music data due to copyright issues and dataset collection difficulties. In this paper, we investigate the use of synthesized training data for the source separation task on real choral music. We make three contributions: first, we provide an automated pipeline for synthesizing choral music data from sampled instrument plugins within controllable options for instrument expressiveness. This produces an 8.2-hour-long choral music dataset from the JSB Chorales Dataset and one can easily synthesize additional data. Second, we conduct an experiment to evaluate multiple separation models on available choral music separation datasets from previous work. To the best of our knowledge, this is the first experiment to comprehensively evaluate choral music separation. Third, experiments demonstrate that the synthesized choral data is of sufficient quality to improve the model's performance on real choral music datasets. This provides additional experimental statistics and data support for the choral music separation study.

1. INTRODUCTION

Choral music is a distinct artistic genre that includes several vocal parts (e.g. soprano, alto, tenor, and bass) arranged into intricate patterns from strict counterpoint to polyphonic echoes and flows of lyrics. One useful tool in the analysis and re-production of choral tracks is the ability to take mixed-down choral music and separate it back into audio tracks of isolated vocal parts: i.e. choral music separation, as a subtask of audio source separation.

Audio source separation is an audio signal processing task that involves separating one or more sound sources from a multi-source audio mixture. This task has a wide range of applications in a variety of domains, including speech separation, vocal-accompaniment separation and musical instrument separation. The latter two tasks are primary tasks in the field of music signal processing and have been adopted for practical use in the entertainment industry [1]. Many models, such as Open-Unmix [2], Demucs [3], and Spleeter [4], achieve great separation performance. Some models [5] further extend the source separation task to a zero-shot or query-based setting. However, choral music separation has received limited attention. Unlike general musical instrument separation, which seeks to separate non-homologous sources (e.g., piano, drums, and singing voice), choral music instrument separation seeks to separate homologous or close-homologous sources (e.g., soprano, alto, tenor, bass). Additionally, the scarcity of data on choral music separation impedes further progress. Choral music separation could be used in a wide variety of scenarios. Individuals could obtain solo tracks from choral recordings for practice, analysis, and re-production. Not only does it fill a void in a particular type of musical instrument separation, but it also provides convenience for music educators.

In this paper, we investigate the choral music separation task from the perspective of addressing the insufficiency of available datasets. We begin by introducing related works in the field of choral music separation. Second, we present the discovery of how to improve the performance of choral music separation using high-quality, synthesized music data. Then, we conduct comprehensive experiments with multiple models and datasets to evaluate the improvement of using synthesized data on choral music separation in real datasets. Finally, we discuss the extensibility of our pipeline to more choral-related separations, such as string quartet separation, as well as its future directions. The code and the dataset are publicly available [1] .

2. RELATED WORK

Research in choral music separation receives relatively less attention. Deep learning methods for audio source separation has already outperformed traditional methods (e.g. Non-negative Matrix Factorization [6]) for a long time. Separation models have been developed following two directions: frequency-domain models and time-domain separation models.

© K. Chen et al.. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** K. Chen et al., "Improving Choral Music Separation through Expressive Synthesized Data from Sampled Instruments", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] https://github.com/RetroCirce/Choral_Music_Separation

Dataset	Minutes	Songs	Public
Choral Singing Dataset [11]	7	3	✓
Dagstuhl ChoirSet [12]	5	2	✓
Cantoria Dataset [13]	20	14	✓
ESMUC Choir Dataset [13]	31	26	✓
Bach and Barbershop Collection [14]	105	48	

Table 1: Existing datasets for choral music separation.

2.1 Frequency-Domain and Time-Domain Models

The traditional method of audio separation is to mask the frequency-domain representation and then inversely transform it to the time-domain signal, referred as frequency-domain separation models. Spec-U-Net [7], based on the U-Net architecture, contains convolutional neural network (CNN) blocks for downsampling the input short-time Fourier transform (STFT) spectrogram and upsampling the bottom feature back into a separation mask. The mask is applied into the input to obtain the separate spectrogram as output. Res-U-Net [8] replaces the original CNN blocks with residual CNN blocks to accelerate convergence speed. On the other hand, time-domain models perform the separation directly on the audio waveform. Wave-U-Net [9] incorporates an end-to-end U-Net structure on the input and output of waveforms. Conv-TasNet [10] applies a CNN encoder-decoder structure to process waveforms into latent features and generates the mask. The masked latent features are decoded back to waveforms as separation results. Bypassing the spectrogram processing, time-domain models can save parameters and perform efficiently in low-latency systems for speech separation. Some hybrid models, such as Demucs v3 [3], can leverage both time-domain and frequency-domain features to achieve the best performance for musical instrument separation, while the size of the model is a little bit large.

2.2 Choral Music Separation

For choral music separation, [15] proposed a score-informed separation model based on Wave-U-Net and performed experiments on 347 (synthesized) Bach Chorale pieces from MIDI files with *MuseScore_General* SoundFont. This model performs well on this SoundFont-Synthesis dataset but poorly on real choral music datasets. [16] proposed a conditional Spec-U-Net to optimize the separation performance by conditioning on the fundamental frequency contour. However, as mentioned in their paper, due to the lack of choral music datasets, the evaluation was conducted on only three songs with a total duration of seven minutes. [14] proposed a harmonic overlap score to increase the model's sensitivity to different choral voices, thereby improving performance. It made use of a relatively large dataset containing 105 minutes of Bach and Barbershop Collections, but this dataset is *not* publicly available due to copyright concerns, which prevents it from being open source. And indeed, 100-minute is still not enough to help achieve an audio separation model with a high generalization ability, we expect to obtain a size more than that.

		Pitch range			
Name	Type	Soprano	Alto	Tenor	Bass
Standard MIDI		A0–C8	A0–C8	A0–C8	A0–C8
Noire [17]	Piano	A0–C8	A0–C8	A0–C8	A0–C8
Grandeur [18]	Piano	A0–C8	A0–C8	A0–C8	A0–C8
Voices Of Rapture [19]	Vocal	B3–D6	E3–G5	B2–C#5	A1–D4
Dominus Choir [20]	Vocal	G3–A5	G3–A5	E2–G4	E2–G4

Table 2: Sample instrument libraries we use for synthesizing choral music separation datasets.

In this paper, we first conduct four fundamental models: Spec-U-Net, Res-U-Net, Wave-U-Net and Conv-TasNet. Our objective is to demonstrate the efficacy of synthesized expressive data in improving separation performance on **real choral music datasets**. As a result, fundamental models enable us to consider the performance gains more directly associated with data changes and augmentations. Also, score-informed and conditional separation models introduce external information, such as musical notes of original songs or multi-pitch estimation results, to guide the separation's goal, while it also limits its applications. In practice, we frequently find ourselves in situations where the only available input is the audio. We continue to demand **unconditioned choral music separation**. As a result, we proceed directly to unconditioned choral music separation in this paper, without relying on any score conditions.

3. METHODOLOGY

3.1 Scarcity of Datasets

Existing datasets for choral music are listed in Table 1, collected from previous works and other public sources. We observe that most of these datasets have short total lengths; three of them are less than 20 minutes. The Choral Singing Dataset [11] and ESMUC Choir Dataset [13] have been used for choral music separation by [16], while the Dagstuhl ChoirSet [12] and Cantoria [13] Datasets were never used for separation tasks but instead for singing performance analysis. The Bach and Barbershop Collection [14] is relatively longer, but is not publicly available. As a result, when a model is trained and tested on such a small amount of data, its generalization and separation capabilities are severely limited. [15] directly synthesized 347 choral pieces of Bach's from MIDI files with *MuseScore_General* SoundFont and trained the model. However, this SoundFont-Synthesis dataset is dissimilar to true choral vocals. Moreover, it lacks lyrics and syllables. As a result, the trained model performs poorly on real datasets [15].

In the next section, we first introduce the pipeline of synthesizing audio datasets for choral music separation from sampled instrument libraries. Then, we train various models on our datasets and compare them to determine the best model. Finally, we transfer the best model weights to the real-world datasets shown in Table 1, fine-tune the model and determine whether it truly improves

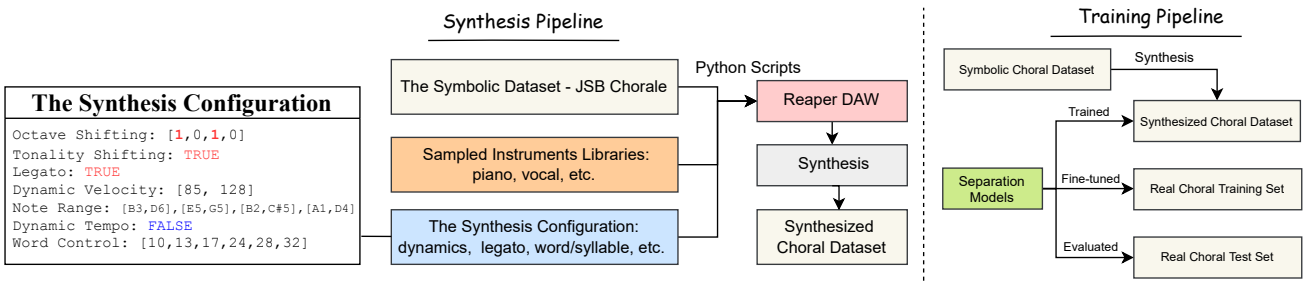


Figure 1: The synthesis pipeline of choral music data from sample instruments and the training pipeline to utilize it.

performance when compared to the previous settings.

3.2 Data Synthesis Pipeline

Figure 1 shows our pipeline of choral music dataset creation and training methods. Generally, we need three collection steps:

1. The symbolic choral music dataset (MIDI, MusicXML)
2. The sampled instrument libraries (standalone VST plugin, or Kontakt sample libraries)
3. The synthesis configuration (syllables or lyrics choices, legato, velocity, and tempo)

Then, our provided code can completely automate the data synthesis process. It is built on top of the python-support and free digital audio workstation (DAW) – Reaper [2] . With the above three steps, one could complete any choral music data synthesis process on supported system platforms.

3.3 Data and Instrument Collection

For Step 1, following [15], we use the JSB Chorales Dataset [21], which contains 347 pieces of choral music in MusicXML format. The total duration is 248 minutes at a tempo of 90 bpm (a.k.a. beat per minute). The data is first transformed into MIDI files, which serves as the symbolic dataset source for the creation of choral music audio. For Step 2, a sampled instrument [3] is a sound source plugin applied in a DAW. Unlike a SoundFont, it contains samples of real instruments recorded in a professional acoustic environment. The human singing voice is also considered as an instrument type. And many vocal sampled instruments support a variety of lyrical or syllabic sets (e.g., vowels, Latin words, etc.). We first choose two types of instruments for our purposes: piano and vocal (soprano, alto, tenor, and bass). Then, we choose two sampled instruments for each type, as shown in Table 2. The reason to choose the piano instrument is to evaluate the separation performance of piano as a common and same-source instrument. In this case, the model needs to consider most on the pitch difference between each voice part. When it comes to the vocal dataset, the model can distinguish the timbres slightly between soprano, alto, tenor, and bass voices, but it is more difficult to model the acoustic features of these four voices than piano. This allows us to determine whether the model can improve performance by exploiting the timbre difference between vocal datasets, or if it fails to model these timbres and perform a bad separation result.

Due to the fact that we have two sampled libraries for each type of instrument, each dataset contains $248 \times 2 = 496$ minutes (8.2 hours) of synthesized choral music data. All sampled instrument libraries that we use have a paid license.

3.4 Synthesis Configuration for Expressiveness

For Step 3, we adopt two methods to further improve the scalability and quality of synthesized data: the basic data augmentation, and expressiveness incorporation. The left of Figure 1 shows a specification.

We perform two operations to augment the data. First, we notice that the pitch ranges of sampled instrument libraries do not always correspond exactly to the pitch ranges of tracks in JSB Chorales Dataset. For instance, some bass melodies in JSB appear to be lower than the lowest note in sampled instrument libraries. Instead of directly discarding these tracks, we implement "octave shifting" by shifting out-of-range notes up or down some octaves until they fall within the range. While it produces non-realistic jumps between some melodies, it saves the whole track to preserve more realistic data. Second, we apply "tonality shifting" to each track. The tonality was shifted upward and downward by three semitones before synthesis. Therefore, the effective length of training data will be further augmented several times.

For expressiveness incorporation, we provide several options, which are supported by sampled instrument libraries, to synthesize audio:

- Legato: for vocal, this includes whether or not to change breath or sing continuously. In vocal instrument libraries, legato is controlled by detecting the presence of an overlap between adjacent notes. To support the legato configuration, we begin by segmenting the track into *musical phrases* using the breath break information in MusicXML (if provided) or the note intervals (if specified). Then, in each phrase, we add overlap to adjacent notes (to activate the legato) if their pitch distance is less than 7 semitones (i.e., a perfect fifth).

[2] https://www.reaper.fm/
[3] A detailed introduction can be found at https://tinyurl.com/2p8trn2u

- Velocity: for each phrase, we provide three types of volume/velocity change curves: *crescendo*, *diminuendo*, and *cresc.→dim.*. The configuration establishes the maximum and minimum velocity ranges.

- Word Control: for vocal, we support the word control of sampled instrument libraries by assigning random combinations of words or syllables to each phrase. Note that real-world performance may not contain random word changes, but for model training, this still increases the data richness on each training batch.

The configuration also supports the reverberation as a designed feature, but currently it is not applied in this work.

4. EXPERIMENTS

In this section, we conduct an experiment on evaluating different separation models on our synthesized datasets. The purpose of this experiment is to identify the best model on synthesized datasets and then transfer it to real choral music datasets.

4.1 Datasets, Models and Hyperparameters

As introduced above, we use two datasets (piano and vocal) to train four models (Spec-U-Net, Res-U-Net, Conv-TasNet, and Wave-U-Net). Each track is in 22,050 Hz sample rate. Each dataset contains 496 minutes data. We use 277 tracks for training, 35 tracks for validation and another 35 tracks for testing. Since we have training combinations among four models, two datasets, and four choral voices, to save training time, we first train models on the dataset *without* the expressiveness incorporation in section 3.4, named as **Standard-Piano** and **Standard-Vocal** datasets. After finding the best model, we will train it on the expressive datasets in section 4.3.

For model hyper-parameters, In Spec-U-Net [7], we use a window size of 2048, FFT size of 2048, and hop size of 441. We apply 7 CNN blocks to downsample the input spectrogram, and another 7 CNN blocks to upsample it into the separation mask. In Res-U-Net, we apply the implementation from [8], with 10 residual CNN blocks to downsample the input spectrogram, then another 6 residual CNN blocks to upsample. In Wave-U-Net, we follow the settings of [9] to adopt 6 downsampling CNN layers and 6 upsampling CNN layers for separation. The filter channels are set from 32 to 1024 in order for each layer. The kernel size is 15 for the first layer and 5 for remaining layers. In Conv-TasNet, we follow the setting of [10] to set hyper-parameters as $N = 512$, $L = 20$, $B = 128$, $H = 512$, $P = 3$, $R = 3$, $X = 8$. Spec-U-Net and Res-U-Net use their default mean absolute error (MAE) loss function; Conv-TasNet uses the default scale-invariant source-to-noise ratio (SI-SDR) loss; and Wave-U-Net with mean squared error (MSE) loss.

For training hyperparameters, the batch size is 8, the learning rate is 1e-3, and each training sample is a 2-sec audio segment randomly chosen from one music track in the training set. The number of steps for each epoch is 700. We apply the Adam optimizer [22] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$, and a learning rate scheduler where the learning rate is reduced with a multiplier $f = 0.65$ if the validation performance does not improve across 3 consecutive epochs. We implemented all methods in Pytorch using NVIDIA RTX 2080Ti GPUs. All models converged within 300 epochs with early stop using a 10-epoch patience.

Standard Dataset	Model	Median Source-Distortion Ratio (dB)				
		Soprano	Alto	Tenor	Bass	Avg.
Piano	Spec-U-Net [7]	**9.78**	**9.46**	**10.35**	10.60	**10.05**
Piano	Res-U-Net [8]	8.53	9.01	9.97	**12.23**	9.94
Piano	Wave-U-Net [9]	6.95	5.36	7.21	9.82	7.34
Piano	Conv-TasNet [10]	7.04	6.98	7.29	7.82	7.28
Vocal	Spec-U-Net [7]	**10.45**	10.19	**12.25**	9.53	**10.61**
Vocal	Res-U-Net [8]	9.35	**10.87**	10.20	**10.77**	10.30
Vocal	Wave-U-Net [9]	2.65	3.08	3.06	3.90	3.17
Vocal	Conv-TasNet [10]	6.60	6.12	6.41	6.58	6.43

Table 3: The separation performance of four models on the test sets of Standard-Piano and Standard-Vocal datasets.

For evaluation, source-to-distortion ratio (SDR) is one of the most widely used metrics for evaluating a source separation system's output, which measures a ratio between the original source track and the noise, interference, added artifacts in the separation track. It is considered to be an overall measure of how good a separation result sounds. We follow the music separation campaign SiSEC 2018 [23] to use the median SDR to evaluate separation performance. The median SDR is obtained by first computing segment-level SDR of each 2-sec segment in each track, then taking the median over them as track-level SDRs, finally taking the median over the track-level SDRs as the final SDR. The computing library is *mus_eval* [23].

4.2 Separation Performance

Table 3 shows all four parts median SDR performance on two standard datasets by four models. We can see that the frequency-domain models Spec-U-Net and Res-U-Net get similar results that are better than those of the time-domain models Conv-TasNet and Wave-U-Net. Spec-U-Net achieves the best average SDRs over four parts on two datasets as 10.05 and 10.61. The Res-U-Net achieves very close performance. When analyzing the results, frequency-domain models can take advantage of spectrograms in choral music to obtain better separation results. The performance on soprano, alto and tenor in vocal is better than that in the piano dataset, suggesting that the timbre difference can also help further discriminate different source tracks. Time-domain models can model the piano acoustic features well to achieve a good performance, but find it hard to model the vocal features solely on the waveform and face the drops in the vocal dataset.

4.3 Fine-Tuning Evaluation on Real Datasets

After comparing models in standard datasets, we chose the best model, Spec-U-Net, to conduct the next experiments. We trained Spec-U-Net on the **Expressive-Vocal** dataset, as we synthesized the data *with* the expressiveness incorporation. Then, as shown in the right of Figure 1, we saved

Fine-Tuning Evaluation Set	Pretraining Set	Avg. Median SDR (Fine-Tuning Ratio)		
		ratio=10%	ratio=40%	ratio=70%
Cantoria Dataset [13]	None	1.42	3.91	4.13
	SoundFont-Synthesis [15]	2.39	3.90	4.03
	Standard-Vocal (ours)	3.03	4.59	5.08
	Expressive-Vocal (ours)	3.73	5.48	5.71
Choral Singing Dataset (CSD) [11]	None	1.98	2.78	5.26
	SoundFont-Synthesis [15]	2.12	3.38	6.20
	Standard-Vocal (ours)	3.43	4.23	6.91
	Expressive-Vocal (ours)	4.19	4.78	7.50
Bach & Barbershop Collection (BBC) [14]	None	4.18	6.08	6.94
	SoundFont-Synthesis [15]	4.19	6.17	6.93
	Standard-Vocal (ours)	4.98	6.71	7.27
	Expressive-Vocal (ours)	5.58	7.17	7.64

Table 4: The fine-tuning performance of three real datasets by our best model – Spec-U-Net.

the best model checkpoints, and conducted a fine-tuning experiment to verify whether our data is useful for transfer learning on real choral music datasets. Table 4 and Figure 2 illustrate the median SDR performance of three real choral music datasets under different fine-tuning ratios with different pretrained models.

For datasets, we chose the Cantoria Dataset, Choral Singing Dataset (CSD), and Bach & Barbershop Collection [4] (BBC). The reason for these choices is that Cantoria contains the best recording quality, CSD is most frequently used in previous works, and BBC contains the longest length. The meta information of each dataset has been described in Table 1.

We considered three ratios for fine-tuning: (1) 10% for training, 90% for evaluation; (2) 40% for training, 60% for evaluation; and (3) 70% for training, 30% for evaluation. The fine-tuning experiments demonstrate if our synthesized datasets can improve the separation performance in real datasets under different settings (e.g., few-shot as 10% and fairly enough as 70%). The intermediate ratio 40% is conducted to further investigate the tendency of the improvement brought by our datasets.

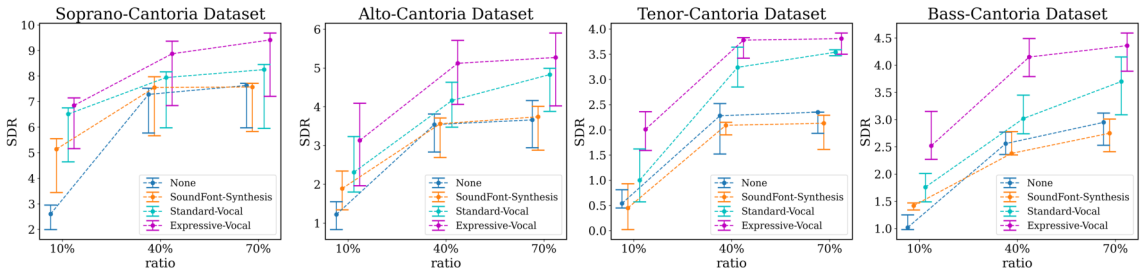
There are four dataset choices on which to pretrain the models: (1) None: without any pretraining; (2) SoundFont-Synthesis: the synthesis dataset in [15] by the *Musescore_General* SoundFont as a baseline; (3) the Standard-Vocal dataset; and (4) the Expressive-Vocal dataset. Since the SoundFont-Synthesis dataset only contains 248 minutes, instead of using two sampled libraries (496 minutes), we only provide the data synthesized from one library – Voices Of Rapture [19] in Standard-Vocal and Expressive-Vocal for the pretraining. Data augmentations of "octave shifting" and "tonality shifting" are applied in all three datasets, except (4) incorporates more expressiveness settings. The fine-tuning learning rate is 1e-4, with the scheduler in section 4.1.

Table 4 shows the average median SDR performance of Spec-U-Net over four voice parts under different fine-tuning ratios and different pretraining settings. We can see that under all three training-test ratios, the performance of the model pretrained on Standard-Vocal and Expressive-Vocal is better than that on SoundFont-Synthesis and none-dataset, where the performance of Expressive-Vocal achieves the best. When the training-test ratio is small as 10%, the performance of SoundFont-Synthesis and non-dataset has the largeest difference, showing that the model learns some priors from SoundFont-Synthesis and converges to a better optimum. However, when the ratio increases to 40% and 70%, their performance is close to each other and does not vary much, especially on Cantoria and BBC. Thus, pretraining on SoundFont-Synthesis dataset provides a very useful initialization – but gains diminish (or even no gain) as the initializer is dominated by larger and larger quantities of real training data.

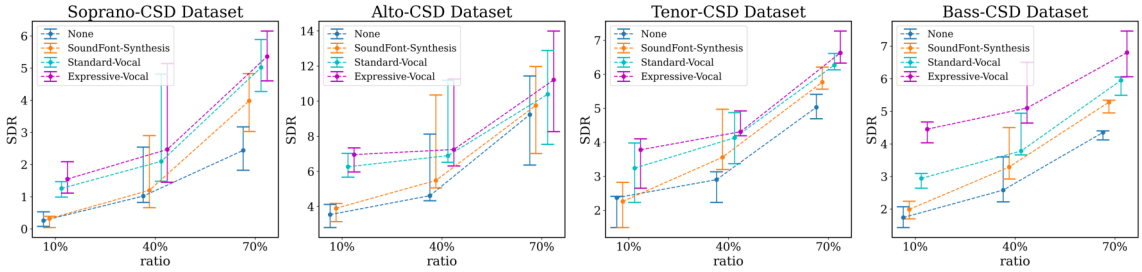
However, when the model is pretrained on Standard-Vocal, it has a strong generalization to real choral music datasets under all fine-tuning ratios, because acoustic features of synthesis tracks share large similarity to the real datasets. This performance is further boosted by Expressive-Vocal as we introduce expressiveness during synthesis, such as lyrics and velocity dynamics. Even under the 70% fine-tuning ratio, as the model has received many real data, Standard-Vocal and Expressive-Vocal pretrained model can still get improvements. In conclusion, our synthesized datasets provide not only additional data volume, but also high-quality and close-to-real choral music samples for boosting the separation performance.

To further verify our analysis, we visualized the trends of median SDRs (blue, orange, cyan, and magenta colors), with a 25th-75th percentile range, for each voice part of three real datasets in Figure 2. We can see the performance of our synthesized datasets (magenta & cyan lines) marks a clear performance increase and a large gap to that of SoundFont-Synthesis and non-dataset (orange & blue lines). However, the trends of SoundFont-Synthesis and non-dataset are close to each other, and even overlap in BBC. When analyzing the percentile range of each model, on Cantoria and CSD, our Standard-Vocal and Expressive-Vocal pretrained models reveal a clear difference of percentile ranges to the left two models, demonstrating that our models get a large improvement. However, the percentile ranges of the SoundFont-Synthesis and non-dataset pretrained models have a large overlap, demonstrating no

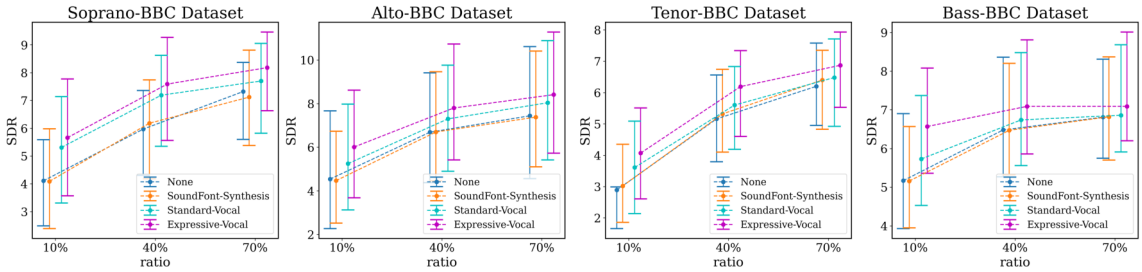
[4] We appreciate the help from authors in [14] to offer the dataset.



(a) The Median SDR performance on the Cantoria Dataset.



(b) The Median SDR performance on the Choral Singing Dataset (CSD).



(c) The Median SDR performance on the Bach & Barbershop Collection Dataset (BBC).

Figure 2: The Median SDR performance, with a 25th–75th percentile range, of soprano, alto, tenor and bass on three datasets by Spec-UNet with different pretrained models and different ratios of training-test sets.

difference even though their median SDRs differ a little. On the BBC dataset, our models yield improvements on both the 25th and 75th percentile values but are not as pronounced as those observed on Cantoria and CSD. The potential reason is because the BBC dataset contains a relatively large data size (105 minutes), which makes the model already achieve a good convergence and hard to get more significant improvements without model designs. These trends further demonstrates that our synthesized dataset plays a role in making up the data scarcity and improving generalization ability.

5. EXTENSIBILITY AND LIMITATIONS

Our provided synthesis pipeline from symbolic datasets to real audio datasets not only benefits choral music separation tasks, but also other choral-related separation tasks. For example, string quartet separation, to separate two violins, viola, and cello parts from a mixed audio, can also be trained with synthesized data of our pipeline. The details of the string quartet separation experiment can be accessed in the code repository. Similarly, our best pretrained model shows a 100%/30% performance increase to the SoundFont-Synthesis and non-dataset pretrained models. This further shows a potential application of our synthesis pipeline to improve other choral-related separation tasks.

There are also some limitations and future improvements to our work. First of all, our implementations of expressiveness are still based on random template modes. Deep learning methods can improve this expressiveness modeling [24]. Second, the design of the choral separation model is needed to learn more priors from weak synthesized data that can be transferred to real data, then it will complement our proposed pipeline better. These limitations are planned for exploration in our future work.

6. CONCLUSION

In this paper, we proposed an automated pipeline for synthesizing choral music data from sampled instrument plugins, and created an 8.2-hour choral music dataset to improve separation performance on real choral music datasets. We comprehensively evaluated multiple separation models to demonstrate that synthesized choral data is of sufficient quality to improve model's performance on real datasets. This provides additional experimental statistics and data support for choral music separation study. In the future, we will focus on the design of timbre-pitch disentanglement model [25] for achieving better separation performance. The application of choral music separation results into other music-related tasks, such as music recommendation [26], is also planned as the future work.

7. REFERENCES

[1] E. Cano, D. FitzGerald, A. Liutkus, M. D. Plumbley, and F. Stöter, "Musical source separation: An introduction," *IEEE Signal Process. Mag.*, 2019.

[2] F. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix - A reference implementation for music source separation," *J. Open Source Softw.*, 2019.

[3] A. Défossez, "Hybrid spectrogram and waveform source separation," in *Workshop on Music Source Separation, ISMIR 2021*.

[4] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, 2020, deezer Research.

[5] K. Chen, X. Du, B. Zhu, Z. Ma, T. Berg-Kirkpatrick, and S. Dubnov, "Zero-shot audio source separation through query-based learning from weakly-labeled data," in *AAAI Conference on Artificial Intelligence, AAAI 2022*.

[6] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Neural Information Processing Systems, NeurIPS 2000*.

[7] A. Jansson and E. J. H. et al., "Singing voice separation with deep u-net convolutional networks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017*.

[8] Q. Kong, Y. Cao, H. Liu, K. Choi, and Y. Wang, "Decoupling magnitude and phase estimation with deep resunet for music source separation," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021*.

[9] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018*.

[10] Y. Luo and N. Mesgarani, "Conv-tasnet: Surpassing ideal time-frequency magnitude masking for speech separation," *IEEE ACM Trans. Audio Speech Lang. Process., TASLP 2019*.

[11] H. Cuesta, E. G. Gutiérrez, A. M. Domínguez, and F. Loáiciga, "Analysis of intonation in unison choir singing," in *Proceedings of the International Conference of Music Perception and Cognition, ICMPC 2018*.

[12] S. Rosenzweig, H. Cuesta, C. Weiß, F. Scherbaum, E. Gómez, and M. Müller, "Dagstuhl ChoirSet: A multitrack dataset for MIR research on choral singing," *Transactions of the International Society for Music Information Retrieval, TISMIR 2020*.

[13] H. Cuesta, "Data-driven pitch content description of choral singing recordings," *PhD Thesis Archive*, 2022.

[14] S. Sarkar, E. Benetos, and M. B. Sandler, "Vocal harmony separation using time-domain neural networks," in *Interspeech 2021*.

[15] M. Gover and P. Depalle, "Score-informed source separation of choral music," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020*.

[16] D. Petermann, P. Chandna, H. Cuesta, J. Bonada, and E. Gómez, "Deep learning based source separation applied to choir ensembles," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020*.

[17] Native-Instruments, "Noire, evocative concert grand," https://www.native-instruments.com/en/products/komplete/keys/noire/, 2021.

[18] Native-Instruments, "The grandeur, vibrant modern grand," https://www.native-instruments.com/en/products/komplete/keys/the-grandeur/, 2020.

[19] SoundIron, "Voice of rapture collection," https://soundiron.com/products/voices-of-rapture, 2015.

[20] FluffyAudio, "Dominus choir," https://www.fluffyaudio.com/shop/dominuschoir/, 2019.

[21] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proceedings of the 29th International Conference on Machine Learning, ICML 2012*.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *the 3rd International Conference on Learning Representations, ICLR 2015*.

[23] F. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation - 14th International Conference, LVA/ICA 2018*.

[24] Y. Wu and E. M. et al., "MIDI-DDSP: detailed control of musical performance via hierarchical modeling," in *the 10th International Conference on Learning Representations, ICLR 2022*.

[25] K. Chen, C. Wang, T. Berg-Kirkpatrick, and S. Dubnov, "Music sketchnet: Controllable music generation via factorized representations of pitch and rhythm," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR 2020*.

[26] K. Chen, B. Liang, X. Ma, and M. Gu, "Learning audio embeddings with user listening data for content-based music recommendation," in *International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021*. IEEE, 2021, pp. 3015–3019.

ETHICS OF SINGING VOICE SYNTHESIS: PERCEPTIONS OF USERS AND DEVELOPERS

Kyungyun Lee[1] **Gladys Hitt**[2] **Emily Terada**[2] **Jin Ha Lee**[2]
[1] Gaudio Lab, Seoul, Korea
[2] University of Washington, Seattle, USA
mo@gaudiolab.com, hittg@uw.edu, eterada@uw.edu, jinhalee@uw.edu

ABSTRACT

Singing Voice Synthesis (SVS) has recently garnered much attention as its quality has improved vastly with the use of artificial intelligence (AI), creating many opportunities for supporting music creators and listeners. Recently, there have been growing concerns about ethical issues related to AI development in general, and to AI-based SVS development specifically. Many questions remain unexplored about how to ethically develop and use such technology. In this paper, we investigate the perception of ethical issues related to SVS from the perspectives of two different groups: the general public and developers. We collected 3,075 user comments from YouTube videos showcasing various uses of SVS as part of a mainstream variety show. Additionally, we interviewed six researchers developing SVS technology. Through thematic analysis, we identify and discuss three different aspects related to ethical issues in SVS development, highlighting the similarities and differences between the perspectives of the general public and developers: (1) Use scenarios, (2) Attitudes towards development, and (3) Meaning of "Creativity", and (4) Concerns about human rights, intellectual property (IP) and legal issues.

1. INTRODUCTION AND BACKGROUND

Artificial intelligence (AI) is developing rapidly and becoming increasingly pervasive in our lives through various forms of technology, including robots and smart devices. The AI field has been thriving over the past decade due to the increasing availability of massive data, computational resources, and deep-learning-based architectures [1, 2]. Related to music, there are various AI technologies to support music organization, production, and enjoyment, including recommendation [3, 4], music generation [5, 6], instrument sound synthesis [7] and singing voice synthesis [8–10]. Singing Voice Synthesis (SVS) has existed since the late 1950s [11] and has been used and enjoyed by many users through applications like singing synthesizers and vocaloids [8]. The significant improvement in the quality of voices due to the use of recent techniques, specifically deep learning, has given rise to more opportunities for the use of SVS, but has also led to more questions regarding its ethical use and increased concern for potential misuse. While the SVS community has started discussing some of the ethical issues related to the development and use of SVS, the discussion is in the early, nascent stage [2]. Furthermore, there is limited research investigating how the general public, as potential users, perceive and feel about the use of SVS, especially since the synthesized singing voices are sounding increasingly realistic and almost indistinguishable from human voices.

The ethical issues encompass multiple areas, including intellectual property (IP) (e.g., the ownership of the media with AI voices) and human rights (e.g., who decides how the voices are used). In addition to understanding how the general public thinks about SVS, it is important to consider what influences their perceptions and whether developers are aware of them. If developers are aware, does that impact their goals and direction? Understanding how the general public perceives AI technology may offer useful insights to developers on how to address negative perception and ensure AI is well-received by users. It is not only important for envisioning the potential commercialization of such technologies, but also for discussing ethical issues and collaboratively thinking about how society should address such issues.

In this paper, we aim to improve our understanding of how people perceive the uses of SVS technology through a content analysis of user comments collected from online videos showcasing various applications of SVS. To examine how user perception on ethical issues related to SVS can influence the development of the technology and vice versa, we also interviewed SVS developers regarding viewpoints on potential ethical issues. We aim to answer the following research questions: (1) How does the general public react to AI-generated singing voices, and what are the implications to SVS developers? (2) What kinds of ethical issues do users and SVS developers consider?

2. LITERATURE REVIEW

2.1 Music-related AI Technologies and SVS

With the advancement of generative models in deep learning, creative applications of music AI technology have be-

© K. Lee, G. Hitt, E. Terada, and J. H. Lee. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** K. Lee, G. Hitt, E. Terada, and J. H. Lee, "Ethics of Singing Voice Synthesis: Perceptions of Users and Developers", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

come a popular research topic in music information retrieval (MIR). Music generation is one area, encompassing various tasks, such as melody generation [5], expressive performance generation [12], sound synthesis [7] and pop song generation [6]. Research in generative music not only varies in tasks, but also in their data formats, which can be at a symbolic-level (e.g., scores and MIDI tracks), at an audio-level (e.g., mp3 and wav files) or a mix of both. In the past years, music AI technologies have become increasingly available and accessible to the public. One example is the BachDoodle, a Bach-like harmonization system from Google Magenta [13], which received much attention when made publicly available as an interactive Google Doodle [1] . Another example is an audio-level pop music generation system, called Jukebox, which generates raw audio of pop songs, including singing voices [6]. This system is conditioned on lyrics and artists, therefore allowing control over such features to obtain desired outcomes. The quality of generated music has vastly improved, even motivating musicians, such as The Flaming Lips, to experiment with AI systems in their music creation process [2] .

SVS has been researched for more than two decades. The goal of SVS is to learn human-like singing voices given input conditions, like speech, lyrics and melodies. A thread of research focuses on creating non-existent voices as seen in popular applications like vocaloids [14]. Other areas handle existing voices: replicating existing singing voices [9,10] such as those of popular artists, correcting the singing expressions for performance improvement [15], and converting speech to singing voice [16]. We have observed significant improvements in the synthesis quality through the adaptation of advanced deep learning models, such as Tacotron [17] and DeepVoice [18]. In particular, the YouTube videos analyzed in this paper replicated voices of popular singers from music scores and lyrics [9], incorporating techniques from text-to-speech (TTS) with adversarial training to achieve state-of-the-art results.

2.2 Ethical Issues Related to Music AI

Concerns regarding intellectual property and AI are growing. Many are concerned about how to protect artists' value in the music industry when "automation" may drive music production costs to zero [19]. While we may still doubt the quality of AI musicality, SKYGGE's Hello World [3] and virtual musicians such as Vocaloid Hatsune Miku [4] are clear signs that the "trend to autonomous and human-like virtual musicians is extremely well established" [20]. As such, ethical concerns are also tied to concerns over excessive loss in human creativity [14]. There is also an aspect of insufficient public knowledge about these technologies, which in turn, informs the lack of legal regulations. Furthermore, such regulations would be heavily influenced by corporations which have "incentives to ignore public input and shift regulation to their benefit" [21]. According to Floridi, there are five digital ethics pitfalls that can be applied when examining case studies and potential consequences of SVS/AI technologies: 1) Ethics Shopping, 2) Ethics Bluewashing, 3) Ethics Lobbying, 4) Ethics dumping, 5) and Ethics Shirking [22].

Beyond the ethics of artist getting compensated for their voices or work, there are also additional questions with regards to human rights – should artists be allowed to "refuse" having their voices or work serve as data sets for generating new voices, and potentially a third party profiting from it [23, 24]?

Existing literature emphasizes the importance of drawing ethical considerations and conclusion within the context of AI's socio-technical systems, rather than on the specific features of AI itself as today's digital technology "ecosystem" necessitates the inclusion of more stakeholders and perspectives [25]. Research finds that Western perspectives often dominate the conversation surrounding AI [26–28] and increasingly emphasizes the importance of considering cultural differences when examining human-AI interactions and attitudes towards AI [29, 30]. There is a pressing need for specific applications of ethics and accountability measures rather than current abstract principles [26] with the argument that stakeholders in the music ecosystem must be the ones responsible for establishing an ethical response for the purpose of a sustainable music industry that includes humans and non-humans [31].

3. STUDY DESIGN AND METHODS

To understand users' and developers' perspectives on AI-based SVS technology, we employed a mixed method approach using thematic analysis of online user comments and interviews. Given that there is currently no widely distributed commercial application using this technology (other than vocaloid communities which are generally considered to be niche), it was challenging to identify a specific community to represent the perception of the general public. Inviting people to react in an experimental setting also has the potential for increased participant response bias [32]. Therefore we opted to find data sources which would allow us to capture user reactions in a more natural setting. We looked into public online videos showcasing state of the art technology and found promising examples from Korean TV shows called "AI versus Humans" [5] and "One More Time," [6] where several different use cases for this technology were presented, such as generating voices of artists (both currently active and deceased), manipulating the artists' voice, generating singing voices in languages different from the language of the original voice samples, and joint performance of artist and AI. A total of 3,075 comments were collected from eight videos from 2020 to 2021 where they presented various applications of AI-based SVS technology.

Most comments were in Korean and a few in English. Two authors who are fluent in Korean independently coded

[1] https://www.google.com/doodles/celebrating-johann-sebastian-bach
[2] https://magenta.tensorflow.org/fruitgenie
[3] https://www.skygge.fr/
[4] https://ec.crypton.co.jp/pages/prod/virtualsinger/cv01
[5] https://programs.sbs.co.kr/enter/aivshuman/about/67201
[6] https://program.genie.co.kr/onemoretime/main

these comments. Taking an inductive approach, the two authors reviewed partial data, used Mural board (a collaborative online tool) to organize the concepts, and came up with the initial codebook through a thematic analysis [33]. Afterwards, the authors coded sample comments together and iterated on the design of the codebook based on discussion. These discussions resulted in addition of a new code (Positive tech-Other) and redefinition of “Commercial” and “Personal" for a clearer distinction. The two coders then used the final codebook (https://osf.io/7em95/) to code all the comments following a consensus model [34], and discrepancy in code application was discussed with the goal of reaching a consensus.

For additional data on developers' perspectives, we reached out to authors of recent publications on this technology asking whether they would participate in an online interview. Each interview was recorded via Zoom and verbal consent was obtained at the start of recording, following the protocol approved by the UW Institutional Review Board. Using Zoom to host and record interviews also enabled us to generate interview transcripts which the authors cleaned and coded for analysis. Interview questions focused on the kinds of ethical issues developers consider, the broader implications to designers of music AI technology, how they perceive the general public's reaction to music AI technology, and what ethical and legal precautions they believe should be implemented.

We reached out to 12 authors, and were able to recruit six developers from three different countries. Five developers were faculty or students building and testing the AI-based SVS technology, and one was an artist who provided their voice samples to build the data set for this technology. Since there is a limited number of researchers who work on this particular technology, the overall pool of users is smaller compared to the number of general AI researchers. The small sample size makes this exploratory in nature.

All interviews were fully transcribed and coded using an inductive approach [35]. Two authors created the initial codebook through thematic analysis, following a similar process as above. The final codebook had 16 codes. Using a qualitative coding software ATLAS.ti, the four authors coded the interviews. We assigned two different coders to each interview, and followed the consensus model [34].

4. FINDINGS

4.1 Perception of the General Public

Eight categories and 27 associated codes emerged from our analysis of online user comments from YouTube videos. "Positive emotions" (e.g., awe) and "Negative emotions" (e.g., fear) categories include responses that reflect users' emotions evoked by seeing the application of SVS. We also created "Positive tech" and "Negative tech" categories that consist of comments about how good or limited they think the current SVS technology is. We dedicated a category, "Conflicted", for comments that showed users' mixed feelings on whether their current emotion is justifiable or not. "Considerations" category contains codes representing different aspects related to ethical issues (e.g., copyright, human right). "Challenges" contains codes that mention realistic problems related to SVS and AI (e.g., misuse). Lastly, "Opportunities" includes comments that suggest new possible use cases and applications of the current SVS technology (e.g,. commercial, personal). Reviewing 3,075 comments, excluding incomprehensible ones, resulted in 1,190 comments that were coded. The result of our analysis as a code distribution can be accessed at: https://osf.io/7em95/.

Overall, we observed positive and negative sentiments in a 65% to 35% ratio. The most observed comment, coded as "awe" (130/1,190), conveyed surprise and fascination towards the level of today's SVS technology. Along with "awe", "moving" (104) and "nostalgia" (84) were commonly appearing positive themes. Several users were touched by the revival of voices of the singers they grew up with (e.g., *"I am thankful for the opportunity to listen to [the artist's] voice again"*). The particular use of the technology did seem to influence audiences' emotion (e.g., nostalgia for deceased artists). The high frequency of the code "moving" implies the potential power and influence of AI on humans. Negative and cynical comments about SVS and AI were represented by codes "fear" (81), "guilt" (25) and "neg-emotions-other" (19). We encountered comments, such as *"this is giving me goosebumps"*, expressing shock and even discomfort towards the quality of reenactments of familiar voices. Feelings of "guilt" were also represented - *"isn't this humiliating the deceased?"*, *"I don't think it's appropriate to do this"* - showing irritation and criticizing lack of morality and respect towards the deceased artists. We observed some conflicted feelings - *"I don't know if I feel good or scared about the regeneration of artists' voice. I feel resistance, as well as curiosity at the same time. I don't know"*, *"It feels weird...I feel scared. Maybe I shouldn't have listened to this"* and *"I do want to see the artist, but not in this way."* Regarding the aspect of technical advancements, there were comments both praising and belittling the results. Some were impressed and rated AI as sounding exactly like the original artists-*"(AI) even replicated the way (the artist uniquely) pronounces 'r'"*- with a few users thinking the AI even sounded better than the original artists.

The "Opportunities" category included comments about audiences' desire to use the presented SVS technology for their personal interest. There were series of comments asking for the regeneration of the voices of other artists or their family members - *"Can they also replicate X?"* and *"I wish to hear my father's voice."* Several comments requested the commercialization of the SVS technology, expressing interest in purchasing the application if made available. There were also comments discussing other potential use cases, such as voice actors in films or saving their own voices to sing songs to their grandchildren posthumously.

From the prominence of the code "misuse," we witnessed growing concerns towards AI in today's society. Often referencing deepfakes, users were afraid of the rise of potential scams (e.g., voice phishing). Some even expressed anger at the irresponsible development of such "dangerous" technology, arguing that it will do more harm

than good. Along this line of negativity, dystopic comments expressed concerns about the potential of AI to replace humans-*"I am a vocal student...Help me, I think my job is disappearing"*,*"I wonder how many jobs will be lost to AI in the near future."* These comments led to sentiments of inevitability and reluctant acceptance- *"we need to make ourselves irreplaceable by AI."* Meanwhile, we observed only one comment discussing the ways to overcome the potential dangers of the technology ("counter").

As reflected in the code "guilt", some addressed issues regarding the legality of SVS usage in terms of human rights and copyright. Many considered it disrespectful and unethical to use voices without the actual owner's consent, invalidating families' and relatives' compliance. They were also critical of the notion that the show could be profiting from the exhibition of deceased artists' voices.

4.2 Perception of Developers

Six interviewees were highly involved in the development of SVS technology, with experience ranging from two to 20 years. Their initial motivation driving their research was curiosity about the technology. All participants envisioned their work potentially helping musicians increase their productivity and creativity. For example, they envisioned SVS being used to quickly generate demo tracks for vocalists, as it is inefficient to invite the vocalists each time to record demos (P1, P5). While participants unanimously predicted that the current SVS technology will reach the level of human voices within the next five to 10 years, they also listed limitations, including data constraint and the need for higher quality results for commercial applications. Developers' thoughts on the "meaning of creativity" were related to the limited data set. They questioned the possibility of AI becoming truly creative, as *"AI can only generate average of the data"* and *"AI is not able to generate something better than humans since it is only trained on the input data"* (P1, P4). P1 mentioned that artists incorporate context and history into the music they are making, which AI lacks. P6 pointed out that the well-known vocaloid Hatsune Miku's fandom is not just towards the virtual character itself, but also towards the human creators behind it.

On the discussion about ethical issues of SVS, a prevalent response was that developers are aware of such issues and their importance, but no one knows specific ways to handle them. Debates have arisen around the unclear standards and practices regarding ownership problems in the music industry. Regarding who is responsible for addressing ethical issues, some participants emphasized that developers should take more caution when it comes to data use and how they make their technology available to the public (P4), while others suggested that there should be a group of experts dealing specifically with ethical issues so the developers can focus solely on the advancement of the technology (P3). However, half of the developers noted that the SVS technology of today is not yet advanced enough to consider ethical issues seriously (P2, P3, P4).

Regarding how to address the ethical and legal implications of this technology, participants indicated that solutions to counteract the misuse of AI are necessary. For example, most participants (all but P5) mentioned the development of counter technologies which can detect AI generated voices as one of the solutions to the potential misuse, and were fairly confident in the power of counter technologies. In general, participants maintained positive views on the future of SVS and considered the general public's negative reactions towards AI to be no different than similar reactions towards other technologies in the past. P5, for instance, talked about the initial negative reactions towards the sound of electronic keyboards and how those attitudes changed as keyboards become more widely used in the music industry. They noted that, similarly, the advancement of AI is inevitable and the general public will slowly accept it. Upon closing the interviews, participants commented on the future of SVS technology, ranging from concerns - *"I'm not sure if it's okay to get comfortable with the mass production of voice through AI"* (P1)-to practical directions to take-*"Humans should try to figure out ways to co-exist with AI, not compete with or hinder its development, because AI will never be able to replace humans"* (P2)-to the need for developers, artists, and the public to maintain an open line of communication regarding differing motivations (P4,P6).

5. DISCUSSION AND IMPLICATIONS

5.1 Perspectives on Use Scenarios

All of the developers interviewed considered supporting creators including composers, performers, and producers, as one of the main goals of their research. When asked to share what kinds of use scenarios they envisioned for the application of their work, they discussed various situations in which AI supports and benefits the creators: *"a tool for the creator like autotune or mixing"* (P2), a tool for generating demo tracks for vocalists (P1), and a means of *"style transfer to correct and improve singing"* (P4). P6, in particular, emphasized that mimicking the human voice to make it sound "natural" is only one aspect of SVS and the true potential lies in generating a variety of voices. They explained how some users may want the voice to sound "artificial" and even prefer that, as exemplified by the popularity of Vocaloid's Hatsune Miku, and the frequent use of Autotune in mainstream music.

The developers also tended to lean towards a more controlled model where a select few have direct access to the use of technology or data set to prevent potential misuse. A question was raised about the commercialization of SVS and its implications. P2, in particular, explained using SVS to recreate the voices of existing or deceased artists will have limitations due to ethical issues beyond just using it as a proof-of-concept or for an event, but generating new voices might have more freedom to be used commercially.

However, the user comments demonstrate a range of desires and ideas for different kinds of commercial uses for SVS technology. Of the 62 comments mentioning potential commercial uses, many expressed that they would be interested in purchasing a song or album by a deceased artist using SVS. Some even envisioned specific apps peo-

ple could download and use like *"a paid app that lets us pick the singer's voice and the song we want to hear in that voice. Maybe in our own voice!"* or *"a business with streaming websites that lets people pay and save the songs.* Many users also imagined the use of the SVS technology beyond the context of music, such as using *"this technology to speak with dead people"*. A user wondered about their future with this technology in relation to human interactions across time (*"Even if I die early, I can sing a lullaby to my grandchildren?"*).

Users also had more diverse ideas about the potential misuse of the technology, including ethical and legal issues that could stem from its development. While this could potentially be attributed to the larger number of comments seen on YouTube compared to those gathered from the interviews, it does indicate that users are definitely considering the commercialization of this technology and are willing to pay for and use it for "personal" goals.

5.2 Attitudes Towards AI Technology Development

Developers tended to be interested in and focused on enhancing the current technology, often recognizing the potential misuse or abuse of technology while accepting the reality that the technology will continue to improve over time regardless of how they feel or act. Despite their initial interest in the research being motivated from the excitement about the technology itself (all but P5), they reported becoming more aware of potential ethical issues when they started seeing "how good" the current level of technology is (e.g., *"I didn't think we'd be able to reproduce it to this level. We'd need technologies to counter the misuse of this technology."* (P1)). P3 shared that while developers are starting to engage in discussion related to these issues, at least in their social circles, people are not yet seriously considering these issues, nor have a clear idea or direction on what to do. Developers also felt that not enough actions are currently being taken. There were different opinions as to who needs to take the lead when it comes to implementing ethical and legal measures to counteract misuse. P3, for instance, viewed that "big players," such as large corporations investing in AI development, should play a bigger role. P4, on the contrary, stated that it will be dangerous for one party to decide how to ethically limit and/or control AI development, be it the government or big corporations, and emphasized that the society as a whole needs to engage in discussion and arrive at social consensus.

User comments showed more mixed opinions. Many were awed and moved by hearing the realistic AI voices, but they also shared fear, guilt, and discomfort caused by "uncanny valley" [8]. Compared to the developers' point of view, users had more pessimistic views on the technology, and a few questioned whether we should be developing such technology in the first place:

"I became fearful as I was watching this [...] I feel lucky that I was born in an era when the AI isn't fully developed." "I think the technology is great, but I also wish we didn't overdo it. Do we really need to listen to songs created by machines? [...] there's no guarantee that the story in movies won't come true where people who wanted support from AI eventually become controlled by them."

These comments suggest that people's fears are influenced by how AI is portrayed in popular media. Several mentioned that the technology reminds them of science fiction TV shows or movies in which AI takes the role of the adversary. Compared to users, developers tended to be over-optimistic about the technology, trusting that counter technologies will exist and work well enough, with some passing the responsibilities to users to some extent (e.g., *"it is not the technology that is bad, but people who misuse it"* (P6), *"AI is just a tool"* (P3)). The discrepancy in how developers and users feel about the technology shows that it is important to encourage discussion on these issues between the two stakeholders, so the general public can be more informed about the actual state of the technology and the developers can understand how the technology is being received by the general public which will inevitably affect the future of such technology. P4 also emphasized the importance of communication with the general public about the technology as the developers' responsibility.

5.3 Meaning of "Creativity" in the Context of AI

A common question raised among developers and users concerns what it means to be "creative" and if AI generated voices can be considered as such. Defining what it means to be creative has ethical implications as determining who or what is responsible for the creative work affects the decision on who should "own" and benefit from the IP. Yet creativity has various definitions; one depends on the "ability to come up with ideas or artifacts that are new, surprising" [36] and another hinges on societal and cultural contexts and thus resists a concrete definition [37]. Some definitions disregard the notion of value and view it as irrelevant, instead emphasizing that creativity must look within the action of being creative itself [38].

As evident by the various perspectives, what is considered creative is widely contested, even more so once AI comes into frame. While it can certainly be agreed that creativity involves processes both cognitive and psychological, the question of whether AI can simulate these processes and the limitations to its approach via simulation remains [39]. P2 pointed out the lack of agency and intention from the AI, and questioned whether AI would be capable of creating something truly novel, stating: *"Even though AI can act perfectly like a human, we are just looking at the outcome. That doesn't mean the AI is really thinking about what it is expressing. It only learned from the data. So at the signal level, it might be similar but that is not an outcome from any kind of reasoning."* P4 also discussed how AI is good at interpolation from existing data sets, thus excelling in giving an "average" performance based on previous performances, but not something truly unique and novel. Some of the user comments also express similar sentiments, stating that the AI voice is "soulless", "lacks emotions", and sounds "too comfortable" or "too honest":

"There isn't something deep or substantial, there's no emotion so I'm not moved [...] It'd sound more natu-

ral if the machine does adlibs or intentionally has trouble singing or barely sings in certain parts."

"The voice is the same but it's missing a soul [...] it's just following exactly what's written in the sheet music."

Audry and Ippolito [40] propose a way of examining the relationship between AI and creativity by redirecting the focus from the artist, both human and artificial, towards the viewer. They draw on Foucault's designation of the "Meta-Artist" to postulate that regardless of whether or not AI can be creative/artists, they most certainly can give rise to an "artist function." As long as viewers continue to construct Meta-Artists, "artists will exist as social constructs" [40].

According to Gioti [39], while AI has generated impressive results in controlled environments, it has yet to break the barrier into autonomy. This leaves room for human artists and suggests the use of AI as an extended intelligence and thus another "actor contributing to a 'networked intelligence' that encompasses both humans and machines" [41]. From this, the concept of computational intelligence emerges where creative responsibility is shared among human and non-human actors where the latter is determined by the "extension of human intentionality through technological intentionality" [41]. The developers we interviewed also primarily considered the collaboration between humans and AI as a big area of opportunity, and none believed that the AI will truly replace the role of humans, partly because of the value ascribed to human skills [42]. Their perspective was focused on seeing AI's role as extending human abilities, similar to how Autotune is commonly used to manipulate singers' voices (P2). While this synergy between AI and humans is posed as ideal, it is limited in its exploration of ethical concerns and considerations of how such a teetering asymmetric relationship will affect both humans and machines.

5.4 Human Rights, IP, and Other Legal Issues

Questions of who should make decisions about the ethics, legitimacy, and legality of SVS remain challenging and unanswered [43]. Sturm et al. [2] asks if the "lack of copyright protection of AI-generated results [is] adequate from a policy point of view" and recommends further "legal and socio-economic analysis." As of 2019, only the "UK, South Africa, Hong Kong, India, Ireland, and New Zealand have envisaged protection for computer-generated works granted to the person by whom the arrangements necessary for the creation of the work have been undertaken" [2]. Users and developers alike question who should be given the rights to the final product in situations where SVS is used to generate new voices using an artist's voice. P5 shared how they wished *"there are some rules about where the profits go to. Currently there are no laws about that."*

Regarding neighboring rights, P4 questioned how the financial gain should be distributed between the AI developer and the artists. Another question was about generating a deceased artist's voice – who should decide how the voice is created and used? P3 shared that family or people close to the deceased artists currently have the rights. In contrast, P6 brought up that the family members are still not the artists themselves. Is it truly acceptable for the third party to make this decision? What if the voice sample is used with other samples to generate a new voice, so the voice data provider does not financially benefit from the result? Users commonly expressed concerns over the worth of human musicians in the event that AI becomes the norm rather than a collaborative tool. Both developers and users expressed conflicted feelings about SVS, with users expressing more negative sentiment such as fear or guilt, using phrases like 'superfluous man/잉여 인간'(i.e., humans with no roles or uses in the society). One proposed solution is to reintroduce scarcity via a Natural Talent certification, which identifies a composition or song as authentically made by human and differentiates it from music produced by AI [19]. However, the implementation is complicated given how human-computer collaboration on music already exists and is pervasive throughout the creation process.

In contrast to end users' concerns, developers are less worried about the "domination" of AI [44]. P2 stated that the individual performers should be able to maintain the performer's rights rather than the entertainment company they belong to, and that any speculated AI programs that will replace the artists will not succeed because of the backlash from the general public; and that coexisting is more likely to lead to success. Ultimately, P2 and P6 did not believe that any outcomes from AI can truly replace music created by humans because people will not want that to happen. As Sturm et al. state "humans still have an important involvement in creating music, even if assisted by an AI system" [2].

6. CONCLUSION AND FUTURE WORK

This study offers initial insights into the perceptions of users and developers regarding ethical issues to consider when developing and implementing SVS technologies. Our findings highlight the discrepancies between user and developer perspectives regarding their envisioned use scenarios and attitudes, but also show that both stakeholders are similarly questioning creativity in the age of AI and concerned about human rights and IP issues.

This is a qualitative exploratory study with limited user data with a goal of enriching our understanding of the topic, not claiming a generalization of the findings. Future research involving more and varied developers and artists should be conducted to gain a more holistic understanding of the different stakeholders' viewpoints. In addition, the 3,075 user comments were predominantly from one culture and one platform and other cultures or platform users will have different perspectives, warranting further investigation. As this study analyzes user perception of the application of AI technologies presented in TV programs, it could have been influenced by how the usage was showcased in the media. However this is also realistically how the user perception is formed on new technology as media plays a significant role in our society. In the future, we also plan to conduct a follow-up study focusing on the perception of ethical issues from the artists' point of view.

7. ACKNOWLEDGEMENTS

We thank the interviewees who shared their honest thoughts and opinions about SVS development.

8. REFERENCES

[1] Y. Goodfellow, I. Bengio and A. Courville, *Deep Learning*. Montréal, Canada: The MIT Press, 2018.

[2] B. L. Sturm, M. Iglesias, O. Ben-Tal, M. Miron, and E. Gómez, "Artificial intelligence and music: open questions of copyright law and engineering praxis," *Arts*, vol. 8, no. 3, p. 115, 2019.

[3] Y. Song, S. Dixon, and M. Pearce, "A survey of music recommendation systems and future perspectives," in *9th International Symposium on Computer Music Modeling and Retrieval*, vol. 4. Citeseer, 2012, pp. 395–410.

[4] A. Van Den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *Neural Information Processing Systems Conference (NIPS 2013)*, vol. 26. Neural Information Processing Systems Foundation (NIPS), 2013.

[5] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[6] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv preprint arXiv:2005.00341*, 2020.

[7] J. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," *arXiv preprint arXiv:2001.04643*, 2020.

[8] M. Goto, T. Nakano, S. Kajita, Y. Matsusaka, S. Nakaoka, and K. Yokoi, "Vocalistener and vocawatcher: Imitating a human singer by using signal processing," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 5393–5396.

[9] J. Lee, H.-S. Choi, C.-B. Jeon, J. Koo, and K. Lee, "Adversarially trained end-to-end korean singing voice synthesis system," *arXiv preprint arXiv:1908.01919*, 2019.

[10] S. Choi, W. Kim, S. Park, S. Yong, and J. Nam, "Korean singing voice synthesis based on auto-regressive boundary equilibrium gan," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7234–7238.

[11] P. R. Cook, "Singing voice synthesis: History, current work, and future directions," *Computer Music Journal*, vol. 20, no. 3, pp. 38–46, 1996.

[12] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *Neural Computing and Applications*, vol. 32, no. 4, pp. 955–967, 2020.

[13] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," *arXiv preprint arXiv:1903.07227*, 2019.

[14] N. Collins, "Trading faures: Virtual musicians and machine ethics," *Leonardo Music Journal*, pp. 35–39, 2011.

[15] S. Yong and J. Nam, "Singing expression transfer from one voice to another for a given song," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 151–155.

[16] T. Saitou, M. Goto, M. Unoki, and M. Akagi, "Speech-to-singing synthesis: Converting speaking voices to singing voices by controlling acoustic features unique to singing voices," in *2007 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2007, pp. 215–218.

[17] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio *et al.*, "Tacotron: Towards end-to-end speech synthesis," *arXiv preprint arXiv:1703.10135*, 2017.

[18] A. Gibiansky, S. Ö. Arik, G. F. Diamos, J. Miller, K. Peng, W. Ping, J. Raiman, and Y. Zhou, "Deep voice 2: Multi-speaker neural text-to-speech." in *NIPS*, 2017.

[19] W. P. Jacobson, "The robot's record: Protecting the value of intellectual property in music when automation drives the marginal costs of music production to zero," *Loy. LA Ent. L. Rev.*, vol. 32, p. 31, 2011.

[20] M. Avdeeff, "Artificial intelligence & popular music: Skygge, flow machines, and the audio uncanny valley," *Arts*, vol. 8, no. 4, p. 130, 2019.

[21] G. Lima and M. Cha, "Descriptive AI ethics: Collecting and understanding the public opinion," p. arXiv:2101.05957, 2021.

[22] L. Floridi, "Translating principles into practices of digital ethics: Five risks of being unethical," *Philosophy & Technology*, vol. 32, no. 2, pp. 185–93, 2019.

[23] K. K. Kimppa and T. I. Saarni, "Right to one's voice?" in *Proceedings of ETHICOMP 2008: Living, Working and Learning beyond Technology*. ETHICOMP, 2008, pp. 480–88.

[24] K. M. Scott, S. Ashby, D. A. Braude, and M. P. Aylett, "Who owns your voice? ethically sourced voices for non-commercial TTS applications," in *CUI '19: Proceedings of the 1st International Conference on Conversational User Interfaces*. Conversational User Interfaces (CUI), 2019, pp. 1–3.

[25] B. C. Stahl, "From computer ethics and the ethics of AI towards an ethics of digital ecosystems," *AI and Ethics*, vol. 2, no. 1, pp. 65–77, 2022.

[26] M. Hickok, "Lessons learned from AI ethics principles for future actions," *AI and Ethics*, vol. 1, no. 1, pp. 41–47, 2021.

[27] R. Huang, B. L. T. Sturm, and A. Holzapfel, "Decentering the west: East Asian philosophies and the ethics of applying artificial intelligence to music," in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, 2021, pp. 301–309.

[28] B. Zhang and A. Dafoe. (2019) Artificial intelligence: American attitudes and trends. [Online]. Available: https://doi.org/10.2139/ssrn.3312874

[29] J. Dang and L. Li, "Robots are friends as well as foes: Ambivalent attitudes toward mindful and mindless ai robots in the United States and China," *Computers in Human Behavior*, vol. 115, 2021.

[30] L.-M. Neudert, A. Knuutila, and P. N. Howard. (2020) Global attitudes towards AI, machine learning automated decision making. [Online]. Available: https://oxcaigg.oii.ox.ac.uk/wp-content/uploads/sites/124/2020/10/GlobalAttitudesTowardsAIMachineLearning2020.pdf

[31] M. Clancy. (2021) Reflections on the financial and ethical implications of music generated by artificial intelligence. [Online]. Available: http://www.tara.tcd.ie/handle/2262/94880

[32] N. Dell, V. Vaidyanathan, I. Medhi, E. Cutrell, and W. Thies, ""Yours is better!" participant response bias in HCI," in *Proceedings of the SIGCHI conference on human factors in computing systems*, 2012, pp. 1321–1330.

[33] J. Corbin and A. Strauss, "Basics of qualitative research (4th ed.)," 2015.

[34] C. E. Hill, B. J. Thompson, and E. N. Williams, "A guide to conducting consensual qualitative research," *The counseling psychologist*, vol. 25, no. 4, pp. 517–572, 1997.

[35] A. Strauss and J. Corbin, "Basics of qualitative research 4th edition. technique and procedures for developing grounded theory," 2015.

[36] M. A. Boden *et al.*, *The creative mind: Myths and mechanisms*. Psychology Press, 2004.

[37] M. A. Boden, *Creativity and art: Three roads to surprise*. Oxford University Press, 2010.

[38] A. Dorin and K. B. Korb, "Creativity refined: Bypassing the gatekeepers of appropriateness and value," in *Computers and creativity*. Springer, 2012, pp. 339–360.

[39] A.-M. Gioti, "From artificial to extended intelligence in music composition," *Organised Sound*, vol. 25, no. 1, pp. 25–32, 2020.

[40] S. Audry and J. Ippolito, "Can artificial intelligence make art without artists? ask the viewer," in *Arts*, vol. 8, no. 1. Multidisciplinary Digital Publishing Institute, 2019, p. 35.

[41] J. Ito, "Extended intelligence," *PubPub*, 2016.

[42] T. J. Pinch and K. Bijsterveld, "'Should one applaud?' breaches and boundaries in the reception of new technology in music," *Technology and Culture*, vol. 44, no. 3, pp. 536–59, 2003.

[43] N. Mirra, "Putting words in your mouth: The evidentiary impact of emerging voice editing software," *Richmond Journal of Law and Technology*, vol. 1, 2018.

[44] G. Velarde. (2021) Artificial intelligence trends and future scenarios: Relations between statistics and opinions. [Online]. Available: https://www.computer.org/csdl/10.1109/CogMI52975.2021.00017

EMOTION-DRIVEN HARMONISATION AND TEMPO ARRANGEMENT OF MELODIES USING TRANSFER LEARNING

Takuya Takahashi **Mathieu Barthet**
Centre for Digital Music, Queen Mary University of London
takahashi@uec.ac.jp, m.barthet@qmul.ac.uk

ABSTRACT

We propose and assess deep learning models for harmonic and tempo arrangement generation given melodies and emotional constraints. A dataset of 4000 symbolic scores and emotion labels was gathered by expanding the HTPD3 dataset with mood tags from last.fm and allmusic.com. We explore how bi-directional LSTM and Transformer encoder architectures can learn relationships between symbolic melodies, chord progressions, tempo, and expressed emotions, with and without a transfer learning strategy leveraging symbolic music data without emotion labels. Three emotion annotation summarisation methods based on the Arousal/Valence (AV) representation are compared: Emotion Average, Emotion Surface, and Emotion Category. 20 participants (average age: 30.2, 7 females and 13 males from Japan) rated how well generated accompaniments matched melodies (musical coherence) as well as perceived emotions for 75 arrangements corresponding to combinations of models and emotion summarisation methods. Musical coherence and match between target and perceived emotions were highest when melodies were encoded using a BLSTM model with transfer learning. The proposed method generates emotion-driven harmonic/tempo arrangements in a fast way, a keen advantage compared to state of the art. Applications of this work include AI-based composition assistant and live interactive music systems for entertainment such as video games.

1. INTRODUCTION

With the burgeoning of video games, user-generated video content, and tv/film productions released on streaming services, the demand of music for media seems to be growing. Although musicians have known for long how to produce music for such media, interactive music production systems can innovate the way producers create dynamic scores responding to contextual and user factors determined prior to or during the media experience. Deep generative models for music composition have made steady improvements but how to control them to support creative agency remains a challenge [1]. In this work, we investigate deep learning techniques to generate musical arrangements controlled by emotional features. Music composition and arrangement are art crafts which involve specialized knowledge and experience. Prior work used artificial intelligence to either fully automate the music composition [2] and arrangement process [3] or develop assistive tools helping producers to compose new material through human-machine interaction [4]. Our work falls into the second category and focuses on generating harmonisation and tempo arrangements for composed melodies given emotional constraints. Deep learning (DL) was recently used to learn relationships between musical attributes (e.g. notes, chords) and associated emotions [5]. As discussed in [6], music emotions can be considered as being communicated by music (perceived emotions), and as being induced or evoked in listeners (felt emotions) [7]. Depending on the nature of the emotional annotations used during training (e.g. Tan et al. [8]), DL models can be aimed at producing music matching perceived or felt emotions. Music emotion recognition (MER) is one of the most challenging music information retrieval challenge, and new developments aim towards personalized and context-sensitive applications [9]. The proposed system generates harmonic and tempo arrangements for input melodies encoded in the symbolic domain so as to express specific emotions controlling the generation. Harmony and tempo were chosen for the inference stage as they have been shown to affect emotional expression: changes in chord progressions influence the emotions expressed by music [10]; tempo can greatly affect music emotions (especially in terms of arousal) [11]. A challenge in stirring DL generative models using emotion controls is the difficulty in finding training datasets containing both a large number of music examples and emotion labels [12]. We produced the HTPD3 Emotion Dataset (HED) released with this paper by collecting crowd-sourced emotion labels for the 4,000 tracks from the HTPD3 dataset [3]. Given the fairly small size of the dataset, we test the effectiveness of transfer learning for emotion-driven music generation using a network pre-trained only considering musical attributes.

Applications include the design of assistant tools helping composers/producers to create different arrangements given input melodies and emotional intentions. This may be of help to musicians who do not have advanced musical knowledge and to find inspiration in musical ideas generated by the machine. Another use case is interactive

© T. Takahashi, and M. Barthet. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** T. Takahashi, and M. Barthet, "Emotion-driven harmonisation and tempo arrangement of melodies using transfer learning", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

music systems which adapts to the user context, defined in [9] as the dynamic aspects from the listener that fluctuate frequently (e.g. physiological signals). If training was conducted using felt emotion labels, the method could be used for generative music produced on the fly driven by a user's felt emotions as predicted from e.g. biosignals. This could support affective gaming for example to produce responsive background music adapting itself to the emotional states of the game player, see e.g. [13].

2. RELATED WORK

A review of affective algorithmic composition dealing with automatic composition of music based on specific emotions can be found in Sulun et al. [5]. Guo et al. [14] proposed a variational autoencoder (VAE) for music generation controlled by tonal tension predicted from low-level symbolic music features. Tan et al. [8] introduced Music FaderNets enabling to stir music generation based on arousal - an emotional dimension related to excitation - using Gaussian Mixture VAEs (GM-VAEs). Makris et al. [12] proposed a method for assigning valence - an emotional dimension linked to pleasantness - to chords based on prior relationships between mood tags and chord qualities. This enabled the generation of lead sheet data (melody and chord) conditioned by valence, phrasing and time signature using a sequence-to-sequence model. Results from subjective evaluations with 42 participants showed consistency between targeted and perceived valence. However, a limitation is that only valence was considered but not arousal. Sulun et al. [5] recently proposed a promising approach for the generation of multi-instrument symbolic music driven by musical emotion using a Music Transformer architecture. The models can be conditioned by continuous-valued valence and arousal labels and yield results representative of current state of the art on a large scale dataset of 34791 songs. However, possible limitations towards generalisation come from the use of machine-predicted valence labels retrieved from Spotify and the modeling of arousal using MIDI note density.

3. DL ARCHITECTURE FOR AUTOMATIC ARRANGEMENT CONDITIONED BY EMOTIONS

The proposed DL architecture is divided into a melody context encoder and an arrangement decoder (Figure 1). The melody context encoder aims to capture information from the input melody taken as a sequence. Based on the encoded melodic context embedding and emotional information, the arrangement decoder predicts chords, harmonic functions, and tempo.

3.1 Melody Context Encoder

The melody context encoder is shown in the top part of Figure 1 and takes a representation of melodies as input and outputs a 128-dimensional embedding (similar to [15]) at every time unit. To reduce the dimensionality of the input, the melody is converted to a pitch class profile (PCP), as in [15]. A PCP is a 12-dimensional vector, in which each element of the vector contains the duration of each pitch class event. We compared the two following models for the melody context encoders (see Section 5):

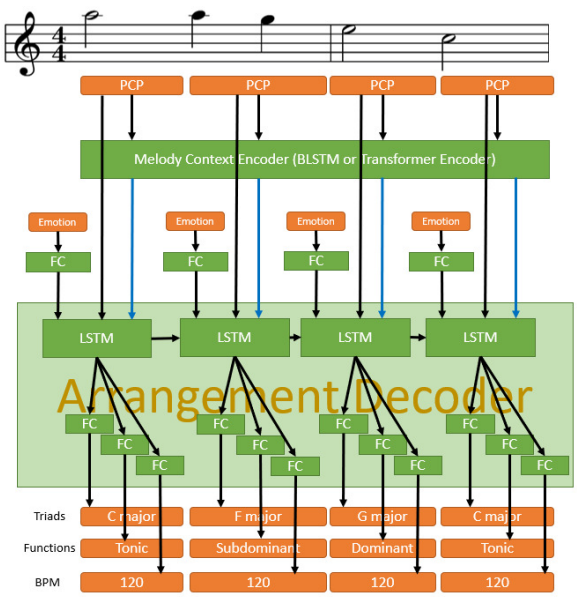


Figure 1. Architecture of the proposed model. The top represents the melodic context encoding, the bottom represents the arrangement generation (LSTM: long short-term memory network; FC: fully connected layers).

Bi-directional LSTM (BLSTM)
Inspired by the melody harmonizer proposed in [15], the same BLSTM model was used in this study with the aim of encoding the context of the melody.

Transformer encoder
The Transformer [16] is a network originally proposed for machine translation. Its self-attention mechanism supports more complex contexts and more efficient computations than BLSTM.

3.2 Arrangement Decoder

As shown at the bottom of Figure 1, the arrangement decoder is constructed using LSTM units only with forward propagation. This is to reduce the amount of computation required, and also to be able to make inferences based only on historical information for near real-time applications. The LSTM unit of each melodic time unit receives as input the hidden state of the past unit, the embedding of the melody context, PCP of the melody, and emotion conditions represented numerically. Finally, for every time units, the arrangement decoder outputs the chord labels and chord functions as a classification problem and the tempo as a regression problem. The output layer of each component consists of a fully connected layer. The loss function is expressed as:

$$L = \mathrm{CCE}(c, c^*) + 1.5\mathrm{CCE}(f, f^*) + 0.001\mathrm{MSE}(t, t^*) \quad (1)$$

where c, f and t are chord labels, chord functions, and tempo, respectively. c^*, f^* and t^* are the related groundtruth attributes. CCE represents the categorical cross-entropy and MSE represents the mean-squared error. The weight of each error was heuristically determined by observing the reduction in loss during training.

3.3 Training Strategies

When a sufficient amount of emotion-labelled musical data is available for training, the model can be commonly trained with a backpropagation algorithm. However, when the amount of emotion-labeled data is not sufficient, transfer learning strategy enabling to include data without emotion labels can be effective. In such case, encoders are pre-trained using music examples without emotion labels, then the pre-trained encoders (weights are fixed) and randomly initialized decoders are concatenated and retrained only for the subset of tracks with emotion labels. However, groundtruth data may not provide the best examples since there are several possible arrangements following music theory and perception considerations [3]. As in [3], training was stopped to a fixed number of epochs (500) without using validation, when the loss was significantly reduced (learning rate = 1e-3) and subjectively consistent arrangements were generated with the test data. Results on the effectiveness of the transfer learning strategy are reported in Section 5.4.

4. MUSIC EMOTION QUANTIFICATION

In order to input emotional conditions to the networks, perceived or felt emotions associated to music have to be represented numerically. We investigated three ways to map emotions into Russell's arousal-valence (AV) space [17].

4.1 Emotion Average Representation

In the Deezer Mood Detection Detaset [18], the emotional tags from last.fm [1] were mapped to arousal and valence values using [19]'s results. In [19], statistics on participant ratings for emotion words are reported for valence, arousal and dominance on a nine-point scale. In most cases, multiple emotion tags can be associated to music content. To address this, one of the methods used e.g. in [18, 20], consists in summarising multiple emotions tags using the geometrical mean of the tag projections in the emotion space (e.g. AV space). We used such method yielding two-dimensional emotional features from a set of mood tags for songs. These were normalized in the range 0-1 for network input. We refer to this emotional representation as emotion average representation (EAR) in the remainder.

4.2 Emotion Surface Representation

The EAR expresses emotions locally in the AV space. However, as investigated e.g. by [20], there is a possibility that a same song suggests/induces multiple emotions to a same listener, or different emotions to different listeners. Therefore, similar to [21], two-dimensional Gaussian mixture models (GMMs) can be used to represent perceived/felt emotions associated to multiple mood tags associated to songs in the AV space. The average and standard deviation corresponding to the mood tags associated to a song are obtained from the experimental results in [19]. Based on the average and standard deviation, random sampling is performed and 10000 samples are generated for each tag. Like for EAR, the emotional features were normalized in the range 0-1. Clustering is performed using two-dimensional GMMs based on the randomly sampled points. Two Gaussian components were assumed sufficient to represent the emotional feature surface in the AV plane as in [21]. Finally, the average and standard deviation of each Gaussian component were used as the network input representation. We refer to this emotional representation as the emotion surface representation (ESR) in the remainder.

4.3 Emotion Category Representation

EAR and ESR are both continuous. However, music emotions may not be best represented by a dimensional model [22]. Studies on music emotion recognition, such as [23], used discrete representations of emotions through categorical variables. We also tested emotional representations with discrete categories. We distinguish four quadrants in the AV space; Q1: high arousal & high valence [joyful], Q2: low arousal & high valence [relaxing], Q3: low arousal & low valence [sad], Q4: high arousal & low valence [angry]). The AV space quadrant with the highest AV annotations determines the emotional category of the music. We refer to this emotional representation as emotion category representation (ECR) in the remainder.

5. EXPERIMENTAL EVALUATION

In this section, the proposed emotion-driven automatic music arrangement systems are evaluated for differences in network architectures and the effect of transfer learning.

5.1 Dataset and training

Network training requires a dataset in which melody, chords, tempo and emotions (perceived or felt) are simultaneously available. As in [3], we rely on the HTPD3 dataset which provides symbolic melodies, chords, and tempo. Time units were set to two beats (half bars in the 4/4 time signature). The chord played for the longest time over two beats was selected as the chord in that time unit. Notes that spanned a segmentation were split. However, tracks in HTPD3 are not labelled with emotion features. We retrieved crowd-sourced mood tags from last.fm [1] and allmusic.com [2] for the song and artists contained in HTPD3. Tags from last.fm [1] and allmusic.com [2] have previously been used in music emotion studies, see e.g. [18]. As some of the last.fm tags are not related to emotions, we filtered these out using the criteria proposed in [24]. This

[1] https://www.last.fm/

[2] https://www.allmusic.com/

method allowed us to tag approximately 4000 tracks available in HTPD3. We refer to this expanded dataset as the HTPD3 Emotion Dataset (HED) available at the link below [3]. There is some uncertainty on whether crowdsourced mood tags relate to perceived or felt emotions. We assume here that these tags relate to perceived emotions, however this can limit the performance of the model in the experiments. Another caveat is that the mood tags relate to audio versions of songs, whereas our model deals with symbolic music. Hence, as in [5], they are considered as "weak labels" for symbolic music. The dataset was divided into 90% training data and 10% test data. As suggested in [25], the use of commercial songs to train AI models for research may be considered fair use.

5.2 Evaluation conditions and music stimuli

We conducted a listening experiment to assess the performance of the proposed model. The study received ethics approval from our institution. Four models (BLSTM without [BL] and with [BT] transfer learning, Transformer without [TR] and with [TT] transfer learning) and the groundtruth (G) were used to create five stimuli for each melody. As we do not investigate the role of key root in this study, the chosen input melodies were converted to either C major or C minor (both training and testing data). Since it is not possible to test all possible EAR and ESR in a continuous way, 15 emotion input presets were prepared for the evaluation. Four types of emotion are represented by EAR, another four by ECR, and seven emotion distributions using the ESR. These input emotions were determined heuristically in order to be able to express as various emotions as possible. The specific configurations are shown on the study website [4]. The chord progressions and tempo generated by the models were converted to MIDI along with the input melody. Audio was rendered from MIDI files using FluidSynth [26] using a SoundFont called SGM-V2.01. As the research [27] has suggested that humans can distinguish the timbre of brass instruments and guitars clearly, the melody part was played by a saxophone and the harmony part by a classical guitar.

5.3 Procedure and Participants

Participants were given instructions on how to complete the study on a dedicated website [5] which can be used to listen to examples of generated accompaniments. The website displayed participants melodies (sampled randomly from the test set) which were represented in the piano roll style. For each of the 15 cases (corresponding to emotional presets not explicitly revealed to participants), participants had to press the "Execute" button to obtain five (four models BT, TR, BT, TT, and groundtruth, G) musical arrangements to rate using three questions:

Q1 The melody and accompaniment were musically coherent. (Likert item: 0 [Disagree] - 6 [Agree]). Participants were instructed that musical coherence here represents a measure of how well the musical accompaniment matches the melody.

Q2 How exciting (arousal) do you perceive the music to be? (Continuous value from 0.0 to 1.0)

Q3 How negative or positive (valence) do you perceive the music to be? (Continuous value from 0.0 to 1.0)

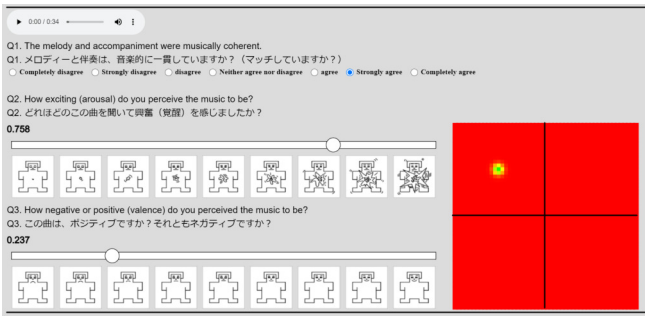


Figure 2. The evaluation interface (five panels had to be completed for each melody; four models plus groundtruth).

For Q2 and Q3, the self-assessment manikin [28] was used to support associations between numerical rating values and corresponding emotions, together with a representation of the rating in the AV space. Figure 2 provides a screenshot of the emotion rating interface. Participants rated 75 songs in total (15 emotion presets x 5 models) each lasting between approximately 15 to 30 seconds. As participants had to rate new arrangements, we assumed that familiarity with the melody, if any, did not affect the ratings (this would have to be assessed in future work). The whole experiment took about 45 minutes to complete.

Participants could not identify the models nor the groundtruth. Different melodies were randomly chosen from the test dataset and used for each emotional preset as, in a pilot testing phase, some participants expressed that listening to the same tune over and over made it difficult to evaluate after a while. For a given emotion preset, the five stimuli (BT, TR, BT, TT, and G) were generated for a same melody, to enable fair comparison between models.

The experiment was completed by 20 participants (7 females, 13 males). All participants were Japanese residents, 19 were Japanese and one was German. Their age ranged between 19 to 59 years (M=30.15, SD=14.05), where M and SD refer to mean and standard deviation, respectively. Six of them had at least one year of formal training in music theory. Eight had more than five years of formal training in their instrument (including voice).

5.4 Results

In statistical analyses, a Type I error of 0.05 was used except when mentioned otherwise.

5.4.1 Perceived musical coherence

Figure 3(a) illustrates the mean and standard error for each model, computed on the perceived musical coher-

[3] http://coconuts-palm-lab.com/EH/HED.zip
[4] http://coconuts-palm-lab.com/EmotionPresets/
[5] http://coconuts-palm-lab.com/EH/

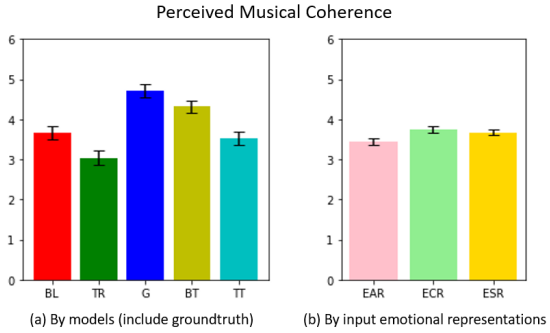


Figure 3. Results of a subjective evaluation experiment, with statistics on the musical coherence across comparisons. The means and standard errors are given.

ence considering all emotion presets. An analysis of variance (ANOVA) shows that the perceived music coherence yielded by the models presented significant differences (df=299, F = 66.750, p<.0001). Tukey's honestly significant difference (HSD) test shows that there is a significant difference between all pairs, except between the BLSTM melody context encoder without transfer learning and the Transformer melody context encoder with transfer learning. The BLSTM model with transfer learning achieves a significantly higher musical coherence compared to the other models. However, compared to the groundtruth, the arrangements generated by the machine learning models were found to be significantly less coherent.

Figure 3(b) displays the mean and standard error of the musical coherence for each emotional expression. ANOVA results showed a significant difference (df=319, F = 3.661, p = 0.025) and Tukey's HSD test showed that there was a significant difference only between EAR and ECR. This suggests that the type of emotional representation impacts perceived musical coherence. In particular, it is suggested that EAR may reduce the coherence of the generated music more than the other representations.

5.4.2 Errors between perceived and target emotions

By observing the error between target and rated emotions, it is possible to assess how well each model expresses the target emotions. The errors for the EAR were obtained using Euclidean distance. The ECR error was defined using the shortest distance between the emotion AV rating and the emotion category AV quadrant. The ESR error was defined by the negative log-likelihood that the rating belongs to the two Gaussians of the GMM component. It should be noted that each absolute emotion error is thus calculated on a different scale.

We also analyzed the relative emotion error, which indicates the degree of reflection of emotion, based on the ratio between the absolute error of the groundtruth and the absolute error of the arrangement generated by the model. If the absolute error for the music generated by the models is smaller than the absolute error for groundtruth, it suggests that the model may have been properly trained. Furthermore, the relative emotion error also makes it easier to compare different emotional representations. The relative error e_r is calculated as follows:

$$e_r = \frac{a_m}{a_g + \epsilon} \tag{2}$$

where a_m is the absolute emotion error for a model and a_g is the absolute emotion error for the groundtruth. ϵ is a small regularising term, avoiding cases where the denominator tends to zero.

The upper part of Figure 4 illustrates the mean and standard error of the absolute emotion error (a_m) for each model and emotional representation. The bottom part of Figure 4 shows relative emotion error (e_r) boxplots. The medians and quartiles are more appropriate to interpret the results than the averages (green triangles) which are strongly influenced by outliers. The blue dotted line is a threshold representing the absolute emotion error yielded by the groundtruth. If the relative emotion error is smaller than the threshold, it suggests that the model has been able to generate arrangements that are closer to the target emotion than the original groundtruth arrangement.

The hypothesis that all means are equal in the absolute error of the EAR is rejected based on the ANOVA (df=79, F=4.388, p=0.004). The results of Tukey's HSD test showed that there was a significant difference in absolute error between the BLSTM with transfer learning and the Transformer without transfer learning (p = 0.003). Moreover, based on the relative emotion errors of the EAR, it was found that the median and quartiles are smaller for BLSTM with transfer learning compared to without transfer learning. In particular, up to the third quartile, the errors were below the threshold, indicating that 75% of the arrangements generated by the model with the transfer-learned BLSTM are closer to the target emotion than the original one. The results show that the BLSTM with transfer learning can reduce the emotion error significantly more than the Transformer one.

ANOVA results suggest that the mean of the absolute error for ECR is not significantly different (df=79, F=1.757, p=0.155). Unlike for EAR, transfer learning does not seem to have had any particular impact for ECR.

According to ANOVA results, the mean of the absolute error for ESR is not significantly different (df=139, F=2.539, p=0.0557). The results of Tukey's HSD test also showed that there were only significant differences between the BLSTM with transfer learning and the Transformer without transfer learning (p = 0.042). When observing the relative error of the ESR, the smallest median, quartiles and mean were obtained with the BLSTM with transfer learning. Thus, when ESR was used, the emotional error was the smallest when BLSTM with transfer learning was used as for the EAR.

In addition, Figure 5 summarises the tempo of the generated arrangements for all the test data. The plotted points represent the actual tempo and the box plot shows the statistics. In all models, the tempo varied significantly when using ESR. This shows that ESR is the emotional representation that generates the most diverse tempi. The

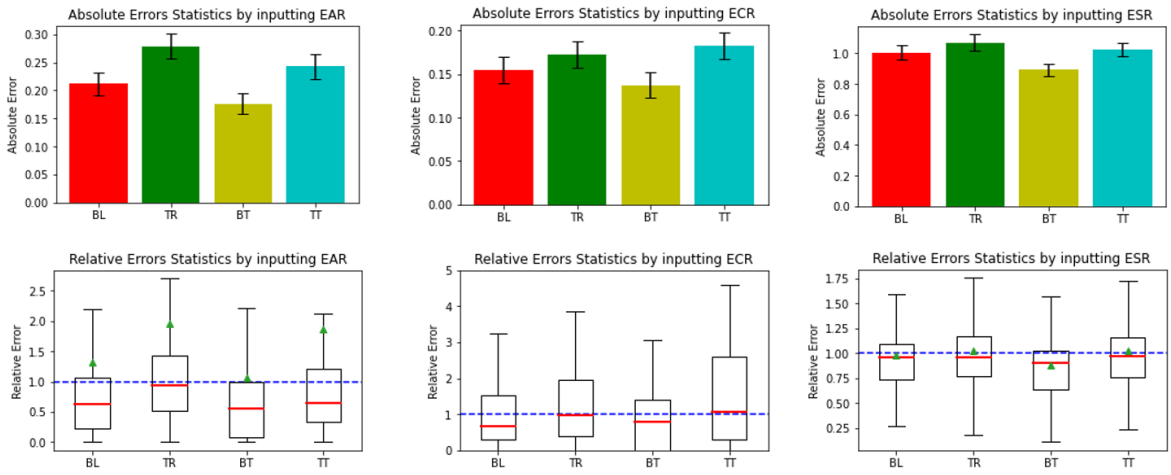


Figure 4. Results of the subjective evaluation experiment, with statistics on the absolute emotion error and relative emotion error across models.

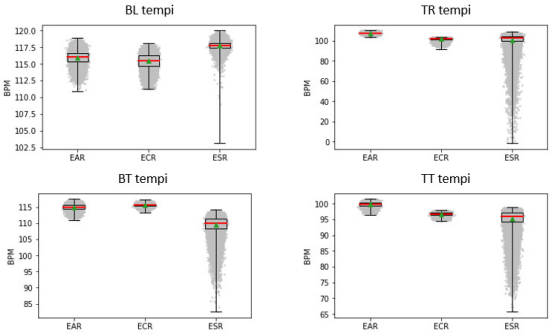


Figure 5. Statistics of the tempi generated for each model from the all melodies of the test data. Each point represents a generated tempo.

use of Gaussian variance, as in ESR, may support more complex emotional expressions.

5.5 Discussion

The highest perceived musical coherence and the lowest absolute and relative emotion errors were obtained when BLSTM with transfer learning was used for the melody context encoder. The results also show that transfer learning, a semi-supervised learning strategy, seems to be effective at generating emotional arrangements. A comparison of the relative errors of the BLSTM with transfer learning showed that the third quartile was not below the threshold only for the ECR, indicating that both EAR and ESR are superior representations for the emotion conditioning.

The generated arrangement could be computed quickly even with Intel(R) Core(TM) i7-9700K CPU (less than approximately 2.5 seconds per 16 bars of melody for any compared models). This could be useful for real-time music generation scenarios. Due to the simplicity of the model, the arrangement may be generated more quickly than state-of-the-art techniques such as that in [12] (approximately 50 seconds per 16 bars of melody for a lead sheet on the same CPU). However, the method in [12] generates the entire leadsheet, so the computation time cannot be directly compared to the ones reported here.

Several limitations should be highlighted. Observing the perceived musical coherence, there was no model in this experiment that could match the groundtruth. The reasons may be overfitting and a lack of data in the dataset. In most cases, the performance of the model was improved by transfer learning, so it is expected that more data and appropriate validation will help to build better models. Emotion errors remain relatively large and it is difficult to prove that the model consistently expresses the desired emotions. As it is unclear whether emotion labels in the training dataset represents perceived or felt emotions, this may contribute to inference errors. More knowledge about the listener's state and context would be needed to gauge more comprehensively emotion perception [9]. Results cannot be generalised since participants were from one provenance (Japan) and the sample size was small (20). More participants of different nationalities would be required to assess the generalisability of the proposed models.

6. CONCLUSION

We devised techniques for automatic harmonic and tempo arrangement of melodies controlled by emotional features, suitable for near real time applications. A network architecture to generate music expressing target emotions by predicting chord progressions and tempo for an input melody was proposed. In addition, three methods to quantify musical emotions were compared. To evaluate the results, we conducted an online listening experiment. For the melodic context encoder, the BLSTM model with transfer learning produced the most coherent arrangement and the one that best reflected the targeted emotions. The proposed method finds applications in assisting tools to create new music arrangements based on emotional directions and affective video games.

7. REFERENCES

[1] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning Interpretable Representation for Controllable Polyphonic Music Generation," in *Proc. of the 21st Int. Society for Music Information Retrieval Conference*, Montréal, Canada, 2020. [Online]. Available: https://github.com/

[2] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever, "Jukebox: A generative model for music," *arXiv Preprint arXiv:2005.00341*, 2020.

[3] Y.-C. Yeh, W.-Y. Hsiao, S. Fukayama, T. Kitahara, B. Genchel, H.-M. Liu, H.-W. Dong, Y. Chen, T. Leong, and Y.-H. Yang, "Automatic melody harmonization with triad chords: A comparative study," *Journal of New Music Research*, vol. 50, no. 1, pp. 37–51, 2021.

[4] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: A steerable model for bach chorales generation," in *International Conference on Machine Learning*. Proceedings of Machine Learning Research (PMLR), 2017, pp. 1362–1371.

[5] S. Sulun, M. E. Davies, and P. Viana, "Symbolic music generation conditioned on continuous-valued emotions," *IEEE Access*, 2022.

[6] C. L. Krumhansl, "An exploratory study of musical emotions and psychophysiology." *Canadian Journal of Experimental Psychology/Revue*, vol. 51, no. 4, p. 336, 1997.

[7] S. Swaminathan and E. G. Schellenberg, "Current emotion research in music psychology," *Emotion Review*, vol. 7, no. 2, pp. 189–197, 2015.

[8] H. H. Tan and D. Herremans, "Music fadernets: Controllable music generation based on high-level features via low-level feature modelling," *arXiv Preprint arXiv:2007.15474*, 2020.

[9] J. S. Gómez-Cañón, E. Cano, T. Eerola, P. Herrera, X. Hu, Y.-H. Yang, and E. Gómez, "Music emotion recognition: Toward new, robust standards in personalized and context-sensitive applications," *IEEE Signal Processing Magazine*, vol. 38, no. 6, pp. 106–114, 2021.

[10] A. J. Blood, R. J. Zatorre, P. Bermudez, and A. C. Evans, "Emotional responses to pleasant and unpleasant music correlate with activity in paralimbic brain regions," *Nature Neuroscience*, vol. 2, no. 4, pp. 382–387, 1999.

[11] E. Coutinho and A. Cangelosi, "Musical emotions: Predicting second-by-second subjective feelings of emotion from low-level psychoacoustic features and physiological measurements." *Emotion*, vol. 11, no. 4, p. 921, 2011.

[12] D. Makris, K. R. Agres, and D. Herremans, "Generating lead sheets with affect: A novel conditional seq2seq framework," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[13] Y. Frachi, T. Takahashi, F. Wang, and M. Barthet, "Design of emotion-driven game interaction using biosignals," in *Proc. HCII*, 2022.

[14] R. Guo, I. Simpson, T. Magnusson, C. Kiefer, and D. Herremans, "A variational autoencoder for music generation controlled by tonal tension," *arXiv Preprint arXiv:2010.06230*, 2020.

[15] H. Lim, S. Rhyu, and K. Lee, "Chord generation from symbolic melody using blstm networks," *arXiv Preprint arXiv:1712.01011*, 2017.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[17] J. A. Russell, "A circumplex model of affect." *Journal of Personality and Social Psychology*, vol. 39, no. 6, p. 1161, 1980.

[18] R. Delbouys, R. Hennequin, F. Piccoli, J. Royo-Letelier, and M. Moussallam, "Music mood detection based on audio and lyrics with deep neural net," *arXiv Preprint arXiv:1809.07276*, 2018.

[19] A. B. Warriner, V. Kuperman, and M. Brysbaert, "Norms of valence, arousal, and dominance for 13,915 english lemmas," *Behavior Research Methods*, vol. 45, no. 4, pp. 1191–1207, 2013.

[20] M. Barthet, D. Marston, C. Baume, G. Fazekas, and M. Sandler, "Design and evaluation of semantic mood models for music recommendation," in *Proc. International Society for Music Information Retrieval Conference*, 2013.

[21] Y. E. Kim, E. M. Schmidt, R. Migneco, B. G. Morton, P. Richardson, J. Scott, J. A. Speck, and D. Turnbull, "Music emotion recognition: A state of the art review," in *International Society for Music Information Retrieval (ISMIR) 2010*, vol. 86, 2010, pp. 937–952.

[22] M. Barthet, G. Fazekas, and M. Sandler, "Music emotion recognition: From content-to context-based models," in *International Symposium on Computer Music Modeling and Retrieval*. Springer, 2012, pp. 228–252.

[23] C. Laurier, O. Lartillot, T. Eerola, and P. Toiviainen, "Exploring relationships between audio features and emotion in music," in *ESCOM 2009: 7th Triennial Conference of European Society for The Cognitive Sciences of Music*, 2009.

[24] M. Buffa, E. Cabrio, M. Fell, F. Gandon, A. Giboin, R. Hennequin, F. Michel, J. Pauwels, G. Pellerin, M. Tikat *et al.*, "The wasabi dataset: Cultural, lyrics and audio analysis metadata about 2 million popular commercially released songs," in *European Semantic Web Conference*. Springer, 2021, pp. 515–531.

[25] OpenAI, "Uspto comment regarding request for comments on intellectual property protection for artificial intelligence innovation," 2019, available: https://www.uspto.gov/sites/default/files/documents/OpenAI_RFC-84-FR-58141.pdf.

[26] J. Newmarch, "Fluidsynth," in *Linux Sound Programming*. Springer, 2017, pp. 351–353.

[27] S. McAdams, "Musical timbre perception," *The Psychology of Music*, pp. 35–67, 2013.

[28] M. M. Bradley and P. J. Lang, "Measuring emotion: The self-assessment manikin and the semantic differential," *Journal of Behaviour Therapy and Experimental Psychiatry*, vol. 25, no. 1, pp. 49–59, 1994.

USING ACTIVATION FUNCTIONS FOR IMPROVING MEASURE-LEVEL AUDIO SYNCHRONIZATION

Yigitcan Özer[1] **Matěj Ištvánek**[2] **Vlora Arifi-Müller**[1] **Meinard Müller**[1]
[1] International Audio Laboratories Erlangen, Germany
[2] Brno University of Technology, Brno, Czech Republic
yigitcan.oezer@audiolabs-erlangen.de, matej.istvanek@vut.cz,
vlora.arifi-mueller@audiolabs-erlangen.de, meinard.mueller@audiolabs-erlangen.de

ABSTRACT

Audio synchronization aims at aligning multiple recordings of the same piece of music. Traditional synchronization approaches are often based on dynamic time warping using chroma features as an input representation. Previous work has shown how one can integrate onset cues into this pipeline for improving the alignment's temporal accuracy. Furthermore, recent work based on deep neural networks has led to significant improvements for learning onset, beat, and downbeat activation functions. However, for music with soft onsets and abrupt tempo changes, these functions may be unreliable, leading to unstable results. As the main contribution of this paper, we introduce a combined approach that integrates activation functions into the synchronization pipeline. We show that this approach improves the temporal accuracy thanks to the activation cues while inheriting the robustness of the traditional synchronization approach. Conducting experiments based on string quartet recordings, we evaluate our combined approach where we transfer measure annotations from a reference recording to a target recording.

1. INTRODUCTION

In music information retrieval (MIR), synchronization techniques are essential for several applications including score following [1], content-based retrieval [2], automatic accompaniment [3], or performance analysis [4, 5]. Beside these applications, music synchronization has a great potential to simplify data augmentation, data annotation, and model evaluation. For example, one can use music synchronization to obtain additional training data for deep learning methods semi-automatically by transferring annotations from one recording to another. Furthermore, using music synchronization, one can transfer measure positions between audio recordings for navigation purposes, structural segmentation, and cross-version analysis [6, 7].

© Yigitcan Özer, Matěj Ištvánek, Vlora Arifi-Müller, Meinard Müller. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Yigitcan Özer, Matěj Ištvánek, Vlora Arifi-Müller, Meinard Müller, "Using Activation Functions for Improving Measure-Level Audio Synchronization", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

While traditional synchronization approaches typically rely on alignment algorithms such as dynamic time warping (DTW) and conventional chroma features used as the input representation [8, 9], the integration of additional onset-related information has proven to enhance the synchronization accuracy [10–12]. Inspired by the combined approach from [12], where *decaying locally adaptive chroma onset* (DLNCO) features are integrated into the synchronization pipeline, we incorporate in this paper onset, beat, and downbeat activation functions to obtain a better temporal accuracy while retaining the robustness of the original chroma-based synchronization approach (see Figure 1 for an illustration of the overall approach). The addition of activation functions results in a grid-like structure in the cost matrix, which guides the alignment through activation cues that point to note onsets or other musical events.

While the integration of DLNCO and spectral flux (SF) have led to substantial improvement of synchronization results [12,13], the detection of soft onsets constitutes a challenging problem due to their long attack phase with a slow rise in energy. To adapt the onset detection task to music recordings which comprise soft onsets and temporal–spectral modulations such as vibrato (e.g., string music), Böck and Widmer [14] introduced the superflux (SF*) feature. Furthermore, deep learning (DL) methods such as bidirectional long short-term memory (BLSTM) networks [15] and convolutional neural networks (CNN) [16] have led to significant improvements compared to conventional onset detectors.

As the main contribution of this paper, we show how one can integrate conventional and DL-based activation functions into the synchronization pipeline. Different from the approach in [12], we do not apply any hard peak picking but directly use onset-related activation cues. Furthermore, we go beyond onsets by integrating activation functions that indicate onset, beat, and downbeat positions. In particular for music with noisy and unreliable onset cues, we show that beat and downbeat cues are more reliable and better suited for improving the synchronization accuracy. For extracting beat and downbeat activation functions, we build on recent work by Böck et al. [17, 18], using recurrent neural network (RNN) models for extracting beat and downbeat activation functions.

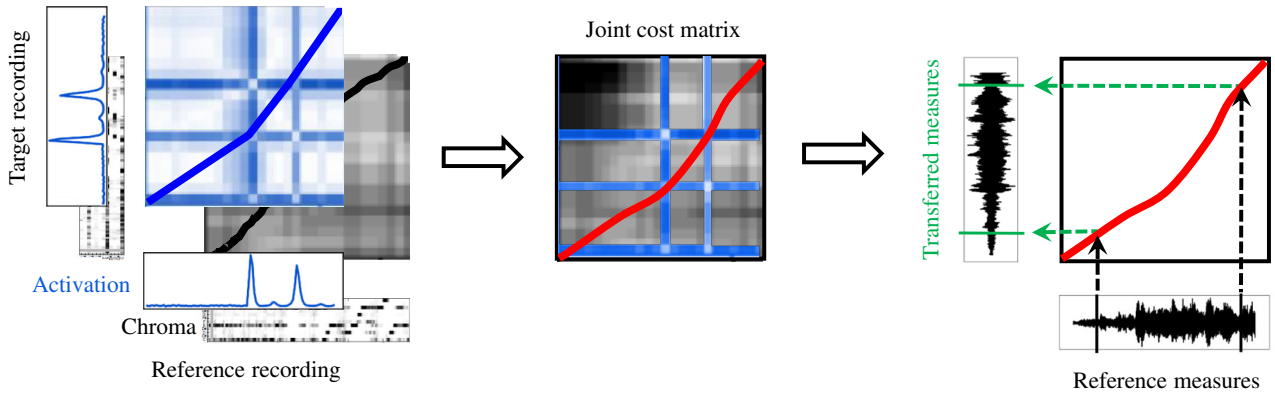


Figure 1: Overview of the combined approach integrating activation functions into a conventional chroma-based synchronization pipeline. The cost matrix computed with activation cues (blue) yields a grid-like structure to guide the alignment of musical events, whereas the chroma-based cost matrix (black) accounts for the robustness of the overall synchronization. The resulting warping path (red) is used for transferring measure positions.

To better understand our improved synchronization pipeline, we compare several synchronization approaches where we transfer measure annotations from a reference recording to a target recording, similar to [19]. In particular, we conduct systematic experiments based on three versions of the String Quartet No. 12 in F major, Op. 96, composed by Antonín Dvořák. As string music generally comprises vibrato, tremolo, rubato, and abrupt tempo changes, which increase the musical complexity, the synchronization of string quartets is a challenging scenario. We show that integrating DL-based activation functions significantly improves the temporal accuracy while retaining the robustness of chroma features.

The remainder of the paper is organized as follows. In Section 2, we introduce our combined synchronization approach, explore conventional and DL-based activation cues, and show how to integrate activation functions into the synchronization pipeline. In Section 3, we present our dataset, the measure transfer between string quartet recordings as our application scenario, and report on our systematic experiments and empirical results. Finally, we conclude in Section 4 with prospects on future work.

2. COMBINED SYNCHRONIZATION APPROACH

In this section, we show how activation functions can be integrated into the synchronization pipeline to enhance the temporal accuracy of traditional chroma-based synchronization approaches. Here, we regard the activation function as a function that yields a value between 0 and 1. Each entry in the function indicates the likelihood of a certain musical event, e.g., onsets, beats, or downbeats, for each frame (time position). In the ideal case, the value of the activation function is one when an event occurs and zero otherwise. Note that we do not apply any peak picking in our approach, but only use the activation functions as temporal cues. This is opposed to onset detection or beat tracking where one needs to apply a temporal decoding method to obtain an explicit representation of onset and beat positions from the activation functions.

In the following, we first explore conventional onset-based activation functions in Section 2.1. Then, in Section 2.2, we investigate DL-based onset, beat and downbeat detectors. Finally, in Section 2.3, we explain how we integrate activation functions into the synchronization pipeline.

2.1 Conventional Onset-Based Activation Functions

2.1.1 DLNCO

DLNCO features are 12-dimensional pitch-based onset features, which combine the robustness of the chroma features with the accuracy of one-dimensional onset features. To compute DLNCO features, we first apply a pitch-wise audio decomposition. Then, we derive pitch-wise onset cues by considering points of energy increase (see Figure 2c for an illustration). DLNCO features are particularly suited for the music with clear note attacks such as piano music. For further details about the computation of DLNCO features, we refer to [12].

2.1.2 SF

Spectral flux (SF) captures the changes in the spectral content of an audio signal, and is widely used for onset detection [9, 20]. For the computation of SF, we apply a first-order differentiator on the log-compressed magnitude spectrogram of a music recording. Half-wave rectification follows the differentiation to keep only the positive differences between subsequent frames. As a final step, we subtract a local average function to enhance the peak structure (see Figure 2d).

2.1.3 SF*

Superflux (SF*) is a modified version of SF for detecting soft onsets [14]. These features are suitable for music recordings with vibrato, such as strings quartets. Similar to

the SF algorithm, SF* also relies on the detection of positive changes in the energy over time. However, it includes a trajectory-tracking stage through maximum filtering, instead of simply calculating the difference between spectral bins over time. Trajectory tracking helps to suppress spurious spectral peaks, especially arising from vibrato. For further information, we refer to [14] (see also Figure 2e).

2.2 DL-Based Activation Functions

2.2.1 CNN Onset Detector

Schlüter and Böck [16] approach the onset detection task as a computer vision problem, where magnitude spectrograms of the audio recordings are used as the input to a CNN. Onsets are often characterized by rapid transient changes in the spectrum, resulting in sharp edges that are clearly visible in a spectrogram. Using convolutional kernels, one can easily detect these sharp edges of onsets. Similar to SF-based methods, the proposed CNN model computes spectro–temporal differences and captures percussive and pitched onsets. The resulting activation function is referred to as `DL-O` (see Figure 2f for an example).

2.2.2 RNN Beat Detector

In the case of unreliable and noisy onset cues, using beat activation functions constitutes a more feasible solution to improve the temporal alignment. To compute such activation functions, we use the BLSTM model by Böck and Schedl [17] for framewise beat detection. BLSTMs can effectively model the temporal context of the data and is therefore suitable for beat tracking. In the proposed approach, magnitude spectrograms computed with three different window lengths, and their first order differences are used as the input to the network. The network outputs encode the likelihoods of beat positions, as illustrated by Figure 2g. In our experiments, the resulting beat activation functions are denoted as `DL-B`.

2.2.3 RNN Downbeat Detector

Böck et. al [18] present an RNN model to jointly detect beat and downbeats. Like the previously mentioned onset and beat detectors, this model also operates on magnitude spectrograms. The downbeat detector uses an RNN similar to the proposed network in [17] to model beats and downbeats. In our experiments, we only use the probability of downbeats as the activation cues, which we denote as `DL-D` (see Figure 2h for an illustration).

2.3 Combined Synchronization with Activation Functions

To find the optimal alignment between two feature sequences $X := (x_1, \ldots, x_N)$ and $Y := (y_1, \ldots, y_M)$, where $n \in \{1, \ldots, N\}$ and $m \in \{1, \ldots, M\}$, we rely on DTW. By comparing each pair of elements in the feature sequences, we obtain a cost matrix $\mathbf{C}(n, m) := c(x_n, y_m)$ of size $N \times M$, where c defines a local cost measure. Then, an *optimal warping path* is determined via dynamic programming. We refer to [9] for a detailed account on DTW

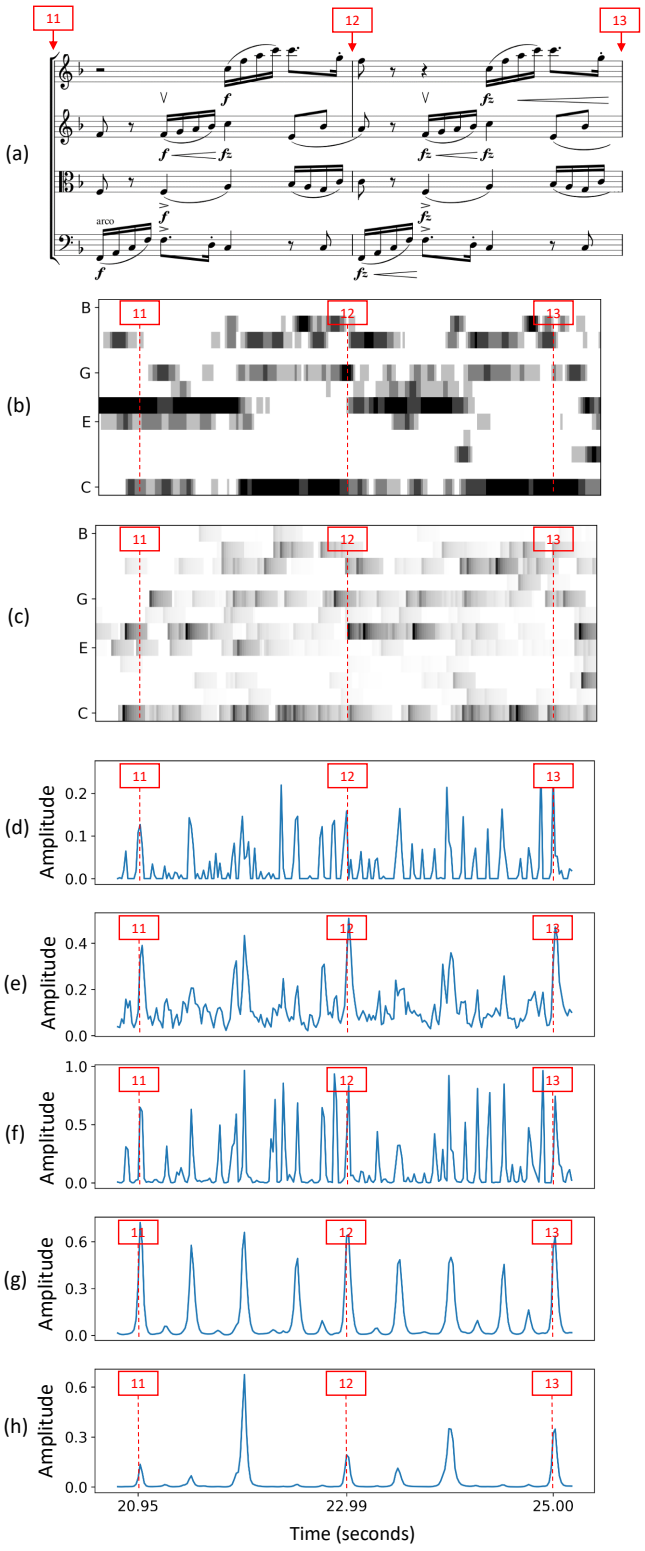


Figure 2: Chroma features and activation functions computed for an excerpt of the String Quartet No. 12 in F major, Op. 96 (first movement) composed by Antonín Dvořák, performed by the Borromeo Ensemble. Activation functions are shown in blue and ground-truth measure positions in red. **(a)** Sheet music representation of the measures 11–13. **(b)** Chroma **(c)** DLNCO **(d)** SF **(e)** SF* **(f)** Onsets (`DL-O`) **(g)** Beats (`DL-B`) **(h)** Downbeats (`DL-D`)

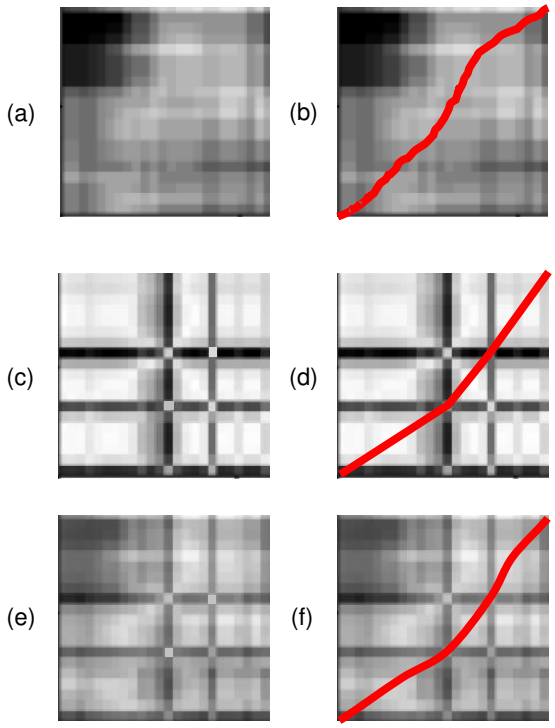


Figure 3: Excerpts from cost matrices and corresponding warping paths computed with DTW **(a) / (b)**: $\mathbf{C}_{\texttt{CHROMA}}$, **(c) / (d)**: $\mathbf{C}_{\texttt{ACT}}$, **(e) / (f)**: $\alpha\mathbf{C}_{\texttt{CHROMA}} + (1-\alpha)\mathbf{C}_{\texttt{ACT}}$, with $\alpha = 0.5$.

for music synchronization. For efficient implementations, we refer to [21, 22].

We now adapt the combined synchronization idea by Ewert et al. [12], integrating conventional onset-based and DL-based activation functions into the synchronization pipeline. Building upon this approach, we introduce three cost matrices. The first one $\mathbf{C}_{\texttt{CHROMA}}$ is a cost matrix based on the normalized chroma features and the cosine distance, see Figure 3a for an example. The second cost matrix $\mathbf{C}_{\texttt{ACT}}$ is computed using the Euclidean distance and conventional onset-based or DL-based activation functions as introduced in Section 2.1 and Section 2.2, respectively. Note that $\mathbf{C}_{\texttt{ACT}}$ exhibits grid-like structures. The visible horizontal and vertical grid lines of high cost (shown in black) correspond to high values in the first and second function, respectively. Only when a horizontal grid line intersects with a vertical grid line, the cost matrix has a small cost value at this intersection point (and also in a small neighborhood). This is where a high activation value of the first sequence meets another high activation value of the second sequence. In other words, these *intersection cells* encode a pair of time positions where two musical events (onsets, beats, downbeats) meet (see Figure 3c). Furthermore, the sections in the activation functions which have low values lead to homogeneous, zero-cost regions in the cost matrix $\mathbf{C}_{\texttt{ACT}}$. The third matrix is the sum of two cost matrices

$$\mathbf{C} = \alpha\mathbf{C}_{\texttt{CHROMA}} + (1-\alpha)\mathbf{C}_{\texttt{ACT}}, \qquad (1)$$

where $\alpha \in [0, 1]$ is a weighting parameter. The sum $\mathbf{C}$ accounts for both harmonic or melodic information of the representations via $\mathbf{C}_{\texttt{CHROMA}}$ and additional activation cues via $\mathbf{C}_{\texttt{ACT}}$. Figure 3e illustrates an example using $\alpha = 0.5$.

Comparing the resulting optimal warping paths using $\mathbf{C}_{\texttt{CHROMA}}$ in Figure 3b, $\mathbf{C}_{\texttt{ACT}}$ in Figure 3d, and $\mathbf{C}$ in Figure 3f, we can observe an enhancement of the temporal alignment. The inclusion of DL-based onset, beat, and downbeat cues leads to an improvement of the warping path guided by grid structure's intersection points. Note that $\mathbf{C}_{\texttt{ACT}}$ remains zero in the regions without any novel events, and the overall alignment of $\mathbf{C}$ is mainly guided by $\mathbf{C}_{\texttt{CHROMA}}$.

3. EXPERIMENTS

3.1 Dataset

The genre of string quartet is composed for a small conductor-less ensemble, which consists of two violins, a viola and a violoncello. In our experiments, we use three versions (performances) of the String Quartet No. 12 in F major, Op. 96, by Antonín Dvořák, which comprises four movements. To give an insight of the musical properties of the string quartet, Table 1 provides the number of measures, time signature, and global tempo for each movement based on the recordings in our dataset. Note that the global tempo does not reflect any local tempo deviations. Its purpose is to indicate at what pace (average of three performances) a given movement is performed. In each recording, the repetitions are played as notated in the sheet music, thus ensuring structural consistency.

Movement	#Measures	Time signature	Global Tempo
M1	239	4/4	100
M2	97	6/8	84
M3	244	3/4	185
M4	382	2/4	140

Table 1: Overview of four movements, including number of measures, time signature, and global tempo in BPM for each movement.

For each recording (in the following referred to as version), we manually annotated the measure positions. In Table 2, the name of the version, the identifier, the recording year for each version, and the duration of each movement are listed. Note that each of the three performances last around 26 minutes in total, whereas durations of the movements may vary across different versions.

			Duration (seconds)				
Version	ID	Year	M1	M2	M3	M4	Σ
Alban Berg	A	1991	599	410	238	323	1570
Borromeo	B	2012	540	465	228	338	1571
Prague	P	1973	584	424	250	322	1580
		Σ	1723	1299	716	983	4721

Table 2: Version, identifier, recording year, and duration of each movement.

	DL-B & DBN			DL-D & DBN		
	P	R	F	P	R	F
M1	0.194	0.806	0.312	0.648	0.586	0.612
M2	0.138	0.820	0.236	0.657	0.643	0.650
M3	0.327	0.539	0.407	0.524	0.224	0.314
M4	0.517	0.908	0.655	0.876	0.647	0.742
ϕ	0.294	0.768	0.403	0.676	0.525	0.580

Table 3: Precision (P), Recall (R), and F-measure (F) based on methods DL-B for beat, and DL-D for downbeat tracking with DBN post-processing, evaluated on reference measure annotations and tolerance $\tau = 70$ ms. ϕ denotes the average accuracy over four movements M1, M2, M3, and M4.

	$\tau = 30$ ms		$\tau = 70$ ms		$\tau = 100$ ms	
	CHROMA	DL-OBD	CHROMA	DL-OBD	CHROMA	DL-OBD
M1	0.408	0.576	0.723	0.813	0.817	0.877
M2	0.396	0.600	0.694	0.842	0.789	0.917
M3	0.528	0.655	0.827	0.912	0.910	0.956
M4	0.485	0.662	0.773	0.884	0.861	0.930
ϕ	0.454	0.624	0.754	0.863	0.844	0.920

Table 4: Accuracy values based on chroma and DL-OBD for different tolerances $\tau = 30,\ 70,\ 100$ ms. ϕ denotes the average accuracy over four movements M1, M2, M3, and M4.

3.2 Beat and Downbeat Tracking

As a baseline, we first introduce how DL-based beat [17] and downbeat trackers [18] perform in the detection of measure positions, where a dynamic Bayesian network (DBN) was used for post-processing (peak picking). Table 3 provides precision, recall, and F-measure values using a tolerance of $\tau = 70$ ms. Here, each entry indicates the mean value over different performances. Due to the higher density of beats, the beat tracker reveals a higher recall than the downbeat tracker for each movement, leading to a low precision and F-measure. Furthermore, each movement has a different time signature, which results in a different number of beats per measure and needs to be taken into consideration as prior information for the DBN.

3.3 Synchronization Results

In this section, we describe our experimental setting and evaluate audio alignments obtained using chroma features and different activation functions. In our experiments, we use the resulting warping path to transfer the measure positions annotated for the reference recording to the target recording. Given two versions of the same music piece with the time-continuous axes $[0, T_1]$ and $[0, T_2]$, the monotonous alignment can be modeled as a function

$$\mathcal{A} : [0, T_1] \to [0, T_2].$$

The pairwise alignment error ϵ_P for a given alignment of two recording is specified as the mean over the values

$$\epsilon_P(g_1) := |\mathcal{A}(g_1) - g_2|,$$

where $(g_1, g_2) \in [0, T_1] \times [0, T_2]$ denotes the ground-truth pairs of measure annotations. As an evaluation metric, we use *accuracy*, which is defined as the proportion of correctly transferred measure positions with a pairwise alignment error below a given tolerance τ [23].

In the following, we use eight different synchronization approaches based on conventional chroma features and the combination of chroma features with DLNCO features, SF, SF*, DL-based onsets (DL-O), DL-based beats (DL-B), DL-based downbeats (DL-D), and finally a 3-dimensional stacked activation function combining DL-based onsets, beats, and downbeats (DL-OBD) (see Section 2 for a detailed overview of activation functions). We use a feature rate of 50 Hz for the computation of chroma, and conventional onset features. To generate DL-based features, we use the *madmom* [24] library, for which we utilize the default setting 100 Hz as the feature rate, and downsample generated features to 50 Hz (after low-pass filtering).

3.3.1 Overall Result

To get a first impression of the alignment behavior of different approaches, Figure 4 illustrates an overview of average accuracy values (averaged over all movements and all pairs of different performances) and for different tolerances τ. Obviously, one can observe that the synchronization accuracy improves with increasing threshold. For example, using $\tau = 30$ ms, the average synchronization accuracy is 0.454 when using only chroma features, and the accuracy increases to 0.624 when integrating the DL-OBD activation cues. This is a substantial improvement.

Next, we focus on the comparison of different approaches for $\tau = 70$ ms, which is a common tolerance value for the evaluation of music synchronization and beat tracking procedures. The inclusion of DLNCO slightly worsens the alignment since DLNCO features are not suited for soft onsets as occuring in string music. However, the integration of SF and SF* into the synchronization pipeline results in a better accuracy. Among conventional onset features, SF* shows a better performance than SF and DLNCO, owing to the fact that SF* features account for the detection of soft onsets and are therefore more suitable for string music. Furthermore, using DL-based methods leads to a better synchronization result compared to the conventional methods. Note that the integration of DL-OBD, which combines DL-O, DL-B, DL-D, reveals the best accuracy among all the synchronization approaches. It is also interesting to observe that DL-B, which is based on beats, is the second-best among DL-based approaches and leads to better accuracy values than downbeat-based DL-D, whereas DL-O reveals the lowest accuracy among the DL-based approaches.

In general, similar trends can be observed when using other thresholds. Using $\tau = 500$ ms, all synchronization approaches yield nearly perfect results.

3.3.2 Dependency on Movement

In our next experiment, we analyze the synchronization accuracy across different movements. Table 4 provides a comparison of alignment results based on the conventional chroma-based approach and our proposed combined

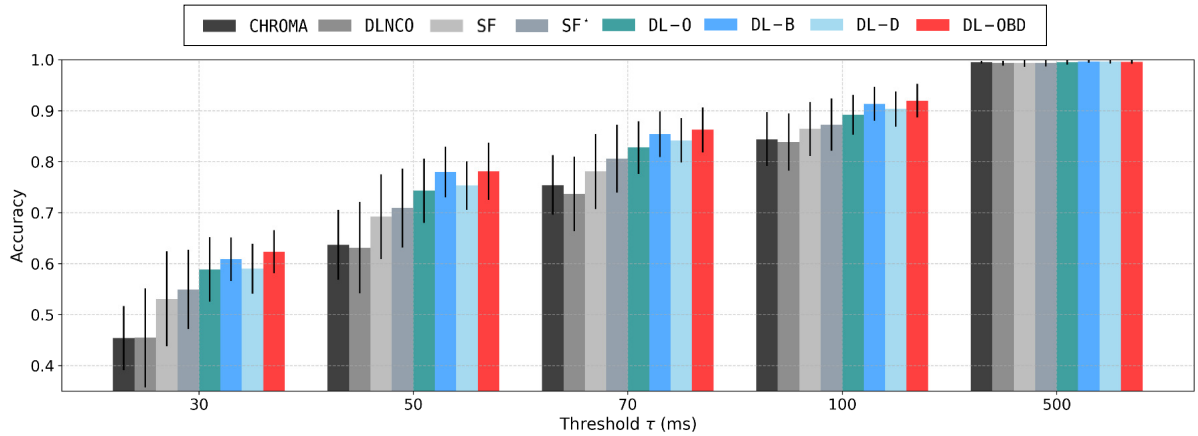


Figure 4: Comparison of the average accuracy values for different synchronization approaches and different threshold parameters τ. The accuracy denotes the proportion of correctly transferred measure positions having an error below a given tolerance τ.

	$\tau = 30$ ms		$\tau = 70$ ms		$\tau = 100$ ms	
	CHROMA	DL-OBD	CHROMA	DL-OBD	CHROMA	DL-OBD
AP	0.494	0.636	0.774	0.864	0.839	0.928
PA	0.500	0.632	0.778	0.864	0.847	0.928
AB	0.434	0.576	0.722	0.849	0.830	0.907
BA	0.451	0.580	0.743	0.863	0.857	0.920
BP	0.429	0.662	0.762	0.870	0.850	0.919
PB	0.417	0.657	0.746	0.866	0.843	0.915
ϕ	0.454	0.624	0.754	0.863	0.844	0.920

Table 5: Accuracy based on chroma and DL-OBD across different synchronization pairs, for different tolerances τ. The first column indicates the pair of versions (see Section 3.1).

method DL-OBD per movement for different tolerances τ. For example, considering the first movement and $\tau = 70$ ms, the accuracy of 0.723 for the chroma-based approach increases to 0.813 when using our combine approach DL-OBD. One can observe a similar trend across different movements for different tolerance parameters τ. The second movement tends to yield the lowest accuracy values when using only chroma features, while the integration of DL-OBD significantly improves the synchronization accuracy from 0.694 to 0.842 for the second movement. One reason may be that the beat and downbeat information leads to a significant improvement in synchronization accuracy of slower sections.

3.3.3 Dependency on Performance

As a final experiment, we provide a comparison of the accuracy values across different performances for different tolerance parameters τ in Table 5. In general, using a combination of chroma and activation functions significantly improves the accuracy for all the synchronization pairs and tolerances. For $\tau = 70$ ms, chroma reveals an average accuracy of 0.774 for AP and DL-OBD improves the accuracy to 0.864. Remarkably, across other synchronization pairs the synchronization accuracy values are similar. Nonetheless, deviations may occur due to soft onsets, slight inconsistencies in ground-truth annotations, and linear interpolation while measure transfer. Note that the synchronization accuracy rather depends on the musical complexity and structure, e.g., across different movements, but not on the performances.

4. CONCLUSION

In this article, we investigated the incorporation of conventional onset features, and activation cues obtained by recent DL-based onset, beat, and downbeat detectors to a conventional chroma-based synchronization pipeline. We showed that the integration of a combined version of onset, beat, and downbeat activation functions significantly improves the synchronization accuracy while maintaining the robustness of the original chroma-based synchronization approach. For the future, we will incorporate other features, which capture smoother note transitions for string quartets. We also aim at extending our string quartet dataset and evaluating the synchronization on beat-level annotations. Moreover, we will further investigate the role of the hyperparameter α (see Equation 1), which can be optimized as a time-dependent parameter.

Acknowledgements: This work was supported by the German Research Foundation (DFG MU 2686/10-2), and "Identification of the Czech origin of digital music recordings using machine learning" grant, which is realized within the project Quality Internal Grants of BUT (KInG BUT), Reg. No. CZ.02.2.69/0.0/0.0/19_073/0016948 and financed from the OP RDE. The International Audio Laboratories Erlangen are a joint institution of the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) and Fraunhofer Institut für Integrierte Schaltungen IIS.

5. REFERENCES

[1] D. Schwarz, N. Orio, and N. Schnell, "Robust polyphonic midi score following with hidden Markov models," in *International Computer Music Conference (ICMC)*, Miami, Florida, USA, 2004.

[2] J. T. Foote, "Content-based retrieval of music and audio," in *Multimedia Storage and Archiving Systems II*, vol. 3229, International Society for Optics and Photonics. SPIE, 1997, pp. 138–147.

[3] R. B. Dannenberg, "An on-line algorithm for real-time accompaniment," in *Proceedings of the International Computer Music Conference (ICMC)*, Paris, France, 1984, pp. 193–198.

[4] C. S. Sapp, "Comparative analysis of multiple musical performances," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Vienna, Austria, 2007, pp. 497–500.

[5] A. Lerch, C. Arthur, A. Pati, and S. Gururani, "Music performance analysis: A survey," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019, pp. 33–43.

[6] V. Konz, M. Müller, and R. Kleinertz, "A cross-version chord labelling approach for exploring harmonic structures—a case study on Beethoven's Appassionata," *Journal of New Music Research*, vol. 42, no. 1, pp. 61–77, 2013.

[7] C. Weiß, H. Schreiber, and M. Müller, "Local key estimation in music recordings: A case study across songs, versions, and annotators," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2919–2932, 2020.

[8] R. B. Dannenberg and N. Hu, "Polyphonic audio matching for score following and intelligent audio editors," in *Proceedings of the International Computer Music Conference (ICMC)*, San Francisco, USA, 2003, pp. 27–34.

[9] M. Müller, *Fundamentals of Music Processing – Using Python and Jupyter Notebooks*, 2nd ed. Springer Verlag, 2021.

[10] B. Niedermayer and G. Widmer, "A multi-pass algorithm for accurate audio-to-score alignment," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, 2010, pp. 417–422.

[11] A. Arzt, G. Widmer, and S. Dixon, "Adaptive distance normalization for real-time music tracking," in *Proceedings of the European Signal Processing Conference (EUSIPCO)*, Bucharest, Romania, 2012, pp. 2689–2693.

[12] S. Ewert, M. Müller, and P. Grosche, "High resolution audio synchronization using chroma onset features," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Taipei, Taiwan, Apr. 2009, pp. 1869–1872.

[13] P. Grosche, M. Müller, and S. Ewert, "Combination of onset-features with applications to high-resolution music synchronization," in *Proceedings of the International Conference on Acoustics (NAG/DAGA)*, 2009, pp. 357–360.

[14] S. Böck and G. Widmer, "Maximum filter vibrato suppression for onset detections," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Maynooth, Ireland, 2013.

[15] F. Eyben, S. Böck, B. Schuller, and A. Graves, "Universal onset detection with bidirectional long short-term memory neural networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Utrecht, The Netherlands, August 2010, pp. 589–594.

[16] J. Schlüter and S. Böck, "Improved musical onset detection with convolutional neural networks," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Florence, Italy, May 2014, pp. 6979–6983.

[17] S. Böck and M. Schedl, "Enhanced beat tracking with context-aware neural networks," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Paris, France, 2011, pp. 135–139.

[18] S. Böck, F. Krebs, and G. Widmer, "Joint beat and downbeat tracking with recurrent neural networks," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, New York City, New York, USA, 2016, pp. 255–261.

[19] C. Weiß, V. Arifi-Müller, T. Prätzlich, R. Kleinertz, and M. Müller, "Analyzing measure annotations for Western classical music recordings," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, New York, USA, 2016, pp. 517–523.

[20] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A tutorial on onset detection in music signals," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 1035–1047, 2005.

[21] C. Tralie and E. Dempsey, "Exact, parallelizable dynamic time warping alignment with linear memory," in *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)*, Montréal, Canada, 2020, pp. 462–469.

[22] M. Müller, Y. Özer, M. Krause, T. Prätzlich, and J. Driedger, "Sync Toolbox: A Python package for efficient, robust, and accurate music synchronization,"

Journal of Open Source Software (JOSS), vol. 6, no. 64, pp. 3434:1–4, 2021.

[23] T. Prätzlich and M. Müller, "Triple-based analysis of music alignments without the need of ground-truth annotations," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Shanghai, China, March 2016, pp. 266–270.

[24] S. Böck, F. Korzeniowski, J. Schlüter, F. Krebs, and G. Widmer, "madmom: A new Python audio and music signal processing library," in *Proceedings of the ACM International Conference on Multimedia (ACM-MM)*, Amsterdam, The Netherlands, 2016, pp. 1174–1178.

A DEEP LEARNING METHOD FOR MELODY EXTRACTION FROM A POLYPHONIC SYMBOLIC MUSIC REPRESENTATION

Katerina Kosta **Wei Tsung Lu** **Gabriele Medeot** **Pierre Chanquion**
ByteDance
{name.surname}@bytedance.com

ABSTRACT

The task of identifying melodic lines in polyphonic music is a known active research topic in the symbolic and audio domain. Its importance has attracted the interest of researchers focusing on Music Information Retrieval and musicological applications and achieving high results is a common goal in industrial applications. Distinguishing the melody from the rest of the material in a written score can be a challenging task, however improvements have been reported in recent years using deep learning methods. In this paper, we present a lightweight deep bidirectional LSTM model for identifying the most salient melodic line of a music piece using handcrafted features without requiring the input score to be separated into multiple parts. We evaluate our model to measure the effectiveness of several data augmentation techniques and to compare performance to other state-of-the-art models. We also identify the features' importance and evaluate their incremental contribution on the model performance using evaluation metrics. Results on the POP909 dataset show that our model approximates or outperforms current state of the art models trained on the same dataset, based on different implemented metrics and observations.

1. INTRODUCTION

The concept of the note-to-note organization of music naturally evolves to perceiving larger structures such as phrases and melodic contours. Identifying what a listener perceives to be the melody in a piece of music and, more generally, examining the musicological concept of a melodic line has emerged in recent years as an important topic in the Music Information Retrieval community [1].

A melodic line incorporates musical properties which are rich in contextual information such as structure and rhythm and it embeds expressive characteristics from the perspective of the composer when constructing a piece as well as the perspective of the performer interpreting it. From a musicological point of view, being able to extract a melodic line with high accuracy can reveal new research directions, from analysing a composition style to classifying a performer's tendencies. In industrial or other research applications, high accuracy in melody extraction can improve the results of music search or recommendation algorithms as well as music generation systems.

In both audio and symbolic domains, the task of retrieving the notes of the melody from a polyphonic piece is not trivial. For example, during the act of listening, one may find it hard to distinguish note intervals from one another. It may happen that an interval judged as a major third when heard by itself, is judged as fourth in a separate music context instead [2] (p. 217). During the act of reading a music score or rendering it using a single type of timbre, the note intervals within separate voices might be perceived as interchangeable. The aforementioned challenges have inspired the research on voice extraction, i.e. partitioning the polyphonic piece into a set of monophonic melodies (more details in [3] and [4]).

In this paper, we are focusing on the task of identifying the main melodic line in a symbolic score. Past work on this specific task can be divided into two broad categories. The first includes models predicting a melody part from a multiple-part score input, meaning that a series of notes have been separated to individual parts and the task is to predict which part, either globally or in segments, contains the main melodic line. Most common features used for this purpose are related to notes pitch, intensity, duration, as well as the total number of notes among a part. This type of melody identification is out of scope for this paper, however we refer to [5–7] for details.

Our approach belongs to the second category that is to identify the notes that constitute the main melody in the score without any prior knowledge of separated score parts. Hence, we process an unsegregated set of notes, as we believe that this approach give more flexibility on how it can be used in various applications.

Related work. There is a limited number of previous work on this matter. One of the first attempts is the skyline algorithm [8] which keeps the highest note starting at any time, from all simultaneous note events. The other existing algorithms can be separated in the way the music is represented as input. We can pinpoint two main strands. One handles the music as a piano-roll visualisation, which is a semantic segmentation where musical scores are treated as two-dimensional images. Melody extraction systems presented in [9] and [10] use this type of representation, with the aim to classify the notes represented as pixels. The

© K. Kosta, W. T. Lu, G. Medeot, and P. Chanquion. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** K. Kosta, W. T. Lu, G. Medeot, and P. Chanquion, "A deep learning method for melody extraction from a polyphonic symbolic music representation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

piano-roll representation is mainly used in deep learning methods using convolutional neural networks. The other strand, which our approach belongs to, treats the music sequentially, representing a series of tokens or low and mid-level features that best describe the input characteristics. Recurrent neural network based models employ the note-sequence representation. In the case of the system in [11], the input representation constitutes note-to-note affinity values coming directly from their contextual notes, constructing a weighted undirected graph, having as edge weights the corresponding affinity values. In order to obtain a single melody outcome, they employ spectral clustering to obtain one cluster over the learned graph.

The use of features is presented in [12], where each note is characterised by a set of note properties such as note dissonance, how close a note is to a note ranked with a high probability of being a melody note, or properties from the score metadata such as the musical instrument that a particular note has been assigned to. In our approach, we adapt a set of features which contain note information such as pitch and duration, as well as properties of a note in its polyphonic context.

A recent approach MIDIBERT [13] adopts the mask language model training strategy which is largely used in the field of natural language processing. The model accepts an input representation of sequence of tokens where each token represents a musical event. By training the model to reconstruct the masked input sequence, the transformer model can learn a latent representation of symbolic music. The experiment results show that with such pretraining, the model can be easily fine-tuned for other down stream tasks such as melody extraction.

Previous work reports high results in melody extraction accuracy, especially on recent systems that use deep learning methods. However, the resources used are either not publicly available or their use is copyright protected. Therefore, it is hard to compare and assess results due to unpublished datasets, different metrics reported and different test sets used. Datasets that clearly identify the primary melodic part from accompaniment are rare. For this publication, we have used the open-source POP909 dataset [14], where the salient melody notes have been distinguished. Pop music has been also used in [6, 9]. Folk music has been used in [10, 11] and classical in [9, 10]. The system in [13] adapts POP909 for the task of melody extraction.

Proposed method. The melody extraction model that we introduce in this paper is a supervised bidirectional Long-Short Term-Memory (biLSTM) model called LStoM, standing for Large Score to Melody. It receives a set of computed features as input derived from a MIDI score and classifies which notes constitute the main melodic line. We adapted the note events representation as a set of features in the form inspired by [4].

For our model input, we compute the following features for each note in our dataset, as described in Table 1. The first two features, `pitch` and `dur`, contain the basic pitch and duration information of a note, respectively, where pitch is expressed in a MIDI note number and the duration in a crotchet level. For `pitch_dist_below` and `pitch_dist_above`, we compute the distance to the pitch of the next (neighbouring) higher or lower note sounded simultaneously with the note in use. `pos_in_bar` records the score beat where the note onset is located within the score bar. `pitch_in_scale` records whether the note pitch is in the diatonic scale of the key signature. The features `pitch_dist_below` and `pitch_dist_above` are inspired by the work in [4]. The feature `pitch_in_scale` is inspired by the work in [12].

Feature name	Description
`pitch`	note pitch (MIDI number)
`dur`	note duration (crotchets)
`pitch_dist_below`	absolute pitch distance (semitones)
`pitch_dist_above`	absolute pitch distance (semitones)
`pos_in_bar`	note onset position in bar (crotchets)
`pitch_in_scale`	note pitch in key scale (boolean)

Table 1. The features that have been selected and computed, along with their description.

More on the data that we have processed and the model we have built in Sections 2 and 3 respectively. Section 4 includes a set of experiments and observations to assess the model's capabilities. Finally, in Section 5 we summarise our findings and suggest potential future directions.

2. DATA DETAILS

For our experiments, we have used the MIDI files from POP909 dataset [14]. This dataset includes piano arrangements of Chinese pop songs created by musicians playing on a MIDI keyboard, and the melody part has been identified by manually transcribing the lead vocal melody. The scores include the remaining parts "bridge" and "accompaniment" which we merge and consider as a single accompaniment part throughout our experiments.

The scores are rich in expressive characteristics, such as the note timing and duration. In our processing, we have used the dataset metadata information regarding the beats and the key. For our purpose, we have pre-processed the MIDI files, to rectify the following dysfunctions. Many scores include a time signature of '1/4', so we have updated the time signature by considering the meter information from the audio beat metadata. Time signatures of "2/4" and "2/2" have been mapped to "4/4". Also, many scores include a misalignment in the downbeat level, so we have fixed the start time of the piece to reduce this issue.

As mentioned before, the scores are performative and this means that they accommodate fine details or imprecision in timing. In order to setup our model, we have created a time-grid of the notes' onset and duration that is flexible enough to keep performative characteristics and strict enough to not explode the dimensions of possible values which would be hard to be trainable later on. The time-grid is setup to align onset times in triplets resolution. The note durations are adjusted to the minimum between the distance of current note onset and next note onset, and the current note duration in semiquaver resolution.

The key information is extracted from the dataset metadata. When a song has more than one key signatures reported, we consider a single key signature extracted from the music21 key detection algorithm [15]. At the inference level of our model, we apply the latter method on the test score input. We do not apply any type of post-processing on the extracted melody. Also, we automatically merge all notes into a single MIDI part per piece, keeping the information of a note being a melody one as ground truth.

To facilitate a fair comparison of the models in our experiments, we use the same dataset subset and train-validation-test set split as in [13], which results to 865 songs. The steps above give us the amount of 1,408,056 notes, from which 19.8% are melody notes.

3. MODEL DETAILS

In this section, we present the details of our model architecture (3.1), the details about our training setup (3.2), as well as the metrics we have established (3.3). We have released the code of the model as open-source [1] .

3.1 Architecture

We implement a deep bidirectional LSTM architecture (biLSTM) [16] for our model, which is a type of Recurrent Neural Network (RNN). In our implementation, the biLSTM model has been setup after implementing a grid-search on the model hyper-parameters, using hyperopt [17]. After this process, the model results in having 6 layers, with hidden size of 140, followed by a forward layer. The Adam optimizer is used with a start learning rate of 0.001. We identified the set of hyper-parameters that produce the highest melody F-measure score (metric described in the following Section 3.3).

One characteristic of our dataset is the imbalance of the data labels (i.e. the two classification classes), meaning that non-melody notes outnumber the melody ones. To overcome this issue in our classification task, we adopt the focal loss [18] as the loss function for the model, which is defined as:

$$FL(p_t) = -a_t(1-p_t)^{\gamma} log(p_t), \tag{1}$$

where p_t denotes the model's estimated probability for an input to be classified to class t and $a_t \in [0, 1]$ is a weighting factor for the imbalanced classes which balances the importance of positive and negative examples. The term $(1-p_t)^{\gamma}$ acts as a modulating factor with γ controlling the rate at which over-weighted examples are down-weighted. We set $a_t = 0.25$ and $\gamma = 2$.

3.2 Training setup

The train-validation-test sets that we used from [13] contain a data percentage of 80-10-10%, respectively. The features have been scaled, given the data points in the train and the validation sets.

[1] https://github.com/bytedance/midi_melody_extraction

3.3 Metrics

The metrics that we have computed for this task are the following: `accuracy` indicating the overall accuracy of the predicted notes in percentage, as well as the `mel_P`, `mel_R` and `mel_F` for melody notes precision, recall and F measure values, respectively:

$$\texttt{mel\_P} = \frac{|\text{Correctly predicted melody notes}|}{|\text{Notes predicted as melody}|} \tag{2}$$

$$\texttt{mel\_R} = \frac{|\text{Correctly predicted melody notes}|}{|\text{Melody notes}|} \tag{3}$$

$$\texttt{mel\_F} = 2 * (\frac{\texttt{mel\_P} * \texttt{mel\_R}}{\texttt{mel\_P} + \texttt{mel\_R}}) \tag{4}$$

Also, we include the metric `Voice False Alarm` (VFA) from mir_eval [19], which is defined as the number of notes predicted as melody notes, although they are not, divided by the number of non-melody notes.

4. EXPERIMENTS

To evaluate the performance of our system, we have prepared a list of separate model setups, given the same train, validation and test sets. Also, we were interested in whether two types of data augmentation techniques would improve the accuracy of LStoM when applied in isolation as well as together. The first type is to shift the score key by altering the note pitches, where each piece has been transposed by n semitones, for n in $\{-6, -5, ..., 4, 5\}$ (the model is tagged as "LStoM PSaugm" standing for pitch shifting augmentation). The second type is to shift the melody notes one octave lower than the original one (the model is tagged as "LStoM MOaugm" standing for melody octave augmentation). Both techniques are common in the literature for the task of symbolic melody extraction [10].

4.1 Comparison with the state of the art

We investigate how other models that report state of the art results in various datasets perform in the case of the specific train, validation and test sets. To this end, we were able to train two models. One is following the default settings of [11] tailored to the task of the melody identification, using the augmentation techniques that are provided from this work and the pitch proximity method as a segment merging mode among the created note clusters (the model is tagged as "Hsiao-Su"). The other is following the default settings of [10], again using the augmentation techniques that are provided from this work (the model is tagged as "Lu-Su").

The skyline algorithm [8], which is essentially picking the highest pitch at a given onset time, has been considered as our baseline. Also, we consider a skyline variation where we keep the duration of the note with the highest pitch, ignoring the lower-pitch notes that start after its onset and before its offset. Both original and variation skyline output are illustrated in Figure 1, using the input example as reported in [8].

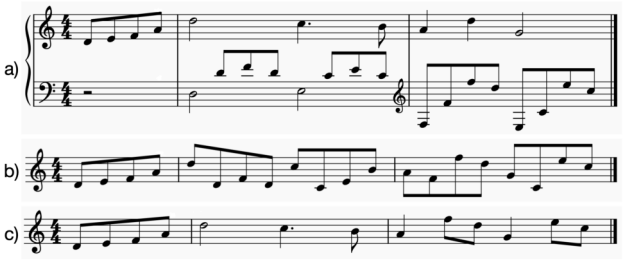


Figure 1. Illustration of our baselines. a) Input example, b) Output of skyline algorithm, c) Output of skyline variation algorithm.

Model	acc (%)	mel_F	mel_P	mel_R	VFA
LStoM*	92.3	0.816	0.778	0.872	0.065
LStoM* PSaugm	91.6	0.788	0.791	0.799	0.054
LStoM* MOaugm	92.3	0.800	0.815	0.811	0.066
LStoM* MO+PSaugm	91.9	0.798	0.797	0.811	0.054
MIDIBERT* [13]	**97.1**	**0.930**	**0.912**	**0.952**	**0.024**
LStoM	90.7	**0.774**	0.751	0.814	0.070
MIDIBERT [13]	**91.9**	0.772	**0.841**	0.727	**0.035**
skyline	63.1	0.499	0.353	**0.881**	0.435
skyline-variation	78.7	0.569	0.485	0.697	0.192
Lu-Su [10]	66.6	0.614	0.600	0.638	0.299
Hsiao-Su [11]	82.5	0.650	0.544	0.821	0.176

Table 2. Basic comparison results among testing models. An '*' is added to indicate that the MIDI files in the test set have been pre-processed.

Both LStoM and MIDIBERT [2] include pre-processing steps for note quantisation and alignment in downbeat level when preparing the MIDI files for training. At the testing stage, we considered as a fair comparison to first replicate the results of [13] where the MIDI files of the test set have been pre-processed, therefore our pre-processing steps were applied to the test set as well for LStoM and its augmentations (metrics reported at the top part of Table 2). The augmentations do not appear to improve the overall performance of LStoM. In the bottom part of Table 2, we report the metrics of the remaining comparison, where the notes of the test set have not been quantised nor aligned to the downbeat. Interestingly, MIDIBERT outperforms in the first scenario, while the results are mixed in the second one.

Lastly, LStoM is significantly lighter than MIDIBERT; the number of parameters are 875,462 compared to 111,298,052, respectively.

In the next two subsections, we are exploring LStoM in terms of how important the selected features for the training process are (Section 4.2) and we are discussing some observations from the models' outcome, respectively (Section 4.3).

[2] Function `align_midi_beats` in `https://github.com/wazenmai/MIDI-BERT/blob/CP/data_creation/preprocess_pop909/preprocess.py`, accessed 31 August 2022

4.2 Importance of features

We were interested to know how relevant the selected features are for our task and whether any of them are more important in a sense that they contain an amount of information that is crucial for such systems. Algorithmically it is feasible to examine and improve the interpretability of a predictive model to a degree, and to identify a type of ranking of importance for the input features. However, most recent feature ranking algorithms assume feature independence (for more examples and details we refer to [20]), an assumption which is not safe in our case.

Ranking	Feature	Ranking	Feature
1	`pitch`	4	`dur`
2	`pitch_distance_above`	5	`pos_in_bar`
3	`pitch_distance_below`	6	`pitch_in_scale`

Table 3. The ranking of feature importance obtained by RELIEF algorithm.

One algorithm that is not based on this assumption is RELIEF [21] which elaborates on a simple comparison idea whereby feature value differences between nearest neighbour instance pairs are identified. If a feature value difference is observed in a neighbouring instance pair with the same class, the feature ranking score decreases. However, the score increases when a feature value difference is observed in a neighbouring instance pair with different class values. We applied RELIEF to our dataset and the resulting feature ranking is reported in Table 3.

Not by surprise, the pitch has been ranked as the most important feature, followed by the one that computes how much is the distance from the pitch above. The third feature in the ranking list is the similar one, computing the pitch distance from the pitch below. The note duration comes to the fourth place, followed by the metrical-related feature of computing the position of the note within the score bar, and finally whether the note pitch is in scale.

In order to evaluate the incremental contribution of each of the features studied, we have considered feature subsets by incrementally adding features to the training set one at a time, starting with the most important one, i.e. the note pitch, and continuing to add features according to their rank order. The results obtained are reported in Table 4.

Model	acc	mel_F	mel_P	mel_R	VFA
LStoM1	86.5%	0.672	0.652	0.708	0.094
LStoM1-2	88.4%	0.718	0.697	0.752	0.082
LStoM1-3	89.4%	0.738	0.723	0.768	0.073
LStoM1-4	90.3%	0.742	0.780	0.724	0.052
LStoM1-5	92.0%	0.797	0.803	0.806	0.051
LStoM1-6	92.3%	0.816	0.778	0.872	0.065

Table 4. Basic comparison results among the variations of LStoM model, where LStoM*x-x* indicates the range of ranked features used at the training process.

It is worth noticing that the highest ranked feature contains on its own substantial predictive power: the melody F-measure score for the model induced with only pitch information alone is 0.672, which is slighter higher than the

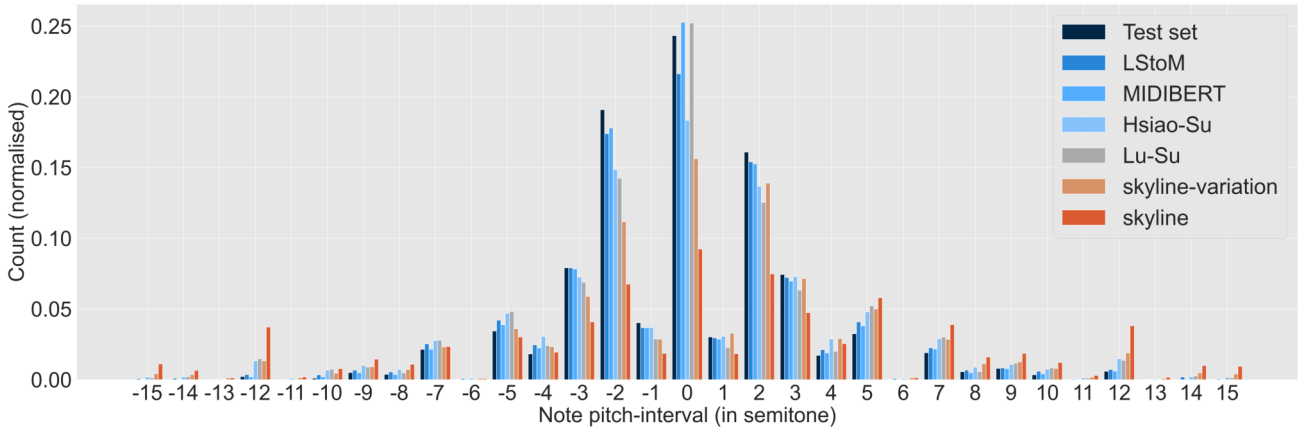


Figure 2. Pitch interval distribution among melodies predicted by LStoM, MIDIBERT, “Hsiao-Su", “Lu-Su", skyline and skyline-variation compared to the melodies from the test set.

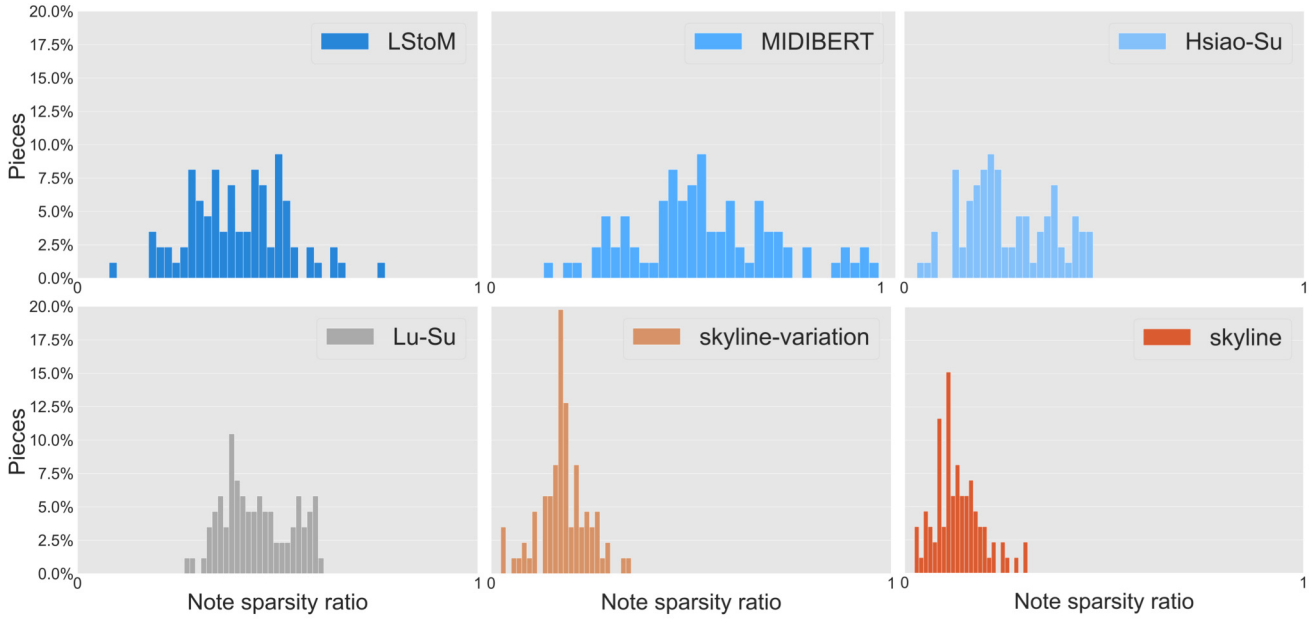


Figure 3a. Sparsity ratio among melodies predicted by LStoM, MIDIBERT, “Hsiao-Su", “Lu-Su", skyline and skyline-variation.

one obtained by our trainings on “Hsiao-Su" and “Lu-Su" systems. Interestingly, the precision is reduced by a very small margin and the voice alarm error value is slightly increased, when adding the feature `pitch_in_scale`, however the accuracy is increased at the same time.

4.3 Observations

Having a closer look to the predicted melodies by LStoM, MIDIBERT, “Hsiao-Su", “Lu-Su" and the baselines, we can highlight two separate statistics. One is the distribution of the note intervals among the consecutive melody note pairs (Figure 2 – the x axis has been limited to the range of around two and a half octaves for paper fitting purposes), where overall the melodies predicted by the models tended to reflect characteristics from the original melody contour. Another statistic is the percentage of predicted melodies with various degrees of sparsity in them. By sparsity, we mean the ratio of the total duration of the silent parts over the total duration of the notes in a score. Fig-

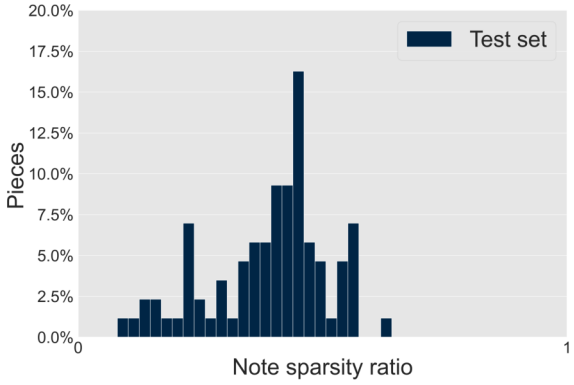


Figure 3b. Sparsity ratio among melodies in the test set.

ure 3a shows the sparsity distribution among the predicted melodies, while Figure 3b highlights the ground truth from the test set. A small amount of melodies from MIDIBERT tends to be more sparse than the scores from the test set.

The most challenging melody notes to classify are those that are not the highest note; i.e., those that the skyline method would certainly miss. We have used the same test set where around 28% of the melody notes are such challenging cases; of these, the LStoM system classifies correctly around 82%. What pattern in the feature data did the model use to achieve this high performance?

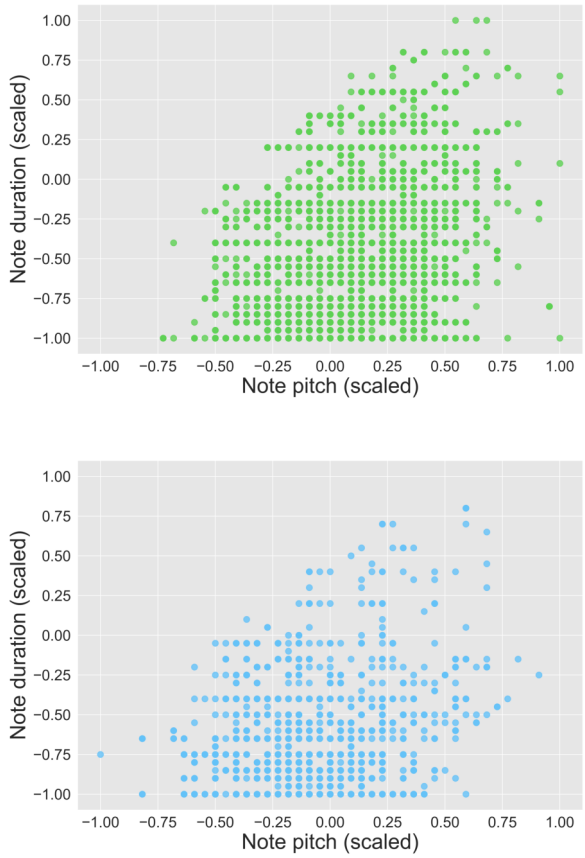


Figure 4. Scaled representation for the pitch-duration feature pair, for the melody notes that are not the highest note and have been either predicted correctly–top– or not (missed)–bottom.

It is not easy to examine the reason why sometimes such notes are not predicted as melody notes, but to investigate, we scaled the feature values of these true melody notes, and compared distributions of features between the correctly labelled (i.e., retrieved) and incorrectly labelled (i.e., missed) notes. One distribution that stood out is shown in Figure 4, a scatter plot of the pitch and duration of the melody notes. One may observe that although the distributions are similar, there are many more correctly labelled notes with high pitch and long duration. Such findings indicate that disentangling such points is not trivial, if at all feasible with our current feature set alone.

5. CONCLUSIONS AND PERSPECTIVES

In this paper we have presented a novel deep learning model of identifying the melody line from a polyphonic symbolic music. The key characteristics of the model is the input data representation as a set of features from the relevant task of multiple voice separation as well as the bidirectional nature of the architecture which provides information of future observations during the prediction process. The features have been examined regarding their ability to contain the information that the model needs to more accurately predict a melody note. Also, two data augmentation techniques are examined in isolation.

Results indicate that the proposed lightweight model which is trained and tested on a set of pop songs is capable of performing at the standard of existing benchmarks. Observations on the results reveal the ability of the model to reflect the concepts of score sparsity or the melodic pitch intervals from the test set. LStoM also performs well in situations where the melody note to be identified is not in the highest pitch of the score. One direction to explore is expanding the set of input features. Note velocity information or metrical representation extracted from the dataset could accommodate additional features for our model. Post-processing techniques have not been explored thoroughly, however they could help improving the results.

Setting research experiments for the task of melody extraction in the symbolic domain is challenging, in terms of data gathering and manipulating separate model architectures to fit the needs of a comparison task. Accumulating the available datasets and the existing systems to a common space for easy approach, such as in a dedicated MIREX page is a promising step forwards. To this end, examining how our model performs using different datasets in the training or inference process, is a step towards a more comprehensive review.

A direction that has been little explored is combining information from the symbolic domain to the prediction of melodies in the audio domain. Most of the existing literature for audio melody transcription relies only on information from the audio signal in isolation from other musical context explicitly. However, in [22], given a polyphonic music audio signal, the model is able to convert it to a piano arrangement; as part of this style transferring process, MIDI scores have been used as ground truth. Also, in [23], symbolic data representation has been used to pre-train a model which then operates to the audio domain. Tasks such as de-noising could be revisited following this perspective.

6. ACKNOWLEDGEMENTS

We thank Jordan B. L. Smith and Janne Spijkervet for their extensive paper review and support before submission.

7. REFERENCES

[1] J. Salamon, E. Gómez, D. P. Ellis, and G. Richard, "Melody extraction from polyphonic music signals: Approaches, applications, and challenges," *IEEE Signal Processing Magazine*, vol. 31, no. 2, pp. 118–134, 2014.

[2] N. Cook, *Music, Imagination, and Culture*, ser. ACLS Humanities E-Book. Clarendon Press, 1990.

[3] E. Cambouropoulos, "Voice and stream: Perceptual and computational modeling of voice separation," *Music Perception*, vol. 26, no. 1, pp. 75–94, 09 2008.

[4] R. de Valk and T. Weyde, "Deep neural networks with voice entry estimation heuristics for voice separation in symbolic music representations," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[5] R. Martín, R. A. Mollineda, and V. García, "Melodic track identification in midi files considering the imbalanced context," in *Pattern Recognition and Image Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 489–496.

[6] Z. Jiang and R. B. Dannenberg, "Melody identification in standard midi files," in *Proceedings of the 16th Sound and Music Computing Conference (SMC)*, 2019, pp. 65–71.

[7] S. Li, S. Jang, and Y. Sung, "Melody extraction and encoding method for generating healthcare music automatically," *Electronics*, vol. 8, no. 11, 2019.

[8] A. L. Uitdenbogerd and J. Zobel, "Melodic matching techniques for large music databases," in *Proceedings of the 7th ACM international conference on Multimedia (Part 1)*, 1999, pp. 57–66.

[9] F. Simonetta, C. E. Cancino-Chacón, S. Ntalampiras, and G. Widmer, "A convolutional approach to melody line identification in symbolic scores," *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[10] W.-T. Lu and L. Su, "Deep learning models for melody perception: An investigation on symbolic music data," in *2018 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2018, pp. 1620–1625.

[11] Y.-W. Hsiao and L. Su, "Learning note-to-note affinity for voice segregation and melody line identification of symbolic music data," *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, pp. 285–292, 2021.

[12] H. Zhao and Z. Qin, "Tunerank model for main melody extraction from multi-part musical scores," in *Proceedings of the 6th International Conference on Intelligent Human-Machine Systems and Cybernetics - Volume 02*, ser. IHMSC '14. USA: IEEE Computer Society, 2014, p. 176–180.

[13] Y.-H. Chou, I.-C. Chen, C.-J. Chang, J. Ching, and Y.-H. Yang, "MidiBERT-Piano: Large-scale pre-training for symbolic music understanding," *arXiv preprint arXiv:2107.05223*, 2021.

[14] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *Proceeding of the 21st International Society for Music Information Retrieval (ISMIR)*, 2020.

[15] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data." in *International Society for Music Information Retrieval*, 2010, pp. 637–642.

[16] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005, iJCNN 2005.

[17] J. Bergstra, D. Yamins, and D. D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *30th International Conference on Machine Learning, ICML 2013*, vol. 28, 2013, pp. 115–123.

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[19] C. Raffel, B. Mcfee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. W. Ellis, C. C. Raffel, B. Mcfee, and E. J. Humphrey, "mir_eval: a transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[20] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc., 2017.

[21] R. J. Urbanowicz, R. S. Olson, P. Schmitt, M. Meeker, and J. H. Moore, "Benchmarking relief-based feature selection methods," *CoRR*, vol. abs/1711.08477, 2017. [Online]. Available: http://arxiv.org/abs/1711.08477

[22] Z. Wang, D. Xu, G. Xia, and Y. Shan, "Audio-to-symbolic arrangement via cross-modal music representation learning," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 181–185.

[23] W. T. Lu, L. Su *et al.*, "Vocal melody extraction with semantic segmentation and audio-symbolic domain transfer learning," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018, pp. 521–528.

A REPRODUCIBILITY STUDY ON USER-CENTRIC MIR RESEARCH AND WHY IT IS IMPORTANT

Peter Knees[1,3] **Bruce Ferwerda**[2] **Andreas Rauber**[1]
Sebastian Strumbelj[1] **Annabel Resch**[1] **Laurenz Tomandl**[1] **Valentin Bauer**[1]
Fung Yee Tang[1] **Josip Bobinac**[1] **Amila Ceranic**[1] **Riad Dizdar**[1]
[1] Faculty of Informatics, TU Wien, Austria
[2] Department of Computer Science and Informatics, Jönköping University, Sweden
[3] School of Music, Georgia Institute of Technology, USA
peter.knees@tuwien.ac.at

ABSTRACT

Reproducibility of results is a central pillar of scientific work. In music information retrieval research, this is widely acknowledged and practiced by the community by re-implementing algorithms and re-validating machine learning experiments. In this paper, we argue for an increased need to also reproduce the results and findings of user studies, including qualitative work, especially since these often lay the foundations and serve as justification for choices taken in algorithmic design and optimization criteria. As an example, we attempt to reproduce the study by Kim et al. [1] presented in the RecSys (2020) paper "Do Channels Matter? Illuminating Interpersonal Influence on Music Recommendations". By repeating this study on how interpersonal relationships can affect a user's assessment of music recommendations on a new sample of $n = 142$ participants, we can largely confirm and support the validity of the original results. At the same time, we extend the analysis and also observe differences with regards to adoption rates between different channels as well as different factors that influences the adoption rate. From this specific reproducibility study, we conclude that potential cultural differences should be accounted for more explicitly in future studies and that systems development should be more explicitly connected to its intended target audience.

1. INTRODUCTION AND CONTEXT

Ensuring the demonstrable reproducibility of experiments is an essential part of the scientific method to acquire knowledge. Reproducibility in music information retrieval research traditionally has a strong focus on the repeatability of experiments that demonstrate the performance of a system. To this end, the formalization and complete documentation of MIR workflows and processes is a top priority (e.g., [2]). This includes proper documentation and provision of access to the data involved, to avoid contributing to the so-called reproducibility crisis found throughout virtually all scientific disciplines. In the MIR community the aspect of data access has been a topic of discussion since the beginning, as dataset sharing is a particularly challenging and sensitive matter, foremost due to copyright issues, cf. [3,4]. Also for evaluation, emphasis is given to applying transparent, open, and sustainable methods, to objectively quantify the performance of systems and consistently compare systems [5]. In short, many efforts of the community are devoted to sharing resources, re-implementing algorithms, and re-validating machine learning experiments to ensure reliable and valid scientific results.

In order to avoid potentially contributing to the reproducibility crisis on another front, we argue that there is urgent need in MIR research to also reproduce the results and findings of user-centric studies, including qualitative work. The system-centric view in MIR has been repeatedly subject to criticism and raised calls for more user-centric approaches (e.g., [6, 7]) as the objectives of the systems developed are ultimately rooted in users' needs. In practice this has led to user-centric work taking the essential roles of requirement engineering (e.g. [8–10]) and/or of a system evaluation vehicle, either by generating ground truth, rating the outcomes of systems, or reflecting on their use (e.g., [11–13]). As their outcomes have (or are supposed to have) a fundamental impact and serve as justification for choices taken in algorithmic design and deployed optimization criteria [14, 15], their validity is by no means of lesser importance than that of system-based experiments. Clearly, reproducibility of experiments involving human subjects is hard, as witnessed in the social sciences [16], and has led to the development of the current evaluation strategies as a workaround [17]. Nonetheless, when it comes to the justification of systems objectives, further attention should be paid to the scientific validity of findings by reproducing and corroborating the findings of studies.

As an example, in this paper, we attempt to reproduce the study by Kim et al. [1], presented in the ACM Rec-

© P. Knees, B. Ferwerda, A. Rauber, S. Strumbelj, A. Resch, L. Tomandl, V. Bauer, F. Y. Tang, J. Bobinac, A. Ceranic, and R. Dizdar. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** P. Knees, B. Ferwerda, A. Rauber, S. Strumbelj, A. Resch, L. Tomandl, V. Bauer, F. Y. Tang, J. Bobinac, A. Ceranic, and R. Dizdar, "A Reproducibility Study on User-centric MIR Research and Why it is Important", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

Sys 2020 paper "Do Channels Matter? Illuminating Interpersonal Influence on Music Recommendations". We have chosen this particular user study for several reasons. First, the study subject is highly relevant for the development and understanding of music recommender systems, therefore clarifying the reproducibility of the experiments is important for future research. Second, the study is well documented with a clearly laid out methodology. Third, as the original data was collected via a web-based survey, to obtain a new set of responses of comparable quality, we can follow the same strategy, with the main difference that recruited participants have a different cultural, i.e. predominantly European, background.

In addition to repeating the original study to investigate the differences of music recommendations from music recommender systems (non-interpersonal) and friends and acquaintances (interpersonal), we carry out further analyses to gain insights regarding gender differences. We further make use of the PRIMAD framework for reproducibility of research in e-Science for positioning our work [18]. In the terminology of the PRIMAD framework, in our reproducibility study, we foremost prime the factor of data and consequently investigate the generalization capabilities of the analysis wrt. another sample of users. Through the additional analyses, we also extend the methodology used.

The remainder of this paper is structured as follows. Section 2 briefly motivates and summarizes the original study by Kim et al. [1]. In Section 3, we describe the methods used to reproduce the study. Section 4 reports on the results and findings of the reproduced study, which are further discussed in relation to the original study in Section 5. In Section 6, we use the PRIMAD model as a tool to systematically categorize the dimensions of reproducibility in this work and suggest its use for future reproducibility studies (not only) of user-centric studies in the MIR research community. Finally, in Section 7 we draw conclusions from this reproducibility study and point out aspects to be considered for future work.

2. ORIGINAL WORK TO BE REPRODUCED

We aim at reproducing the findings of Kim et al. [1], who have conducted a web survey to investigate the perceptions, evaluations, and differences of music recommendations received through two different channels: *interpersonal*, i.e., music recommendations received from friends or acquaintances, and *non-interpersonal*, i.e. music recommendations received from recommendation systems and through updates on artists followed. The rationale behind that study is to gain a deeper understanding of the characteristics of interpersonal music recommendations and how the interpersonal relationship affects the evaluation of the recommendation.

To this end they recruited 175 Korean-speaking participants (56% female; all above age 18, with 91% of participants between 18 and 32). [1] Furthermore, they collected information that 85% of the participants were active users of a music subscription service for more than one year and that 62% of participants were actively sharing music with others more than once per week (see Table 1).

Study	**Kim et al. [1]**	**Current study**
Number of participants (n)	175	142
Female participants	56%	45%
Age between 18 and 32	91%	86%
Use music subscription service	85%	69%
Sharing music with others	62%	16%

Table 1. Demographics and characteristics of survey participants in the original study and the reproducibility study. Values for the current study refer to percentages in recorded responses.

The study shows an interesting trend, namely that music recommendations obtained from a system are typically considered to be more relevant for a user's own taste (which is the primary objective of recommender systems), more convenient, and more frequently used and adopted than interpersonal recommendations. On the other hand, aspects of diversity, novelty, and serendipity were evaluated higher when recommendations were obtained through interpersonal channels. More detailed results of the study are also included in this paper for comparison in Section 4.

From these results, the authors conclude that indeed interpersonal and non-interpersonal channels have different characteristics that should be accounted for when designing music recommendation platforms, particular to take advantage of their individual strengths (e.g., relevance vs. diversity and novelty). They further highlight a limitation of their study and point to a possible next step: "Since the study was conducted in Korea, Korean cultural background was reflected in the results. Future studies with participants from other cultures (e.g., with different norms, technologies available, etc.) will enable us to explore how cultural differences can influence users' perceptions and behaviors towards music recommendations from different channels." In this paper, we aim at gaining such insights by conducting the same study again with a fresh set of participants, with a different background (i.e., predominantly European), as described next.

3. METHODS

Following the method of Kim et al. [1], we used a web-based survey to investigate interpersonal and non-interpersonal music channels in the context of music playlist creation and consumption. A similar survey was designed with a section regarding interpersonal channels (i.e., from friends or acquaintances) and a section regarding non-interpersonal channels (i.e., from recommendation systems) and sections with the respective questions were presented in a random order to the participants (see Table 2 for the survey questions). The survey was conducted in English with all participants. The questions for each

[1] Despite highlighting that all participants spoke Korean fluently, it is not explicitly stated whether the survey was conducted in Korean or English.

section consisted of the same questions, the only difference is whether they are related to interpersonal or non-interpersonal channels. Upon presenting a section to the participants, they were first asked if they had ever received recommendations through the respective channel. If they had not, they would skip the questions of that section and would move on to the next section.

The survey questions were divided into two parts: 1) evaluation criteria, and 2) usage behavior. Participants were asked to respond to the four evaluation criteria (i.e., relevance, diversity, novelty, and serendipity) using a 5-point Likert scale. Participants were then asked to respond to the usage behavioral questions of frequency and convenience in a similar 5-point Likert scale fashion. Additionally, participants were asked to estimate the adoption rate of the recommended music to their playlist (i.e., adding the music to their playlist) on a percentage level. As with the original survey, no actual recommendations were presented, as the survey solely focused on past experiences of received recommendations.

4. RESULTS

We recruited a total of 142 participants for this study in the context of three Master's course projects at TU Wien via open calls for participation advertised to other students in the course, as well as other peers and acquaintances in their extended networks, cf. [19–21]

Out of the disclosed and collected information of the total 142 participants, 45% identify as female, 55% as male, with 86% aged between 18 and 32. When it comes to the use of music subscription services, 69% of the participants indicated to use music subscription services for more than a year, and 16% shared music with others more than once a week (a side-by-side comparison with the original study can be found in Table 1). To investigate the reproducibility of Kim et al. [1], similar statistical tests were used. In terms of cultural background, 89% of reporting participants in our study state their origin to be in a European country. In contrast to the original study, where 100% of participants reported their background as Asian (specifically, Korean), in our study, only 7% report their origin to be in an Asian country (see Table 3).

4.1 Evaluation of music recommendation channels

Initial mean comparison show that diversity, novelty, and serendipity on average score higher for interpersonal music channels, while relevance, convenience, frequency, and adoption rate are higher for the non-interpersonal channels (see Table 4). When conducting significance testing on the four evaluation criteria, paired t-tests confirm that diversity ($p < .001$) and novelty ($p = .002$) are rated higher for interpersonal channels and relevance ($p < .001$) is rated higher for non-interpersonal channels (see Figure 1A).

An additional principal component analysis (PCA) was conducted to investigate the relationship between the four evaluation criteria. The correlation matrix show that some of the evaluation criteria appeared to be similar than others. The PCA show two components to be extracted from the data with a total accounted variance of 67%: component 1 accounts for 42% and component 2 accounts for 24%. PCA bi-plot (see Figure 2) show that relevance is mainly explained by component 1, and diversity, novelty, and serendipity by component 2. The bi-plot indicate that diversity, novelty, and serendipity are more closely aligned, while relevance is almost orthogonal to these three evaluation criteria.

4.2 Usage behavior of music recommendation channels

Non-interpersonal channels rated higher for convenience, frequency, and adoption rate (see Table 4). Significance testing by using paired t-tests show that frequency was significantly rated higher ($p < .001$) for non-interpersonal channels than for interpersonal channels (see Figure 1B). When it comes to the convenience of using a channel, non-interpersonal channels were rated higher ($p = .004$) as well (see Figure 1C). However, when it comes to the adoption rate, although the non-interpersonal channel scored slightly higher, there was no significant difference ($p = ns$) between non-interpersonal and interpersonal channels.

Additionally, a linear regression was conducted to investigate the effect of frequency on adoption rate. Frequency has a significant effect on adoption rate for interpersonal channels ($F(1, 110) = 4.416, p = .038$) as well as for non-interpersonal channels ($F(1, 90) = 4.971, p = .028$). Whereas convenience plays an additional role on adoption rate of non-interpersonal channels ($F(1, 90) = 4.312, p = .041$).

Furthermore, a univariate regression was conducted to investigate the four evaluation criteria on the adoption rate. For the interpersonal channels, diversity ($F(1.68) = 1.574, p = .003$), novelty ($F(1.68) = 2.220, p = .002$) show to have a significant effect on adoption rates, while for the non-interpersonal channels only novelty ($F(1.68) = 1.596, p = .06$) appeared to be significant.

4.3 Gender differences

We conducted an additional multivariate test to investigate the role of gender between interpersonal and non-interpersonal channels. Gender differences were found when it comes of frequency of the receiving recommendations through non-interpersonal channels ($F(1, 90) = 4.475, p = .035$), indicating that males receive more frequent recommendations through non-interpersonal channels than females. However, no significant gender differences ($p = ns$) were found within interpersonal channels.

Furthermore, a marginal significant effect was found of gender on novelty within interpersonal channels ($F(1, 68) = 3.138, p = .061$), indicating that males in particular receive more novel recommendations through interpersonal channels such as friends and acquaintances than females. No significant effects ($p = ns$) of gender were found within non-interpersonal channels.

Evaluation criteria	Relevance	Is the music recommend through the channel consistent with the context and theme of the playlist?
	Diversity	Does the music recommended through the channel diversify genre or mood of the playlist?
	Novelty	Is the music recommended through the channel new or novel?
	Serendipity	Is the music recommended through the channel unexpected but good in terms of the context and theme of the playlist?
Usage behavior	Frequency	How often do you receive music recommendations through the channel?
	Convenience	How convenient is the process of listening to the music recommended through the channel?
	Adoption rate	What percentage of the recommended music through the channel is added to the playlist?

Table 2. Survey questions. Same questions were used for both interpersonal and non-interpersonal channels. Adopted from Kim et al. [1].

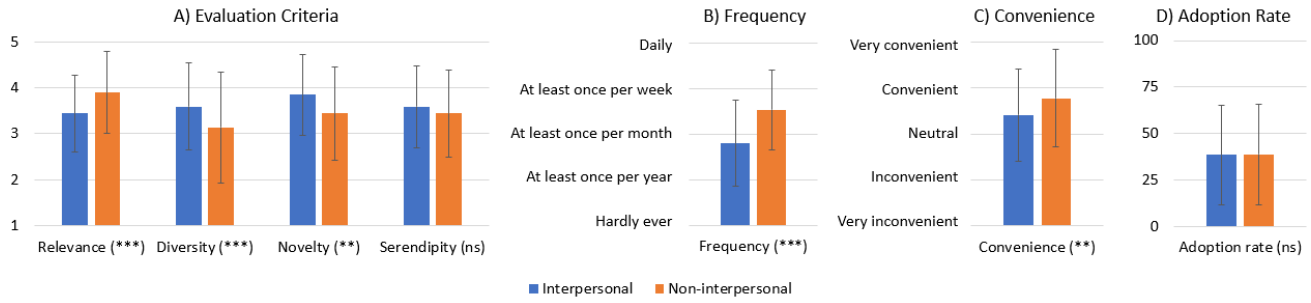


Figure 1. Mean scores with standard deviation error bars ($*\, p < 0.01$, $**\, p < 0.001$, $***\, p < 0.0001$).

Continent	Percentage
Europe	89%
Asia	7%
Africa/Middle East	3%
Oceania	1%

Table 3. Distribution of participants of the current study according to disclosed origin per continent.

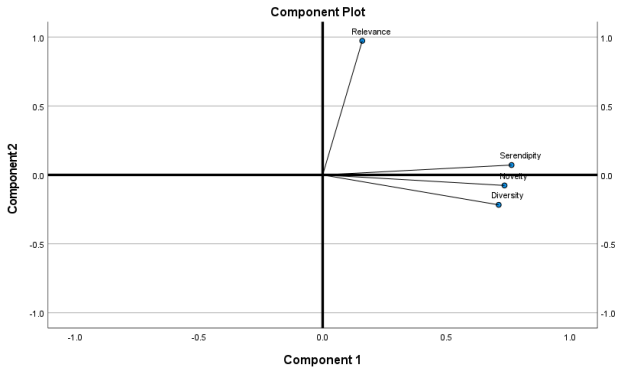


Figure 2. PCA biplot with the two extracted components.

5. DISCUSSION

In this section, we discuss the results obtained with regards to reproducibility, deviations from the original study, and additional analyses carried out to gain further insights.

5.1 Reproducibility

For this reproducibility study we followed similar procedures and methods of the original work by Kim et al. [1]. Following the same procedures we recruited 142 participants to fill in questions on their music consumption behaviors through interpersonal and non-interpersonal music channels.

Looking at the mean values, the overall strength of the mean values suggest that similar trends are found (see Table 4) in the current study as in the original study of Kim et al. [1] between interpersonal channels and non-interpersonal channels. By additionally conducting paired t-tests, we tested whether there are significant differences between the interpersonal and non-interpersonal channels. Also here, we found in general similar results aside of that novelty returned significant, but serendipity and adoption rate became not significant. These results indicate that the sample in this reproducibility study showed less pronounced difference in the areas of serendipity and adoption rate between interpersonal and non-interpersonal channels, but more differences in novelty.

When looking at the relationship between the four evaluation criteria (i.e., relevance, diversity, novelty, and serendipity), we also found similar results as in the original study. The PCA results show the extraction of two components with a total of 67% of the variance accounted for: component 1 accounting for 43% of the variance and component 2 accounting for 24% of the variance. Our results show that the four evaluation criteria account for a

	Kim et al. [1]			Current study		
	Interpersonal ($n = 155$)	Non-Interpersonal ($n = 143$)	p	Interpersonal ($n = 118$)	Non-Interpersonal ($n = 100$)	p
Relevance	$3.47(0.86)$	**4.05**(0.70)	< 0.0001	$3.45(0.88)$	**3.90**(0.88)	< 0.0001
Diversity	**3.92**(0.77)	$3.59(0.90)$	< 0.001	**3.59**(0.95)	$3.14(1.2)$	< 0.001
Novelty	**4.09**(0.71)	$3.94(0.82)$	ns	**3.85**(0.88)	$3.44(1.0)$	< 0.01
Serendipity	**3.65**(0.96)	$3.23(0.98)$	< 0.001	**3.59**(0.85)	$3.44(0.94)$	ns
Convenience	$2.64(0.86)$	**3.05**(0.83)	< 0.0001	$3.41(1.0)$	**3.78**(1.0)	< 0.01
Frequency	$2.03(0.71)$	**2.65**(0.89)	< 0.01	$2.82(0.89)$	**3.54**(0.83)	< 0.001
Adoption rate	$38.34(22.47)$	**45.76**(22.51)	< 0.01	$38.57(26.80)$	**38.68**(27.24)	ns

Table 4. Mean values of each factor with standard deviations and significance levels. Boldfaced values indicate the higher mean values between interpersonal vs. non-interpersonal.

little less of the variance compared to the original study (75.2% with component 1 accounting for 49% and component 2 accounting for 26.2%). Although the accounted variance of the PCA is lower than of the original study, the extracted components still account for a large part of the variance.

Further linear regression show similar results as in the original study in which frequency plays a significant role in both interpersonal and non-interpersonal channels when it comes to adoption rates. Additional univariate testing show that adoption rates are influenced by diversity and novelty through interpersonal channels, while the novelty factor only plays a role on adoption rates in non-interpersonal channels. In this respect, our sample differs significantly compared to the Korean sample of the original study in which relevance and novelty factors play a role on adoption rates in interpersonal channels and relevance and diversity influenced adoption rates in non-interpersonal channels.

5.2 Deviations

Although in general similar results were found in line with the original study, there are obvious differences regarding the effect of certain evaluation criteria on the adoption rate. Where the original study found relevance and novelty to play a role in interpersonal channels and relevance and diversity in non-interpersonal channels on adoption rates, the current study found that it is especially diversity and novelty for interpersonal channels and novelty for non-interpersonal channels on adoption rates. This indicates that for the participants in the current study, they believe that non-system recommendations are able to provide them with more diverse music to listen to. Hence, music that gets recommended through friends or acquaintances consist of better ways to find new music to listen to in order to deepen or broaden people's music taste. This would suggest that people feel that music recommended through a system or service is more homogeneous (e.g., recommending music from the short tail). The need to include more diversified recommendations (or at least the perception of diversification) in those systems may play an important role on the satisfaction of users [22].

The found differences need to have further exploration to find the root-cause of occurring. A possible cause could be that cultural differences play a role [23]. The sample in the current study consisted of European participants whereas the original study only had Korean participants. Another difference in the demographics of the sample in the current study is the low amount of participants that is sharing music with others (merely 16% opposed to 62% in the original study), which could contribute to the differences in findings.

5.3 Additional findings

Aside of testing the reproducibility of the work of Kim et al. [1], we took this opportunity to further investigate other effects within interpersonal and non-interpersonal channels. Our findings show that the usability of non-interpersonal channels plays a significant role on the adoption rate of the music. In this case adding the recommended music to the user's playlist. Hence, even though improving algorithms is given a lot of attention, the usability still plays a vital role on whether recommendations of systems are being adopted by its users.

We furthermore found that gender plays a role in some aspects of interpersonal and non-interpersonal channels. In particular, males were found to receive more frequent music recommendations than females. This suggest that males in general might be using music recommendations systems more frequently than females. Additionally we found an effect of gender on receiving novel music recommendations through interpersonal channels. These results suggest that males believe that music recommendations received through friends or acquaintances are more often new or novel than females believe that they receive through interpersonal channels. Hence, it seems that the friends and acquaintances of males share and recommend more often music that is new or novel with each other.

6. ANALYSIS USING THE PRIMAD MODEL

In order to further analyze and systematically place the conducted reproducibility study, we apply the PRIMAD model [18]. The goal of PRIMAD is to systematically vary ("prime") the factors (P)latform, (R)esearch Objective, (I)mplementation, (M)ethod, (A)ctor, and (D)ata, and to categorize the various types of insights gained via the

resulting reproducibility characteristics by analyzing the variations made.

One of the core principles of the model is the underlying observation that a simple replication of experiments under completely identical conditions does not lead to any real knowledge gains, save from confirming a deterministic behavior of the process. A substantial knowledge gain, however, originates from specific priming and variation of factors. In the concrete case, we are *foremost interested in the effect of priming (D)ata*, with obvious deviations regarding (A)ctor, i.e. the persons executing the experiments, and (P)latform and (I)mplementation, which can be considered robust due to the well-established and replicable method, and are hence of lesser interest. By extending the analyses, we also vary the (M)ethod to gain further insights with regards to the unchanged overall (R)esearch Objective.

The repetition of user experiments using a different data sample with a different background provides insights on the generalization capabilities of the analysis. This offers insights on whether the original observations also hold in (slightly) different data settings. Simply using the same data but priming only the parameter settings, on the other hand, uncovers the parameter sensitivity of an analysis. While such a sensitivity evaluation should actually be performed by the initial investigation we frequently observe that studies report on the process of meta-search for the optimal parameter setting, yet sometimes failing to report on the performance variations observed. These, however, are crucial for understanding whether an insight holds as general observation and can thus be deployed, or whether a highly sensitive parameter that causes huge performance variations on minor changes to that parameter basically would limit application to predefined test settings.

More generally, the question of which factors to prime is difficult to answer. Priming of (D)ata (parameters, raw data) should be a given to derive conclusions that are not based on singular events and configurations. (M)ethod priming is essential for true confirmation of findings—and is finding additional support from the area of explainable AI, where (in principle) interpretable surrogate models are used to understand and explore the behavior of more powerful black-box models. Priming of the (P)latform, at the least, leads to information on the correctness of the (I)mplementation and completeness of the description. [2]

In the study investigated, the process was sufficiently documented and even could be reproduced without availability of the original code for the user study or for subsequent data analysis. In general, for user studies, the standards for description of study design, method description, and participants are high and expected to suffice for reproducing the study. Furthermore, as the number of settings and parameters to be documented and taken into account is much smaller than, e.g., in machine learning experiments, this should give further incentive to increasingly reproduce user studies and gather even further insights by systematically "priming" factors.

[2] Unfortunately, this aspect seems to see decreasing relevance through the trend of publicly sharing code as extensive testing of code before reuse appears to have lower priority. An investigation of the potentially harmful effects of code sharing on scientific practice are still pending, however.

7. CONCLUSION

In this work, we reproduced the study by Kim et al. [1] on differences of music recommendations obtained through non-interpersonal channels, i.e., recommender systems, and recommendations obtained through interpersonal channels, i.e. friends or acquaintances, with newly collected data (sample size $n = 142$). By re-implementing the analyses as carried out in the original study, we could largely confirm the original results and support their validity. Possibly caused by the differences in music sharing behavior and/or cultural differences (predominantly European vs. Korean study participants), we also found differences in our results mostly with regards to novelty in recommendations and their impact on adoption rate. From this we conclude that the aspect of cultural differences in music consumption should be accounted for more explicitly in future studies, i.e., by repeating and extending existing studies at larger, global scale, or specifically addressing cultures. As MIR systems ultimately are designed for users and user studies (if carried out) are the departing point for the design of these systems, findings of one existing study might not be reliable enough and generalize beyond the group it was conducted with.

For conducting the various types of reproducibility studies in MIR, using the PRIMAD model seems to be a meaningful framework and we suggest its use in future studies. We should stress, however, that in many reproducibility studies the priming does not happen due to according study design, but to compensate for lack of information available, be it a lack of parameter descriptions, the lack of sharing the data sets involved in training and evaluating systems, failure to make (thoroughly tested) code available or have sufficiently detailed information on architecture and design of, e.g., deep neural network models. Such incomplete information forces the actors in reproducibility studies to make documented assumptions on the according setting, leading most likely to slight variations in the according setting compared to the original study design.

8. ACKNOWLEDGEMENTS

This research was funded in whole, or in part, by the Austrian Science Fund (FWF) [P33526]. For the purpose of open access, the author has applied a CC BY public copyright license to any Author Accepted Manuscript version arising from this submission.

9. REFERENCES

[1] H. J. Kim, S. Y. Park, M. Park, and K. Lee, "Do channels matter? illuminating interpersonal influence on music recommendations," in *Proceedings of the 14th ACM Conference on Recommender Systems*. Virtual Event, Brazil: ACM, Sep. 2020, p. 663–668.

[2] K. R. Page, B. Fields, D. D. Roure, T. Crawford, and J. S. Downie, "Capturing the workflows of music information retrieval for repeatability and reuse," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 435–459, 2013.

[3] W. Chen, J. Keast, J. Moody, C. Moriarty, F. Villalobos, V. Winter, X. Zhang, X. Lyu, E. Freeman, J. Wang, S. Cai, and K. Kinnaird, "Data Usage in MIR: History & Future Recommendations," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*. Delft, The Netherlands: ISMIR, Nov. 2019, pp. 25–32.

[4] R. Bittner, M. Fuentes, D. Rubinstein, A. Jansson, K. Choi, and T. Kell, "mirdata: Software for Reproducible Usage of Datasets," in *Proceedings of the 20th International Society for Music Information Retrieval Conference*. Delft, The Netherlands: ISMIR, Nov. 2019, pp. 99–106.

[5] B. McFee, E. J. Humphrey, and J. Urbano, "A plan for sustainable mir evaluation," in *Proceedings of the 17th International Society for Music Information Retrieval Conference*, New York City, United States, Aug. 2016, pp. 285–291.

[6] M. Schedl, A. Flexer, and J. Urbano, "The neglected user in music information retrieval research," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 523–539, 2013.

[7] P. Knees, M. Schedl, B. Ferwerda, and A. Laplante, "User awareness in music recommender systems," in *Personalized Human-Computer Interaction*. De Gruyter Oldenbourg, 2019, pp. 223–252.

[8] S. J. Cunningham, N. Reeves, and M. Britland, "An ethnographic study of music information seeking: Implications for the design of a music digital library," in *Proceedings of the 3rd ACM/IEEE-CS Joint Conference on Digital Libraries*. Houston, Texas: IEEE, May 2003, p. 5–16.

[9] J. H. Lee and R. Price, "Understanding Users of Commercial Music Services through Personas: Design Implications," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*. Málaga, Spain: ISMIR, Oct. 2015, pp. 476–482.

[10] K. Andersen and P. Knees, "Conversations with Expert Users in Music Retrieval and Research Challenges for Creative MIR," in *Proceedings of the 17th International Society for Music Information Retrieval Conference*. New York City, United States: ISMIR, Aug. 2016, pp. 122–128.

[11] E. L. M. Law, L. von Ahn, R. B. Dannenberg, and M. Crawford, "Tagatune: A game for music and sound annotation," in *Proceedings of the 8th International Conference on Music Information Retrieval*. Vienna, Austria: ISMIR, Sep. 2007, pp. 361–364.

[12] J. H. Lee, "Crowdsourcing Music Similarity Judgments using Mechanical Turk," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*. Utrecht, Netherlands: ISMIR, Aug. 2010, pp. 183–188.

[13] C. Inskip and F. Wiering, "In Their Own Words: Using Text Analysis to Identify Musicologists' Attitudes towards Technology," in *Proceedings of the 16th International Society for Music Information Retrieval Conference*. Málaga, Spain: ISMIR, Oct. 2015, pp. 455–461.

[14] J. H. Lee and J. S. Downie, "Survey Of Music Information Needs, Uses, And Seeking Behaviours: Preliminary Findings." in *Proceedings of the 5th International Conference on Music Information Retrieval*. Barcelona, Spain: ISMIR, Oct. 2004.

[15] J. H. Lee and S. J. Cunningham, "Toward an understanding of the history and impact of user studies in music information retrieval," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 499–521, 2013.

[16] S. Schmidt, "Shall we really do it again? the powerful concept of replication is neglected in the social sciences," *Review of General Psychology*, vol. 13, no. 2, pp. 90–100, 2009.

[17] J. Urbano, M. Schedl, and X. Serra, "Evaluation in music information retrieval," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 345–369, 2013.

[18] A. Rauber, V. Braganholo, J. Dittrich, N. Ferro, J. Freire, N. Fuhr, D. Garijo, C. Goble, K. Järvelin, B. Ludäscher, B. Stein, and R. Stotzka, "PRIMAD – information gained by different types of reproducibility," in *Reproducibility of Data-Oriented Experiments in e-Science (Dagstuhl Seminar 16041)*, J. Freire, N. Fuhr, and A. Rauber, Eds. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016, vol. 6, no. 1, pp. 128–132.

[19] A. Resch, S. Strumbelj, and L. Tomandl, "A Reproducibility Analysis on: Do Channels Matter? Illuminating Interpersonal Influence on Music Recommendations," Jan. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4459832

[20] V. Bauer, J. Bobinac, and F. Y. Tang, "Reproducibility Study: Do Channels Matter? Illuminating Interpersonal Influence on Music Recommendations," Jan. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5920550

[21] A. Ceranic, R. Dizdar, and A. Sokoli, "Reproduce experimental results from a paper," Jan. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.5921006

[22] B. Ferwerda, M. P. Graus, A. Vall, M. Tkalcic, and M. Schedl, "How item discovery enabled by diversity leads to increased recommendation list attractiveness,"

in *Proceedings of the Symposium on Applied Computing*. Marrakech, Morocco: ACM, Apr. 2017, p. 1693–1696.

[23] B. Ferwerda and M. Schedl, "Investigating the relationship between diversity in music consumption behavior and cultural dimensions: A cross-country analysis," in *Extended Proceedings of the 24th ACM Conference on User Modeling, Adaptation and Personalisation*. Halifax, Canada: CEUR-WS, Jul. 2016.

MUSIC SEPARATION ENHANCEMENT WITH GENERATIVE MODELING

Noah Schaffer[⋆1] **Boaz Cogan**[⋆1] **Ethan Manilow**[1]
Max Morrison[1] **Prem Seetharaman**[2] **Bryan Pardo**[1]
[1] Interactive Audio Lab, Northwestern University, Evanston IL, USA
[2] Descript, Inc.

ABSTRACT

Despite phenomenal progress in recent years, state-of-the-art music separation systems produce source estimates with significant perceptual shortcomings, such as adding extraneous noise or removing harmonics. We propose a post-processing model (the Make it Sound Good (MSG) post-processor) to enhance the output of music source separation systems. We apply our post-processing model to state-of-the-art waveform-based and spectrogram-based music source separators, including a separator unseen by MSG during training. Our analysis of the errors produced by source separators shows that waveform models tend to introduce more high-frequency noise, while spectrogram models tend to lose transients and high frequency content. We introduce objective measures to quantify both kinds of errors and show MSG improves the source reconstruction of both kinds of errors. Crowdsourced subjective evaluations demonstrate that human listeners prefer source estimates of bass and drums that have been post-processed by MSG.

1. INTRODUCTION

Audio source separation is the problem of isolating a sound producing source (e.g., a singer) or group of sources (e.g., a backing band) in an audio scene (e.g., a music recording). Source separation is a core problem in computer audition that can facilitate music remixing and other Music Information Retrieval (MIR) tasks such as music instrument labeling [1, 2] and transcription [3, 4].

Current state-of-the-art source separation systems often produce source estimates that contain perceptible artifacts, such as high-frequency noise, source leaking (e.g., drum hits heard in the bass source estimate), unnatural transients, or missing overtones. For many downstream tasks in MIR or music creation, it is preferable for source separators to minimize these errors. Given that we have observed these artifacts to be endemic to the separators themselves, we propose an additional post-processing step to clean up the initial outputs of these separators.

In this work, we introduce Make it Sound Good (MSG), a post-processing neural network for enhancing the quality of music source separation. MSG combines elements of off-the-shelf architectures from generative modeling tasks in speech vocoding and denoising to enhance the output of pre-trained source separation models in both the waveform and spectrogram domains.

The main contributions of this work are:

- A source separation post-processor (MSG) that performs imputation and denoising to enhance the output of both waveform and spectrogram models for music audio source separation.
- A subjective listener study that confirms MSG improves the perceptual quality of bass and drum source estimates on a set of five separation models, including one on which it was not trained.
- An in-depth exploration of the kinds of errors produced by different classes of source separators and how MSG affects these errors.

Audio examples and code can be found at `https://interactiveaudiolab.github.io/project/msg.html`.

2. RELATED WORK

Deep learning is the dominant approach for music source separation. For example, all entries to the 2021 Sony Music Demixing Challenge [5] were deep learning based separators. Most separators fall into one of two classes. *Waveform models* [6–9] take audio waveform input and produce an audio waveform for each separated source. *Spectrogram models* [10–18] take a mixture spectrogram as input and output a mask to apply to the spectrogram for each source being separated. Despite the recent successes of these deep learning methods, state of the art systems continue to exhibit perceptible artifacts in their outputs. We show in Section 5 that waveform models tend to introduce more high-frequency noise, while spectrogram models tend to lose transients and high frequency content.

⋆ Equal contribution

© N. Schaffer, B. Cogan, E. Manilow, M. Morrison, P. Seetharaman, and B. Pardo. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** N. Schaffer, B. Cogan, E. Manilow, M. Morrison, P. Seetharaman, and B. Pardo, "Music Separation Enhancement with Generative Modeling", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

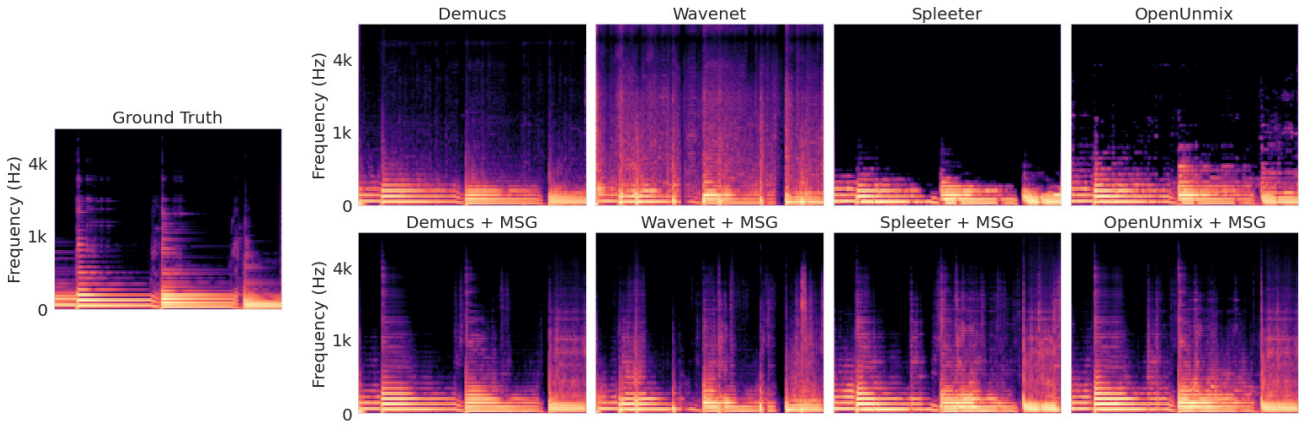


Figure 1: Spectrograms of ground-truth (left), source estimates (top), and MSG output (bottom) for the bass source. MSG is able to simultaneously infer missing frequencies and remove noise from the output of common source separation systems.

Recent works in adversarial audio synthesis [19–25] and end-to-end speech enhancement [26–32] show that the adversarial loss of Generative Adversarial Networks (GANs) [33] is effective in generating high-fidelity audio. Although such systems have been effective in audio synthesis and denoising, no previous work has explored using these systems for enhancing source separation output. Recent work has also shown that adversarial loss is effective for training source separation systems [34, 35], however this work does not look at using adversarial loss to enhance existing separation output.

While many recent works for speech enhancement have been proposed, music enhancement (e.g., denoising and artifact removal) is less common. There are only a few recent works in music denoising [36–38] and bandwidth expansion [39, 40]. Some work has also looked at potential causes of [41] and remedies for [42] audio artifacts in untrained source separation networks. Our work is most similar to Kandpal et. al [43], who proposed a generative model that can enhance the audio quality of a low-quality music recording taken on a consumer device. We are unaware of a prior system for enhancing the output of trained music separation systems.

3. "MAKE IT SOUND GOOD" POST-PROCESSOR

Here we describe our "Make it Sound Good" (MSG) post-processor, which perceptually improves a source estimate by removing artifacts that the separator introduced and imputing elements the separator omitted. We use the adversarial loss of Generative Adversarial Networks (GANs) [33] due to its success denoising many types of audio.

The generator of MSG is a waveform-to-waveform U-Net with 1D convolutions. This is very similar to the Demucs v2 [7] architecture with the exception that Demucs has two BLSTM layers at the bottleneck, which we omit.

We train the generator using three loss functions. The first is the LSGAN [44] generator loss,

$$L_G = \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}\left[\left(D_k(G(\hat{s}))-1\right)^2\right], \qquad (1)$$

where $\hat{s}$ is the raw source estimate from the separator, D_k is the k-th discriminator, K is the total number of discriminators, and G is the generator.

Next is deep feature matching loss [45], which is the L_1 distance between the intermediate activations of the discriminators on corresponding real and generated data. The last loss function we use is a multi-scale Mel-spectrogram reconstruction loss [46, 47], which is the average Mel reconstruction loss over three different Short-time Fourier transforms (STFTs), each of which uses different parameters for the number of STFT bins, window lengths, and hop sizes.

We use two types of discriminators: the multi-period discriminators from HiFi-GAN [24] and the multi-resolution spectrogram discriminators from UnivNet [25]. The multi-period discriminators operate on the waveform, and reshapes the waveform to a 2D tensor with a prime-valued stride before processing the reshaped waveform with 2D convolutional layers. The multi-resolution spectrogram discriminators process a spectrogram with different STFT window sizes (see Section 4.3). We use five multi-period discriminators with strides $[2, 3, 5, 7, 11]$, respectively. We also use three multi-resolution discriminators with FFT windows $[512, 1024, 2048]$, for a total of eight discriminators. Each discriminator uses the LSGAN [44] loss,

$$L_D = \mathbb{E}\left[\left(D(s)-1\right)^2 + \left(D(G(\tilde{s}))\right)^2\right], \qquad (2)$$

where $\tilde{s}$ is the cleaned up source estimate from the MSG generator and s is the ground-truth source audio. Further details on the discriminator architectures are provided in the original papers [24, 25].

We use an adversarial loss typical of Generative Adversarial Networks, but do not condition on a random input vector. This produces a deterministic model that is not

technically generative. However, prior work has shown that the unique mode-selecting behaviors of these adversarial loss models are highly effective for densely-conditioned generative modeling tasks such as vocoding [20,24,25] and speech denoising and enhancement [26–32]. Our goal is not an exact reconstruction of ground truth, but an output that is perceptually improved. This means a distribution of viable outputs exists and the task can be framed as one of generative modeling.

4. EXPERIMENTAL VALIDATION

We conducted experiments to understand whether MSG perceptually enhances the raw output of a set of music source separation models. The remainder of this section is devoted to outlining the details of our experiments.

4.1 Models

We trained MSG post-processors using the output of four existing source separation models as the input to MSG. We train on two waveform-based separators: Demucs v2 [7] and Wavenet [8]; and two spectrogram-based separators: Spleeter [17] and OpenUnmix [16]. To investigate whether MSG can learn to correct the artifacts of different separators, we created one enhancement model that is trained and evaluated on all four separators instead of creating separator-specific models. We evaluated the MSG post-processor using the output of each of the four separators on a held out test set (see Section 4.2). Furthermore, to understand whether MSG can also reduce the artifacts produced by an unseen separator, we evaluate our post-processor on the output of a fifth separator that it was never trained on: Hybrid Demucs (v3) [48], which operates in both waveform and spectral domains. For all separation models we used the trained, frozen weights released by the authors, with no alterations. We refer readers to the papers on each separator for architectural and training details.

4.2 Data

All experiments were run with the MUSDB18 dataset [49]. MUSDB18 contains 150 songs: 100 in the training set and 50 in the test set. Each song in MUSDB18 has a full mixture and isolated source audio stems for vocals, bass, drums and a fourth catch-all category called "other". We omitted this catch-all category because we find that attempting to enhance many instruments at once with the same model greatly increases the difficulty of the task. We performed source separation on every song in MUSDB18 using all five of our source separators, producing source estimates of bass, drums and vocals.

The input audio was peak normalized before passing it through the network. Since Wavenet operates at 16 kHz we use this sample rate. We downsampled all systems to 16 kHz so that there was a uniform sample rate across all separation models. Here, we focus solely on enhancement and leave the task of bandwidth extension for future work. Thus, the output of MSG on all systems was at a sample rate of 16 kHz.

4.3 Training

We trained one MSG model on the MUSDB18 training set for each source class (bass, drums, or vocals). Each model was trained using source estimates from four separators (Demucs v2, Wavenet, Spleeter, and OpenUnmix) as input and the ground-truth sources as training targets. We segmented the audio into 1-second clips and rejected silent clips where the ground-truth source has an RMS below -60dB FS, resulting in over 100,000 training examples per model. On each training iteration, we randomly swapped the input data with ground-truth with a 10% probability. This encourages the model to leave high-quality audio unaltered.

We computed the three resolutions of STFTs passed to our multi-resolution spectrogram discriminators (Section 3) using window sizes of 512, 1024, and 2048 samples and hop sizes of 128, 256, and 512 samples, respectively. We used one Adam optimizer [50] for the generator and another for the discriminator. We used a learning rate of 2e-4 and beta values of $(.5, .9)$. To find suitable loss weights for all 3 types of losses on the generator (LSGAN loss, deep matching loss, and multi-scale spectral loss; see Section 3), we solved a least squares equation to weigh all loss terms equally for the first 1k training iterations. After that, we froze those weights applied to the losses for the remainder of training.

4.4 Subjective evaluation

The goal of our research is to improve the perceptual quality of source separation output. Therefore, we evaluate our MSG post-processor using a crowdsourced subjective evaluation (Section 4.4) rather than reporting an objective metric like Signal-to-Distortion Ratio (SDR) [51,52], since widely-used objective metrics for source separation are imperfect proxies for human perception [34, 52–57].

For evaluation data, we used one seven-second segment from each of the 50 songs from the MUSDB18 test set. We performed source separation on each seven-second segment using each of the five source separation systems (Section 4.1) to create source estimates of bass, drums, and vocals. Each output was then processed with MSG, resulting in 50 matched pairs for each combination of separator and source class: the raw output, and the output processed by MSG.

There are 15 unique combinations of the five separators and three sources (bass, drums, and vocals). For each combination, we performed a two-way forced-choice listening test between the raw output and the output processed by MSG. We initially recruited 20 participants for each test and omitted responses from participants that failed a prescreening listening test. This resulted in a minimum number of 15 participants in any test. Each participant evaluated 25 randomly-selected pairs from the 50 examples for that combination of source and separator.

A two-tailed binomial test was performed where the null hypothesis was that there was no difference between MSG-enhanced and raw separator output. If the results of a particular test showed no difference (i.e., $p < 0.05$) we

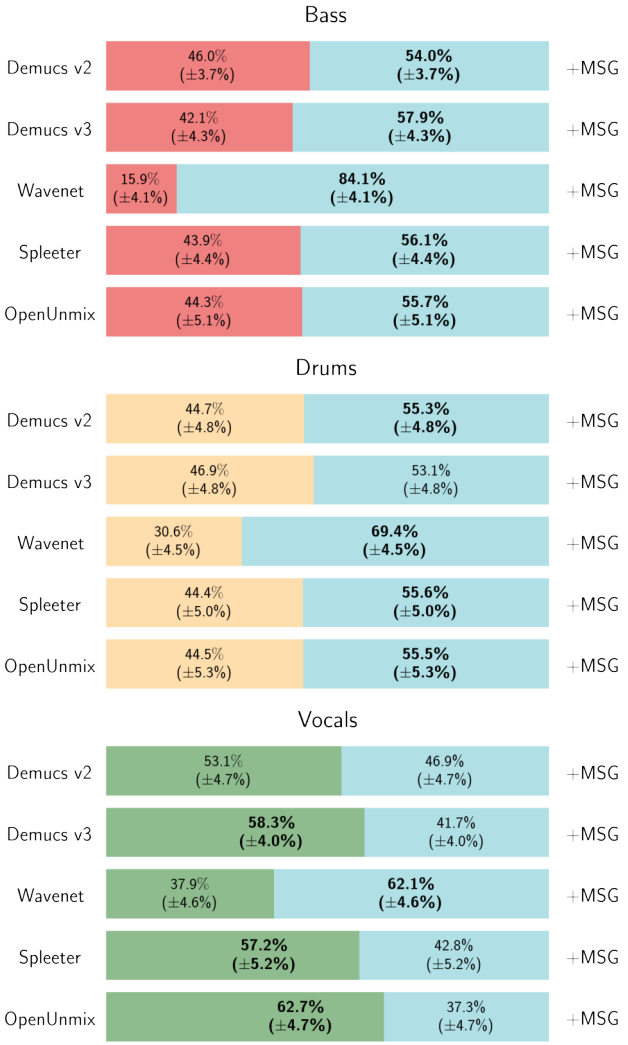


Figure 2: Subjective pairwise test results for bass, drums, and vocals. Each row contains the percent of listeners selecting that option as higher quality in a two-way forced choice listening test. A bold-faced value indicates a statistically significant difference.

recruited an additional 10 participants to see if a difference could be determined.

For each pairwise comparison, participants were given the following instructions, where <*source*> is one of "bass", "drums" or "vocals":

> Listen to both recordings of a <*source*>. After listening to both, select the recording that sounds like a higher-quality <*source*>. The higher-quality recording is the one that is more natural sounding, or has fewer audio artifacts (e.g., noise, clicks, or other instruments).

We used Reproducible Subjective Evaluation (ReSEval) [58] to set up our listener studies. We recruited participants via Amazon Mechanical Turk (MTurk). Our participants were US residents at least 18 years old that completed 20 or more tasks on MTurk with an approval rating

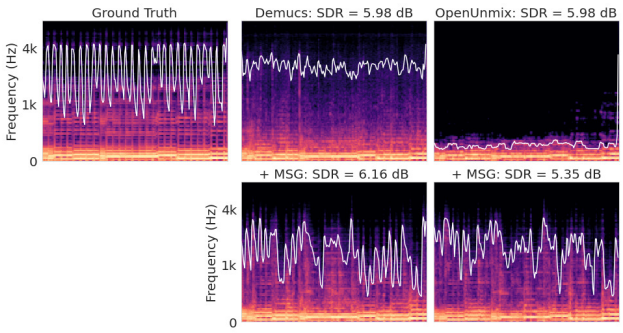


Figure 3: Spectrograms of the ground truth (left) and source estimates from Demucs v2 and OpenUnmix (top) and corresponding MSG output (bottom) for the bass source. The 98% spectral rolloff frequency is overlaid in white.

of at least 97%. Participants who passed the listening test and completed our evaluation were paid $3.00.

For each of the 15 tests, we collected between 308 and 696 pairwise evaluations from between 15 and 30 participants who passed the prescreening listening test. The number of evaluations is not a multiple of 25 because a few participants did not finish all 25 examples in their set of pairwise evaluations.

4.4.1 Subjective evaluation results

Each listening test evaluates one combination of source class and source separator. Figure 2 shows the results for each of the 15 tests. Listeners preferred the MSG output to the raw source separator output in 11 out of 15 combinations of separator and source. This difference was statistically significant (using a binomial test) in 10 of the 11 combinations. Listeners preferred the quality of separators with MSG on bass source estimates and had a moderate preference for MSG on drums. For vocals, listeners had a slight preference for the source estimate without MSG.

MSG performed best on the Wavenet separator, where it significantly improved the perceptual audio quality of all sources. MSG was also able to improve on the quality of source estimates of a separator not seen during training, Demucs v3, for bass sources—as well as an improvement on Demucs v3 drum sources that was not statistically significant. Note that Demucs v3 is a hybrid approach that operates in both the waveform and spectrogram domains. Our performance on vocals indicates that MSG is not able to enhance the quality of the source estimate of vocals. We are unaware of prior work that attempts to enhance source separation output, let alone a separated vocal source. Vocal separation has been optimized by many years of existing research, potentially leaving less room for a post-processing system to improve compared to, e.g., drum and bass sources.

5. FURTHER ANALYSIS

Our listener study indicates whether a separation is relatively "good" or "bad", but it does not clarify *why* one sep-

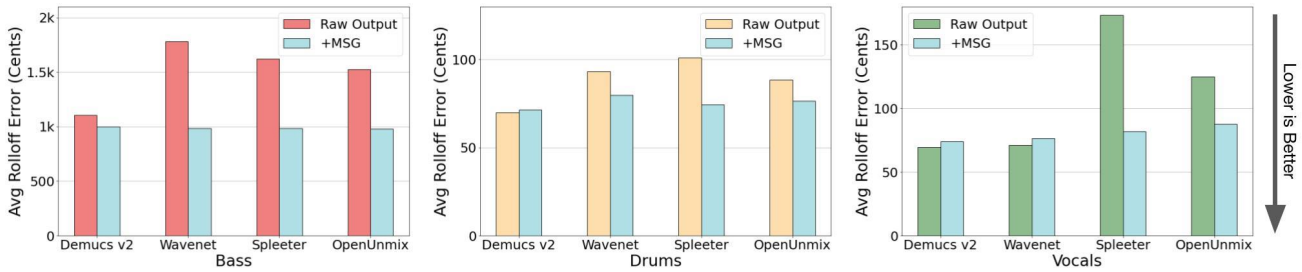


Figure 4: Mean spectral rolloff error for bass, drums, and vocals for separators with and without MSG post-processing.

arated source is better or worse than another. Similarly, the widely-used Signal-to-Distortion Ratio (SDR) [51, 52], as well as the related SIR and SAR, are not designed to capture the specific types of errors we focus on in this work. See, for example, the top row of Figure 3, which shows source estimates produced by Demucs v2 and OpenUnmix for the same bass source from the same mixture. Demucs adds additional high-frequency noise not present in the ground truth, while OpenUnmix removes many of the upper harmonics. Visually, the difference between these two systems is plain, however their SDR values (using `mus_eval` [59]) are equal to two decimal places: *5.98 dB!*

In this section, we examine the output of the four state-of-the-art separation systems used in the training of MSG models: (Demucs v2 [7], Wavenet [8], Spleeter [17], and OpenUnmix [16]), as well as the MSG-processed outputs for those four systems. As before, we use the MUSDB18 [49] test set, and we omit the "other" source.

Anecdotally, we have noted that waveform separators tend to add extra high-frequency noise and spectrogram separators tend to remove high-frequency partials, especially in bass estimates (see Figure 1). Spectrogram separators also tend to smooth out transients. While these are not the only issues that current separation systems exhibit, the rest of this section will be dedicated to analysis of these two issues.

5.1 Added and Missing Frequency Content

Following our anecdotal observation that waveform separators tend to add extra high-frequency noise and spectrogram separators tend to remove high-frequency partials, we seek to formalize these notions.

One statistic that can be a good proxy for whether a source estimate has excess high-frequency content or is missing desirable high-frequency content is spectral rolloff. For a given time frame in a spectrogram, the spectral rolloff at $X\%$ is the frequency below which $X\%$ of the energy of the signal lies. For example, the white line on each spectrogram in Figure 3 shows the spectral rolloff at 98%.

For every song in the MUSDB18 [49] test set, we compute the spectral rolloff at 98% every 32 ms (a hop size of 512 samples at 16 kHz) for every ground truth isolated source, every estimate produced by one of the four training separators and every MSG-enhanced source estimate. To calculate our statistics, we omit any frames that have an RMS less than -40 dBFS, in the ground-truth source, so as not to examine rolloff in relatively silent regions. We report the error between a source estimate's rolloff and a ground-truth source's rolloff in cents, which is $1200 \times (\log_2 x - \log_2 y)$, where x and y are rolloff frequencies in Hz. We chose to use cents over Hz because it better correlates to how humans perceive audio.

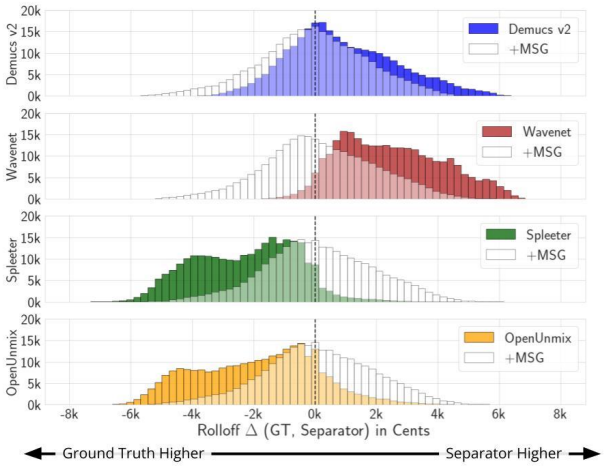


Figure 5: Histogram of the difference in spectral rolloff values between a given separator's bass estimate and ground truth bass source over the MUSDB18 test set. The vertical dotted line shows the desired difference of 0. MSG reduces the difference between the rolloff values of source estimate and ground-truth.

In Figure 4, we show the mean error, in cents, between the ground truth rolloff and the source estimate's rolloff. We see that the source estimates of vocals and drums have spectral rolloff errors on the order 100-200 cents, whereas source estimates of bass have errors of roughly 1000 cents. MSG reduces this error for all separators on bass sources, for three of the four separators on drums, and two of the four separators on vocals.

Because the bass estimates have such large errors, we examine them further in Figure 5, where we show a histogram of the per-frame differences between the spectral rolloff of source estimates and ground truth. The top two rows of the histogram show results for the two waveform separators (i.e., Demucs v2 and Wavenet), which each show an error distribution that is strongly skewed towards

positive error. This corroborates our observation of high-frequency noise introduced by waveform separators, as shown in the bass spectrogram in Figure 3. The bottom two rows show the error distribution for two spectrogram separators, Spleeter and OpenUnmix. Both exhibit an error distribution for spectral rolloff that is strongly skewed toward negative values. This quantifies the effect illustrated in Figures 3 and 1, where the spectrogram separators remove the higher partials of the bass source.

We further observe that, when MSG is applied to the output of all four separators, the resulting error distribution is less biased and, as was already shown in Figure 4, reduces the mean error magnitude. Figure 1 illustrates the effect of MSG on a single bass example, showing improved spectral rolloff reconstruction for both waveform and spectrogram models.

5.2 Improving Transient Reconstruction

While listening to the source estimates from spectrogram separators, we noticed that the transients of source estimates for drums and bass did not sound as clear as in the ground truth source estimates. To quantify these observations, we measure the location and strength of onsets in the estimated sources relative to the ground-truth. We use librosa's [60] `onset_strength()` function [61], which computes the spectral flux onset strength envelope at every frame in a spectrogram. We approximate an onset by identifying every frame with a strength above a certain threshold. We select an onset strength threshold via manual tuning. We set the threshold value to a constant value of 0.75 for both bass and drums on the MUSDB18 dataset. We manually tuned this threshold to find a value that best corresponds with our perception of relevant peaks in the signal. We chose to threshold `onset_strength()` instead of using librosa's `onset_detect()` because we found that matching up onsets between two signals using the latter method was hard to correctly tune.

We run this onset strength thresholding on both the ground-truth source and a source estimate and then calculate the F1 between the binary threshold arrays of the raw source estimate and the MSG post-processed estimate as a proxy for how well a separator preserves transients. A true positive (TP) is when a detected onset exists at the same spectrogram frame in the ground-truth and source estimate, a false positive (FP) is when an onset is detected at a frame in the source estimate but not the ground truth, and a false negative is when an onset is detected (FN) at a frame in the ground truth but not in the source estimate. We report the F1 score of onset reconstruction, $TP/(TP + \frac{1}{2}(FP + FN))$.

We report the F1 scores for onset detection on bass, drums, and vocals in Table 1. The results for vocals are not in favor of MSG for 3 of the 4 separators. We include the vocals results for completeness. Evaluating the transients for vocals might be slightly unusual, but the observed results align with the listener studies. In contrast with vocals, bass and drums both see improved F1 scores across multiple separators: MSG improves the F1 score in 7 out of the 8 combinations of source and separator, with the sole exception of drums separated by Demucs v2. This indicates that the ability to represent transients is generally improved by applying MSG-based post processing on bass and drums.

	Model	Type	Onset Strength F1 Raw	Onset Strength F1 + MSG
Bass	Demucs v2	Wave	0.52	**0.54**
	Wavenet	Wave	0.36	**0.44**
	Spleeter	Spec	0.36	**0.52**
	OpenUnmix	Spec	0.39	**0.49**
Drums	Demucs v2	Wave	**0.84**	0.82
	Wavenet	Wave	0.73	**0.74**
	Spleeter	Spec	0.78	**0.82**
	OpenUnmix	Spec	0.78	**0.79**
Vocals	Demucs v2	Wave	**0.58**	0.57
	Wavenet	Wave	**0.51**	0.49
	Spleeter	Spec	**0.71**	0.66
	OpenUnmix	Spec	0.41	**0.57**

Table 1: F1 scores for thresholded onset strength for bass, drums, and vocals for four separators with and without MSG post-processing. "Raw" means that F1 is computed between the separator's raw output and ground truth. "+MSG" means that MSG post-processing is applied to the raw source estimates. According to this measure, MSG is able to better preserve onsets in 7 out of 8 cases between the bass and drums sources, which most clearly demonstrate artifacts with transients.

6. CONCLUSION

State-of-the-art music source separators create audible perceptual degredations, such as missing frequencies and transients. In this work, we propose Make it Sound Good (MSG), a post-processing neural network that leverages generative modeling to enhance the perceptual quality of music source separators. In listening studies, users prefer bass and drum source estimates produced with MSG post-processing—even on a state-of-the-art separator not seen during training. We analyze the errors of waveform-based and spectrogram-based separators with and without MSG. Without MSG, we show that waveform-based separators induce high-frequency noise and spectrogram-based separators fail to reconstruct high-frequencies in the bass source, and have trouble reconstructing transients. We measure these artifacts via spectral rolloff and onset detection and show that, for both bass and drums, MSG generally improves reconstruction of spectral rolloff and onsets of the source estimate relative to the ground-truth sources. Fruitful directions for future work include using more modern techniques for sound enhancement (e.g., diffusion models [43]), making post-processors for the vocals and "other" sources, and deeper analyses of the issues with separation systems.

7. REFERENCES

[1] E. Vincent and X. Rodet, "Instrument identification in solo and ensemble music using independent subspace analysis," in *International Society for Music Information Retrieval (ISMIR)*, 2004.

[2] J. F. Woodruff, B. Pardo, and R. B. Dannenberg, "Remixing stereo music with score-informed source separation." in *International Society for Music Information Retrieval (ISMIR)*, 2006.

[3] M. D. Plumbley, S. A. Abdallah, J. P. Bello, M. E. Davies, G. Monti, and M. B. Sandler, "Automatic music transcription and audio source separation," *Cybernetics & Systems*, vol. 33, no. 6, pp. 603–627, 2002.

[4] E. Manilow, P. Seetharaman, and B. Pardo, "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[5] Y. Mitsufuji, G. Fabbro, S. Uhlich, and F.-R. Stöter, "Music demixing challenge 2021," *International Society for Music Information Retrieval (ISMIR)*, 2021.

[6] Y. Luo and N. Mesgarani, "Tasnet: time-domain audio separation network for real-time, single-channel speech separation," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018.

[7] A. Défossez, N. Usunier, L. Bottou, and F. Bach, "Music source separation in the waveform domain," *arXiv e-prints*, pp. arXiv–1911, 2019.

[8] F. Lluís, J. Pons, and X. Serra, "End-to-end music source separation: is it possible in the waveform domain?" *Interspeech*, 2019.

[9] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," in *International Society for Music Information Retrieval (ISMIR)*, 2018.

[10] A. M. Reddy and B. Raj, "Soft mask estimation for single channel speaker separation," in *ISCA Tutorial and Research Workshop on Statistical and Perceptual Audio Processing (SAPA)*, 2004.

[11] R. J. Weiss and D. P. Ellis, "Estimating single-channel source separation masks: Relevance vector machine classifiers vs. pitch-based masking," in *ISCA Tutorial and Research Workshop on Statistical and Perceptual Audition (SAPA)*, 2006.

[12] Z. Rafii and B. Pardo, "A simple music/voice separation method based on the extraction of the repeating musical structure," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011.

[13] A. Liutkus, Z. Rafii, R. Badeau, B. Pardo, and G. Richard, "Adaptive filtering for music/voice separation exploiting the repeating musical structure," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.

[14] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe, "Deep clustering: Discriminative embeddings for segmentation and separation," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016.

[15] Y. Luo, Z. Chen, J. R. Hershey, J. Le Roux, and N. Mesgarani, "Deep clustering and conventional networks for music separation: Stronger together," in *International conference on acoustics, speech and signal processing (ICASSP)*, 2017.

[16] F.-R. Stöter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, "Open-unmix-a reference implementation for music source separation," *Journal of Open Source Software*, vol. 4, no. 41, p. 1667, 2019.

[17] R. Hennequin, A. Khlif, F. Voituret, and M. Moussallam, "Spleeter: a fast and efficient music source separation tool with pre-trained models," *Journal of Open Source Software*, vol. 5, no. 50, p. 2154, 2020.

[18] N. Takahashi and Y. Mitsufuji, "D3net: Densely connected multidilated densenet for music source separation," *arXiv e-prints*, pp. arXiv–2010, 2020.

[19] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *International Conference on Learning Representations (ICLR)*, 2018.

[20] K. Kumar, R. Kumar, T. de Boissiere, L. Gestin, W. Z. Teoh, J. Sotelo, A. de Brebisson, Y. Bengio, and A. Courville, "Melgan: generative adversarial networks for conditional waveform synthesis," in *Neural Information Processing Systems (NeurIPS)*, 2019.

[21] M. Bińkowski, J. Donahue, S. Dieleman, A. Clark, E. Elsen, N. Casagrande, L. C. Cobo, and K. Simonyan, "High fidelity speech synthesis with adversarial networks," *International Conference on Learning Representation (ICLR)*, 2020.

[22] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "Gansynth: Adversarial neural audio synthesis," *International Conference on Learning Representations (ICLR)*, 2019.

[23] R. Yamamoto, E. Song, and J.-M. Kim, "Parallel wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[24] J. Kong, J. Kim, and J. Bae, "Hifi-gan: Generative adversarial networks for efficient and high fidelity speech synthesis," *Neural Information Processing Systems (NeurIPS)*, 2020.

[25] W. Jang, D. Lim, J. Yoon, B. Kim, and J. Kim, "Univnet: A neural vocoder with multi-resolution spectrogram discriminators for high-fidelity waveform generation," *Interspeech*, 2021.

[26] S. Pascual, A. Bonafonte, and J. Serrà, "Segan: Speech enhancement generative adversarial network," *Interspeech*, 2017.

[27] S.-W. Fu, C.-F. Liao, Y. Tsao, and S.-D. Lin, "Metricgan: Generative adversarial networks based black-box metric scores optimization for speech enhancement," in *International Conference on Machine Learning (ICML)*, 2019.

[28] J. Su, Z. Jin, and A. Finkelstein, "Hifi-gan: High-fidelity denoising and dereverberation based on speech deep features in adversarial networks," in *Interspeech*, 2020.

[29] A. Pandey and D. Wang, "Densely connected neural network with dilated convolutions for real-time speech enhancement in the time domain," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[30] J. Su, Z. Jin, and A. Finkelstein, "Hifi-gan-2: Studio-quality speech enhancement via generative adversarial networks conditioned on acoustic features," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2021.

[31] S.-W. Fu, C. Yu, T.-A. Hsieh, P. Plantinga, M. Ravanelli, X. Lu, and Y. Tsao, "Metricgan+: An improved version of metricgan for speech enhancement," in *Interspeech*, 2021.

[32] P. Andreev, A. Alanov, O. Ivanov, and D. Vetrov, "Hifi++: a unified framework for neural vocoding, bandwidth extension and speech enhancement," *arXiv e-prints*, pp. arXiv–2203, 2022.

[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Neural information processing systems (NeurIPS)*, 2014.

[34] E. Gusó, J. Pons, S. Pascual, and J. Serrà, "On loss functions and evaluation metrics for music source separation," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 306–310.

[35] D. Stoller, S. Ewert, and S. Dixon, "Adversarial semi-supervised audio source separation applied to singing voice extraction," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2391–2395.

[36] Y. Li, B. Gfeller, M. Tagliasacchi, and D. Roblek, "Learning to denoise historical music," *International Society for Music Information Retrieval (ISMIR)*, 2020.

[37] E. Moliner and V. Välimäki, "A two-stage u-net for high-fidelity denoising of historical recordings," *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[38] J. Imort, G. Fabbro, M. A. M. Ramírez, S. Uhlich, Y. Koyama, and Y. Mitsufuji, "Removing distortion effects in music using deep neural networks," *arXiv preprint arXiv:2202.01664*, 2022.

[39] S. Sulun and M. E. Davies, "On filter generalization for music bandwidth extension using deep neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 15, no. 1, pp. 132–142, 2020.

[40] S. Kim and V. Sathe, "Bandwidth extension on raw audio via generative adversarial networks," *arXiv e-prints*, pp. arXiv–1903, 2019.

[41] J. Pons, S. Pascual, G. Cengarle, and J. Serrà, "Upsampling artifacts in neural audio synthesis," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3005–3009.

[42] J. Pons, J. Serrà, S. Pascual, G. Cengarle, D. Arteaga, and D. Scaini, "Upsampling layers for music source separation," *arXiv e-prints*, pp. arXiv–2111, 2021.

[43] N. Kandpal, O. Nieto, and Z. Jin, "Music enhancement via image translation and vocoding," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022.

[44] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *International Conference on Computer Vision (ICCV)*, 2017.

[45] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," in *International Conference on Machine Learning (ICML)*, 2016.

[46] A. Defossez, G. Synnaeve, and Y. Adi, "Real time speech enhancement in the waveform domain," *Interspeech*, 2020.

[47] X. Wang, S. Takaki, and J. Yamagishi, "Neural source-filter waveform models for statistical parametric speech synthesis," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 402–415, 2019.

[48] A. Défossez, "Hybrid spectrogram and waveform source separation," *ISMIR Workshop on Music Demixing (MDX)*, 2021.

[49] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.1117372

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, 2014.

[51] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[52] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "Sdr–half-baked or well done?" in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[53] B. Fox, A. Sabin, B. Pardo, and A. Zopf, "Modeling perceptual similarity of audio signals for blind source separation evaluation," in *International Conference on Independent Component Analysis and Signal Separation*. Springer, 2007, pp. 454–461.

[54] M. Cartwright, B. Pardo, G. J. Mysore, and M. Hoffman, "Fast and easy crowdsourced perceptual audio evaluation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 619–623.

[55] M. Cartwright, B. Pardo, and G. J. Mysore, "Crowdsourced pairwise-comparison for source separation evaluation," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 606–610.

[56] U. Gupta, E. Moore, and A. Lerch, "On the perceptual relevance of objective source separation measures for singing voice separation," in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2015, pp. 1–5.

[57] E. Cano, D. FitzGerald, and K. Brandenburg, "Evaluation of quality of sound source separation algorithms: Human perception vs quantitative metrics," in *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE, 2016, pp. 1758–1762.

[58] M. Morrison, B. Tang, G. Tan, and B. Pardo, "Reproducible subjective evaluation," in *ICLR Workshop on ML Evaluation Standards*, April 2022.

[59] F.-R. Stöter, A. Liutkus, and N. Ito, "The 2018 signal separation evaluation campaign," in *Latent Variable Analysis and Signal Separation: 14th International Conference, LVA/ICA 2018, Surrey, UK*, 2018, pp. 293–305.

[60] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Python in Science (SciPy)*, 2015.

[61] S. Böck and G. Widmer, "Maximum filter vibrato suppression for onset detection," in *Digital Audio Effects (DAFx)*, 2013.

SAMPLEMATCH: DRUM SAMPLE RETRIEVAL BY MUSICAL CONTEXT

Stefan Lattner
Sony Computer Science Laboratories (CSL), Paris, France
stefan.lattner@sony.com

ABSTRACT

Modern digital music production typically involves combining numerous acoustic elements to compile a piece of music. Important types of such elements are *drum samples*, which determine the characteristics of the percussive components of the piece. Artists must use their aesthetic judgement to assess whether a given drum sample fits the current musical context. However, selecting drum samples from a potentially large library is tedious and may interrupt the creative flow. In this work, we explore the automatic drum sample retrieval based on aesthetic principles learned from data. As a result, artists can rank the samples in their library by fit to some musical context at different stages of the production process (i.e., by fit to incomplete song mixtures). To this end, we use contrastive learning to maximize the score of drum samples originating from the same song as the mixture. We conduct a listening test to determine whether the human ratings match the automatic scoring function. We also perform objective quantitative analyses to evaluate the efficacy of our approach.

1. INTRODUCTION

Machine-learning (ML) powered tools are increasingly finding their way into modern music production [1]. Commercial tools already use machine learning for different applications, like mixing/mastering [1] or data visualization. [2] In public perception, ML and artificial intelligence (AI) is often associated with *content generation*. Indeed, also in music production, generative AI models are about to transition from research labs to the studio, in the form of tools that integrate seamlessly into the artist's workflow, like Magenta Studio [3] and others [2–6].

Besides content generation, another frequent task in modern music production is *content retrieval*, where commercial solutions (based on expert systems) also already exist. [4] In manual content retrieval (i.e., content selection), artists select "loops" (short, typically bar-aligned audio files of any content) or "samples" (usually shorter, single notes or drum hits) that fit the current musical context. Content selection is a creative act in its own right, but it is challenging to browse through potentially large loop or sample libraries while – crucially – keeping the current context in mind.

To support the selection process, we envision a retrieval system that learns aesthetic principles of matching drum samples to musical context from data. As a result, it is able to sort a drum sample library according to a *score* it assigns to each sample. This lets users start their search with the most promising options first and may help turn a cumbersome curation process into a more artistic activity.

We use the cosine similarity in a learned latent space as a scoring function. Two encoders are trained with a contrastive loss to maximize the score between mixtures and drum samples originating from the same song. The encoders are trained with an electronic and acoustic data set of songs whose tracks are available separately. We perform a user study to evaluate if human listeners agree with our proposed scoring function. To that end, we test if participants prefer samples that obtain a high rating by the scoring function over randomly selected samples. Furthermore, we perform an ablation study to evaluate the different components of our method. In addition, we want to understand better how the learned space is organized. In particular, we want to gain some insights into the relationship of drum samples that fit well in the same context. For that, we perform a correlation analysis between audio features of samples that are close in the learned latent space.

The paper is organized as follows. Related works are discussed in Section 2, and Section 3 describes the used method. The used data and data preprocessing are described in Section 4. Section 5 explaines how the model training is performed. In Section 6, we introduce the performed experiments, including the objective evaluation, a user study and correlation analysis. We present and discuss the results in Section 7 and conclude in Section 8.

2. RELATED WORK

Self-supervised learning has been a hot topic in recent years, with well-known works in the visual domain, like MoCo [7] and SimCLR [8] that adopt contrastive learning approaches, and BYOL [9], or VICReg [10] that aim to eliminate negative samples. Examples for contrastive approaches in the speech domain are Wav2Vec [11], Wav2Vec 2.0 [12], as well as SpeechSimCLR [13]. Inspired by these works, self-supervised learning in non-speech (i.e., musical and

[1] https://www.izotope.com/
[2] https://algonaut.audio/
[3] https://magenta.tensorflow.org/studio/
[4] https://jamahook.com/


© Stefan Lattner. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Stefan Lattner, "SampleMatch: Drum Sample Retrieval by Musical Context", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

environmental) audio was also introduced with contrastive approaches [14,15], Audio2Vec [16] and Wav2Vec for non-speech audio using a conformer architecture [17]. An example for self-supervised learning for audio based on the BYOL method is [18]. Other audio-related contributions based on contrastive learning are multimodal contrastive learning (for audio and video) [19], as well as multi-format contrastive learning (between raw audio and spectrogram representations) [20]. We adopt ideas from self-supervised learning methods mentioned above, using a dictionary look-up approach like in MoCo [7], we adopt the variance- and co-variance regularization of VICReg [10], and we use decoupled contrastive learning, as proposed in [21].

Other works on drum sample retrieval involve the contrastive learning-based retrieval of single drum samples by their mixed versions [22] and retrieval by vocalization [23, 24]. Other academic works that tackle the task of matching artistic content based on *aesthetic principles* are the Neural Loop Combiner [25], and a stem mashup creation approach [26] (both based on contrastive learning). Other methods for computational mashup creation are based on optimization [27] or expert systems [28].

3. METHOD

3.1 Scoring Function

The goal of our proposed method is to estimate how well a given drum sample fits a specific musical context. This task can be modeled by contrastive learning as a dictionary look-up task, as described in [7]. A musical context $\mathbf{q}_i \in \mathbb{R}^m$ acts as a query, and drum samples $\mathbf{k}_j \in \mathbb{R}^m$ as keys. We define a scoring function $s(\mathbf{q}_i, \mathbf{k}_j)$ that outputs a high score if the sample fits the query. To that end, we use two encoders g_q and g_k to produce a query encoding $\mathbf{u}_i = g_q(\mathbf{q}_i), \mathbf{u}_i \in \mathbb{R}^d$ and a sample encoding $\mathbf{v}_j = g_k(\mathbf{k}_j), \mathbf{v}_j \in \mathbb{R}^d$. Finally, we define the scoring function as the cosine similarity $\mathrm{sim}(\cdot,\cdot)$ between the resulting encodings $s(\mathbf{q}_i, \mathbf{k}_j) = \mathrm{sim}(\mathbf{u}_i, \mathbf{v}_j)$. We train the encoders using a contrastive loss called NT-Xent in [8]:

$$\mathcal{X}(Z) = -\log \frac{\exp\left(\mathrm{sim}(\mathbf{u}_i, \mathbf{v}_j)/\tau\right)}{\sum_{l \neq j} \exp\left(\mathrm{sim}(\mathbf{u}_i, \mathbf{v}_l)/\tau\right)}, \qquad (1)$$

where $\{\mathbf{u}_i, \mathbf{v}_j\}$ is a positive pair, $Z \in \mathbb{R}^{n \times d}$ are all representations of a training batch, τ is the temperature parameter, and we adopt the decoupled contrastive learning variant, that has shown to work better for smaller batch sizes, by removing the positive pair from the denominator (i.e., $l \neq j$) [21].

3.2 Regularizations

Furthermore, we combine the contrastive loss with the variance and covariance regularization used in VICReg [10]. The variance regularization term is defined as a hinge function that penalizes variances of latent features along the batch dimension that are smaller than 1 as

$$\mathcal{V}(Z) = \frac{1}{d} \sum_{j=1}^{d} \max\left(0, 1 - S(\mathbf{z}_{:,j}, \epsilon)\right), \qquad (2)$$

where Python slicing notation is used, and S is the regularized standard deviation

$$S(x, \epsilon) = \sqrt{\mathrm{Var}(x) + \epsilon}. \qquad (3)$$

The covariance regularization penalizes non-zero off-diagonal entries in the covariance matrix of each batch, leading to a decorrelation of the latent dimensions:

$$\mathcal{C}(Z) = \frac{1}{d} \sum_{i \neq j} [C(Z)]_{i,j}^2, \qquad (4)$$

where C is the covariance matrix

$$C(Z) = \frac{1}{d-1} \sum_{i=1}^{d} (\mathbf{z}_{:,i} - \bar{z}_i)(\mathbf{z}_{:,i} - \bar{z}_i)^T, \; \bar{z}_i = \frac{1}{n} \sum_{j=1}^{n} z_{j,i}. \qquad (5)$$

Even though the cosine distance renders the norms of the network outputs irrelevant, they can still grow due to optimization dynamics. To keep the norms in a moderate range, we also add a norm regularization in form of a hinge function as

$$\mathcal{N}(Z) = -\frac{1}{n} \sum_{i=1}^{n} \min\left(0, c - ||\mathbf{z}_i||\right), \qquad (6)$$

where we fix $c = 4$ in our experiments (because most norms are smaller than 4 at initialization). Putting all the above terms together (and weighting the regularization terms with factors γ, δ and η), yields the final loss

$$\mathcal{L}(Z) = \mathcal{X}(Z) + \gamma \mathcal{V}(Z) + \delta \mathcal{C}(Z) + \eta \mathcal{N}(Z). \qquad (7)$$

4. DATA

We used a dataset of electronic music and pop/rock songs of 44.1 kHz sample rate for training and evaluation. The electronic music portion consists of 4830 so-called "remix packs", with durations ranging from several seconds to several minutes. Each remix pack consists of multiple audio tracks that contain the individual (mutually aligned) instruments (such as synth, bass, guitar, pad, strings, choir, brass, keyboard, vocals, and different percussion instruments).

The Pop/Rock dataset is a proprietary dataset of 885 well-known rock/pop songs of the past decades, including artists such as Lady Gaga, Coldplay, Stevie Wonder, Lenny Kravitz, Metallica, AC/DC, and Red Hot Chili Peppers. The stems (guitar, vocals, bass guitar, keyboard, misc, and different percussion instruments) are available as individual audio tracks for each song.

From every percussion track in the dataset, we extract so-called "one-shots", single hits with the respective percussion instrument. To that end, we picked those samples where the subsequent onset is of maximal distance (to retain most of the percussion's potential reverb). From the pop/rock songs, we extract 5 one-shots per percussion track, while from the electronic songs, we extract 1 one-shot per percussion track (as most of the time they are all identical). Altogether, this results in 63042 one-shot drum samples.

We categorize the extracted drum samples into 6 categories which are {kick, snare, hi-hat, ride, crash, toms}. The assignment is based on the filenames of the stems the samples originate from, using keyword dictionaries of typical drum-type expressions (like "hat" for hi-hat or "BD" for kick drum). We split the dataset in train / eval / test set with proportions $0.85/0.05/0.1$ (on a track level, meaning that samples of one track do not spread over different sets). All reported results are computed on the test set.

5. TRAINING

As an encoder, we use the EfficientNet-B4 [29], where we start training from weights that have been pre-trained on the ImageNet dataset. Even though ImageNet is far from the audio spectrum domain, it turned out that using ImageNet weights is crucial in our experiments (see Section 7.1, and it has also been shown in a previous study that cross-domain pre-training can be beneficial [30]).

The 1000 output units are reduced to 256 with a single linear layer. The EfficientNet expects a 2D input, which we provide by converting the audio signals into mel-scaled spectrograms with an STFT window length of 2048, a hop length of 512, and 128 resulting mel bins, considering the whole frequency range (fmax $=$ 22050). Also, we log-scale the values of the resulting mel spectrograms. We input the resulting spectrograms in each of the three RGB input channels of the EfficientNet (after normalization to the ImageNet color statistics).

A positive pair consists of an audio query and a corresponding drum sample. Given a drum sample, we select the stems of the corresponding song and remove the stem from which the drum sample was extracted. Then, we randomly choose at least 2, at most all of the remaining stems (the number of stems is uniformly sampled) and mix them on the fly during training (note that the mel spectrogram transformation is part of the model architecture using the *nnAudio* library). [5] All audio inputs are of length 4 seconds. If a drum sample is shorter than that, it is zero-padded. To obtain a query, we are cutting a 4-second-long snippet from a random position of the mixture.

The encoders are trained by the ADAM optimizer, with a batch size of 190, a learning rate of 3e-4, and a weight decay factor of 3e-5. The temperature parameter τ of the NT-Xent loss is set to 0.2. The factors for the variance and covariance regularization terms γ and δ are set to 1, and the norm regularization factor η is set to 10.

We use some data augmentation on both the queries and the drum samples. First, Gaussian noise is added (up to -12 dB). Furthermore, we perform time-stretch with a ratio between 0.9 and 1.15 of the original tempo. Another augmentation is reducing the gain down to a minimum attenuation of -6 dB, and we perform a time shift of up to 800ms to the right and 200ms to the left (to make the models invariant to the exact onset position of one-shot audios). All these augmentations occur with a probability of 50% for each instance.

[5] `https://github.com/KinWaiCheuk/nnAudio`

6. EXPERIMENTS

In this section, we introduce the conducted experiments to validate our method including objective metrics (Section 6.1), the user study (Section 6.2), and we describe the method of the correlation analysis (Section 6.3).

6.1 Objective Evaluation

As the actual goal of the study is to sort the candidate samples given an audio query $\mathbf{q}_i$, the primary evaluation metric is the rank_i of the ground-truth samples for a given query (i.e., the samples that originate from the same song as the query). For that, we sort all drum samples in descending order according to the cosine similarity to a given encoded query and determine how early in the list a ground-truth sample is located. We also divide the rank_i by N list entries to obtain normalized ranks $\in (0, 1]$. As we can extract several queries (4-second-long random excerpts) for each song in the test set, and as there are several drum samples assigned to each song, we perform this test for every song in the test set 50 times (resulting in $|Q| = 27$k evaluations) and average the resulting ranks. The explanation above results in the Mean Normalized Rank summarized as

$$R_{\text{mn}} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{\text{rank}_i}{N}. \tag{8}$$

In addition, we also report the *Median* Normalized Rank R_{md}, as this is an indicator of how well the model performs *most of the time*, ignoring possible outliers.

Besides also reporting the contrastive loss $\mathcal{X}$ (see Equation 1), we also evaluate how well the model has learned to cluster electronic and acoustic queries (i.e., mixtures) and keys (i.e., drum samples). To that end, we perform a binary (acoustic/electronic) k-nn classification (with $k = 50$) on the encodings separately for both the queries and the keys. From this classification task, we report the likelihood (L_q and L_k, respectively) of the data under the predictions.

In order to better understand the influence of different design choices, we perform ablation studies and report the metrics for each model and training variants. The different ablation scenarios are described in the following.

- `2Enc` denotes our proposed setup with 2 separate encoders instead of a shared encoder for both, queries and samples.
- `PTrain` denotes a variant that starts training from an EfficientNet that was pre-trained on ImageNet, instead of from randomly initialized weights.
- In `Aug`, data augmentation is used.
- `VCReg`, is using the variance and co-variance regularization terms (see Equations 2 and 4).
- In `SMix`, we randomly mix different numbers n of random stems to form a query (where $n > 1$). If `SMix` is not used, we always mix all stems of a song.

- For `QSInv`, in addition to sampling positive pairs from queries and corresponding samples, we also include as positive pairs two queries and two samples originating from the same song.

6.2 User Study

To evaluate the quality of the scoring function, we perform a user study in which we ask 10 experts (with musical education or concerned with music production) to rate the quality of selected drum samples given a musical context. More precisely, we test their preference between drum samples that scored well according to our system and samples that are randomly picked from the dataset. The possible choices are to select one of the two proposed mixtures, or "equal" (if there is no preference for one of the choices), or "skip" (if the samples to be rated are not of the expected class or any other problem occurred). Every participant obtains 12 single comparisons (presenting each of the 6 percussion types twice) and provides 12 ratings accordingly. After that, they can decide if they want to enter a further session (to provide 12 further ratings). Participants are asked not to perform more than 4 sessions to obtain a balanced result.

The procedure to obtain one comparison presented to a participant for rating is as follows. We pick a percussion type (e.g., snare) and a song from the test set that contains such a type as a single stem. We remove the stem containing that type, mix the remaining stems and pick ten 4-second-long excerpts from randomly sampled positions of the resulting mixture. All these excerpts are then fed through the query encoder, and the mean is taken from the resulting encodings. The resulting mean vector acts as the query encoding for the current song. Using this query, the cosine similarities to the latent representations of all 63k drum samples in the data set (obtained as described in Section 6) are computed, and the ten samples with the *highest similarities* are selected. Furthermore, ten drum samples of the corresponding percussion type are *randomly picked* from the data set. The thereby obtained samples are used to generate 20 new versions of the current song (a version for each sample).

To create a new version of a song, we perform an onset detection on the previously removed stem (containing the percussion type in question), position the selected samples in the estimated positions, and mix the resulting stem with the remaining stems (omitting the original stem of the percussion type in question). That way, we replace the original drum track with a track containing the drum sample to be evaluated. For a single user rating, we then contrast the mixtures of a *randomly picked* and a *highly rated* percussion sample and have participants indicate their preference between the two.

6.3 Correlation Analysis

In order to gain some insights into the latent space learned by the encoders, we perform a correlation analysis between audio features of neighboring data points. For that, different perceptual and spectral features are computed for the drum samples in the data set using the *Audio Commons timbre models* [6] for perceptual features (e.g., boominess, brightness, depth, hardness, roughness, warmth) and *librosa* [7] for spectral features (e.g., spectral centroid or spectral contrast). We also add an indicator if the respective drum sample originates from an electronic or acoustic song. In the former case, the "electronic" feature value is set to 1, in the latter case to 0.

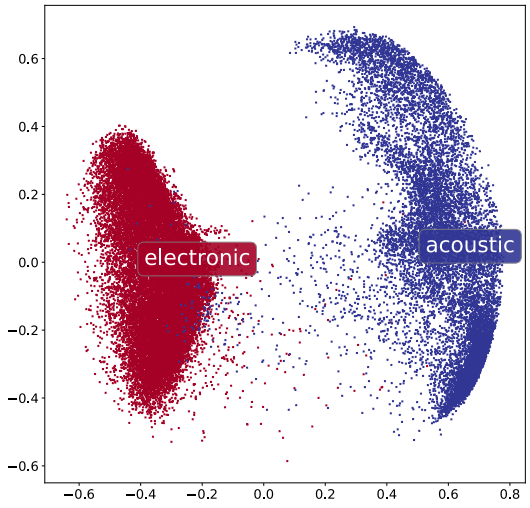


Figure 1: Principal Component Analysis (PCA) of drum sample encodings. Red dots indicate samples originating from electronic music and blue dots indicate samples originating from acoustic music.

The drum samples are mapped into the sample encoder's latent space, and k-means clustering is performed using $k = 24$. In each cluster, the mean of each audio feature is used as a specific observation of that audio feature variable. Then, the Pearson correlation coefficient is computed between all such variables (separately for every percussion type). As a result, we can derive statements like "whenever the loudness of a kick drum is high, the boominess of the snare tends to be low". Note that for computing the audio features, we normalize every sample to 0.5 seconds (i.e., cut or pad) so that the audio feature computation is not influenced by the length of the audio file (as some features are computed by averaging several spectrogram frames). To obtain the latent encodings used to compute the clusters, we use the regular 4-second-long samples.

7. RESULTS AND DISCUSSION

7.1 Objective Evaluation

In Table 1, we report the contrastive loss, the Mean and Median Normalized Rank, and the query and sample electronic/acoustic classification likelihood for different archi-

[6] `https://github.com/AudioCommons/ac-audio-extractor`

[7] `https://librosa.org/`

	Queries: full mixtures					Queries: sparse mixtures				
Variant	$\mathcal{X}$	R_{mn}	R_{md}	L_q	L_k	$\mathcal{X}$	R_{mn}	R_{md}	L_q	L_k
`2Enc+PTrain+Aug+VCReg+SMix`	3.614	**0.105**	**0.032**	0.9940	0.9767	**3.761**	**0.124**	0.043	**0.9905**	0.9763
`2Enc+PTrain+Aug+VCReg+SMix+QSInv`	3.718	0.120	0.037	0.9900	0.9782	3.818	0.136	0.047	0.9862	0.9768
`2Enc+PTrain+Aug+VCReg`	4.001	0.124	0.061	0.9825	0.9732	4.389	0.183	0.100	0.9635	0.9724
`2Enc+PTrain+VCReg+SMix`	3.742	0.128	0.037	0.9945	**0.9895**	3.780	0.137	**0.042**	0.9901	**0.9898**
`2Enc+PTrain+Aug+SMix`	**3.575**	0.116	**0.032**	**0.9946**	0.9774	3.812	0.135	0.043	0.9893	0.9790
`2Enc+Aug+VCReg+SMix`	5.235	0.458	0.432	0.7268	0.7629	5.237	0.470	0.451	0.7188	0.7480
`2Enc+Aug+VCReg+SMix+QSInv`	4.174	0.164	0.079	0.9826	0.9566	4.387	0.181	0.091	0.9768	0.9585
`PTrain+Aug+VCReg+SMix`	3.853	0.121	0.047	0.9812	0.9809	4.000	0.137	0.058	0.9768	0.9819
`2Enc+PTrain`	3.883	0.140	0.053	0.9925	0.9795	4.399	0.205	0.089	0.9821	0.9803

Table 1: Ablation study for different architectures and training scenarios tested on queries from full mixtures and queries from sparse mixtures (a sparse mixture is based on a random number n of stems, where $n > 1$). $\mathcal{X}$ denotes the contrastive loss and R_{mn} and R_{md} is the mean and median normalized rank, respectively, of ground truth samples. L_q is the likelihood of the binary k-nn (electronic/acoustic) classification task for query, and L_k for key encodings in the latent space.

tectures and training scenarios (as described in Section 6.1). For each run, we train for 600 epochs and pick the saved checkpoint of the epoch with the best performance (regarding the Mean Normalized Rank R_{mn}) on the evaluation set for evaluating the test set. We report the results when evaluating the models with queries that are drawn from mixtures where all stems of a song are used (denoted as "full mixtures") and mixtures with a random number n of stems (denoted as "sparse mixtures", where $n > 1$). The latter scenario is crucial in music production, where sample selection may occur at an intermediate state of the project, when several instruments are still missing.

The results show that our proposed architecture and training procedure (first row in Table 1) performs best in the mean and median rank metrics (R_{mn} and R_{mn}) for both query scenarios (note that the random guessing baseline is 0.5). It is somewhat surprising to us that `QSInv` (i.e., imposing additional invariance for queries/samples originating from the same song) does not improve but rather worsens the result (cf. row 2 of the table). Apparently, the relaxation of the space caused by not using this regularization helps to perform the main objective.

We can see for `SMix` that it is vital to mix different (numbers of) stems on the fly during training (cf. third row in Table 1 where this has not been done). `SMix` does not only help for the "sparse mixtures" scenario at test time (right part of the table) but also in the "full mixtures" scenario (left part of the table).

Using a pre-trained EfficientNet (`PTrain`) makes a considerable difference in the overall performance and training dynamics. In fact, when not using pre-trained weights, it is necessary to add `QSInv` in our experiments. Otherwise, the encoders do not learn at all (see row 6 in Table 1).

The last row of Table 1 presents the results of a scenario where all training optimizations (augmentation, variance, co-variance regularization, and sparse mixing) are turned off. It can be seen that the performance deteriorates substantially, particularly in the "sparse mixtures" test scenario.

Finally, the results of the classification likelihood (mostly around 99%) show that the models have implicitly learned to separate electronic and acoustic content (cf. Figure 1, showing a PCA of the drum sample encodings). Interestingly, this separation works consistently better if no data augmentation is used (see row 4 in Table 1). Electronic samples are generally "cleaner" than those extracted from acoustic music. Augmentation (e.g., adding noise) may remove some of these characteristics, making it harder to discriminate between the two classes.

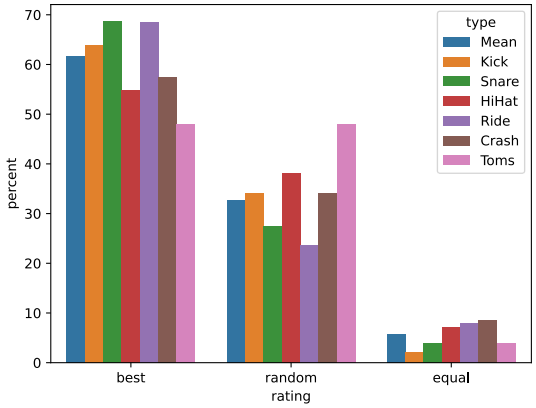


Figure 2: Preference ratings of participants in the user study, separated by percussion type (the blue bar "Mean" shows the mean of all ratings). "best" are mixtures with samples that *scored highest* by our method, and "random" denotes mixtures with *random samples* from the data set. An "equal" rating means no particular preference.

7.2 User Study

Figure 2 shows the results of the listening test (based on 300 ratings in total, omitting skipped ratings). On average ("Mean"), the samples that are ranked "best" by our method were preferred approximately twice as often as random samples (61.57% to 32.64%). It is also interesting that for most percussion types, the human preference for "best" is similar to or better than average (with Snare and Ride being the most salient), while human ratings disagree with our system's choices, particularly often for HiHat and Toms. A hypothesis why HiHats perform worse is that we merged both open and closed hi-hat in one group. When replacing the original hi-hats with the selected ones, there

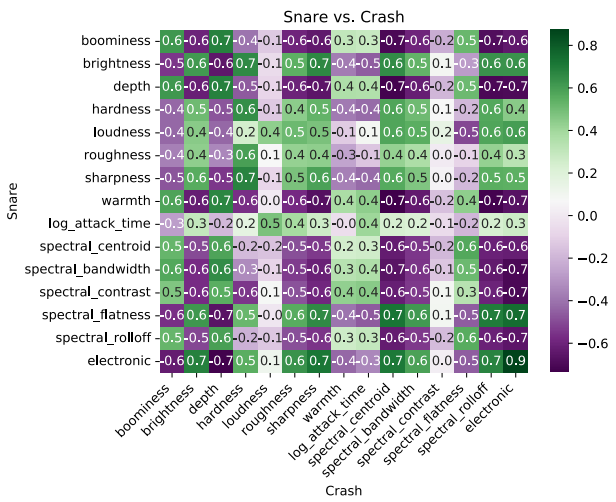


Figure 3: Correlations between perceptual and spectral features (and electronic / acoustic indicator) of Snare and Crash drum samples that are close in the latent space (i.e., scored to fit well in the same musical context).

is an equal chance that an open hi-hat is used for a rhythm that is meant to be instantiated with a closed hi-hat and vice versa. This is particularly problematic in electronic music, where closed hi-hats are often used with minimal temporal intervals. The reason why Toms do not work well is probably due to less available training data (many tracks do not have tom stems). Examples of how the user study was presented can be found on the accompaniment website. [8]

7.3 Correlation Analysis

Figure 3 shows the correlation coefficients between audio features of snare and crash drum samples over clusters in the latent space (as described in Section 6.3, note that the entirety of the percussion-type combinations is shown on the accompaniment website). [8]

To our knowledge, there is no theory about the aesthetic rules of drum sample selection. Therefore, this analysis is particularly informative. It is interesting to see that strong correlations exist at all between the audio features of snare and crash found in the same clusters of the latent space (as is true for most other percussion-type combinations). Consequently, the characteristics that make drum samples fit in the same musical contexts can (also) be explained by such lower-level features. Using such correlation coefficients makes it possible to understand how the learned space is organized. More importantly, it is also possible to derive rules and make the method explainable. For example, it is possible to extract statements like "if the snare of a song sounds warm, the crash should not sound bright" (because "warmth" of the snare and "brightness" of the crash are negatively correlated with a coefficient of -0.6).

When looking at the diagonal of the correlation matrix in Figure 3, we see that the perceptual features tend to be positively correlated (from "boominess" to "warmth"), while the spectral features tend to be negatively correlated (from "log attack time" to "spectral rolloff"). Possibly, as snare and crash have similar perceptual characteristics, they should not get into each other's way regarding their frequency ranges. Whenever the snare occupies the higher frequencies, the crash occupies the lower frequencies and vice versa (according to the "spectral centroid" entry of -0.6 in the diagonal).

Finally, the correlations with the "electronic" indicator are also informative. It shows that a snare (and partly a crash) in electronic music tends to be more intense than in acoustic music, with more brightness, loudness, and sharpness. At the same time (looking at the spectral centroid), in electronic music, a snare tends to occupy rather lower frequencies than in acoustic music, while the opposite is true for crash (where the spectral centroid is positively correlated with electronic snares - having a correlation coefficient of 0.7). The high correlation of "electronic" (0.9) in the diagonal shows that electronic snares and crashes tend to be in the same clusters of the latent space.

8. CONCLUSION AND FUTURE WORK

We introduced a method that automatically scores the fit of drum samples to a given musical context. As the method is thought to be used in a music production context, the audio queries used for training are based on "sparse mixtures", which allows scoring samples at different stages of the music production process. Results show that this form of data augmentation generally improves performance, also when queries are computed from complete mixtures at test time. Critically, the automatic system should agree with human judgment. Therefore, we performed a user study that tests this agreement. It could be shown that the drum samples that are highly rated by our system are also preferred by human experts approximately twice as often as randomly selected samples.

As the architecture and training procedure combine different ideas, we performed an ablation study, where it became clear that starting with a pre-trained model is crucial for a good final performance and that additional choices (like using variance and co-variance regularization, data augmentation, and on-the-fly sparse mixing) improve the results considerably. It was also shown that audio features of drum samples whose encodings are close in the latent space are highly correlated, and we demonstrated that the observed correlations can also be interpreted to derive rules.

Our proposed method is not limited to drum samples but can potentially be used to retrieve different kinds of musical material based on learned aesthetic principles. Also, in the future, we want to scale up the system by using bigger data sets and a more generic pre-processing strategy. Currently, single drum samples are extracted before training to obtain a controlled experiment setup and application scenario. Developing a system that uses stems directly would apply better to audio sources of any type and would therefore greatly increase the applicability of our method in music production.

[8] https://sites.google.com/view/samplematch

9. ACKNOWLEDGEMENTS

I want to thank the reviewers for their extraordinarily thorough suggestions to improve this work and my colleagues at Sony CSL that assisted me with technical expertise and emotional support.

10. REFERENCES

[1] E. Deruty, M. Grachten, S. Lattner, J. Nistal, and C. Aouameur, "On the development and practice of AI technology for contemporary popular music production," *Trans. Int. Soc. Music. Inf. Retr.*, vol. 5, no. 1, p. 35, 2022. [Online]. Available: https://doi.org/10.5334/tismir.100

[2] J. Nistal, C. Aouameur, I. Velarde, and S. Lattner, "Drumgan VST: A plugin for drum sound analysis/synthesis with autoencoding generative adversarial networks," in *Proceedings of the 39th International Conference on Machine Learning, Baltimore, Maryland, USA*, 2022. [Online]. Available: https://arxiv.org/pdf/2206.14723.pdf

[3] J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *8th International Conference on Learning Representations, ICLR, Addis Ababa, Ethiopia, April 26-30*, 2020. [Online]. Available: https://openreview.net/forum?id=B1x1ma4tDr

[4] T. Bazin, G. Hadjeres, P. Esling, and M. Malt, "Spectrogram inpainting for interactive generation of instrument sounds," in *Joint Conference on AI Music Creativity*, 2020. [Online]. Available: https://arxiv.org/abs/2104.07519

[5] S. Lattner and M. Grachten, "High-level control of drum track generation using learned patterns of rhythmic interaction," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA, New Paltz, NY, USA, October 20-23*, 2019, pp. 35–39. [Online]. Available: https://doi.org/10.1109/WASPAA.2019.8937261

[6] G. Hadjeres and L. Crestel, "The piano inpainting application," *CoRR*, vol. abs/2107.05944, 2021. [Online]. Available: https://arxiv.org/abs/2107.05944

[7] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick, "Momentum contrast for unsupervised visual representation learning," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, Seattle, WA, USA, June 13-19*, 2020, pp. 9726–9735. [Online]. Available: https://doi.org/10.1109/CVPR42600.2020.00975

[8] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning, ICML, 13-18 July*, 2020, pp. 1597–1607. [Online]. Available: http://proceedings.mlr.press/v119/chen20j.html

[9] J. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Á. Pires, Z. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent - A new approach to self-supervised learning," in *Annual Conference on Neural Information Processing Systems, NeurIPS, December 6-12*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html

[10] A. Bardes, J. Ponce, and Y. LeCun, "VICReg: Variance-invariance-covariance regularization for self-supervised learning," *CoRR*, vol. abs/2105.04906, 2021. [Online]. Available: https://arxiv.org/abs/2105.04906

[11] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," in *Interspeech, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September*, 2019, pp. 3465–3469. [Online]. Available: https://doi.org/10.21437/Interspeech.2019-1873

[12] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Annual Conference on Neural Information Processing Systems, NeurIPS, December 6-12*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/92d1e1eb1cd6f9fba3227870bb6d7f07-Abstract.html

[13] D. Jiang, W. Li, M. Cao, W. Zou, and X. Li, "Speech SimCLR: Combining contrastive and reconstruction objective for self-supervised speech representation learning," in *Interspeech, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September*, 2021, pp. 1544–1548. [Online]. Available: https://doi.org/10.21437/Interspeech.2021-391

[14] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, Toronto, ON, Canada, June 6-11*, 2021, pp. 3875–3879. [Online]. Available: https://doi.org/10.1109/ICASSP39728.2021.9413528

[15] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR, Online, November 7-12*, 2021, pp. 673–681. [Online]. Available: https://archives.ismir.net/ismir2021/paper/000084.pdf

[16] M. Tagliasacchi, B. Gfeller, F. de Chaumont Quitry, and D. Roblek, "Pre-training audio representations with self-supervision," *IEEE Signal Process. Lett.*,

vol. 27, pp. 600–604, 2020. [Online]. Available: https://doi.org/10.1109/LSP.2020.2985586

[17] S. Srivastava, Y. Wang, A. Tjandra, A. Kumar, C. Liu, K. Singh, and Y. Saraf, "Conformer-based self-supervised learning for non-speech audio tasks," *CoRR*, vol. abs/2110.07313, 2021. [Online]. Available: https://arxiv.org/abs/2110.07313

[18] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "BYOL for audio: Self-supervised learning for general-purpose audio representation," in *International Joint Conference on Neural Networks, IJCNN, Shenzhen, China, July 18-22*, 2021, pp. 1–8. [Online]. Available: https://doi.org/10.1109/IJCNN52387.2021.9534474

[19] L. Wang, P. Luc, A. Recasens, J. Alayrac, and A. van den Oord, "Multimodal self-supervised learning of general audio representations," *CoRR*, vol. abs/2104.12807, 2021. [Online]. Available: https://arxiv.org/abs/2104.12807

[20] L. Wang and A. van den Oord, "Multi-format contrastive learning of audio representations," *CoRR*, vol. abs/2103.06508, 2021. [Online]. Available: https://arxiv.org/abs/2103.06508

[21] C. Yeh, C. Hong, Y. Hsu, T. Liu, Y. Chen, and Y. LeCun, "Decoupled contrastive learning," *CoRR*, vol. abs/2110.06848, 2021. [Online]. Available: https://arxiv.org/abs/2110.06848

[22] W. Kim and J. Nam, "Drum sample retrieval from mixed audio via a joint embedding space of mixed and single audio samples," in *Audio Engineering Society Convention 149*. Audio Engineering Society, 2020.

[23] A. Delgado, C. Saitis, E. Benetos, and M. B. Sandler, "Deep conditional representation learning for drum sample retrieval by vocalisation," *CoRR*, vol. abs/2204.04651, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2204.04651

[24] A. Mehrabi, K. Choi, S. Dixon, and M. B. Sandler, "Similarity measures for vocal-based drum sample retrieval using deep convolutional auto-encoders," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, Calgary, AB, Canada, April 15-20*, 2018, pp. 356–360. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8461566

[25] B. Chen, J. B. L. Smith, and Y. Yang, "Neural loop combiner: Neural network models for assessing the compatibility of loops," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR, Montreal, Canada, October 11-16*, 2020, pp. 424–431. [Online]. Available: http://archives.ismir.net/ismir2020/paper/000225.pdf

[26] J. Huang, J. Wang, J. B. L. Smith, X. Song, and Y. Wang, "Modeling the compatibility of stem tracks to generate music mashups," in *Thirty-Fifth AAAI Conference on Artificial Intelligence, February 2-9*, 2021, pp. 187–195. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16092

[27] G. Bernardo and G. Bernardes, "Leveraging compatibility and diversity in computational music mashup creation," in *AM '21: Audio Mostly, Virtual Event / Trento, Italy, September 1-3*, 2021, pp. 248–255. [Online]. Available: https://doi.org/10.1145/3478384.3478424

[28] M. E. P. Davies, P. Hamel, K. Yoshii, and M. Goto, "AutoMashUpper: Automatic creation of multi-song music mashups," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 22, no. 12, pp. 1726–1737, 2014. [Online]. Available: https://doi.org/10.1109/TASLP.2014.2347135

[29] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning, ICML, Long Beach, California, USA, 9-15 June*, 2019, pp. 6105–6114. [Online]. Available: http://proceedings.mlr.press/v97/tan19a.html

[30] A. Guzhov, F. Raue, J. Hees, and A. Dengel, "ESResNet: Environmental sound classification based on visual domain models," in *25th International Conference on Pattern Recognition, ICPR, Virtual Event / Milan, Italy, January 10-15*, 2020, pp. 4933–4940. [Online]. Available: https://doi.org/10.1109/ICPR48806.2021.9413035

A TRANSFORMER-BASED "SPELLCHECKER" FOR DETECTING ERRORS IN OMR OUTPUT

Timothy de Reuse **Ichiro Fujinaga**
Centre for Interdisciplinary Research in Music Media and Technology, McGill University
{timothy.dereuse, ichiro.fujinaga}@mcgill.ca

ABSTRACT

The outputs of Optical Music Recognition (OMR) systems require time-consuming human correction. Given that most of the errors induced by OMR processes appear "non-musical" to humans, we propose that the time to correct errors may be reduced by marking all symbols on a score that are musically unlikely, allowing the human to focus their attention accordingly. Using a dataset of Romantic string quartets, we train a variant of the Transformer network architecture on the task of classifying each symbol of an optically-recognized musical piece in symbolic format as correct or erroneous, based on whether a manual correction of the piece would require an insertion, deletion, or replacement of a symbol at that location. Since we have a limited amount of data with real OMR errors, we employ extensive data augmentation to add errors into training data in a way that mimics how OMR would modify the score. Our best-performing models achieve 99% recall and 50% precision on this error-detection task.

1. INTRODUCTION

An enormous amount of music scores exist only in the form of digital images, where its musical content is not machine-readable. Among the fields of research that that assist with large-scale processing of musical documents is Optical Music Recognition (OMR), which investigates how to transform images of music scores into symbolic music files (e.g., MusicXML files) that can be searched, played back, and edited. OMR has the potential to be a valuable tool for music archives and libraries, but it is not yet reliable enough; all methods require a human correction step after the OMR process, as the raw outputs of currently-available OMR algorithms contain too many errors to be acceptable for musicians or music researchers. This is an issue when using OMR on printed Western music notation, and is worse for other types of music notation, especially when training data is scarce. While custom interfaces exist for the purpose of facilitating human correction, such as the correction interface of MuRET [1], this is still a time-consuming task even for experts. While OMR speeds up symbolic encoding in many cases [2], the process of scanning documents, performing OMR, and correcting OMR is not always faster than that of transcribing pieces from scratch on complex, polyphonic scores [3].

We observe that OMR systems tend to introduce musically unlikely phenomena into scores. A typical example of the errors induced by the OMR process of PhotoScore [1] (a leading proprietary, commercial OMR software) is illustrated in Figure 1. In the OMR'ed score, consider the pattern of missing staccato articulations in measure 1, the eighth note misidentified as a sixteenth note in measure 2, the strangely placed grace note in measure 3, and the accidental misidentified as a grace note in measure 6. Also note how the slurs in each instrument of the OMR'ed score do not match up with each other, though the correct phrasing could probably be guessed from the melodic content alone. While not all of these phenomena are necessarily impossible in a string quartet, a human looking over the score would notice their presence.

Given the musical "incorrectness" of many of the errors introduced by OMR, we propose that a machine-learning algorithm could flag most OMR errors automatically given information on the conventions of the genre. This could reduce the correction time of a score; instead of checking every measure of OMR output against the corresponding measure in the score image, the user would only have to check those regions flagged by the algorithm as possibly erroneous. Even if the algorithm flags some non-erroneous regions of the score, it could potentially still exclude large portions of it from needing human review.

The reader may question why we aim for *detection* of errors as opposed to *correction* of errors outright; this could further reduce the time to correct OMR output even further, and there exists a large body of work on error correction for natural language that we could draw on. We posit, however, that any amount of correction that is not guaranteed to address nearly *all* errors in an OMR output would not significantly save time on the part of the human corrector, as they would still be required to manually review the result anyway. Current studies on correcting errors in polyphonic music (discussed in Section 2) do not attain high enough performance to meet this bar.

© Timothy de Reuse, Ichiro Fujinaga. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Timothy de Reuse, Ichiro Fujinaga, "A Transformer-Based "Spellchecker" for Detecting Errors in OMR Output", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] /www.neuratron.com/photoscore.htm

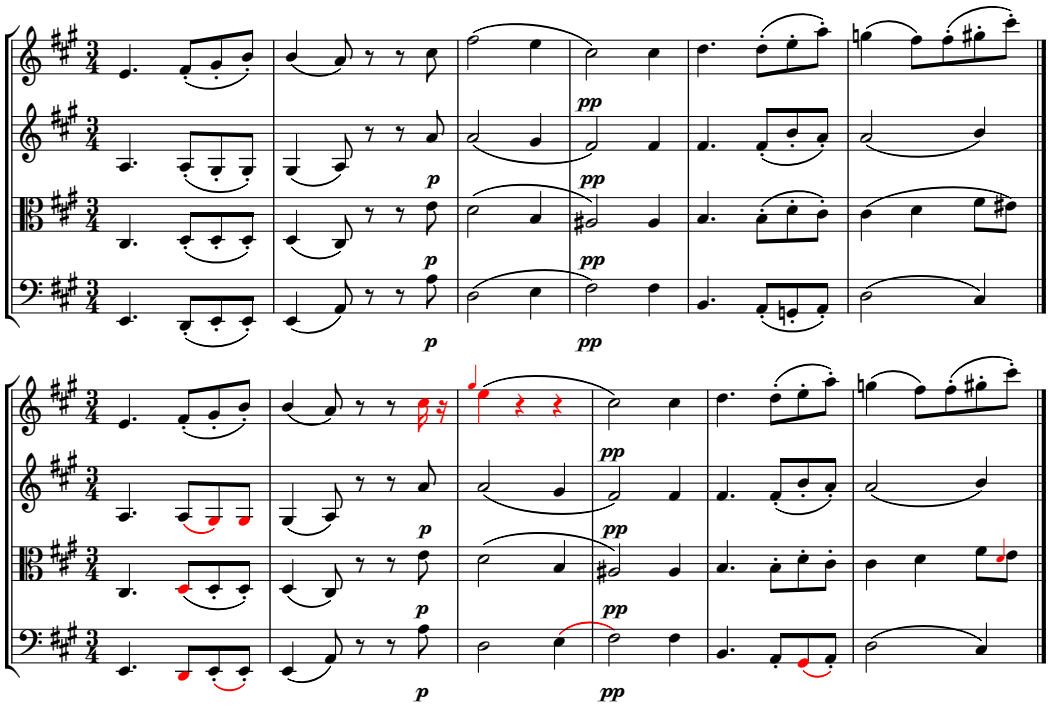


Figure 1: Two versions of Felix Mendelssohn's Quartet No. 2 in A Major, Op. 13, mm. 4-9. The upper system is an engraving of the correct score. The lower system is an output from PhotoScore's OMR Process. Some prominent errors are highlighted in red.

2. PREVIOUS WORK

There is little other research that seeks to detect or correct errors in symbolic music in the context of OMR. Most studies that discuss errors made by OMR systems seek to characterize them as a part of a method of evaluating their performance [4, 5]. As part of an end-to-end OMR system, Rossant and Blach [6] explicitly encode some musical rules (such as ensuring the number of beats in a measure matches the most recent time signature) that let them highlight areas that are likely incorrect, and also use confidence measures of detection steps from earlier in their OMR process to highlight areas that may have been detected incorrectly. More research focuses on error correction in music for other applications. McLeod et al. [7] focus on the task of analyzing errors in the output of Automatic Music Transcription (AMT) methods, introducing rule-based and machine leaning-based models to solve error detection and correction tasks, though these are intended as proof-of-concept baselines to define the tasks. Ycart et al. [8] compare a number of machine-learning models on the closely related task of Symbolic Music transduction, which involves processing the raw per-note probabilities output by an AMT model and producing a valid symbolic music file as output. Sidorov et al. [9] analyse music using formal grammars, taking the assumption that "correct" music tends to be highly compressible, and so any note whose presence alone significantly increases the amount of information in a piece is likely to be an error.

Significantly more literature exists in the fields of Grammar Error Detection (GED) and Grammar Error Correction (GEC) in natural language. The fields are far too broad to cover here; we direct readers to reviews on these subjects by Wang et al. [10], covering GEC, and Madi and Al-Khalifa [11], covering GED. Early models in both of these fields were rule-based, wherein every type of error had to be manually codified [12]. The current state of the art in these areas is in sequence-to-sequence deep-learning models similar to those used for machine translation, reaching human-level performance on common English-language GEC benchmarks [13]. To achieve this level of performance, however, a large amount of training data is required; there is much less research on GEC and GED in low-resource languages, the vast majority being on either English or Chinese [11]. Our task faces a similar issue; there exists orders of magnitude fewer training data for any one genre of music than there exists for the English language.

Data augmentation is commonly used in GEC and GED where data availability is low. Errors are added to correct examples and the models are then directed to correct those errors; we refer to errors created in this way as *synthetic errors* and errors created by the process of interest (e.g. errors induced by an OMR process) as *natural errors*. Early work on this process used simple statistics about distributions and types of errors to add errors semi-randomly [14, 15]. Later work, inspired by back-translation techniques from machine translation, trained models that *add* synthetic errors simultaneously with ones that *remove* them. Htut and Tetreault [16] evaluate a number of these models, finding that they provide significant improvements to training on natural errors when the dataset containing natural errors is small. Notably, they

find these gains in performance even when the synthetic errors are qualitatively different than the natural errors; that is, a GEC method can still perform well even when it is trained on grammatical errors that a human would never make. However, when a large dataset of natural errors is available, the improvements obtained by augmenting with synthetic errors are significantly diminished.

3. METHODOLOGY

In developing our method, we aim for it to have the following qualities:

- **High Recall Rate**: Our detection method should ideally make *no* false negatives; that is, whenever an error is present in its input, it highlights it. It is acceptable for the precision to be significantly lower than the recall as long as the user is assured that no errors lie outside the flagged regions.
- **Localization of Errors**: Within the constraint of having a high recall rate, the method should try to be as specific as possible when pointing out errors.
- **Trainable on a Small Dataset**: Many of the most common use-cases of OMR are on genres that do not have high-quality, large, digitized datasets on which we could train.
- **Takes Long Inputs**: Many errors introduced by OMR are unlikely to be identifiable in isolation from a small snippet of music, and may only be recognized on the basis of comparison with the rest of the piece.

3.1 Data Representation

What is "an error" in a musical score? Rigorously defining this notion requires some procedure to take the "difference" between two scores (in our case, an OMR output and a correct score). The characteristics of the errors that we find will depend heavily on the underlying representation we use. We also face the constraint that whatever representation we use must be suitable for input into a machine-learning model, preferably as a sequence; this disqualifies the use of methods that operate on hierarchically-structured MusicXML data [17, 18].

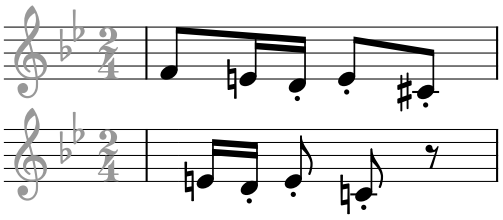
A piano roll-like representation, representing a snippet of music as a two-dimensional image, easily admits a difference operation by way of taking the pixel-wise difference of two two-dimensional piano rolls, but is memory-intensive for long inputs. Another possibility would be to use a token-based representation such as NoteTuple [19], which represents polyphonic music as an ordered list of multi-valued tokens; for example, a piece might be represented as a sequence of triples of the form `(MIDI Pitch, Onset Time, Duration)`. This kind of representation is common in deep learning applications, and there exist many notions of difference on one-dimensional sequences that could be adapted. However, to align with our goal of localizing errors, it would be better for each unit of our representation to contain as little information as possible, so that in calling a single element erroneous we pinpoint the location of the error down to a smaller region in the score.

To accomplish this, and to choose a representation that matches the domain of OMR, we opt to use an *agnostic representation* [20]. An agnostic representation of a musical score lists each of the symbols on a staff without considering their underlying musical meaning; this stands in contrast to *semantic representations* like MusicXML. An agnostic representation encodes less information per sequence element; a dotted eighth note with an accidental would typically be encoded by a single token in a representation like NoteTuple, whereas the accidental, the note, and the dot would each be a separate element in an agnostic representation, and each could be individually flagged as an error.

We write our own utility (included with the source code of this project) to convert Musicxml or Humdrum **kern files into an agnostic encoding by first parsing them with the `music21` package, and then heuristically defining an agnostic encoding that matches the resulting textttmusic21 stream. The encoding lists each symbol on a staff in order from left to right. When two or more symbols are in the same horizontal position (such as when multiple notes sound simultaneously) we list them in ascending order and add a caret symbol ^ between each of them. To handle multiple staves, we interleave them: a measure of the first violin, a measure of the second violin, the viola, the bass, and repeat. This scheme cannot encode all polyphonic musical scores into unique agnostic representations; staff lines where two rhythmically distinct polyphonic voices are notated on the same staff may contain unresolvable ambiguities, where it is not clear which notes belong to which voice. This is not an issue for our task, as we aim not to generate valid semantically-encoded scores from agnostic representations but only to *classify* notes in an existing score.

3.2 Sequence Alignment

To define a difference between two agnostically-encoded musical excerpts, we use the Needleman-Wunsch (NW) sequence alignment algorithm [21]. This algorithm takes as input two sequences and computes the shortest list of operations necessary to turn one into another, where an operation is defined as insertion, deletion, or replacement of a sequence element. By comparing a corrected score with its OMR'ed version, we can see *where* operations must be performed in order to correct it. Our strategy will be to train a model to answer the question: *Given a musical score that contains errors, where would we need to perform operations to correct those errors, according to a NW alignment?* We ignore information on *which* operations need to be performed. Preliminary experiments showed degradation in overall performance when our model was trained to identify the type of error along with its position. We surmise that to a human corrector, reliable detection of the location error itself is more important.



(a)

Original	OMR	Op.
barline.regular	barline.regular	-
8th.pos2.startBeam	–	I
accid.pos1.natural	accid.pos1.natural	-
16th.pos1.continueBeam	16th.pos1.startBeam	R
dot.pos-2	dot.pos-2	-
^	^	-
16th.pos0.endBeam	16th.pos0.endBeam	-
dot.pos-1	dot.pos-1	-
^	^	-
8th.pos1.startBeam	8th.pos1.noBeam	R
accid.pos-1.sharp	accid.pos-1.natural	R
dot.pos-3	dot.pos-3	-
^	^	-
8th.pos-1.endBeam	8th.pos-1.noBeam	R
–	8th.rest	D
barline.regular	barline.regular	-

(b)

Figure 2: (a): A correct score (top) and version with OMR errors (bottom): Mendelssohn's Quartet No. 1 in E♭ Major Op. 12, Mvt. 2, measure 10, 2nd violin part. (b): The Needleman-Wunsch alignment between the agnostic encodings of the two scores on the left. The **Op.** column shows the operations needed to correct the OMR column: **I**nsertion, **R**eplacement, and **D**eletion. Note that during training, all these operations will simply be marked as Errors.

Figure 2a contains an illustration of two snippets of music: one original and one with OMR errors. Figure 2b shows the result of their alignment with the NW algorithm. Our training data will mark each note as an error if it has an operation assigned to it; for example, the eighth rest at the end of the OMR snippet in Figure 2a will be marked as an error in training, since it must be removed to correct the score. Under this scheme we have no way of flagging a note as an error if it is not present in the OMR output. As a workaround, we flag a symbol as erroneous in training if, to correct the input, it would be necessary to insert one or more items *after* that symbol in the sequence. For example, we would mark the very first barline of the OMR output in Figure 2b as an erroneous symbol, since there is a missing `8th.pos2.start` immediately succeeding it.

3.3 Dataset and Data Augmentation

Ideally, our dataset would comprise a large, well-curated corpus of symbolic music in a single genre, each with accompanying score files in some image format, and another dataset containing versions of those score images passed through an OMR process. We could match up each symbolic music file with its OMR'ed counterpart and perform an NW alignment between them. Then, during training, the OMR'ed symbolic music would be the input to our model, and the resulting alignment would be our target.

To our knowledge, there are no large, publicly available sources of data that match these requirements. Datasets intended for training OMR systems do not include erroneous OMR outputs, as their purpose is to aid in developing the OMR process itself. Given a set of scores in symbolic and image formats, we could automate a OMR application to process a new dataset in this way, but the commercial OMR software available to us does not have the requisite batch processing functionality that would make this feasible. Instead, we take a small dataset of symbolic music files paired with their OMR'ed counterparts, and supplement it with a large dataset containing music of the same genre, but without accompanying OMR'ed files. We add synthetic errors to this larger dataset to mimic the natural errors in the small dataset.

Our main dataset, from which we derive natural OMR errors, is a set of Mendelssohn's string quartets comprising 24 movements [22], containing a total of 175 thousand tokens when encoded agnostically. This dataset contains OMR'ed versions of each movement generated using the PhotoScore software, partially-corrected versions of the OMR'ed files for each movement, and fully-corrected versions of each movement. The OMR'ed files contain a high proportion of errors; the character error rate between the fully-corrected and uncorrected files is 0.29, and the same statistic for the partially-corrected files is 0.11 [2] . We supplement this with a corpus of the string quartets of Beethoven, Haydn, Mozart, and Schubert, taken from KernScores [3] and the Annotated Beethoven Corpus [23], comprising a total of 361 movements and a total of 9.25 million agnostic tokens. It would be ideal to focus specifically on a single era of music, but we posit that for the sake of ease of training, consistency in instrumentation (restricting ourselves to only string quartets) is a more important consideration.

To augment the dataset, we add errors to the agnostic representation of our dataset in a way that mimics how natural OMR errors are distributed in the smaller dataset. To this end, we set aside a small portion of the Mendelssohn string quartet dataset and run a NW alignment between these snippets and their manually-corrected versions. From these alignments we can obtain statistics

[2] This statistic may sound high compared to the number of errors shown in Figure 1. This is explained by the verbosity of agnostic encodings; something that *looks* intuitively like a single error on the page may actually constitute several.

[3] `http://kern.ccarh.org/`

	Precision on raw OMR Output				Precision on partially corrected OMR			
	R = 0.80	R = 0.90	R = 0.95	R = 0.99	R = 0.80	R = 0.90	R = 0.95	R = 0.99
Baseline LSTUT	0.49	0.47	0.47	0.46	0.33	0.32	0.31	0.31
Architectures								
LSTM	0.49	0.47	0.47	0.46	0.11	0.11	0.10	0.10
Transformer	0.51	0.50	0.50	0.49	**0.14**	**0.12**	**0.12**	0.10
Synthetic Errors								
Random	0.47	0.46	0.46	0.45	0.10	0.10	0.10	0.10
50% fewer errors	0.53	**0.51**	0.50	0.46	0.10	0.10	0.10	0.09
Sequence Length								
64	0.47	0.47	0.46	0.45	0.10	0.10	0.10	0.09
128	0.49	0.47	0.47	0.46	0.10	0.10	0.10	0.09
512	0.50	0.49	0.49	0.48	0.11	0.10	0.10	0.10
1024	**0.56**	**0.51**	**0.51**	**0.50**	0.11	0.11	0.11	**0.11**

Table 1: A table comparing precision scores for given recall (R) scores on our error detection task. The "Baseline" model uses an LSTUT, synthetic errors made with our data augmentation method described in Section 3.3, and a sequence length of 256. Each section of the table changes a single variable from the baseline.

on how each type of symbol tends to be affected by the OMR process. For example, in the OMR process, the symbol `rest.quarter` remains the same with probability 0.527, is deleted with probability 0.142, is replaced with an eighth rest with probability 0.013, and so on. We use these statistics as a probability distribution, sampling from it for every single symbol in an agnostically-encoded piece, and then apply the resulting operation to each symbol.

This is a highly simplified model of how OMR introduces errors. It does not take into account the effects of adjacent symbols on one another, or how errors tend to group together on parts of a page that are badly scanned or damaged. However, we have a small amount of real OMR data to work from, which we plan to use for validation and testing, so we cannot spare much for the purpose of training the error generator. A more sophisticated error-generation model would likely overfit on such a small fraction of this dataset.

We train with the large dataset augmented with synthetic errors, reserving our dataset of Mendelssohn string quartets containing natural OMR errors for validation and testing. Each batch of training data in agnostic format is tokenized and is augmented with errors. Then, each training example in the batch is NW-aligned with its correct version (Section 3.2), and the alignment results are used to create a sequence of binary flags (error or no error) the same length as the training example. We train the model using the synthetic training data as input and these binary flags as the target. Validation and testing follow the same process but use existing OMR'ed data instead of adding synthetic errors.

4. EXPERIMENTS

As a baseline model, we employ the Long Short-Term Universal Transformer (LSTUT) [24] as an encoder-only model. This architectre consists of a 4-layer Universal Transformer [25] (a Transformer network where all the layers share the same weights) with 5 self-attention heads, bookended by two bidirectional Long Short-Term Memory (LSTM) layers in lieu of the positional encoding scheme used by vanilla Transformers. This architecture was designed specifically to learn from short- and long-term repetitive musical structure, and outperformed other variants tested on our tasks. We use the linear self-attention mechanism by Vyas et al. [26], implemented as part of their `fast-transformers` package for use with the Pytorch deep learning framework; this is a modification to the vanilla Transformer network with memory usage that scales linearly with input size instead of quadratically. The source code for our full workflow is available on GitHub [4] .

We test on two alternate architectures: a bidirectional LSTM and a vanilla Transformer. In the baseline LSTUT, the LSTM layers and attention heads all have an internal hidden dimension of 128. The LSTM network uses four layers, each with a hidden dimension of 512, and the vanilla Transformer network has 6 layers each containing 4 self-attention heads, all with a hidden dimension of 128. All models end in a fully-connected layer that reduces each sequence element down to a single dimension. We designed these architectures such that all would have a similar number of trainable parameters (around 8 million).

We also test on different input sequence lengths, and the type of synthetic errors used for data augmentation; the "Random" run applies uniformly random deletions, insertions, and replacements in the training data, and the "50% fewer" run uses our statistically-informed data augmentation scheme but only adds half as many errors, which we use to see if it improves performance on partially-corrected OMR. All models were trained with early stopping based on validation loss, using four NVIDIA P100-PCIE GPUs with a combined 48 GB of memory. When testing dif-

[4] `github.com/timothydereuse/transformer-omr-spellchecker`

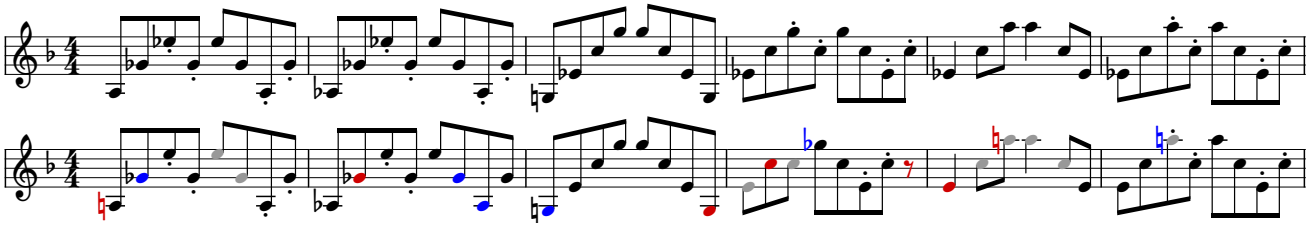


Figure 3: An excerpt from Mendelssohn's String Quartet in E♭ Major, Op. 12, Mvt 1, mm. 212-217 (1st violin part). The upper staff contains the original excerpt, and the bottom one contains OMR errors, along with the predictions of our model. Notes marked in red are errors that our model correctly identified, those marked in blue are errors that our model missed (false negatives), and those marked in grey are correct notes that our model marked as erroneous (false positives). Keep in mind that a correct note may be marked as erroneous if, to correct the score, one needs to insert a symbol *after* it.

ferent sequence lengths, we alter the batch size so as to always use the entirety of the available GPU memory. Lastly, we test on two different sets of data: one is the OMR'ed Mendelssohn quartets, and one is the same quartets after a single pass of human review. This second test set allows us to see if the model can detect the kinds of errors that humans miss when reviewing a score.

In testing, we must apply a threshold to the raw output of the network to turn it into a binary prediction. A common way to do this is to choose a threshold that maximises F1 score on the validation set and then apply that threshold to the test set. However, since we seek to maximise precision *given* a recall score near 1.0, we instead fix several scores for recall, and report the corresponding precision scores on the test set for those thresholds. We anticipate that in a real error-correction scenario, a user would be able to set a threshold themselves based on their tolerance for error; such forms of user interaction are common in high-recall information retrieval applications [27].

4.1 Results and Discussion

Table 1 shows a summary of our results. The Baseline model uses an LSTUT, synthetic errors generated with our data augmentation method, and an input sequence length of 256. We show the precision scores associated with different recall scores to represent different tolerances for missed errors. For example, our baseline model has a precision of 0.47 when the recall is 0.9, meaning that if a user wishes to be sure that 90% percent of the errors are caught, then 53% of the model's detections will be false positives.

The LSTUT model performs slightly better than the LSTM and the vanilla Transformer, though this performance difference is most notable at lower recall scores. Our statistically-informed error generation method outperforms the addition of random errors. At the highest recall score (R=0.99), our best precision score is 0.50, which we obtain at a sequence length of 1024. Precision drops only slightly as our target recall score increases, indicating that a significant fraction of each string quartet is deemed to be correct, and is ignored by the model with high confidence. Our models to perform well at identifying single errors surrounded by correct musical content, while long runs of errors, common in OMR'ed passages with tightly-spaced notes, are sometimes ignored. This may be a consequence of how our data augmentation method does not produce dense clusters of errors.

All models performed poorly on detecting errors in the partially-corrected files, though the vanilla Transformer network did marginally better. Examining our results, we note that our model struggles to notice the *absence* of elements like articulations and accidentals; these kinds of omissions comprise a majority of the errors in the partially-correct OMR files.

Figure 3 shows a violin passage from a Mendelssohn quartet and its OMR'ed version, indicating where our baseline model (at 0.90 recall) predicts errors. A common tendency of our method is illustrated here: when the model cannot pin an error to one single symbol definitively, it will mark a region around the error as erroneous (e.g., the many false positives in measures 4–5 of the excerpt). False negatives do happen (e.g., the missing staccato markings in measure 2), but when notes are missing or inserted, the model tends to make a detection that is at least in the vicinity of the true error.

5. CONCLUSION

For a machine-learning method to be useful in low-resource situations, such as those faced by many archivists and musicologists trying to make long-neglected music more accessible, it must perform a task consistently, given a small amount of data, and be easily generalizable. For this reason we have set ourselves a simple task, in hopes that we can perform it reliably. Even a low precision score may be helpful to a corrector if errors are rare; if our method flags 50% of the score as potentially erroneous, that halves the amount of material the human must check.

While our method achieves a reasonable amount of precision on Mendelssohn's string quartets, we have not proven that it can actually reduce OMR correction time. It would be useful to find ways of evaluating this type of system that speak more directly to its utility for human correctors; for example, by using the results of experiments by Pugin et al. [28] that test how long it takes to correct different types of errors. We plan to integrate our method into an existing notation application or correction interface, to see how it can best aid the symbolic encoding process.

6. REFERENCES

[1] D. Rizo, J. Calvo-Zaragoza, and J. M. Iñesta, "MuRET: A Music Recognition, Encoding, and Transcription Tool," in *Proc. of the 5th Int. Conf. on Digital Libraries for Musicology*. New York, NY: ACM, 2018, pp. 52–56.

[2] M. Alfaro-Contreras, D. Rizo, J. M. Iñesta, and J. Calvo-Zaragoza, "Omr-Assisted Transcription: A Case Study with Early Prints," in *Proc. of the 22nd Int. Society for Music Information Retrieval Conf.*, Online, 2021.

[3] A. Daigle, "Evaluation of Optical Music Recognition Software," Master's thesis, McGill University, 2020.

[4] J. Hajic Jr, J. Novotný, P. Pecina, and J. Pokorný, "Further Steps Towards a Standard Testbed for Optical Music Recognition." in *Proc. of the 17th Int. Society for Music Information Retrieval Conf.*, New York, NY, 2016.

[5] D. Byrd and J. Simonsen, "Towards a Standard Testbed for Optical Music Recognition: Definitions, Metrics, and Page Images," *Journal of New Music Research*, vol. 44, Jul. 2015.

[6] F. Rossant and I. Bloch, "Robust and Adaptive OMR System Including Fuzzy Modeling, Fusion of Musical Rules, and Possible Error Detection," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, 2006.

[7] A. McLeod, J. Owers, and K. Yoshii, "The MIDI Degradation Toolkit: Symbolic Music Augmentation and Correction," *arXiv:2010.00059 [cs, eess]*, 2020.

[8] A. Ycart, D. Stoller, and E. Benetos, "A Comparative Study of Neural Models for Polyphonic Music Sequence Transduction," in *Proc. of the 19th Int. Society for Music Information Retrieval Conf.*, Delft, Netherlands, 2019.

[9] K. Sidorov, A. Jones, and D. Marshall, "Music Analysis as a Smallest Grammar Problem," in *Proc. of the 15th Int. Society for Music Information Retrieval Conf.*, Taipei, Taiwan, 2014.

[10] Y. Wang, Y. Wang, J. Liu, and Z. Liu, "A Comprehensive Survey of Grammar Error Correction," *arXiv:2005.06600 [cs.CL]*, p. 35, 2020.

[11] N. Madi and H. S. Al-Khalifa, "Grammatical Error Checking Systems: A Review of Approaches and Emerging Directions," in *Proc. of the Thirteenth Int. Conf. on Digital Information Management*. Berlin, Germany: IEEE, 2018, pp. 142–147.

[12] D. Naber, "A Rule-Based Style and Grammar Checker," Doctoral Thesis, Universität Bielefeld, 2003.

[13] T. Ge, F. Wei, and M. Zhou, "Reaching Human-level Performance in Automatic Grammatical Error Correction: An Empirical Study," *arXiv:1807.01270 [cs]*, 2018.

[14] C. Brockett, W. B. Dolan, and M. Gamon, "Correcting ESL Errors Using Phrasal SMT Techniques," in *Proc. of the 21st Int. Conf. on Computational Linguistics and the 44th annual meeting of the ACL*, Sydney, Australia, 2006, pp. 249–256.

[15] A. Rozovskaya and D. Roth, "Training Paradigms for Correcting Errors in Grammar and Usage," in *Human Language Technologies: The 2010 Annual Conf. of the North American Chapter of the Association for Computational Linguistics*, Los Angeles, CA, 2010.

[16] P. M. Htut and J. Tetreault, "The Unbearable Weight of Generating Artificial Errors for Grammatical Error Correction," *arXiv:1907.08889 [cs]*, 2019.

[17] F. Foscarin, F. Jacquemard, and R. Fournier-S'niehotta, "A Diff Procedure for Music Score Files," in *Proc. of the 6th Int. Conf. on Digital Libraries for Musicology*. The Hague, Netherlands: ACM, 2019, pp. 58–64.

[18] I. Knopke and D. Byrd, "Towards MusicDiff: A Foundation for Improved Optical Music Recognition Using Multiple Recognizers," in *Proc. of the 8th. Int. Society for Music Information Retrieval Conf.*, Vienna, Austria, 2007.

[19] C. Hawthorne, A. Huang, D. Ippolito, and D. Eck, "Transformer-NADE for Piano Performances," in *Proc. of the 32nd Conf. on Neural Information Processing Systems*, Montreal, Canada, 2018.

[20] D. Rizo, J. Calvo-Zaragoza, J. Iñesta, and I. Fujinaga, "About Agnostic Representation of Musical Documents for Optical Music Recognition," in *Proc. of the Music Encoding Conf.*, Tours, France, 2017.

[21] S. B. Needleman and C. D. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins," *Journal of Molecular Biology*, vol. 48, no. 3, pp. 443–453, 1970.

[22] J. Degroot-Maggetti, T. de Reuse, L. Feisthauer, S. Howes, Y. Ju, S. Kokobu, S. Margot, N. Nápoles López, and F. Upham, "Data Quality Matters: Iterative Corrections on a Corpus of Mendelssohn String Quartets and Implications for MIR Analysis," in *Proc. of the 21st Int. Society for Music Information Retrieval Conf.*, Montreal, Canada, 2020.

[23] M. Neuwirth, D. Harasim, F. C. Moss, and M. Rohrmeier, "The Annotated Beethoven Corpus (ABC): A Dataset of Harmonic Analyses of All Beethoven String Quartets," *Frontiers in Digital Humanities*, vol. 5, p. 16, 2018.

[24] J. de Berardinis, S. Barrett, A. Cangelosi, and E. Coutinho, "Modelling Long- and Short-Term Structure in Symbolic Music with Attention and Recurrence," in *Proc. of the 1st Joint Conf. on AI Music Creativity*, Stockholm, Sweden, 2020.

[25] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser, "Universal Transformers," *arXiv:1807.03819 [cs, stat]*, 2019.

[26] A. Vyas, A. Katharopoulos, and F. Fleuret, "Fast Transformers with Clustered Attention," *arXiv:2007.04825 [cs, stat]*, 2020.

[27] H. Zhang, M. Abualsaud, N. Ghelani, M. D. Smucker, G. V. Cormack, and M. R. Grossman, "Effective User Interaction for High-Recall Retrieval: Less is More," in *Proc. of the 27th ACM Int. Conf. on Information and Knowledge Management*, Torino, Italy, 2018, pp. 187–196.

[28] L. Pugin, J. A. Burgoyne, and I. Fujinaga, "Reducing Costs for Digitising Early Music with Dynamic Adaptation," in *Research and Advanced Technology for Digital Libraries*, L. Kovács, N. Fuhr, and C. Meghini, Eds. Berlin: Springer Berlin Heidelberg, 2007, vol. 4675, pp. 471–474.

"MORE THAN WORDS": LINKING MUSIC PREFERENCES AND MORAL VALUES THROUGH LYRICS

Vjosa Preniqi[1] **Kyriaki Kalimeri**[2] **Charalampos Saitis**[1]
[1] Centre for Digital Music, Queen Mary University of London, London UK
[2] ISI Foundation, Turin, Italy
{v.preniqi, c.saitis}@qmul.ac.uk, kyriaki.kalimeri@isi.it,

ABSTRACT

This study explores the association between music preferences and moral values by applying text analysis techniques to lyrics. Harvesting data from a Facebook-hosted application, we align psychometric scores of 1,386 users to lyrics from the top 5 songs of their preferred music artists as emerged from Facebook Page Likes. We extract a set of lyrical features related to each song's overarching narrative, moral valence, sentiment, and emotion. A machine learning framework was designed to exploit regression approaches and evaluate the predictive power of lyrical features for inferring moral values. Results suggest that lyrics from top songs of artists people like inform their morality. Virtues of hierarchy and tradition achieve higher prediction scores ($.20 \leq r \leq .30$) than values of empathy and equality ($.08 \leq r \leq .11$), while basic demographic variables only account for a small part in the models' explainability. This shows the importance of music listening behaviours, as assessed via lyrical preferences, alone in capturing moral values. We discuss the technological and musicological implications and possible future improvements.

1. INTRODUCTION

The field of music recommender systems has a lot to gain from the fields of music psychology and sociology [1, 2], where researchers have found converging evidence that people listen to music that reflects their personality needs [3–7] and helps express their values [8–10]. For example, extroverted people tend to choose more energetic and rhythmic tunes, while listeners holding values of understanding and tolerance prefer more sophisticated and complex music. Indeed, operationalising knowledge of how personality traits relate to listener taste and preferences has already been shown to improve music recommendations [11, 12] and to make them more diverse [13]. Yet personality dispositions alone may not suffice to explain, and thus model, our music listening behaviours.

© V. Preniqi, K. Kalimeri, and C. Saitis. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** V. Preniqi, K. Kalimeri, and C. Saitis, ""More than words": Linking Music Preferences and Moral Values through Lyrics", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

Aiming to advance an integrative view of the music listener, which may benefit music recommender system scenarios, we set to explore the less attended relation between moral values and music preferences. If personal values are conceived as intrinsic motivational goals, moral values reflect traits learned under the influence of society, culture, and religion, amongst others, which bond people together into groups. Considering music as an evolved tool of social affiliation and bonding [14, 15], it is reasonable to speculate that people may like certain music styles and genres because they provide stimuli that match their morality-related needs.

We further hypothesise that moral values are expressed more clearly in a verbal rather than non-verbal manner and examine their influence on musical taste through lyrics. When people listen to sung music, their preferences are driven by not only the audio content but also the content of lyrics [16]. Lyrics convey rich, multifaceted messages about societal issues such as love, life and death, but also political or religious concepts, often independently from melodic and other audio information [17]. Lyrical messages can support listeners' mental health [18]. Nonetheless, little is known about whether lyrical information manifests links between psychological traits and music preferences [19]. To what extent are moral values reflected in the lyrics of one's favourite songs? Do lyrics predict the moral traits of listeners?

To tackle these questions, we used data from the *LikeYouth.org* project, a Facebook-hosted application developed specifically for research purposes as a surveying tool and was mainly deployed in Italy. Upon providing their informed consent, participants completed validated psychometric questionnaires for personality, moral traits and basic human values, basic demographic information such as age and gender, while agreed to share their Page Likes (see [20] for a detailed description of the complete dataset). For the purpose of this study we only analysed moral values scores and Likes on music artist Pages. Combining these with information from the *genius.com* music database, we obtained the lyrics from the five most popular songs per artist. We performed both sentiment [21] and emotion [22] analysis on the obtained lyrics, assessed their moral narratives employing the MoralStrength lexicon [23], and examined themes and overarching narratives through topic modelling [24].

We built a series of regression models that infer moral traits from lyrical content, demographics, and Likes-based

features (e.g., artist popularity). Our findings show that peoples' worldviews and moral values are indeed reflected in their music preferences as modelled through lyrics, in line with recent literature [25, 26]. Extracting topic and moral features from lyrics specifically increased model performance over sentiment, emotion, and demographic features.

We contribute to the growing literature studying the interplay between music and psychology, with findings that clearly link the preferences of people to artists and songs that are in line with their moral values. Personalised recommendations for streaming on-demand music can be greatly enriched by including notions of moral worldviews about their listeners instead of only shallow psychological attributes [1, 5, 10]. Such knowledge can be directly implemented in psychologically aware music recommender systems, improving music streaming services and contributing to listener wellbeing [27]. On a different key, the relationship between moral worldviews and music preference is crucial to inform communication experts about their choice of the most appropriate music piece to accompany a social campaign.

2. BACKGROUND

We operationalise morality via the Moral Foundations Theory (MFT) [28], which expresses the psychological basis of moral reasoning in terms of five innate foundations, namely Care/Harm, Fairness/Cheating, Loyalty/Betrayal, Authority/Subversion, and Purity/Degradation. These can further collapse into two superior foundations: *Individualising* (Care and Fairness), indicative of a more liberal perspective, and *Binding* (Purity, Authority and Loyalty), indicative of a more conservative outlook.

Moral foundations are considered to be higher psychological constructs than the more commonly investigated personality traits [29]. They have been associated with attitudes towards complex situations such as politics [30, 31], climate change [32], and vaccination [33, 34].

However, moral values have attracted less attention from music scientists. Using data from an ad-hoc online survey comprising, among other items, MFQ scores and preferences ratings on 13 music genres, Preniqi et al. [25] found that people with higher levels of Binding foundations (e.g., more authoritarian individuals) tend to listen to country and Christian music, the lyrics of which often foster notions of tradition [3]. Those with lower levels of Binding traits tend to prefer music genres such as punk and hip-hop, where lyrics are known to challenge traditional values, and the status quo [8]. Individualising foundations were overall harder to predict (cf. [35]). Furthermore, including demographic information (e.g., age, gender, political views, education) improved MFT predictions marginally, indicating the ability of music preferences alone to explain one's moral values.

In the computational social science field, recent work has demonstrated the predictability of MFT traits from a variety of digital data, including gameplay [36], smartphone usage and web browsing [35]. Moral values can also be explained by verbal data, as they can be more clearly communicated through thoughts and opinions [23, 34, 37]. Several dictionary-based approaches for predicting moral values expressed in texts such as tweets and other social media posts have been proposed, including the Moral Foundations Dictionary [37, 38] and the MoralStrength lexicon [23]. Here we employ the latter to uncover moral narratives in song lyrics, which we then use to predict the moral traits of listeners.

		Census	**MFT All data** $n = 3,920$	**MFT & ≥10 Page Likes** $n = 1,386$
Gender	M	48%	54%	53%
	F	52%	46%	47%
Age	<25	23%	21%	29%
	≥25	77%	79%	71%

Table 1. Demographic breakdown of our data according to gender and age. The "Census" column reports the national distribution per attribute according to the statistics provided by the official census bureau [45].

The relation between lyrics and music preferences has only recently started to receive attention across music and social psychology disciplines. Some studies have suggested associations between the personality or mental health of songwriters and their lyrics [39, 40]. On the listener side, neurotic individuals tend to listen to songs with more complex and less repetitive lyrics that express negative emotions [19, 41]. More conscientious individuals tend to prefer lyrics talking about achievements [19] but also about love [42]. Importantly, preferences for lyrics are found to be predictive of personality traits distinctly from audio or melodic preferences [19, 42].

Concerning moral values, in recent work, they have been found to explain a unique and significant portion of the variance in the lyrical preferences of different metal music sub-genre fans that was not already accounted for by personality traits [26]. For example, preferring lyrics about celebrating metal culture and unity was related to higher levels of the Loyalty foundation and higher levels of extroversion. In U.S. popular music, an increase in lyrics related to self-focus and -promotion since the 1980s has been shown to manifest the increasing individualism of American society [43, 44].

3. DATA COLLECTION

The LikeYouth Facebook-hosted application was initially launched in March 2016, while the data used here were downloaded in September 2019. It was deployed mainly in Italy, where approximately 64,000 people entered the platform, from whom 3,920 users (90% geolocated in Italian territory) filled out the MFT questionnaire correctly.

Of those, 47% did not provide their age due to the facultative nature of LikeYouth. Because we wished to include age as a demographic predictor variable, we inferred the missing values from all (e.g., not just music artist related) Page Likes of the 3,920 users. Similar to [46] we created

a sparse matrix representation of Page Likes per user and applied sparse singular value decomposition to reduce dimensionality, while binning the age attribute (median $= 25$) as "younger" (< 25) and "older" (≥ 25) allowed to approximate the official census distribution [45]. We then employed an XGBoost classifier, to predict missing age values [35], with an estimated $AUROC = 0.79$ and standard deviation $= 0.018$. Acknowledging that age inference might add bias to our models, we only use age as a predictor in isolated experiments (see Table 4). We also run the same experiments keeping only users who provided their age. Predictions were similar for Binding and slightly lower for Individualising.

To ensure the stability of our regression models, we applied a simple activity threshold. After extensive experimentation we chose to drop users with less than 10 Facebook Page Likes related to music artists (Page category selection), resulting in a reduced final dataset of 1,386 users. Table 1 reports the demographic breakdown of our data sample in terms of gender and age, which follows closely the population distribution of the official Italian census [45].

For the final 1,386 users, we retrieved song lyrics corresponding to their music artist Page Likes using *genius.com*. Querying the Genius API, we initially obtained the 10 most popular songs per artist alongside the respective lyrics. We assume that if a user liked the Page of a specific artist, then that artist's most famous songs (as per Genius) reflect the music preferences of the user. We carried out predictive tasks using the $n = 10$, 5, or 3 most popular songs from an artist and found that $n = 5$ gave the best compromise in terms of predictions, computational resources, and within-musician variability in lyrical and audio content (see future work discussion) while maintaining an optimal number of musicians and songs for our lyrics data. Finally, we used the spaCy library [47, 48] to identify songs with English lyrics only, resulting in 3,179 artists and 15,895 songs.

We also considered two additional, more shallow digital trace features that can potentially convey information about user's music habits, namely the number of Page Likes per user (mean $= 35.11$, standard deviation $= 33.95$) and a built-in feature of artist popularity from LikeYouth, based on the number of Page followers.

We use LikeYouth because, to our best knowledge, it is the only dataset providing MFT scores of individuals alongside a potential proxy of their music preferences (e.g., artist Page Likes). A limitation of this approach is that the data provided by LikeYouth are static and may thus refer to a snapshot of music interests in time. Streaming platforms could offer richer information about habitual music listening [7, 42]. Nonetheless, there is substantial evidence that Facebook Page Likes can capture personality needs and personal values [5,20,46]. Another limitation is that LikeYouth user MFT scores and thus our predictive models cannot be made publicly available due to privacy implications [34]. Instead, we have shared the lyrics data and related source code for lyrical feature modeling in a GitHub repository. [1]

[1] https://github.com/vjosapreniqi/lyrics-content-features

Type	Method	Features
Topics	LDA	Death/Fear/Violence, Obscene, Romantic, World/Time/Life
Morals	MoralStrength	Care, Fairness, Loyalty, Authority, Purity
Sentiment	VADER	Negative, Positive, Neutral, Compound
Emotions	NRC	Anger, Disgust, Fear, Sadness, Anticipation, Surprise, Joy, Trust

Table 2. Summary of lyrical features used in this study.

4. LYRICS CONTENT ANALYSIS

We extracted a set of textual features related to each song lyrics' overarching narrative (topic modelling), moral valence, sentiment, and emotion. Based on the corresponding feature modeling method, we applied different levels of text preprocessing. Sentiment detection required only a general cleanup while keeping punctuation and capitalization within the text. For the other methods, we extracted Part Of Speech (POS) lemmas using the spaCy lemmatizer [47]. On average, each lyrics contained 273 words and 108 lemmas.

4.1 Topic Modelling

Initially, we aimed to uncover common patterns in the lyrics narratives by applying a topic modelling approach based on Latent Dirichlet Allocation (LDA) [24]. We used LDA due to its simplicity, high accuracy in topic modelling, and good computational efficiency [49]. The input of the LDA model is a term frequency matrix of the corpus created by the song lyrics. To eliminate very common terms that can lead to irrelevant topics, we ignored words with frequency higher than 90%.

To derive the optimum number of topics k, we optimized the topic coherency (C_v metric [50]) for models with $k \in [2, 16]$ using a step size of 2. The number of topics for which coherency was maximised was $k = 4$. For $k > 4$, we obtained topics that were either generic or hard to characterise due to the mixture of different words belonging to multiple topics. While for $k = 4$, the topics obtained were in line with previous literature [51,52]. Table 3 depicts examples of manually selected songs of 5 artists for each topic, ranked by descending weight in the specific topic.

4.2 Moral Valence

We assess the moral narratives by employing the MoralStrength lexicon [23], which holds the state-of-the-art performance in moral text prediction. This expands the Moral Foundation Dictionary by offering three times more moral-annotated lemmas. The lexicon provides, along with each lemma, the *moral valence score*, a numeric assessment that indicates both the polarity and the intensity of the lemma

Topic	Artist	Song Title	%
Romantic (0.39)	Mike Williams	Give it up	99
	Marc Anthony	I need to know	99
	NSYNC	I want you back	99
	Willie Nelson	Always on my mind	98
	Alexia	Because I miss you	97
Obscene (0.24)	Tyga	Rack city	98
	Fat Joe	Yellow tape	96
	Cardi B	Bartier cardi	95
	Chamillionaire	Ridin'	95
	21 Savage	Bank account	91
World/Time/ Life (0.22)	Holly Herndon	Morning sun	99
	Noisecontrollers	The day	97
	Nathan East	Finally home	96
	Dave Gahan	Tomorrow	94
	Gabrielle Aplin	Start of time	90
Death/Fear/ Violence (0.15)	Hatebreed	Destroy everything	99
	Fear Factory	Edgecrusher	97
	Eomac	Mandate for murder	95
	Destruction	Thrash till death	92
	Sabaton	Attack of dead men	91

Table 3. LDA topic modelling: overall topic prevalence (in brackets below topic descriptions) and 5 manually selected songs per topic as ranked by descending topic proportion.

in each of the five moral foundations (MFT traits). Moral valence is expressed on a Likert scale from one to nine, with five considered neutral. When lower than 5, scores reflect notions closer to Harm, Cheating, Betrayal, Subversion, and Degradation, while values higher than 5 indicate Care, Fairness, Loyalty, Authority, and Purity, respectively.

We obtained a moral valence score for each lemma in a song's lyrics and each MFT trait, which is then averaged across lemmas for each song. Negation correction was not applied, as moral foundation polarities do not directly translate as opposites (e.g., "not care" is not the same as "harm"). The MoralStrength lexicon has a limited linguistic coverage; as a result, we could not predict moral valence for 16% of the collected lyrics. Instead, we assigned them the value 5, the neutral point of the moral valence Likert scale. This approach pushes the observed mean towards the center of the scale, but captures the variability of the moral values across all the lyrical data.

4.3 Sentiment and Emotion Analysis

In textual data, emotions, as brief and preconscious phenomena, can be defined via descriptions of appraisal, physiological reaction, expressive display, feeling, or action tendency, while sentiments, as lasting and conscious emotional dispositions, tend to be modelled in terms of text polarity (positive, negative, neutral) [53].

We applied the commonly used VADER (Valence Aware Dictionary and sEntiment Reasoner) model [21] on the lyrical text to obtain information about the sentiment of each song. The VADER model is shown to perform well both with long and short text, providing for each song a score for positive, neutral, negative, and compound sentiment

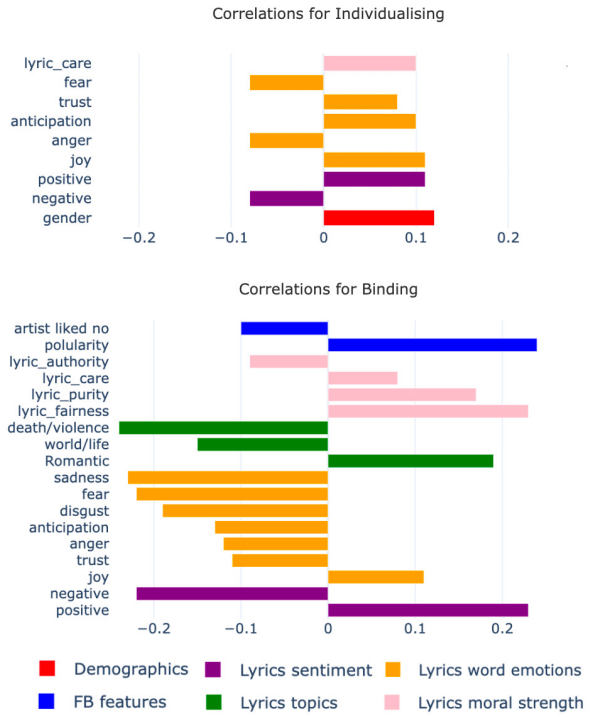


Figure 1. Spearman's Rank correlations of MFT super foundations with demographics, artist likes and lyrics features. We report only ones that were significant at $p \leq .01$. "Artist liked no" refers to the number of artist Likes per user.

(see Table 2). We also estimated the eight basic emotions defined in the Plutchik wheel of emotions [54] employing the NRC Word–Emotion Association Lexicon [22]. This lexicon was shown to be efficient with unlabeled data [55]. Each song lyrics was annotated with the eight emotions (see Table 2) by averaging its word emotion association scores.

5. EXPERIMENTS AND RESULTS

Initially, we explored the relationship between users' moral values as emerged from the self-reported questionnaires, basic demographic attributes, and their respective music preferences, as expressed in the linguistic components of the lyrics. Figure 1 depicts the statistically significant correlations ($p \leq .01$) obtained for the two superior foundations, namely Individualising and Binding. We observed that people who value more Individualising foundations prefer artists whose songs prevalently talk about anticipation and trust. On the other side, those concerned more about social order and Binding foundations tend to prefer artists who deal with more romantic topics in their songs instead of existential and social issues. Overall, participants with strong Binding foundations display a tendency to dislike songs with negative valence and emotions such as sadness, fear, or disgust. Yet both the Individualising and Binding groups resonate with positive and joyful songs, showing that despite often profound differences in sociopolitical stances, music is a shelter to everyone.

Next, we proposed a series of experimental designs to

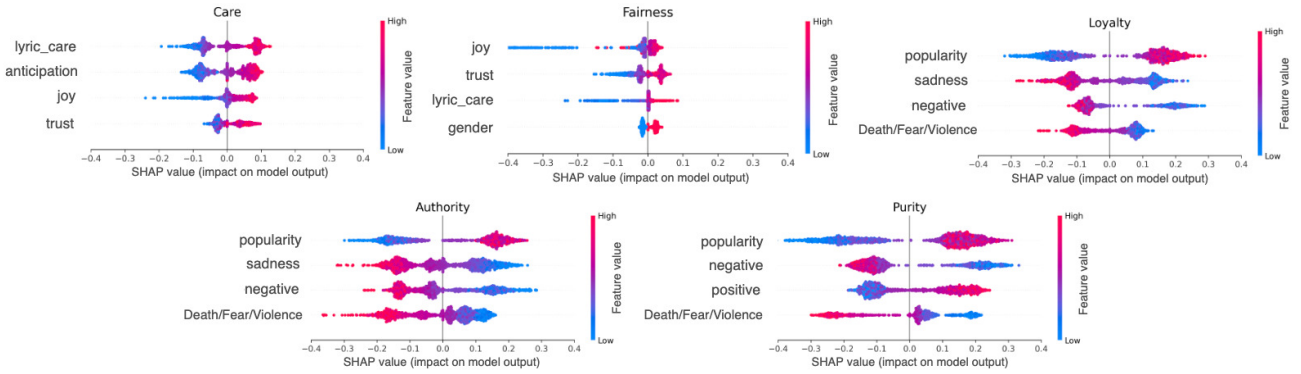


Figure 2. Top 4 individual feature contributions (via SHAP values) for the five basic moral foundations from experiment EX8 (see Table 4). The higher the SHAP value, the more the feature contributes to the prediction model.

ID	Features
EX1	Sentiment (VADER)
EX2	Emotions (NRC)
EX3	Sentiment + Emotions
EX4	Best of {EX1, EX2, EX3} + Morals
EX5	Best of {EX1, EX2, EX3} + Topics
EX6	Best of {EX1, EX2, EX3} + Morals + Topics
EX7	EX6 + Age + Gender
EX8	EX7 + Artist Likes + Artist Popularity

Table 4. Summary of performed experiments with corresponding features used as predictors.

infer moral values of the participants from their music preferences and the respective linguistic content. Table 4 summarises the performed experiments. We employed four algorithms, namely Support Vector Regressor, Random Forest, XGBoost, and ElasticNet, to predict moral values using a multivariate regression approach over a 5-fold cross-validation setting. For each participant, the features were aggregated and normalised. Here, we report only the results from the Random Forest since it slightly outperformed the rest. We used the Pearson's correlation coefficients between the predicted and actual moral values scores to measure the model's goodness fit. This metric was commonly used in papers that predicted personality based on users' music preferences and listening behaviours [5, 56].

To comprehend the general behaviour of our models and evaluate the importance of each feature, we estimated the SHAP values. SHAP (SHapley Additive exPlanations) is a game theory approach designed to illustrate the features' contribution to the final output of any machine learning model [57].

Following the incremental experimental design reported in Table 4, we trained one model per each moral foundation and presented the best results obtained by each feature in Table 5. In line with recent literature [25] that shows higher prediction accuracy for Binding rather than Individualising foundations, we noticed a similar behaviour also when inferring from linguistic features of song lyrics.

When adding demographics and artist Facebook information (EX8), the results slightly improved for both super foundations, implying that the more information we have about users' demographics and music preferences, the more precise our models become. Despite that, the model trained on just emotions, sentiment and moral information (EX4) achieved almost as good results as those who are aware of the demographics and the general artist information (EX7 and EX8). This highlights the importance of music preferences in portraying our goals and decisions whose motivations go far beyond basic demographic knowledge.

Figure 2 depicts the most important individual (only top 4 due to page restrictions) features for predicting each of the five moral foundations. While Figure 3 illustrates the impact individual (top 8) and grouped features in inferring the two superior foundations when considering all predictor variables (EX8). In line with observed correlations, feature importance representations for regression models show that lyrics linked to objective and subtle emotions (e.g., joy, trust, and anticipation) effectively predict Care and Fairness. Whereas more intense and opposite polarities of sentiment and emotions (e.g., fear, sadness, lyrics positive and negative valence) account for better predictions of Loyalty, Authority and Purity. We noticed that those who value more the Binding foundations appear to be sensitive to the popularity of the song, which reflects their worldview of prioritising group-focus over self-focus.

6. CONCLUSION

This paper discussed the link between lyrical information and moral values. We presented a wide range of lyrics processing techniques and features for measuring the power of linguistic aspects in predicting complex psychological traits such as moral values. Besides, we explored and compared the impact of user demographics and shallow digital traces in inferring moral foundations against the song lyrics components.

We noticed that Binders express their views throughout their music preference and lyrical styles. In contrast, Individualising views are more complex to be captured solely by people's music lyrics preferences. Thus, using the proposed framework, it was easier to infer moral values of Binding

	Moral Foundations - Regression Models							
	EX1	EX2	EX3	EX4	EX5	EX6	EX7	EX8
C	.07 [-.05, .19]	.10 [-.02, .21]	.10 [-.02, .22]	.11 [-.01, .22]	.11 [-.01, .22]	.11 [-.01, .22]	**.12 [.01, .24]**	.12 [.01, .23]
F	.04 [-.08, .15]	.06 [-.05, .18]	.06 [-.05, .18]	.07 [-.05, .18]	.06 [-.06, .17]	.06 [-.06, .17]	**.07 [-.05, .19]**	.05 [-.07, .17]
L	.11 [-.01, .23]	.16 [.04, .27]	.18 [.06, .29]	.20 [.08, .31]	.19 [.08, .30]	.20 [.08, .31]	.20 [.08, .31]	**.21 [.09, .32]**
A	.18 [.06, .29]	.22 [.10, .32]	.24 [.12, .34]	.26 [.15, .37]	.24 [.13, .35]	.26 [.15, .36]	.26 [.15, .37]	**.28 [.16, .38]**
P	.18 [.06, .29]	.20 [.09, .31]	.23 [.12, .34]	.25 [.14, .36]	.23 [.11, .34]	.25 [.13, .36]	.25 [.13, .36]	**.27 [.16, .37]**
I	.08 [-.04, .19]	.09 [-.03, .21]	.10 [-.01, .22]	.09 [-.03, .21]	.10 [-.02, .21]	.09 [-.03, .21]	.10 [-.02, .22]	**.11 [-.01, .22]**
B	.19 [.07, .30]	.23 [.11, .34]	.26 [.15, .37]	.29 [.17, .39]	.27 [.15, .37]	.28 [.17, .39]	.28 [.17, .39]	**.30 [.19, .41]**

Table 5. Moral foundations regression with Random Forest using different feature combinations (see Table 4): Pearson's correlation [95% confidence intervals] between predicted and the actual values averaged across 5-fold cross-validation. C: Care; F: Fairness; L: Loyalty; A: Authority; P: Purity; I: Individualising; B: Binding.

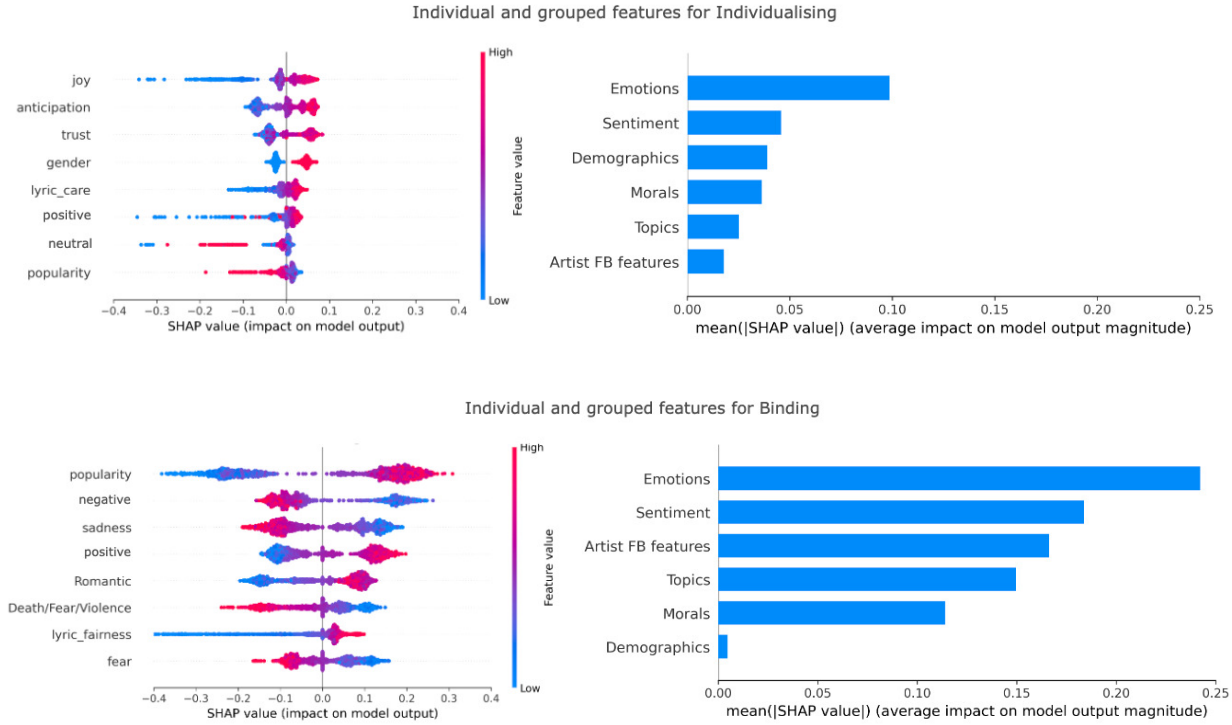


Figure 3. Individual (top 8) and grouped feature contributions (via SHAP values) for the two superior moral foundations from experiment EX8 (see Table 4). The higher the SHAP value, the more the feature contributes to the prediction model.

($.20 \leq r \leq .30$) between predicted and target values than Individualising foundations ($.08 \leq r \leq .11$).

We demonstrated that lyrics features extracted from the naturally emerging music preferences in social media, to some extent, allow for constructing reliable inferences of moral values. Considering the expanded presence of online music streaming services our findings may have direct implications for music recommendation and personalisation algorithms [1,5,58]. Since moral values are a key element of the decision making process in several societal issues [35, 59] and highly linked to political leanings [60], our research implications can help future studies to tackle aspects of why and how music is or can be used for mass stimulation and persuasion in social and political campaigns, raising awareness on what our digital music behaviours can reveal.

In future work, we intend to combine audio and lyrical content analysis together in a multimodal framework to further expand our understanding of music and moral affiliations, especially for Individualising foundations that remain hard to predict. Recent work highlights that preferences for both lyrics and audio features are important in predicting, often distinctly, personality traits [42]. We will also use additional data from LikeYouth to investigate if moral foundations can explain variance in music preferences that cannot be accounted for by personality traits and personal values (cf. [26]). We ultimately aim to integrate our findings into novel psychologically aware music recommender systems, but also beyond the music domain to other media.

7. ACKNOWLEDGEMENTS

This work was supported by the QMUL Centre for Doctoral Training in Data-informed Audience-centric Media Engineering (2021–2025) as part of a PhD studentship awarded to VP. KK acknowledges support from the Lagrange Project of the Institute for Scientific Interchange Foundation (ISI Foundation) funded by Fondazione Cassa di Risparmio di Torino (Fondazione CRT). We would like to thank the three anonymous reviewers and the meta-reviewer for their thoughtful comments.

8. REFERENCES

[1] L. Porcaro, C. Castillo, and E. Gómez Gutiérrez, "Diversity by design in music recommender systems," *Transactions of the International Society for Music Information Retrieval. 2021; 4 (1).*, 2021.

[2] A. Laplante, "Improving music recommender systems: What can we learn from research on music tastes?" in *ISMIR*, 2014, pp. 451–456.

[3] P. J. Rentfrow and S. D. Gosling, "The do re mi's of everyday life: the structure and personality correlates of music preferences." *Journal of personality and social psychology*, vol. 84, no. 6, p. 1236, 2003.

[4] D. M. Greenberg, M. Kosinski, D. J. Stillwell, B. L. Monteiro, D. J. Levitin, and P. J. Rentfrow, "The song is you: Preferences for musical attribute dimensions reflect personality," *Social Psychological and Personality Science*, vol. 7, no. 6, pp. 597–605, 2016.

[5] G. Nave, J. Minxha, D. M. Greenberg, M. Kosinski, D. Stillwell, and J. Rentfrow, "Musical preferences predict personality: evidence from active listening and facebook likes," *Psychological science*, vol. 29, no. 7, pp. 1145–1158, 2018.

[6] S. P. Devenport and A. C. North, "Predicting musical taste: Relationships with personality aspects and political orientation," *Psychology of Music*, vol. 47, no. 6, pp. 834–847, 2019.

[7] I. Anderson, S. Gil, C. Gibson, S. Wolf, W. Shapiro, O. Semerci, and D. M. Greenberg, ""just the way you are": Linking music listening on spotify and personality," *Social Psychological and Personality Science*, vol. 12, no. 4, pp. 561–572, 2021.

[8] A. Gardikiotis and A. Baltzis, "'rock music for myself and justice to the world!': Musical identity, values, and music preferences," *Psychology of Music*, vol. 40, no. 2, pp. 143–163, 2012.

[9] V. Swami, F. Malpass, D. Havard, K. Benford, A. Costescu, A. Sofitiki, and D. Taylor, "Metalheads: The influence of personality and individual differences on preference for heavy metal." *Psychology of Aesthetics, Creativity, and the Arts*, vol. 7, no. 4, p. 377, 2013.

[10] S. Manolios, A. Hanjalic, and C. C. Liem, "The influence of personal values on music taste: Towards value-based music recommendations," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 2019, pp. 501–505.

[11] R. Hu and P. Pu, "Enhancing collaborative filtering systems with personality information," in *Proceedings of the fifth ACM conference on Recommender systems*, 2011, pp. 197–204.

[12] Y. Jin, N. Tintarev, N. N. Htun, and K. Verbert, "Effects of personal characteristics in control-oriented user interfaces for music recommender systems," *User Modeling and User-Adapted Interaction*, vol. 30, no. 2, pp. 199–249, 2020.

[13] F. Lu and N. Tintarev, "A diversity adjusting strategy with personality for music recommendation." in *IntRS@ RecSys*, 2018, pp. 7–14.

[14] C. Loersch and N. L. Arbuckle, "Unraveling the mystery of music: Music as an evolved group process." *Journal of Personality and Social Psychology*, vol. 105, no. 5, p. 777, 2013.

[15] P. E. Savage, P. Loui, B. Tarr, A. Schachner, L. Glowacki, S. Mithen, and W. T. Fitch, "Music as a coevolved system for social bonding," *Behavioral and Brain Sciences*, vol. 44, 2021.

[16] A. M. Demetriou, A. Jansson, A. Kumar, and R. M. Bittner, "Vocals in music matter: the relevance of vocals in the minds of listeners." in *ISMIR*, 2018, pp. 514–520.

[17] S. O. Ali and Z. F. Peynircioğlu, "Songs and emotions: are lyrics and melodies equal partners?" *Psychology of music*, vol. 34, no. 4, pp. 511–534, 2006.

[18] J. H. Lee, A. Bhattacharya, R. Antony, N. K. Santero, and A. Le, "Finding home: Understanding how music supports listener mental health through a case study of bts," in *ISMIR*, 2021, pp. 358–365.

[19] L. Qiu, J. Chen, J. Ramsay, and J. Lu, "Personality predicts words in favorite songs," *Journal of Research in Personality*, vol. 78, pp. 25–35, 2019.

[20] A. Urbinati, K. Kalimeri, A. Bonanomi, A. Rosina, C. Cattuto, and D. Paolotti, "Young adult unemployment through the lens of social media: Italy as a case study," in *International Conference on Social Informatics*. Springer, 2020, pp. 380–396.

[21] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Proceedings of the international AAAI conference on web and social media*, vol. 8, no. 1, 2014, pp. 216–225.

[22] S. M. Mohammad and P. D. Turney, "Crowdsourcing a word–emotion association lexicon," *Computational intelligence*, vol. 29, no. 3, pp. 436–465, 2013.

[23] O. Araque, L. Gatti, and K. Kalimeri, "Moralstrength: Exploiting a moral lexicon and embedding similarity for moral foundations prediction," *Knowledge-based systems*, vol. 191, p. 105184, 2020.

[24] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[25] V. Preniqi, K. Kalimeri, and C. Saitis, "Modelling moral traits with music listening preferences and demographics," *arXiv preprint arXiv:2107.00349*, 2021.

[26] K. J. Messick and B. E. Aranda, "The role of moral reasoning & personality in explaining lyrical preferences," *PLoS one*, vol. 15, no. 1, p. e0228057, 2020.

[27] Y. Mejova and K. Kalimeri, "Effect of values and technology use on exercise: implications for personalized behavior change interventions," in *Proceedings of the 27th ACM Conference on User Modeling, Adaptation and Personalization*, 2019, pp. 36–45.

[28] J. Graham, J. Haidt, S. Koleva, M. Motyl, R. Iyer, S. P. Wojcik, and P. H. Ditto, "Moral foundations theory: The pragmatic validity of moral pluralism," in *Advances in experimental social psychology*. Elsevier, 2013, vol. 47, pp. 55–130.

[29] D. P. McAdams and J. L. Pals, "A new big five: fundamental principles for an integrative science of personality." *American psychologist*, vol. 61, no. 3, p. 204, 2006.

[30] A. Miles and S. Vaisey, "Morality and politics: Comparing alternate theories," *Social Science Research*, vol. 53, pp. 252–269, 2015.

[31] J. Haidt and J. Graham, "When morality opposes justice: Conservatives have moral intuitions that liberals may not recognize," *Social Justice Research*, vol. 20, no. 1, pp. 98–116, 2007.

[32] C. Wolsko, H. Ariceaga, and J. Seiden, "Red, white, and blue enough to be green: Effects of moral framing on climate change attitudes and conservation behaviors," *Journal of Experimental Social Psychology*, vol. 65, pp. 7–19, 2016.

[33] A. B. Amin, R. A. Bednarczyk, C. E. Ray, K. J. Melchiori, J. Graham, J. R. Huntsinger, and S. B. Omer, "Association of moral values with vaccine hesitancy," *Nature Human Behaviour*, vol. 1, no. 12, pp. 873–880, 2017.

[34] K. Kalimeri, M. G. Beiró, A. Urbinati, A. Bonanomi, A. Rosina, and C. Cattuto, "Human values and attitudes towards vaccination in social media," in *Companion Proceedings of The 2019 World Wide Web Conference*, 2019, pp. 248–254.

[35] K. Kalimeri, M. G. Beiró, M. Delfino, R. Raleigh, and C. Cattuto, "Predicting demographics, moral foundations, and human values from digital behaviours," *Comput. Hum. Behav.*, vol. 92, pp. 428–445, 2019.

[36] E. Kim, R. Iyer, J. Graham, Y.-H. Chang, and R. Maheswaran, "Moral values from simple game play," in *International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction*. Springer, 2013, pp. 56–64.

[37] Y. Lin, J. Hoover, G. Portillo-Wightman, C. Park, M. Dehghani, and H. Ji, "Acquiring background knowledge to improve moral value prediction," in *2018 ieee/acm international conference on advances in social networks analysis and mining (asonam)*. IEEE, 2018, pp. 552–559.

[38] J. Graham, J. Haidt, and B. A. Nosek, "Liberals and conservatives rely on different sets of moral foundations." *Journal of personality and social psychology*, vol. 96, no. 5, p. 1029, 2009.

[39] D. M. Greenberg, S. C. Matz, H. A. Schwartz, and K. R. Fricke, "The self-congruity effect of music." *Journal of Personality and Social Psychology*, 2020.

[40] E. J. Lightman, P. M. McCarthy, D. F. Dufty, and D. S. McNamara, "Using computational text analysis tools to compare the lyrics of suicidal and non-suicidal songwriters," in *Proceedings of the Annual Meeting of the Cognitive Science Society*, vol. 29, no. 29, 2007.

[41] J. Shriram, S. Paruchuri, and V. Alluri, "How much do lyrics matter? analysing lyrical simplicity preferences for individuals at risk of depression," *arXiv preprint arXiv:2109.07227*, 2021.

[42] L. Sust, G. Kudchadker, R. Schoedel, T. Schuwerk, M. Bühner, and C. Stachl, "Personality computing with naturalistic music listening data," 2022.

[43] C. N. DeWall, R. S. Pond Jr, W. K. Campbell, and J. M. Twenge, "Tuning in to psychological change: Linguistic markers of psychological traits and emotions over time in popular us song lyrics." *Psychology of Aesthetics, Creativity, and the Arts*, vol. 5, no. 3, p. 200, 2011.

[44] P. McAuslan and M. Waung, "Billboard hot 100 songs: Self-promoting over the past 20 years." *Psychology of Popular Media Culture*, vol. 7, no. 2, p. 171, 2018.

[45] I.Stat, "Italian statistics," http://dati.istat.it/, accessed: 2019-09.

[46] M. Kosinski, Y. Bachrach, P. Kohli, D. Stillwell, and T. Graepel, "Manifestations of user personality in website choice and behaviour on online social networks," *Machine learning*, vol. 95, no. 3, pp. 357–380, 2014.

[47] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing," 2017, to appear.

[48] M. Neumann, D. King, I. Beltagy, and W. Ammar, "ScispaCy: Fast and robust models for biomedical natural language processing," in *Proceedings of the 18th BioNLP Workshop and Shared Task*. Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 319–327. [Online]. Available: https://aclanthology.org/W19-5034

[49] A. Lancichinetti, M. I. Sirer, J. X. Wang, D. Acuna, K. Körding, and L. A. N. Amaral, "High-reproducibility and high-accuracy method for automated topic classification," *Physical Review X*, vol. 5, no. 1, p. 011007, 2015.

[50] M. Röder, A. Both, and A. Hinneburg, "Exploring the space of topic coherence measures," in *WSDM*, 2015, pp. 399–408.

[51] S. Sasaki, K. Yoshii, T. Nakano, M. Goto, and S. Morishima, "Lyricsradar: A lyrics retrieval system based on latent topics of lyrics." in *Ismir*, 2014, pp. 585–590.

[52] L. Misael, C. Forster, E. Fontelles, V. Sampaio, and M. França, "Temporal analysis and visualisation of music," in *Anais do XVII Encontro Nacional de Inteligência Artificial e Computacional*. SBC, 2020, pp. 507–518.

[53] M. Munezero, C. S. Montero, E. Sutinen, and J. Pajunen, "Are they different? affect, feeling, emotion, sentiment, and opinion detection in text," *IEEE transactions on affective computing*, vol. 5, no. 2, pp. 101–111, 2014.

[54] R. Plutchik, "A psychoevolutionary theory of emotions," 1982.

[55] E. Çano and M. Morisio, "Moodylyrics: A sentiment annotated lyrics dataset," in *Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics & Swarm Intelligence*, 2017, pp. 118–124.

[56] A. Anderson, L. Maystre, I. Anderson, R. Mehrotra, and M. Lalmas, "Algorithmic effects on the diversity of consumption on spotify," in *Proceedings of The Web Conference 2020*, 2020, pp. 2155–2165.

[57] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[58] J. R. Ogden, D. T. Ogden, and K. Long, "Music marketing: A history and landscape," *Journal of Retailing and Consumer Services*, vol. 18, no. 2, pp. 120–125, 2011.

[59] J. Haidt and C. Joseph, "Intuitive ethics: How innately prepared intuitions generate culturally variable virtues," *Daedalus*, vol. 133, no. 4, pp. 55–66, 2004.

[60] G. Lakoff, *Moral politics: How liberals and conservatives think*. University of Chicago Press, 2010.

Papers - Session VII

A UNIFIED MODEL FOR ZERO-SHOT SINGING VOICE CONVERSION AND SYNTHESIS

Jui-Te Wu[1] **Jun-You Wang**[2] **Jyh-Shing Roger Jang**[1,2] **Li Su**[1,3]
[1]NTU-AS Data Science Degree Program, National Taiwan University, Taiwan
[2]Department of Computer Science and Information Engineering, National Taiwan University, Taiwan
[3]Institute of Information Science, Academia Sinica, Taiwan
Project page: `https://github.com/bryan051003/USVG`

ABSTRACT

Recent advances in deep learning not only facilitate the implementation of zero-shot singing voice synthesis (SVS) and singing voice conversion (SVC) tasks but also provide the opportunity to unify these two tasks into one generalized model. In this paper, we propose such a model that generate the singing voice of any target singer from any source singing content in either text or audio format. The model incorporates self-supervised joint training of the phonetic encoder and the acoustic encoder, with an audio-to-phoneme alignment process in each training step, such that these encoders map the audio and text data respectively into a shared, temporally aligned, and singer-agnostic latent space. The target singer's latent representations encoded at different granularity levels are all trained to match the source latent representations sequentially with the attention mechanisms in the decoding stage. This enables the model to generate unseen target singer's voice with fine-grained resolution from either text or audio sources. Both objective and subjective experiments confirmed that the proposed model is competitive with the state-of-the-art SVC and SVS methods.

1. INTRODUCTION

Singing voice generation is an innovative technology in contemporary music and multimedia content production. There are two major approaches to singing voice generation, which are 1) singing voice synthesis (SVS) [1–7] and 2) singing voice conversion (SVC) [8–12]. SVS features generation from a given musical score and lyrics (texts) and SVC from the given content (e.g., pitch, rhythm) of a source audio recording. Both approaches aim at imitating the performance style (e.g., expression, timbre) of a target singer's recordings in the generation result. In recent years, the SVS and SVC tasks have both witnessed a great improvement thanks to the advances in deep learning. Training a model that generates singing voice in different singer identities is no longer an issue. However, most of these works only support the generation of preset singers' voice, and learning the voices of new singers usually requires a sufficient amount of data to fine-tune the model [4, 7].

A practical singing voice generation system requires the capability to adapt to arbitrary singers, also known as the *zero-shot* scenario. Recent works on zero-shot SVC/SVS have utilized speaker encoders which pretrained on speaker verification tasks [13, 14] to generate speaker embeddings that enable the system to adapt to unseen singers' voices [15–17]. However, this approach has proven insufficient for unseen voice adaptation in recent speech generation studies. An advanced zero-shot voice conversion (VC) and text-to-speech (TTS) approach have utilized the attention mechanism to extract target speaker information from reference audio according to the desired content [18, 19], which outperforms the speaker encoder method [20, 21]. Such advances show potential in zero-shot singing voice generation, which is more challenging than VC and TTS due to the high variation of pitch, timbre, and expression in the singing voice, as well as the lack of publicly available datasets containing a large number of singers. These issues hinder a zero-shot singing voice generation model from sufficient generalization.

Recently, a semi-supervised SVS model combining an acoustic encoder and a phonetic encoder is proposed to deal with both audio and text inputs [7]. Jointly training the two encoders allows the SVS model to learn new singing voice identities in the fine-tuning stage with audios without text annotation. This approach provides new perspectives of zero-shot singing voice generation: from the aspect of technical concerns, joint training of both audio and phonetic encoders might improve the generalizability of zero-shot SVS/SVC; from the aspects of the application, SVS and SVC are no longer regarded as independent but unified into one task.

While [7] only supports SVS for preset singers' voice, in this paper we propose a unified model for SVC and SVS, which is able to generate arbitrary singing voice by either text or audio input. In addition, the system does not require well-aligned lyrics for synthesis since the phoneme duration is learned in a self-supervised manner. We utilize the attention mechanism to extract the target singer information, and introduce two variations of implementation, depending on the similarity or naturalness of the generated

© J.-T. Wu, J.-Y. Wang, J.-S. R. Jang and L. Su. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J.-T. Wu, J.-Y. Wang, J.-S. R. Jang and L. Su, "A unified model for zero-shot singing voice conversion and synthesis", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

audio. Results show that for both SVC and SVS on unseen singers, the proposed model outperforms state-of-the-art.

2. METHOD

Figure 1 illustrates the whole diagram of the proposed system. The left side of the diagram (blue part) is the source encoder, which encodes the content information of the desired output from either audio or text input. It contains an acoustic encoder (E_a) which encodes the spectrogram and a phonetic encoder (E_p) which encodes the phoneme sequence. The monotonic alignment search (MAS) module [22] encourages the output latent spaces of the two encoders to be in a similar distribution such that the phoneme embeddings are temporally aligned with the spectrogram. SVC or SVS can be achieved by choosing which encoder output to process. On the right side of the diagram, the target encoder (E_t) encodes the spectrograms of the target singer's audios into singer information, where the decoder (D) fuses it into the content information and generates the final output. The functions of all these building blocks will be introduced hereafter in this section.

2.1 Source encoder

Inspired by [7], our source encoder supports joint training for both audio and text inputs. The audio input $\mathbf{x}_s \in \mathbb{R}^{t_a \times n}$ is the log-scale mel-spectrogram of an input audio segment from the source singer s, where t_a is the number of frames and $n = 80$ is the number of mel-bands. The text input $\mathbf{p}_s$, which is corresponding to $\mathbf{x}_s$ is a phoneme index sequence with the length of t_p.

The acoustic encoder E_a maps $\mathbf{x}_s$ to a latent representation $\mathbf{z}_a \in \mathbb{R}^{t_a \times d}$ of source audio, where d is the dimension of this latent representation. Two methods are adopted to train the singer-agnostic $\mathbf{z}_a$. First, we employ a singer classifier C, which works as a domain confusion network [8] that attempts to predict the one-hot singer embedding s from $\mathbf{z}_a$ in a frame-by-frame manner by minimizing the loss function $\mathcal{L}_C := \text{CE}(C(\mathbf{z}_a), s)$, in which CE is the categorical cross-entropy. On the other hand, E_a is trained by maximizing $\mathcal{L}_C$ by using a gradient reversal layer. Second, after having $\mathbf{z}_a$ from E_a, we add Gaussian noise $\mathcal{N}(0, \alpha)$ to $\mathbf{z}_a$ to prevent E_a from overfitting the mel-spectrogram details and to encourage the stability of phoneme embedding [7]. We set $\alpha = 0.1$ in this paper.

The phonetic encoder E_p firstly maps $\mathbf{p}_s$ to a phonetic latent space $\mathbf{z}_p \in \mathbb{R}^{t_p \times d}$ at one of its intermediate layer. Then, the MAS module [22] is adopted to find the alignment between the acoustic embedding and the phonetic embedding and obtains the duration of each phoneme, which is required in the synthesis stage. Denote each frame of $\mathbf{z}_p$ as $\mathbf{z}_{p,i}$ for $i = 1, \cdots t_p$. Each $\mathbf{z}_{p,i}$ is fed into a linear layer which outputs a phonetic prior distribution $\hat{\mathbf{z}}_{p,i} \sim \mathcal{N}(\mu_i, \sigma_i)$. In the MAS process, the phoneme prior distribution sequence $\hat{\mathbf{z}}_p := \{\hat{\mathbf{z}}_{p,i}\}_{i=1}^{t_p}$ is aligned to $\mathbf{z}_a$ by finding the monotonic and surjective alignment path $A : [1 : t_a] \to [1 : t_p]$ such that the objective function

$$\mathcal{L}_{\text{mle}} := \sum_{j=1}^{t_a} \log \mathcal{N}(\mathbf{z}_{a,j}; \mu_{A(j)}, \sigma_{A(j)}) \tag{1}$$

is maximized. $\mathcal{N}(z; \mu, \sigma)$ is the likelihood function of a Gaussian variable z parametrized by μ and σ. In each training step, the alignment path is first searched by the Viterbi algorithm, and then the maximum likelihood estimation over this path updates the parameters. $\hat{\mathbf{z}}_p$ therefore approximates $\mathbf{z}_a$ iteratively along the training process. The optimal alignment path A^* indicates the temporal duration $d^* := \{d_i^*\}_{i=1}^{t_p}$ of every phoneme $\mathbf{p}_{s,i}$ in terms of frames, such that the texts could be well aligned with the log-scale mel-spectrogram.To avoid inaccurate pronunciation of the generated voice in some words, we duplicate expand $\mathbf{z}_{p,i}$ by d_i^* frames and pass it into the downstream layers of E_p, which finally outputs the phonetic latent representation $\mathbf{z}_p^* \in \mathbb{R}^{t_a \times d}$. To encourage the audio and the phoneme models to encode common information, the l_2 loss between $\mathbf{z}_p^*$ and $\mathbf{z}_a$ is minimized.

$$\mathcal{L}_{\text{enc}} := \|\mathbf{z}_p^* - \mathbf{z}_a\|_2^2 . \tag{2}$$

The final source embedding $\mathbf{z}$ is obtained by randomly switching between $\mathbf{z}_p^*$ and $\mathbf{z}_a$, that means,

$$\mathbf{z} = k\mathbf{z}_p^* + (1 - k)\mathbf{z}_a \tag{3}$$

where $k \sim \text{Bernoilli}(0.5)$.

To encode the pitch information, we extract the F0 contours of the source audio data using CREPE [23] and represent them in terms of MIDI pitch numbers. In preliminary experiments, we found that the output of CREPE caused inconsistent spikes in some audio segments. Therefore, we apply a median filter with the size of three to remove the noise, and obtain our input pitch sequence $\mathbf{f}_s$. $\mathbf{f}_s$ is then converted to a sequence of pitch embedding and concatenated with $\mathbf{z}$ as the decoder input.

2.2 Target encoder

Similar to E_a, the target encoder E_t is also constructed mainly with stacking three Conv Blocks (see Figure 1 for the diagram of a Conv Block). E_t takes the log-scale mel-spectrogram $\mathbf{x}_t$ of the audio segments from the target singer's corpus as input and then outputs timbre features with different granularity levels from each Conv Block. D then processes singer information following the order from the highest to the lowest level. Inspired by [19], these Conv Blocks in E_t are linked to the Style-Adaptive Layer Normalization (SALN) Feed Forward Transformer (FFT) blocks [24,25] in the decoder D (see Figure 2 Section 2.3), resembling the encoder-decoder pathway in the U-net [26].

2.3 Decoder

The decoder D incorporates two decoding steps. First, the *feature transformation module* fuses the target singer information into the content latent representation. Second, the *mel-spectrogram generator* reconstructs the log-scale

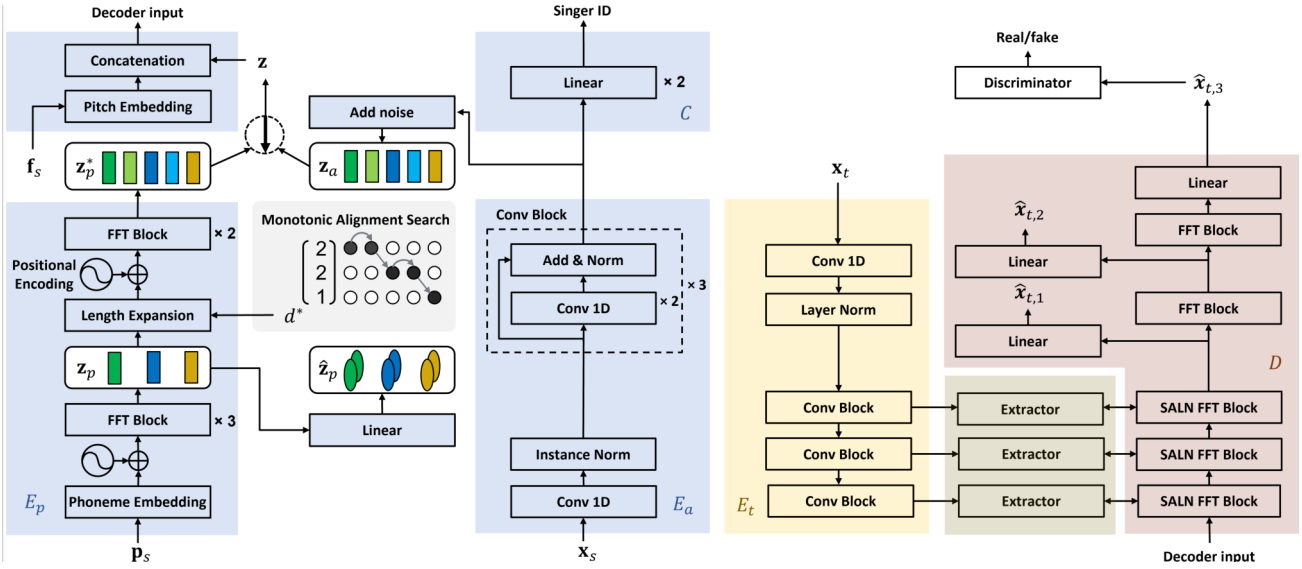


Figure 1. The architecture of the proposed model.

mel-spectrogram. The architecture of the feature transformation module is the same as in [25], in which the layer normalization layer in the FFT block [24] is replaced by a SALN layer [25]. Given an intermediate output h of the FFT and the target singer information w, the SALN operation can be formulated as follows:

$$y = g(w) \odot \text{LayerNormalization}(h) + f(w)\,, \quad (4)$$

where g and f are both linear layers and $\odot$ denotes element-wise multiplication.

The mel-spectrogram generator is comprised of two stacked FFT blocks and three output linear layers linked from the input/output nodes of the FFT blocks. This is similar to the idea of post-network [20, 27] using the FFT and linear layer [6, 24]. Each of the linear layers then outputs the log-scale mel-spectrogram. The outputted mel-spectrograms are denoted as $\hat{\mathbf{x}}_{t,1}$, $\hat{\mathbf{x}}_{t,2}$, and $\hat{\mathbf{x}}_{t,3}$. During training, suppose $\mathbf{x}_s$ is the ground-truth log-scale mel-spectrogram and the decoder is trained to minimize the average mel-spectrogram l_2 reconstruction loss:

$$\mathcal{L}_{\text{recon}} := \frac{1}{3}\sum_{i=1}^{3} \|\mathbf{x}_s - \hat{\mathbf{x}}_{t,i}\|_2^2\,. \quad (5)$$

2.4 Extractor

The extractor is an attention and layer normalization block [1] that connects each pair of Conv Block of E_t and the SALN FFT block of D, as shown in the right side of Figure 1. The extractor controls the mapping from the latent representation of the target singer outputted from the Conv Blocks of E_t to the latent representations of the source utterance such that they are structurally aligned. In other words, the attention mechanism guides E_t to extract the local timbre/style patterns that match the hidden latent states of D layer by layer in the training process [19].

[1] The affine transformation is removed in this block.

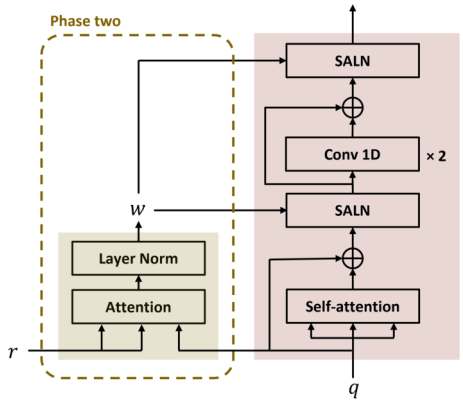


Figure 2. The SALN FFT block (right) with the Extractor (left). For the FFT blocks not using the SALN layers (e.g., the FFT blocks in E_p), the SALN layer is replaced by a layer normalization layer.

The processing of the extractor is illustrated in Figure 2. Let q be the input of one SALN FFT block of D and r be the latent representation of the target from its corresponding Conv Block of E_t. Each frame of q attempts to find local timbre/style patterns in r that have a similar structure to itself. And by aggregating them together, the extractor outputs an aligned local timbre/style representation w. More specifically, the attention mechanism is as follows:

$$Q = qW_q\,, \quad K = rW_k\,, \quad V = rW_v\,,$$
$$w = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V\,, \quad (6)$$

where W_q, W_k and W_v are learnable parameters.

It should be noted that for each generated result, its corresponding w and also r are usually learned from the concatenation of multiple target audios, which is a strategy for training speech synthesis and conversion systems [18, 19]. However, such a strategy still tends to generate discontinuous outputs in SVS and SVC, especially when deal-

ing with unseen singers. This is because singing exhibits much more diverse and combined expressions than speaking. Under this situation, an effective extractor should be able to recombine the features of various expressions from different audio recordings into a sentence. The softmax function in (6), however, fails to achieve this since it tends to give parsimonious attention maps.

To alleviate the situation mentioned above, we adopt the idea from video style transfer [28], which used an alternative way of computing the attention map by changing the Softmax function in Equation (6) to cosine similarity. The cosine similarity attention can be represented as:

$$M_{i,j} = \frac{S_{i,j}}{\sum_j S_{i,j}}, \quad S_{i,j} = \frac{Q_i \cdot K_j}{\|Q_i\|\|K_j\|} + 1, \\ w = MV, \tag{7}$$

where Q_i and K_j are the ith frame of Q and jth frame of K respectively, and $M_{i,j}$ is the (i,j)th element of the resulting new attention map M. Since the softmax function may overemphasize the contents of vocal fragments with similar phonetic structures [28], using cosine similarity ensures that the model combines much more diversified timbre structures and generates a smoother output.

2.5 Discriminator

Over-smoothing of the mel-spectrogram predictions is a common problem in singing voice generation due to the use of the reconstruction loss [29]. Like many previous studies, we add a discriminator (denoted as Disc) at the output stage of our model to alleviate this problem.

The discriminator architecture we use is similar to [30] but with two slight modifications. First, we remove the conditional projections that were intended to make predictions for multiple singers, and second, we divide the number of channels by 4 to balance the generator and discriminator performances. The input of the discriminator is a 128-frame mel-spectrogram segment randomly sampled from $\hat{\mathbf{x}}_{t,3}$, and the output is a single value. It should be noted that $\hat{\mathbf{x}}_{t,1}$ and $\hat{\mathbf{x}}_{t,2}$ are not taken as the input of the discriminator in our experimental setting. Based on the principle of the Least Square Generative Adversarial Network (LSGAN) [31], the proposed model is trained with the adversarial loss $\mathcal{L}_{\text{adv}}$ while the discriminator is trained with the discriminator loss $\mathcal{L}_{\text{Disc}}$ ($a = 0.3$ in this paper):

$$\mathcal{L}_{\text{adv}} := (\text{Disc}(\hat{\mathbf{x}}_{t,3}) - a)^2, \\ \mathcal{L}_{\text{Disc}} := (\text{Disc}(\hat{\mathbf{x}}_{t,3}))^2 + (\text{Disc}(\mathbf{x}_s) - a)^2. \tag{8}$$

2.6 Two-phase training process and inference

We adopt the two-phase training process [7] to train the whole system. During phase one, the source encoder and D are jointly trained. Since E_t and the extractors are inactive in this phase, the latent representation of target singer w is set to a fixed-size 1-dimensional vector, which is derived from a singer embedding lookup table followed by two linear layers with ReLU activation. The 1-dimensional w is then expanded to t_a frames and fed into the SALN module. The total loss during this phase is as follows:

$$\mathcal{L}_{\text{Phase-I}} := \mathcal{L}_{\text{recon}} + \lambda_{\text{enc}}\mathcal{L}_{\text{enc}} - \lambda_{\text{mle}}\mathcal{L}_{\text{mle}} - \mathcal{L}_C, \tag{9}$$

where the last two terms of the equation use a minus sign to represent maximization, and $\mathcal{L}_C$ is also used for updating C. We set the loss weights with $\lambda_{\text{enc}} = 0.5$ and $\lambda_{\text{mle}} = 0.1$.

During phase two, E_t, D, and the extractors are jointly trained, while the parameters of the source encoder are fixed. For the input of E_t, we randomly sampled 5 utterances from the source singer s and concatenate their log-scale mel-spectrogram along the time axis as $\mathbf{x}_t$. The total loss for updating the model during this phase is as follows:

$$\mathcal{L}_{\text{Phase-II}} := \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{adv}}, \tag{10}$$

and the discriminator Disc is updated by $\mathcal{L}_{\text{Disc}}$.

The process of randomly switching (see Equation (3)) is applied in both phases, and for the utterance without textual notation, we directly pass $\mathbf{z}_a$ as $\mathbf{z}$, letting the corresponding loss functions $\mathcal{L}_{\text{enc}}$ and $\mathcal{L}_{\text{mle}}$ not counted.

During inference, we specify either $\mathbf{z}_p^*$ or $\mathbf{z}_a$ as $\mathbf{z}$, depending on whether SVS or SVC is to be performed. $\mathbf{x}_t$ can be multiple utterances of a target singer. Since the input pitch $\mathbf{f}_s$ may not be within the target singer's pitch range, we shift $\mathbf{f}_s$ by the difference in median pitches:

$$\mathbf{f}_{\text{shift}} = \mathbf{f}_s - \text{median}(\mathbf{f}_s) + \text{median}(\mathbf{f}_t), \tag{11}$$

where $\mathbf{f}_t$ is the pitch sequence of $\mathbf{x}_t$ and $\mathbf{f}_{\text{shift}}$ is the shifted pitch for model input [15]. We take the last layer output $\hat{\mathbf{x}}_{t,3}$ of the model as the resulting mel-spectrogram.

Dataset	d	p	d/p	Type	Text
MPOP600	10	4	150	singing	✓
Musdb-V	2.3	86	1.6	singing	✗
NUS-48E	2.8	12	14	singing/speech	✓
VCTK	44	109	24.2	speech	✓

Table 1. The datasets employed in this paper. From left to right: dataset name, total duration (d, in hours), number of speaker/singer (p), average duration per speaker/singer (d/p, in minutes), type of content (singing, speech, or both), and the availability of text labels.

3. EXPERIMENTAL SETUP

3.1 Datasets

Four public datasets are used in our experiment. First, the MPOP600 [32] dataset contains 600 Mandarin pop songs sung by two male and two female vocalists. Second, the NUS-48E [33] dataset consists of 48 English popular songs performed by 12 singers. Third, the VCTK corpus [34] is a multi-speaker speech dataset for TTS and voice conversion tasks and has been widely used in many singing voice generation systems.Finally, Musdb-V is the vocal tracks manually collected from MUSDB18 [35], a dataset for music

source separation, containing 150 songs in different genres along with their isolated drums, bass, vocals, and other stems.Details of these datasets are listed in Table 1. The total duration of the data achieves 59.1 hours.

We randomly select 10 singers from Musdb-V as our test set; they are the so-called unseen singers during training. The remaining data are then partitioned into training and validation sets by 95:5 for each singer/speaker. All the audios are resampled to 24kHz, and the log-scale mel-spectrograms with 80 bins are computed by short-time Fourier transformation (STFT) using the size of Fast Fourier Transform, window size, and hop size of 2,048, 1,200, and 300, respectively. For MPOP600 and VCTK, we convert the text into phoneme sequence using pypinyin [2] and phonemizer [3], respectively. As for NUS-48E, we use the phoneme labels provided in the dataset.

3.2 Implementation details

The dimension of all hidden layers is set as 128 for the source encoder and 256 for both D and E_t. In each block, the kernel size in the two-layer 1D-convolution are set to 3 and 1, respectively, and the kernel sizes in the first 1D-convolution layer of E_a and E_t are both set to 3. The number of attention heads is set to 2. We use the AdamW optimizer with $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$, and follow the same learning rate schedule in [19]. Both phase one and phase two are trained for 250k steps with batch size being 8, and the discriminator joins the training process of phase two after 150k steps. We use Parallel WaveGAN (PWG) [36] to convert log-scale mel-spectrograms to waveforms. The PWG [4] is trained for 1M steps using MPOP600, NUS-48E and Musdb-V.

3.3 Evaluation methods

We conduct the evaluation with four scenarios: 1) **Unseen SVC**, conversion between singers in the test set; 2) **Seen SVC**, conversion between singers in the training set; 3) **Unseen SVS**, synthesizing singers in the test set from the utterance in the validation set; [5] and 4) **Seen SVS**, synthesizing singers in the training set from the utterance of another speaker/singer also in the training set. In addition, the VCTK and the speech part of NUS-48E are excluded from the evaluation. For each round of generation, one utterance of the source singer and five utterances of the target singer are randomly selected.

For objective evaluation, we use a speaker verification (SV) system [37] to evaluate the similarity between the target singer and the generated singing voice. The SV system [6] is trained from scratch using all four datasets for 500 epochs. The loss function and the model we use is AM-Softmax and ResNetSE34L, respectively. The resulting model takes an utterance as input and outputs a fix-dimensional embedding. We then compute the cosine similarity between the embedding of the generated singing voice and the average embedding of the five target utterances. The averaged cosine similarity of the randomly generated 1,000 utterances is then reported.

For subjective evaluation, we conduct a Mean Opinion Score (MOS) test. For each evaluation scenario, 50 utterances are randomly generated for evaluation. Each participant was asked to listen to the source utterance, target utterances, and the generated audios , and rated the generated singing voice in terms of two metrics: perceptual naturalness and similarity to the target. Both metrics are rated from 1 to 5, with 5 representing the best performance and 1 being the worst. We report the averaged scores with the 95% confidence intervals for each model.

Model	Unseen		Seen	
	SVC	SVS	SVC	SVS
Baseline SVC	0.059	–	0.213	–
Baseline SVS	–	0.084	–	–*
Fragment SVC/SVS	0.129	0.122	0.237	0.244
Proposed (S)	**0.294**	**0.232**	**0.536**	**0.479**
Proposed (S\Disc)	0.257	0.232	0.420	0.398
Proposed (C)	0.115	0.079	0.451	0.446

Table 2. Result of objective evaluation. *Result of seen Baseline SVS is not counted due to the inconsistency of training data (not supporting text-free training) compared to other seen SVS models.

Datasets	SVC	SVS
All	0.290	0.233
w/o MPOP600	0.292	0.225
w/o Musdb	0.178	0.145
w/o MPOP600 and w/o Musdb	0.194	0.149

Table 3. Objective similarity scores with reduced training data. All four settings are trained with Proposed (S) under the unseen-to-unseen scenario.

3.4 Models for comparison

Three variants of the proposed model are considered in the evaluation. They are the model using the softmax-based attention in the extractor (denoted as **Proposed (S)**), the model using cosine similarity attention (denoted as **Proposed (C)**), and the model without the discriminator while using softmax-based attention (denoted as **Proposed (S\Disc)**). These model variants are used to compare how the attention types and the discriminator affect the performance. Besides, three baseline models are considered and described as follows.

Fragment (SVC/SVS) is adapted from FragmentVC, a state-of-the-art voice conversion model which also employs the softmax attention in the extractors [19].In our implementation, we simply replace the pretrained Wav2Vec [38] source encoder in FragmentVC with our pretrained source encoder. The remaining architecture and the train-

[2] https://github.com/mozillazg/python-pinyin
[3] https://github.com/bootphon/phonemizer
[4] https://github.com/kan-bayashi/ParallelWaveGAN
[5] It should be noted that there is no text annotation in our test set.
[6] https://github.com/clovaai/voxceleb_trainer

Task	Model	Unseen singers		Seen singers	
		Similarity	Naturalness	Similarity	Naturalness
SVC	Baseline (SVC)	3.14 ± 0.17	3.29 ± 0.16	3.20 ± 0.18	3.21 ± 0.16
	Proposed (S)	$\mathbf{3.61 \pm 0.16}$	3.27 ± 0.17	3.63 ± 0.16	3.27 ± 0.16
	Proposed (C)	3.56 ± 0.17	$\mathbf{3.53 \pm 0.17}$	$\mathbf{3.70 \pm 0.16}$	$\mathbf{3.46 \pm 0.16}$
SVS	Baseline (SVS)	3.14 ± 0.18	2.98 ± 0.16	$\mathbf{3.88 \pm 0.15}$	$\mathbf{3.39 \pm 0.16}$
	Proposed (S) w/o Musdb-V	$\mathbf{3.18 \pm 0.17}$	$\mathbf{3.06 \pm 0.17}$	3.87 ± 0.15	3.32 ± 0.17

Table 4. Subjective test of both the SVC and SVS tasks. MOS and the 95% confidence interval are shown for each case.

ing process follow the original implementation.

Baseline (SVC) is adapted from [15], the state-of-the-art zero-shot SVC model, which is an adaptation of AutoVC [20] and uses the WORLD vocoder [39] to synthesize the singing voice. In our modification of [15], we replace the WORLD vocoder with the PWG vocoder by changing the output layer to generate mel-spectrogram and concatenating a pitch embedding with the same dimension of the content encoding. This is to ensure fair comparison since WORLD vocoder may lead to sound quality degradation in comparison with PWG [40]. During training and inference, the singer embedding is generated by feeding the five target utterances to the speaker encoder and averaging the resulting embeddings.

Baseline (SVS) is adapted from the singing model of DeepSinger, also a state-of-the-art SVS model [6]. In our implementation, we change the output layer of this model to generate mel-spectrograms, and the phoneme duration is extracted by a commonly used open source tool [41]. The major difference between Baseline (SVS) and our proposed model is that the baseline model needs phoneme duration provided by another speech-text alignment system, while our model estimates phoneme duration directly from our self-supervised source encoder.

For the objective test, the six models described above are all compared under the four scenarios. As for the subjective test, to reduce participants' loading, we only compare Baseline (SVC), Proposed (S), and Proposed (C) for SVC, and compare Baseline (SVS) and Proposed (S) (w/o Musdb-V) for SVS. it should be noted that in SVS, Proposed (S) is particularly trained without Musdb-V; this is again for a fair comparison since Baseline (SVS) does not support training with text-free data. [7]

4. RESULTS

Table 2 shows the objective evaluation results in terms of similarity score. The three proposed models generally outperform the baselines. The effectiveness of using the attention-based extractor is verified by comparing Baseline SVC/SVS to the other four models. Also, the advantage of the SALN layer is shown by comparing Fragment SVC/SVS and Proposed (S). The improvement using the discriminator is also shown. The softmax-based attention achieves higher similarity than cosine similarity attention.

Table 3 compares the objective similarity scores trained on reduced sizes of data for the unseen-to-unseen case. It shows that the Musdb-V dataset plays a crucial role in improving the similarity score, and implies that a training dataset with more singers might benefit from singing voice generation more than a large dataset in zero-shot singing voice generation.

Table 4 shows the MOS results responded by 62 subjects for both the SVC and SVS tasks. For the SVC task, both Proposed (S) and Proposed (C) outperform the Baseline (SVC) considerably in terms of perceptual similarity for both seen and unseen cases. As for naturalness, Proposed (C) outperforms both Baseline (SVC) and Proposed (S) substantially. The cosine similarity attention achieves better naturalness and verifies the argument that it can better combine timbre structures and generate smoother outputs, while softmax-based attention achieves better similarity, in line with objective evaluation (see Table 2).

For the SVS subjective test, our degenerated model (Proposed (S) without Musdb-V) shows results comparable to Baseline (SVS) in general, while it still slightly outperforms the baseline for both similarity and naturalness in the unseen-to-unseen case. Such a comparable result can be explained by the finding in Table 3: having a singer-diverse dataset (e.g., Musdb-V) is critical for zero-shot singing voice generation. Since lyrical annotations are not always available for such datasets, a unified model that supports both audio- and text-based training is apparently more flexible, and could also jointly improve the performance of the SVS and SVC models.

5. CONCLUSION

We have presented a unified model which jointly supports the zero-shot SVC and SVS tasks, and has achieved state-of-the-art performance on both tasks. Our experiments also show that the design of the attention mechanism determines the trade-off between perceptual similarity and naturalness, and that a dataset containing a large number of singers is critical in improving zero-shot SVS and SVC. These two directions are suggested for future research on zero-shot singing voice generation.

[7] Since our proposed model allows either audio or text input, we can surely incorporate the text-free Musdb-V set but train the model for the SVS task. In our pilot study, we also observed that incorporating Musdb-V did improve the perceptual quality of SVS results. Such improvement with Musdb-V is also seen in the ablation study (Table 3). However, in order not to take advantage of the Baseline (SVS) model, we opt to degenerate Propose (S) (i.e. without Musdb-V) in our subjective test.

6. REFERENCES

[1] J. Bonada and X. Serra, "Synthesis of the singing voice by performance sampling and spectral models," *IEEE Signal Process. Mag.*, vol. 24, no. 2, pp. 67–79, 2007.

[2] T. Nakano and M. Goto, "Vocalistener2: A singing synthesis system able to mimic a user's singing in terms of voice timbre changes as well as pitch and dynamics," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2011, pp. 453–456.

[3] P. Chandna, M. Blaauw, J. Bonada, and E. Gómez, "WGANsing: A multi-voice singing voice synthesizer based on the wasserstein-gan," in *27th European Signal Processing Conference (EUSIPCO)*, 2019, pp. 1–5.

[4] M. Blaauw, J. Bonada, and R. Daido, "Data efficient voice cloning for neural singing synthesis," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 6840–6844.

[5] S. Choi, W. Kim, S. Park, S. Yong, and J. Nam, "Korean singing voice synthesis based on auto-regressive boundary equilibrium gan," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 7234–7238.

[6] Y. Ren, X. Tan, T. Qin, J. Luan, Z. Zhao, and T. Liu, "DeepSinger: Singing voice synthesis with data mined from the web," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2020, pp. 1979–1989.

[7] J. Bonada and M. Blaauw, "Semi-supervised learning for singing synthesis timbre," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 7083–7087.

[8] E. Nachmani and L. Wolf, "Unsupervised singing voice conversion," *Interspeech*, pp. 2583–2587, 2019.

[9] A. Polyak, L. Wolf, Y. Adi, and Y. Taigman, "Unsupervised cross-domain singing voice conversion," in *Interspeech*, 2020, pp. 801–805.

[10] Y. Luo, C. Hsu, K. Agres, and D. Herremans, "Singing voice conversion with disentangled representations of singer and vocal technique using variational autoencoders," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 3277–3281.

[11] N. Takahashi, M. K. Singh, and Y. Mitsufuji, "Hierarchical disentangled representation learning for singing voice conversion," in *International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–7.

[12] S. Liu, Y. Cao, N. Hu, D. Su, and H. Meng, "Fastsvc: Fast cross-domain singing voice conversion with feature-wise linear modulation," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*, 2021, pp. 1–6.

[13] L. Wan, Q. Wang, A. Papir, and I. Lopez-Moreno, "Generalized end-to-end loss for speaker verification," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4879–4883.

[14] Y. Jia, Y. Zhang, R. J. Weiss, Q. Wang, J. Shen, F. Ren, Z. Chen, P. Nguyen, R. Pang, I. Lopez-Moreno, and Y. Wu, "Transfer learning from speaker verification to multispeaker text-to-speech synthesis," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018 (NeurIPS)*, 2018, pp. 4485–4495.

[15] S. Nercessian, "Zero-shot singing voice conversion," in *Proceedings of the 21th International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 70–76.

[16] ——, "Improved zero-shot voice conversion using explicit conditioning signals," in *Interspeech*, 2020, pp. 4711–4715.

[17] L. Zhang, C. Yu, H. Lu, C. Weng, C. Zhang, Y. Wu, X. Xie, Z. Li, and D. Yu, "DurIAN-SC: Duration informed attention network based singing voice conversion system," in *Interspeech*, 2020, pp. 1231–1235.

[18] S. Choi, S. Han, D. Kim, and S. Ha, "Attentron: Few-shot text-to-speech utilizing attention-based variable-length embedding," in *Interspeech*, 2020.

[19] Y. Y. Lin, C.-M. Chien, J.-H. Lin, H.-y. Lee, and L.-s. Lee, "Fragmentvc: Any-to-any voice conversion by end-to-end extracting and fusing fine-grained voice fragments with attention," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 5939–5943.

[20] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson, "AutoVC: Zero-shot voice style transfer with only autoencoder loss," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, vol. 97, 2019, pp. 5210–5219.

[21] E. Cooper, C. Lai, Y. Yasuda, F. Fang, X. Wang, N. Chen, and J. Yamagishi, "Zero-shot multi-speaker text-to-speech with state-of-the-art neural speaker embeddings," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6184–6188.

[22] J. Kim, S. Kim, J. Kong, and S. Yoon, "Glow-tts: A generative flow for text-to-speech via monotonic alignment search," *Advances in Neural Information Processing Systems*, vol. 33, pp. 8067–8077, 2020.

[23] J. W. Kim, J. Salamon, P. Li, and J. P. Bello, "Crepe: A convolutional representation for pitch estimation," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 161–165.

[24] Y. Ren, Y. Ruan, X. Tan, T. Qin, S. Zhao, Z. Zhao, and T.-Y. Liu, "Fastspeech: Fast, robust and controllable text to speech," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[25] D. Min, D. B. Lee, E. Yang, and S. J. Hwang, "Meta-stylespeech: Multi-speaker adaptive text-to-speech generation," in *International Conference on Machine Learning*, 2021, pp. 7748–7759.

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, 2015, pp. 234–241.

[27] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Ryan, R. A. Saurous, Y. Agiomyrgiannakis, and Y. Wu, "Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 4779–4783.

[28] S. Liu, T. Lin, D. He, F. Li, M. Wang, X. Li, Z. Sun, Q. Li, and E. Ding, "Adaattn: Revisit attention mechanism in arbitrary neural style transfer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 6649–6658.

[29] J. Chen, X. Tan, J. Luan, T. Qin, and T.-Y. Liu, "HiFiSinger: Towards high-fidelity neural singing voice synthesis," *arXiv preprint arXiv:2009.01776*, 2020.

[30] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo, "StarGAN-VC2: Rethinking conditional methods for stargan-based voice conversion," in *Interspeech*, 2019.

[31] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2794–2802.

[32] C. Chu, F. Yang, Y. Lee, Y. Liu, and S. Wu, "Mpop600: A mandarin popular song database with aligned audio, lyrics, and musical scores for singing voice synthesis," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2020, pp. 1647–1652.

[33] Z. Duan, H. Fang, B. Li, K. C. Sim, and Y. Wang, "The NUS sung and spoken lyrics corpus: A quantitative comparison of singing and speech," in *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. IEEE, 2013, pp. 1–9.

[34] J. Yamagishi, C. Veaux, and K. MacDonald, "CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit (version 0.92)," 2019.

[35] Z. Rafii, A. Liutkus, F.-R. Stöter, S. I. Mimilakis, and R. Bittner, "The MUSDB18 corpus for music separation," Dec. 2017.

[36] R. Yamamoto, E. Song, and J. Kim, "Parallel Wavegan: A fast waveform generation model based on generative adversarial networks with multi-resolution spectrogram," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6199–6203.

[37] J. S. Chung, J. Huh, S. Mun, M. Lee, H. Heo, S. Choe, C. Ham, S. Jung, B. Lee, and I. Han, "In defence of metric learning for speaker recognition," in *Interspeech*, 2020, pp. 2977–2981.

[38] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, (NeurIPS)*, 2020.

[39] M. Morise, F. Yokomori, and K. Ozawa, "WORLD: A vocoder-based high-quality speech synthesis system for real-time applications," *IEICE Trans. Inf. Syst.*, vol. 99-D, no. 7, pp. 1877–1884, 2016.

[40] J. Liu, C. Li, Y. Ren, F. Chen, P. Liu, and Z. Zhao, "DiffSinger: Diffusion acoustic model for singing voice synthesis," *CoRR*, vol. abs/2105.02446, 2021.

[41] M. McAuliffe, M. Socolof, S. Mihuc, M. Wagner, and M. Sonderegger, "Montreal forced aligner: Trainable text-speech alignment using kaldi," in *Interspeech*, 2017, pp. 498–502.

SEMANTIC CONTROL OF GENERATIVE MUSICAL ATTRIBUTES

Stewart Greenhill **Majid Abdolshah** **Vuong Le**
Sunil Gupta **Svetha Venkatesh**
Applied Artificial Intelligence Institute, Deakin University, Australia
s.greenhill@deakin.edu.au

ABSTRACT

Deep generative neural networks have been successful in tasks such as composing novel music and rendering expressive performance. Controllability is essential for building creative tools from such models. Recent work in this area has focused on disentangled latent space representations, but this is only part of the solution. Efficient control of semantic attributes must handle non-linearities and holes that occur in latent spaces, whilst minimising unwanted changes to other attributes. This paper introduces SeNT-Gen, a neural traversal algorithm that uses a secondary neural network to model the complex relationships between latent codes and musical attributes. This enables precise editing of semantic attributes that adapts to context. We demonstrate the method using the dMelodies dataset, and show strong performance for several VAE models.

1. INTRODUCTION

Deep generative models show promise for music, with systems that can compose melodies and accompaniments, render expressive performances, or synthesise instrumental sounds and singing voices [1, 2]. While many of these work as "black boxes," *controllability* is an essential factor in creative tools for musicians. This raises important challenges such as controlling musically meaningful aspects of the composition, balancing creativity versus imitation, providing interactivity and refinement, and producing a convincing temporal structure that has a sense of direction [3].

In generative applications a neural network is trained to find a low-dimensional representation for complex data. A common approach uses a Variational Auto-Encoder (VAE) in which an *encoder* learns to transform the data space X into a simpler "latent space" Z. For music X is a representation of a score such as a piano-roll that defines the pitch and duration of notes over time. A *decoder* recovers the original representation from the latent samples. In learning the latent representation the network identifies the essential attributes needed to construct the melody. The dimensions of the latent space represent semantic attributes such as note density [4], syncopation [5], genre [6] or arousal [7]. Novel melodies can be generated by decoding samples drawn from the latent space. Moving in the latent space adjusts the semantic attributes, for example: to smoothly "morph" between two melodies by interpolating a path between their latent positions.

Recent work on control of generative music has focused on regularisation of the latent space, and disentanglement of the semantic attributes. The aim is to relate each semantic attribute of the melody to a unique dimension of the latent space, which allows attributes to be adjusted by adding an "attribute vector" [8, 9] in latent space. This approach involves some challenges. First, there is often a tradeoff between regularisation of the latent space and reconstruction quality. Second, effective disentanglement can only be achieved through supervision [10] and unsupervised approaches are sensitive to inductive biases in both the data set and learning model (such as network model, hyperparameters, random seeds). Third, the relationship between the latent dimension and corresponding semantic attribute value may be non-linear. Fourth, latent spaces may contain "holes" where the decoder produces invalid results [11] and these must be avoided when adjusting attributes, or interpolating paths in the latent space. This means that some additional work is required beyond disentanglement to properly control the values of semantic attributes.

This paper proposes a new method to efficiently traverse latent spaces frequently used in generative music. We introduce the Semantic Neural Latent Traversal method for Generative models (SeNT-Gen), which employs a secondary neural network to model the complex relationship between latent codes and semantic attributes. The SeNT-Gen traversal function predicts the new latent position given a starting position and attribute change, supporting non-linear relationships and adapting to the context of the traversal. We evaluate the performance of SeNT-Gen for musical control using the dMelodies [12] data set.

Our key contributions are:

1. A neural method to traverse the latent space, targeting precise changes to musical attributes, and supporting non-linear contextual relationships between the VAE latent space and semantic attributes;

2. Experiments and analysis of the proposed method using the established dMelodies data set. The method is independent of learning model, and shows best performance for strongly regularised models;

© S. Greenhill, M. Abdolshah, V. Le, S. Gupta, S. Venkatesh. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** S. Greenhill, M. Abdolshah, V. Le, S. Gupta, S. Venkatesh, "Semantic Control of Generative Musical Attributes", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

3. Objective quantitative metrics to measure performance of the method through targeted attribute changes, which improves on notions of "controllability" that are tied to interpolation heuristics.

2. RELATED WORK

Control of generative music is usually based on latent space representations. Some of these approaches are surveyed here, while alternative approaches for images do not directly translate between image and musical domains [12]. Music-VAE [5] uses a hierarchical recurrent VAE to model temporal structure in music. Attribute vectors are shown to partially control attributes such as *note density*, *melodic interval* and *rhythmic syncopation*. MIDI-VAE [6] learns songs using a a parallel VAE with shared latent space, and an attached style classifier enables a chosen *style* (classic, jazz, pop, Bach, Mozart) to be applied to the output. GLSR-VAE [4] uses a novel regularisation method to control *note density* of chorale-style melodies. Music FaderNets [7] proposes a Gaussian mixture VAE for learning piano performance, with control over the *arousal* or "energy level" based on rhythm and note density. AR-VAE [13] defines a supervised regularisation method which is applied to controlling *rhythmic complexity*, *pitch range*, *note density*, and *contour* for melodies trained from chorales and folk tunes. Kawai and colleagues [14] use a VAE with adversarial classifier-discriminator to condition the decoder on semantic attributes. This model is trained on folk tunes to control orthogonal attributes such as *number of notes*, *pitch variability*, *chromatic motion* and *amount of arpeggiation*.

Apart from latent space approaches, other methods generally require ad-hoc modifications to the probability distribution of a model. Transformers are used to generate music with long-term structure [15], but control is limited to providing a prompt stimulus for the system to extend. Pop Music Transformer [16] uses a Transformer-XL model to learn expressive piano performances. Control over *tempo* and *chord* is achieved by masking out corresponding event probabilities in the model output. Coconet [17] uses a convolutional model to generate chorales in the style of Bach. Cococo [18] adds "semantic sliders" for *conventional/surprising* and *happy/sad* output using "soft priors" to modify the model's sampling distribution.

Several factors make it difficult to compare the results of these studies. Firstly, there are a wide variety of model architectures, and each implements its own encoding of the training data into a form suitable for model training. This in turn depends on the data-sets used, which vary from simple scores in ABC notation, to traditional scores, to MIDI transcriptions of actual performances either recorded from a digital keyboard or automatically extracted from audio recordings. Secondly, while there is a focus on disentangled representations, these are not in themselves control methods. Even in a disentangled latent space, more work is required to accurately achieve a specific value of an attribute, dealing with potential non-linearities and holes in the latent space. In the absence of a particular control algorithm some works assess "controllability" through interpolation [7, 11, 14] though there is no consistency in the metrics used.

Feature	Values	Description
Tonic	12	Notes C, C#, D ..., B
Octave	3	Octave 4, 5, or 6
Scale	3	major, minor, or blues
Rhythm Bar 1, 2	28	$\binom{8}{6}$ codes for 6 note onsets
Arp Chord 1, 2, 3, 4	2	arpeggio direction up or down

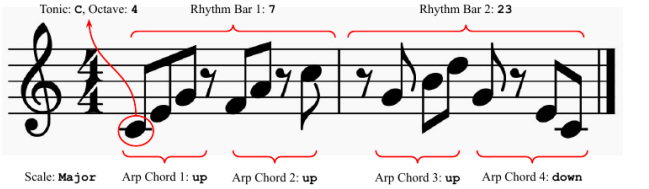


Figure 1. Each dMelodies two-bar melody is described by 9 attributes. *Tonic*, *Octave* and *Scale* apply to the entire melody. Separate *Rhythm* attributes apply to each bar. *Arpeggio* attributes apply to each of the 4 chords. The table (top) shows the number of values for each attributes, and an example from [12] is shown below.

The dMelodies dataset [12] has been proposed for evaluation of musical disentanglement learning, similar to dSprites in the image domain [19]. dMelodies includes 1,354,752 unique two-bar melodies, algorithmically constructed with independent factors of variation, with each comprised of 4 arpeggios in a I-IV-V-I chord pattern (see Table 1). Attributes are discrete, and most (except for *Tonic* and *Octave*) are categorical. dMelodies provides three unsupervised reference models including β-VAE [20], and a subsequent paper [11] adds three supervised models: I-VAE [21], AR-VAE [13], and S2-VAE [22]. Three measures of disentanglement are implemented [12]: *Mutual Information Gap* [23], *Modularity* [24], and *Separated Attribute Predictability* [25]. On these measures, supervised learning is shown to give better disentanglement, without sacrificing reconstruction quality [11].

2.1 Metrics

Suppose the VAE decoder $\mathcal{G}$ generates an object $x = \mathcal{G}(z)$ for latent sample z, and let $c = [c_1, c_2, ..., c_K]$ be the K semantic attributes of x. Disentanglement is formalised through the concepts of *consistency* and *restrictiveness* [26] and establishes a relationship between a latent dimension z_i and a corresponding semantic attribute c_i. *Consistency* says that when z_i is fixed, c_i of the generated x never changes. *Restrictiveness* says that when only z_i is changed, the change is restricted to c_i and there is no change to other attributes c_j for $j \neq i$.

Attributes may be controlled through a traversal function $\mathcal{T}_k(z, c_k, c_k^*)$ which predicts the latent value z^* required to change attribute k of z from c_k to c_k^*. The accuracy of $\mathcal{T}_k$ can be evaluated for a set of changes (z, c_k, c_k^*) by measuring the deviation of the result from the target, and any side effects on non-target attributes. Alternative approaches assess "controllabilty" in the absence of such a traversal function. Instead, interpolation is used to tra-

VAE	To	Oc	Sc	R1	R2	A1	A2	A3	A4
β	1	.95	.14	.64	.64	.28	.23	.28	.24
AR	1	.95	.34	.09	.09	.02	.02	.02	.02
I	1	.99	.36	.73	.77	.50	.52	.54	.53
S2	1	.94	.12	.65	.70	.29	.23	.34	.25

Table 1. Latent Density Ratio (LDR) for four dMelodies VAE models. Attributes are Tonic (To), Octave (Oc), Scale (Sc), Rhythm Bar (R) and Arp Chord (A).

verse from a position z in the latent space, along the corresponding dimension z_d between a minimum and maximum value over a number of steps. This is repeated over a small batch of points. Three measures are defined by [7] based on the work of [26]: *Consistency* measures how constant the attribute is across the batch for the same value of z_d, *Restrictiveness* measures how constant other attributes are when z_d changes, and *Linearity* measures how linear the attribute is with respect to the latent dimension z_d. An *attribute change matrix* $A(d, n)$ is proposed by [11] which computes the net change in the n^{th} attribute while traversing regularised dimension d. When an attribute is well controlled A should be high on the diagonal, and low elsewhere, and these relationships can be inspected by visualising the matrix. Another measure of controllability is the correlation between the interpolated value z_d and the resulting attribute [14], which measures linearity but not restrictiveness.

Another important factor is the *Latent Density Ratio* (or *LDR*), the proportion of a batch of random latent samples that decode with valid attributes [11]. As shown in Table 1, errors in dMelodies occur most frequently with the *Scale*, *Rhythm Bar* and *Arp Chord* attributes. If notes are generated that are outside the three defined scales (major, minor, blues) the *Scale* attribute is invalid. If a bar does not have exactly 6 note onsets the *Rhythm Bar* attribute does not match one of the 28 codes, and is invalid. If the chord notes are not consistently ascending or descending, the *Arp Chord* attribute is invalid. A traversal function should aim to be more accurate than these base-line LDR rates.

3. METHOD

3.1 Neural Latent Traversal

This section describes SeNT-Gen as applied to musical VAE models. Further details are available for a GAN-based image synthesis application [27].

Let x be a sample of the data space $X = \mathbb{R}^N$, a piano-roll encoding of a melody. Let z be a sample of the latent space $Z = \mathbb{R}^D$ of the VAE model. Let $c = [c_1, c_2, ..., c_K]$ be the vector of K semantic attributes of x, and $R_1(.)$, ..., $R_k(.)$ be K functions that compute the value of attribute k of x in the normalised range $[0, 1]$: $c_k = R_k(x)$. Let $\mathcal{F}(.)$ be the encoder, and $\mathcal{G}(.)$ be the decoder of the VAE, so that $\mathcal{G}(\mathcal{F}(x)) \approx x$. We use x^* for the *target value* of x, and $\hat{x}$ for its *predicted* or *achieved* value.

Given a sample x, we aim to find the modified x^* such that the value of attribute k is changed from c_k to c_k^* while all other attributes remain unchanged: $c_i^* = c_i, i \neq k$.

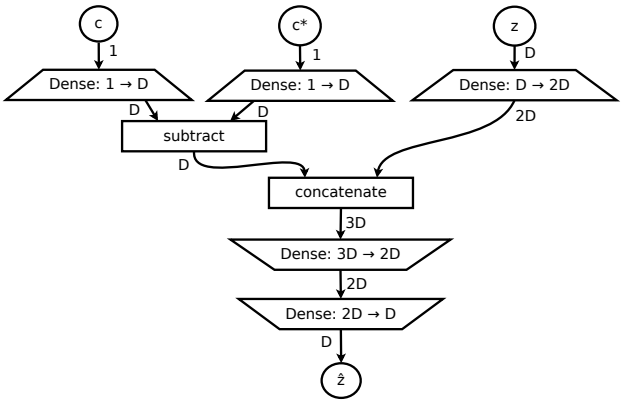


Figure 2. Implementation of SeNT-Gen neural traversal function $\mathcal{T}_k(z, c_k, c_k^*)$. Input is the latent code z, and the desired attribute change from c_k to c_k^*. Each Dense layer includes ReLU activation, except for the Tanh activation on the final layer. Output is the latent code $\hat{z}$ which approximates the solution z^*.

This adjustment will be done in the latent space. Given the latent encoding $z = \mathcal{F}(x)$, we seek a modified z^* that generates $x^* = \mathcal{G}(z^*)$, with the desired attribute value $c_k^* = R_k(\mathcal{G}(z^*))$.

SeNT-Gen implements contextual traversal of the latent space for each of the attributes. The traversal function $\mathcal{T}_k$ predicts the new value $\hat{z}$ given the old value z, and the old and new values of the attribute c_k and c_k^*:

$$\hat{z} = \mathcal{T}_k(z, c_k, c_k^*)$$

The traversal function $\mathcal{T}_k$ is implemented by training the neural network shown in Figure 2 which outputs the value $\hat{z}$, an approximate solution for z^*.

Three constraints are imposed during training. The first constraint is to minimise the perceptual loss by ensuring that the changed attribute value $\hat{c}_k = R_k(\mathcal{G}(\hat{z}))$ is close to the target c_k^*:

$$\mathcal{L}_c = \mathbb{E}_{z \sim P(Z)} \left[\| R_k(\mathcal{G}(\hat{z})) - c_k^* \|_2^2 \right] \quad (1)$$

The term $\| R_k(\mathcal{G}(\hat{z})) - c_k^* \|_2$ represents the deviation between $\hat{c}_k$ and the target c_k^*, and is in the range [0,1]. When R_k is invalid for $\mathcal{G}(\hat{z})$ a deviation of 1 is used instead.

The other two constraints are required to align the attributes of the traversal function with the relevant dimensions of the latent space. In a disentangled latent space each semantic attribute corresponds to one latent dimension, but in general there will likely be a small number of relevant dimensions that are strongly related to each attribute. For *relevant* dimensions r, $\hat{z}_r$ should be be close to the correct solution z_r^*. For the other *irrelevant* dimensions i, $\hat{z}_i$ should be close to the original location z_i.

Relevance is expressed using the vector $\rho_k \in \mathbb{R}^D$, which is 1 for relevant dimensions and 0 otherwise. Under supervised training, this relation will be known and for a consistent numbering of attributes and dimensions $\rho_k[i] = 1$ for $i = k$, and 0 for $i \neq k$. Otherwise, for unsupervised models ρ_k can be calculated using mutual

information as described below. The two remaining loss functions are thus:

$$\mathcal{L}_I = \mathbb{E}_{z,z^* \sim P(Z)} \left[\| {\rho_k}^T (\hat{z} - z^*) \|_2^2 \right] \quad (2)$$

$$\mathcal{L}_{\neg I} = \mathbb{E}_{z,z^* \sim P(Z)} \left[\| (1 - \rho_k)^T (\hat{z} - z) \|_2^2 \right] \quad (3)$$

Equation 2 pulls $\hat{z}$ towards z^* for relevant dimensions, and Equation 3 pulls $\hat{z}$ towards z for irrelevant dimensions.

When ρ_k is not known a-priori it can be calculated using the mutual information (MI) between the dimensions $d \in \{1, ..., D\}$ of z and the c_k values:

$$\rho_k = \text{threshold} \left(\text{softmax}(\text{MI}(z_d, c_k)), \gamma \right) \quad (4)$$

Where γ is a hyper-parameter controlling the number of latent dimensions related to each semantic attribute. Equation 4 selects γ dimensions of z that have the most mutual information with each attribute c_k.

3.2 Latent Traversal Training

The traversal function $\mathcal{T}_k$ is trained on similar pairs of latent vectors (z, z^*) which differ only in attribute c_k. Such pairs are good examples of attribute modifications. We sample pairs of the data space (x, x^*), and compute their attributes (c_k, c_k^*), where $c_k = R_k(x)$, and $c_k^* = R_k(x^*)$. We use the encoder compute their latent representation (z, z^*), where $z = \mathcal{F}(x)$ and $z^* = \mathcal{F}(x^*)$. We discard pairs where there is a significant difference in attributes other than k. A training pair is considered valid if:

$$\sum_{j \neq k} \| c_j - c_j^* \|_2 \leq \epsilon, \quad \text{for } j \in \{1, ..., K\} \quad (5)$$

Where $\epsilon \geq 0$ is a slack parameter. One implementation is to enumerate all pairs from a set of candidate samples, and adjust ϵ to be as small as possible while also yielding enough similar pairs.

After generating the training pairs, we train the traversal model $\mathcal{T}_k$ by minimising the loss:

$$\mathcal{L}_k \triangleq \lambda_1 \mathcal{L}_I + \lambda_2 \mathcal{L}_{\neg I} + \lambda_3 \mathcal{L}_c \quad (6)$$

where λ_1, λ_2, and λ_3 are hyper-parameters that balance perceptual loss versus semantic relevance.

4. EXPERIMENTS

We train each of the four dMelodies learning models (β-VAE, AR-VAE, I-VAE and S2-VAE) for three different hyperparameter settings and 3 different random seeds, a total of 36 models. Input is a two-bar melody, with a latent space of $D = 32$ dimensions. For β-VAE we vary $\beta \in \{0.2, 1, 4\}$, and for the other models we vary the regularisation strength $\Gamma \in \{0.1, 1, 10\}$ for $\beta = 0.2$. The default settings use 1016k melodies (75%) for training the original models, which leaves 338k melodies (25%) as candidates for training and testing the SeNT traversal function $\mathcal{T}_k$. Setting $\epsilon = 0$ yields between 42k and 157k pairs (see Equation 5) and from these we allocate 80% for training and 20% for testing. Since dMelodies is generated by combinatorial expansion, attributes with the most states (*Tonic*, *Rhythm Bar*) have the most pairs, while those with the fewest states (*Arp Chord*) have the least. Latent codes z are normalised to the range $[-1, 1]$, and attributes c_k to $[0, 1]$. Neural traversal is trained for $\gamma = 3$ (see Equation 4), $\lambda_1 = \lambda_2 = \lambda_3 = 1$ (Equation 6).

4.1 Performance Metrics

Performance of the traversal function $\mathcal{T}_k$ is evaluated by measuring accuracy on a set of attribute changes. A set of testing pairs (z, z^*) is generated using the same method that generated the training pairs (see Equation 5). These define a starting state z and an attribute change from c_k to c_k^* that leaves other attributes unchanged. Using the traversal function we predict the new latent code $\hat{z} = \mathcal{T}_k(z, c_k, c_k^*)$, and compute its attributes $\hat{c}_i = R_i(\mathcal{G}(\hat{z}))$ for all i. This approach ensures that only feasible attribute changes are tested, and that these have been unseen during training.

For a targeted change of an attribute, the important measures are: (1) How far is the edited attribute value $\hat{c}_k$ from the desired target c_k^*? and (2) How far are the unedited attributes $\hat{c}_u$ from the original values c_u? The *target deviation* Δ_k for target attribute k is defined as:

$$\Delta_k = |\hat{c}_k - c_k^*| \quad (7)$$

where $\hat{c}_k = R_k(\mathcal{G}(\hat{z}))$. The *non-target deviation* ∇_u for non-target attribute u is defined as:

$$\nabla_u = |\hat{c}_u - c_u| \quad (8)$$

When Δ_k is low, the traversal achieves the desired attribute k value. When ∇_u is low, the traversal avoids unintended changes to other attributes u. When combined, these measures are similar to the attribute change matrix of [11], except that Δ_k is with respect to specific target values, rather than arbitrary interpolation points.

Occasionally errors occur in the generated melodies, due to the decoder quality and the proximity of samples to holes in the latent space. Ideally $\mathcal{T}_k$ should avoid these holes, and to measure this we define the *Target Density Ratio* (or TDR) to be the proportion of the decoded target $\hat{z}$ values with valid attributes. We aim for TDR to be substantially higher than the underlying LDR (defined in 2.1).

5. RESULTS

For brevity we summarise the 36 models by selecting the best performing hyper-parameters, and aggregate over the three random seeds. For supervised models these are the settings with the most regularisation $\Gamma = 10$, and for β-VAE, the median $\beta = 1$. Source code is available online with further results and technical details. [1]

Figure 3 shows Mutual Information between semantic attributes (vertical axis) and latent dimensions (horizontal axis). The supervised S2-VAE (top) shows good disentanglement, with each attribute uniquely related to one

[1] https://github.com/stewartgreenhill/sentgen

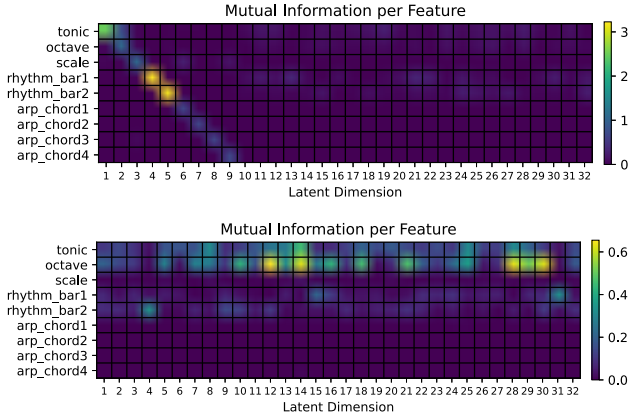


Figure 3. Mutual Information between semantic attributes and dimensions of the latent space for supervised S2-VAE (top), and unsupervised β-VAE (bottom).

of the first 9 latent dimensions. The strongest relationships are for the attributes with the most states: *Tonic* and *Rhythm Bar*. The unsupervised β-VAE (bottom) shows some strong but less well separated relationships, and some very weak relationships: *Scale*, *Arp Chord*. Mutual Information is important in SeNT-Gen for determining ρ_k which aligns semantic attributes to the latent space (Equations 2–4).

Figure 4 shows SeNT-Gen traversal accuracy for the S2-VAE. The bottom chart shows mean target deviation Δ_k for target attributes k. Smaller deviations are better. For most attributes the deviation is 2% or less, and the worst performance is 8% for attribute *Tonic*. The top chart shows the mean non-target deviation ∇_k, with non-target attributes on the vertical axis, and target attributes on the horizontal axis. The *Scale* attribute is most influenced by changes to other attributes, particularly *Tonic*, and *Octave* where the influence approaches 60%. This makes sense since these attributes are both changing the overall pitch of the melody, and it only requires one transposition "error" amongst the 12 melody notes to cause the scale to be invalid or altered. To a lesser extent the *Arp Chord* attributes are also susceptible for the same reason. Changes to *Arp Chord* attributes are the most accurate, with no deviation in the target or non-target attributes other than *Scale*.

Figure 5 shows SeNT-Gen traversal accuracy for the β-VAE which is the worst performing model. Rhythm attributes show a good target deviation of 3%, with the pitch based attribute deviations ranging from 7 to 22%. Non-target errors are lowest for changes to *Rhythm Bar 1 & 2*, but are generally much higher than for the S2-VAE. Here *Scale* and *Arp Chord* attributes are most influenced by changes to other attributes. This is expected since these attributes are only weakly related to dimensions of the latent space (see Figure 3) so are essentially invisible to the traversal constraints.

Another way to evaluate traversal accuracy is to look at correlation R^2 between target and achieved attribute values. Figure 6 shows the target c_k^* value (horizontal) versus the achieved $\hat{c}_k$ (vertical) for rhythm_bar1, the attribute

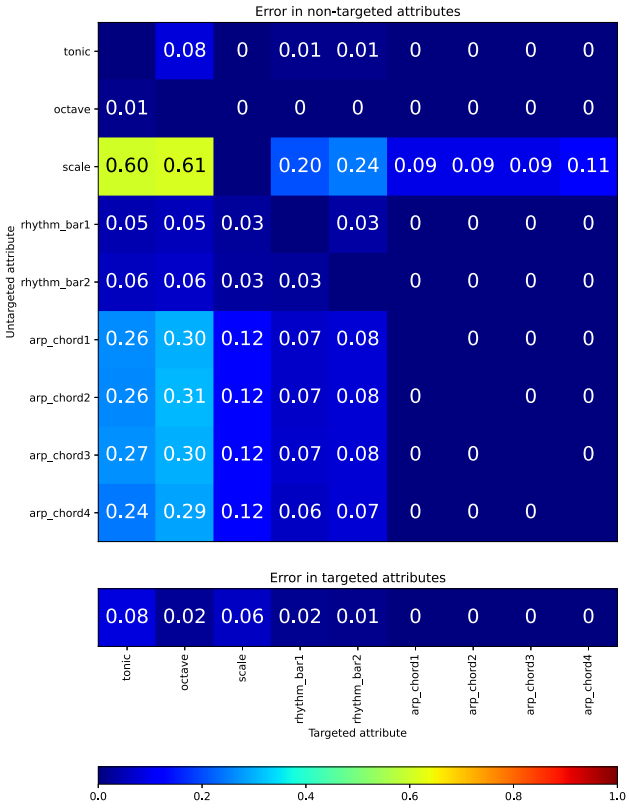


Figure 4. Accuracy of S2-VAE traversal, showing *target deviation* Δ (bottom), and *non-target deviation* ∇ (top). Target attributes are on the horizontal axis, and non-targets on the vertical.

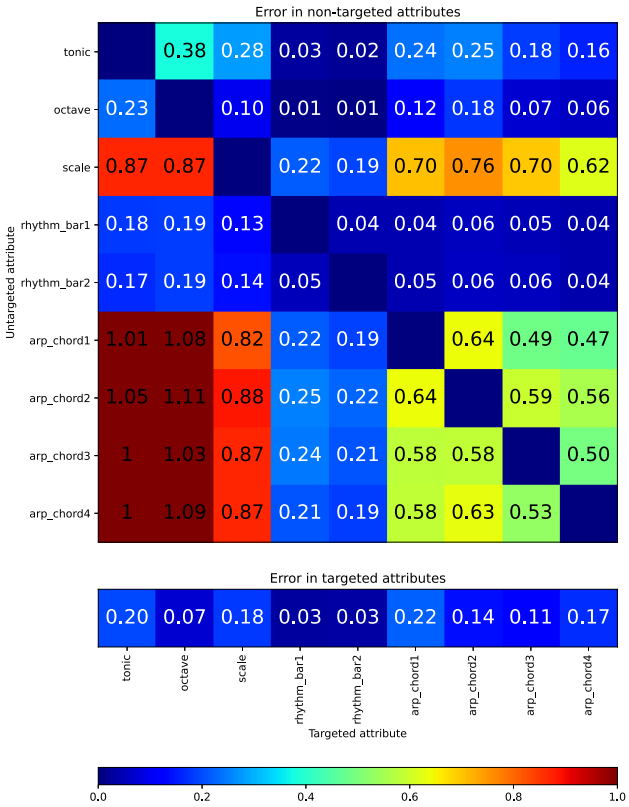


Figure 5. Accuracy of β-VAE traversal, showing *target deviation* Δ (bottom), and *non-target deviation* ∇ (top). Target attributes are on the horizontal axis, and non-targets on the vertical.

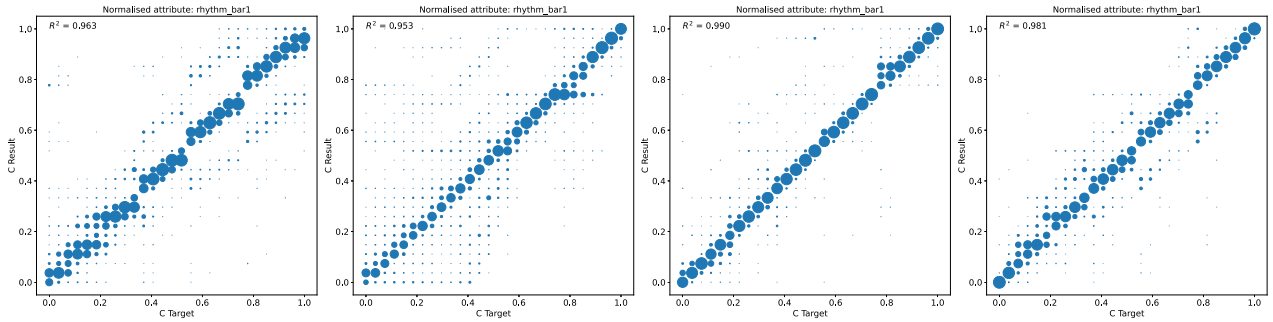


Figure 6. Correlation R^2 between normalised target c_k^* (horizontal axis) and result $\hat{c}_k$ (vertical axis) for $k = \text{rhythm\_bar1}$. Dot area is proportional to number of samples. Models are (left to right) β-VAE, AR-VAE, I-VAE, and S2-VAE.

VAE	To	Oc	Sc	R1	R2	A1	A2	A3	A4
β	.09	.78	.19	.96	.97	.11	.45	.56	.32
AR	.07	0	-1.3	.95	.97	.98	.99	1	1
I	.26	.34	.98	.99	.97	1	1	1	1
S2	.77	.93	.78	.98	.99	1	1	1	1

Table 2. Correlation R^2 between $\hat{c}_k$ versus c_k^* for four dMelodies VAE models. Attributes are Tonic (To), Octave (Oc), Scale (Sc), Rhythm Bar (R) and Arp Chord (A).

VAE	To	Oc	Sc	R1	R2	A1	A2	A3	A4
β	1	1	.30	.92	.92	.57	.52	.64	.62
AR	1	1	.49	.76	.82	.72	.73	.73	.73
I	1	1	.92	.95	.91	.90	.91	.90	.92
S2	1	.99	.65	.93	.93	.91	.90	.91	.91

Table 3. Target Density Ratio (TDR) for four dMelodies VAE models. See Table 1 for Latent Density Ratio (LDR).

with the most states. The frequency and location of errors can be visualised by inspecting the off-diagonal elements. Table 2 shows the R^2 values for each of the attributes over four dMelodies VAE models, excluding samples which yield invalid results. As expected, this measure shows high correlations where target deviation Δ is low. The three supervised models show strong control over the *Rhythm Bar* and *Arp Chord* attributes with weaker control over *Tonic*, *Octave* and *Scale*.

Table 3 shows the Target Density Ratio (TDR), the proportion of the decoded targets $\hat{x} = \mathcal{G}(\hat{z})$ that have valid attributes, thus avoiding potential holes in the latent space. The best performer from this perspective is the I-VAE which exhibits good TDR for all attributes. All models score better than the corresponding latent density ratio (LDR, Table 1). Notably, the AR-VAE shows good control over *Arp Chord* with TDR=0.73 despite scoring very low LDR=0.02 for these attributes.

6. CONCLUSION

This paper presents a novel algorithm to control musical attributes of deep generative models. The SeNT-Gen method implements a neural traversal function $\mathcal{T}_k(z, c_k, c_k^*)$ that predicts the latent position z^* required to change attribute k of z from c_k to c_k^*. Previous works in this field focus on disentanglement but do not implement traversal functions and instead assess controllability via latent space interpolation which does not allow the specification of a particular target value c_k^* for the controlled attribute.

The SeNT-Gen method is demonstrated using the dMelodies data set and various VAE models. Performance is strongest for highly regularised models. The best performance was obtained using the S2-VAE model, which shows strong control over most attributes, and few side-effects except for *Scale*. Some attributes that depend on note pitch (*Scale* and *Arp Chord*) are significantly less stable when adjusting overall pitch via *Tonic* or *Octave*, and also show low LDR scores. The definition of these attributes may be fragile to small changes in the melody notes. The Target Density Ratio (TDR) measures the robustness of the method to holes in the latent space, aggregating factors related to the latent space regularisation, the traversal algorithm, and the fidelity of the decoder. TDR for I-VAE and S2-VAE is good for most attributes, and AR-VAE shows very strong improvement over a poor baseline LDR.

Although each SeNT-Gen traversal function controls only one attribute, more complex operations involving multiple attributes could be performed using a sequence of separate attribute changes. Future improvements to the algorithm could include better support for categorical variables, as well as mixtures different variable types. Categorical variables would normally be one-hot encoded, but alternative distance metrics might be required, for example Jaccard distance where Euclidean distance is currently used (Equations 1 and 5).

While demonstrated here using VAE models, SeNT-Gen can also be used in other latent space models such as Generative Adversarial Networks (GAN). There is an underlying assumption that the relationship between semantic attributes and the latent dimensions will be fairly sparse. This is normally true for supervised models, but can also apply in other models too. In any case the hyper-parameter γ should be chosen to reflect this. An extensive study of the other hyper-parameters λ_1, λ_2, and λ_3 was outside the scope of this work, but fine-tuning these values through meta-optimisation may improve the overall performance.

7. REFERENCES

[1] S. Ji, J. Luo, and X. Yang, "A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions," *arXiv preprint arXiv:2011.06801*, 2020.

[2] G. Peeters and G. Richard, "Deep learning for audio and music," in *Multi-faceted Deep Learning*. Springer, 2021, pp. 231–266.

[3] J.-P. Briot and F. Pachet, "Music generation by deep learning – challenges and directions," *arXiv preprint arXiv:1712.04371*, 2017.

[4] G. Hadjeres, F. Nielsen, and F. Pachet, "GLSR-VAE: Geodesic latent space regularization for variational autoencoder architectures," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2017, pp. 1–7.

[5] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *International conference on machine learning*. PMLR, 2018, pp. 4364–4373.

[6] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer, "MIDI-VAE: Modeling dynamics and instrumentation of music with applications to style transfer," in *Proceedings 19th International Society for Music Information Retrieval Conference*, 2018.

[7] H. H. Tan and D. Herremans, "Music FaderNets: Controllable music generation based on high-level features via low-level feature modelling," in *Proc. of the International Society for Music Information Retrieval Conference*, 2020.

[8] T. White, "Sampling generative networks," *arXiv preprint arXiv:1609.04468*, 2016.

[9] D. T. Chang, "Latent variable modeling for generative concept representations and deep generative models," *arXiv preprint arXiv:1812.11856*, 2018.

[10] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem, "Challenging common assumptions in the unsupervised learning of disentangled representations," in *International Conference on Machine Learning*. PMLR, 2019, pp. 4114–4124.

[11] A. Pati and A. Lerch, "Is disentanglement enough? on latent representations for controllable music generation," in *Proceedings 22nd International Society for Music Information Retrieval Conference*, 2021.

[12] A. Pati, S. Gururani, and A. Lerch, "dMelodies: A music dataset for disentanglement learning," in *Proceedings 21st International Society for Music Information Retrieval Conference*, 2020.

[13] A. Pati and A. Lerch, "Attribute-based regularization of latent spaces for variational auto-encoders," *Neural Computing and Applications*, vol. 33, no. 9, pp. 4429–4444, 2021.

[14] L. Kawai, P. Esling, and T. Harada, "Attributes-aware deep music transformation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 2020.

[15] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer," *arXiv preprint arXiv:1809.04281*, 2018.

[16] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[17] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *Proceedings 18th International Society for Music Information Retrieval Conference*, 2017.

[18] R. Louie, A. Cohen, C.-Z. A. Huang, M. Terry, and C. J. Cai, "Cococo: AI-steering tools for music novices co-creating with generative models." in *Proceedings 25th International Conference on Intelligent User Interfaces*, 2020.

[19] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner, "dSprites: Disentanglement testing sprites dataset," 2017. [Online]. Available: https://github.com/deepmind/dsprites-dataset/

[20] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[21] T. Adel, Z. Ghahramani, and A. Weller, "Discovering interpretable representations for both deep generative and discriminative models," in *International Conference on Machine Learning*. PMLR, 2018, pp. 50–59.

[22] F. Locatello, M. Tschannen, S. Bauer, G. Rätsch, B. Schölkopf, and O. Bachem, "Disentangling factors of variation using few labels," in *Proceedings 8th International Conference on Learning Representations*, 2020.

[23] R. T. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[24] K. Ridgeway and M. C. Mozer, "Learning deep disentangled embeddings with the f-statistic loss," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[25] A. Kumar, P. Sattigeri, and A. Balakrishnan, "Variational inference of disentangled latent concepts from unlabeled observations," *arXiv preprint arXiv:1711.00848*, 2017.

[26] R. Shu, Y. Chen, A. Kumar, S. Ermon, and B. Poole, "Weakly supervised disentanglement with guarantees," in *International Conference on Learning Representations*, 2019.

[27] M. Abdolshah, H. Le, T. K. George, V. Le, S. Gupta, S. Rana, and S. Venkatesh, "Neural latent traversal with semantic constraints," 2022. [Online]. Available: https://openreview.net/forum?id=ODdaICh-7dK

MUSIC REPRESENTATION LEARNING BASED ON EDITORIAL METADATA FROM DISCOGS

Pablo Alonso-Jiménez
Music Technology Group
Universitat Pompeu Fabra
pablo.alonso@upf.edu

Xavier Serra
Music Technology Group
Universitat Pompeu Fabra
xavier.serra@upf.edu

Dmitry Bogdanov
Music Technology Group
Universitat Pompeu Fabra
dmitry.bogdanov@upf.edu

ABSTRACT

This paper revisits the idea of music representation learning supervised by editorial metadata, contributing to the state of the art in two ways. First, we exploit the public editorial metadata available on Discogs, an extensive community-maintained music database containing information about artists, releases, and record labels. Second, we use a contrastive learning setup based on COLA, different from previous systems based on triplet loss. We train models targeting several associations derived from the metadata and experiment with stacked combinations of learned representations, evaluating them on standard music classification tasks. Additionally, we consider learning all the associations jointly in a multi-task setup. We show that it is possible to improve the performance of current self-supervised models by using inexpensive metadata commonly available in music collections, producing representations comparable to those learned on classification setups. We find that the resulting representations based on editorial metadata outperform a system trained with music style tags available in the same large-scale dataset, which motivates further research using this type of supervision. Additionally, we give insights on how to preprocess Discogs metadata to build training objectives and provide public pre-trained models.

1. INTRODUCTION

Developing robust representations is crucial to improving the performance of existing supervised Music Information Retrieval (MIR) tasks on datasets with few annotations. While conventional approaches are based on classification [1–3], other directions include self-supervised strategies [4–9] including leveraging generative models [10], supervision by editorial metadata [11–15], playlist co-ocurrences [16], co-listen statistics [15], natural language text [17], or combinations of them [10, 14–16]. While self-supervised systems are narrowing the performance gap with the supervised approaches, the latter seem to be reaching a performance ceiling in the academic research, partially due to the tremendous difficulty to collect larger annotated datasets.

Using editorial metadata as a source of supervision was already considered in other domains, for example, in scientific publication classification [18] or film recommendation [19]. Likewise, music is naturally rich in metadata. For example, physical formats typically contain detailed editorial information on their covers, digital audio files support containers such as ID3 for this purpose, and most streaming platforms offer album or artist-level browsability. Most music digital service providers routinely require such metadata from content uploaders, and therefore it is often available for music collections in the industry by default. Furthermore, editorial metadata does not suffer from the subjectivity problems common for music tags and tends to be more consistent. Because of these reasons, such metadata allows the creation of potentially larger and less noisy datasets than tag annotations.

Park *et al.* showed that certain types of editorial metadata (artist name) could be used to learn music representations using the triplet loss [11]. Motivated by their study and following the recent success of unsupervised representation learning approaches such as SimCLR [9, 20], we are interested whether modifying such systems to operate with similarity relations based on editorial metadata improves the learned features.

Our proposal differs from previous works on metadata-based music representation learning in two aspects. First, we use editorial metadata from Discogs, [1] a website containing an extensive metadata database including the artists, year, country, record label, and genres/styles of each release. Discogs publishes monthly dumps of their data under the Creative Commons CC0 license that we use to label 3.3 million tracks from our in-house data collection, allowing us to extract conclusions about systems trained on large datasets. Second, we adopt a contrastive approach already performant in a self-supervised setup instead of siamese networks. Specifically, we choose COLA [21] for its simplicity, operating directly on spectral representations, and requiring no data augmentation, which makes it efficient and suitable for large-data regimes. COLA works by maximizing the bilinear similar-

© P. Alonso, X. Serra, and D. Bogdanov. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0).
Attribution: P. Alonso, X. Serra, and D. Bogdanov, "Music Representation Learning Based on Editorial Metadata From Discogs", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] https://www.discogs.com/search

ity between a pair of mel-spectrogram patches (anchor and positive) cropped from the same audio clip while minimizing it for the rest of the patches in the batch. We propose to modify this self-supervised approach by constructing the anchor/positive pairs according to the different types of metadata considered (i.e., same track, release, artist, and record label). Additionally, we explore whether different metadata associations generate complementary information by combining the embeddings produced by their respective models and creating multi-task systems jointly optimized to learn them.

To summarize, we propose investigating the usability of metadata as an inexpensive source of supervision to train contrastive feature extractors for music classification. On top of the original self-supervised COLA approach, we experiment with associations based on editorial metadata to construct the positive pairs. We report results on public benchmarking datasets to facilitate comparison with the SOTA and provide publicly available pre-trained embedding and downstream models.

2. RELATED WORK

This paper combines three ideas: supervision based on editorial metadata, usage of the Discogs database, and the recent advances in contrastive learning for MIR. In this section, we review relevant works on these topics.

2.1 Metadata-based music representation learning

Park *et al.* shows that the artist name can be used as training target, with the resulting features being close in performance to those obtained from systems trained on crowdsourced tags [11]. As the artist information is too sparse to be learned efficiently in a classification setup, they approached it as a metric learning task by creating triplets with anchor and positive samples belonging to the same artist and a negative sample belonging to a different one. In a posterior study, they extend the approach to learning track-, album-, artist-level associations, and a combination of all, finding that the latter two provide the best representation [13].

Kim *et al.* propose dealing with the high dimensionality of artist vectors by summarizing them into Latent Dirichlet Allocation topics [22] to create targets suitable for a traditional classification setup [12]. Other works exploit editorial metadata (release year) among other learning sources in multi-task setups [14].

2.2 Discogs in MIR research

The Discogs database has already been used for research. Bogdanov *et al.* propose using it in the context of music recommendation [23], MIR, and computational musicology [24]. The former study proposes a recommendation system based on the similarity between artist representations in the form of a tag cloud of the associated genre, style, record label, and release year and country metadata. The latter illustrates the potential uses of the database on various cultural analysis examples including the evolution of physical distribution formats, genre and style trends, and their co-occurrences. Similarly, some studies analyze music artist collaboration networks [25–28].

In the context of music genre classification, the AcousticBrainz Genre dataset contains mappings across different music genre taxonomies, including the one from Discogs [29]. Hennequin *et al.* use the genre and style labels from Discogs for genre tag disambiguation [30].

2.3 Contrastive Learning in MIR

The MIR community has lately adopted contrastive learning approaches, both supervised and self-supervised. Regarding supervised methods, Favory *et al.* use a contrastive objective to align embedded representations of tags and audio [31, 32]. In a subsequent study, the system is enriched with playlists-track interactions as an additional modality to align [16].

On the self-supervised side, several contrastive systems from the computer vision domain have been adapted and evaluated for music-related tasks. CLMR [5] adapts SimCLR [20] to operate on waveforms using musically-meaningful augmentations. BYOL-A [6] relies on the BYOL [33] framework and adapts it to the audio domain by proposing specific augmentations and evaluating it on several audio tasks, including instrument classification. S3T [8] combines the MoCo framework [34] with Swin Transformers [35] to learn music classification features. Wang *et al.* [9] modify SimCLR by using a normalization-free SlowFast [36] backbone and improve the performance in several audio tasks, including music-auto-tagging. PEMR [7] deals with the lack of temporal resolution of existing systems by learning to mask important or irrelevant parts of the mel-spectrogram to produce self-augmented positive/negative samples. COLA [21] is a simple contrastive model for audio that takes random mel-spectrogram patches of the same clip instead of augmented versions of the same patch as anchor/positive pairs. It relies on the bilinear similarity to compute a distance matrix between all the anchors and positives and uses the categorical cross-entropy loss to maximize the similarity between correspondent anchor/positive pairs while using the rest of the positives in the batch as negatives.

We follow a straightforward implementation of COLA, but instead of relying on the same audio clip (purely self-supervised), we form the anchor/positive pairs according to relationships from the metadata. The complete pipeline is represented in Figure 1 (left). This is, to the best of our knowledge, the first usage of this framework in the context of MIR.

3. METHODOLOGY

We are interested in assessing the representation power of features derived from associations of commonly available editorial metadata. This goal is motivated by previous works that already showed the usability of the track, artist, and album similarities as supervision targets [13]. As these works already studied the influence of the dataset size and

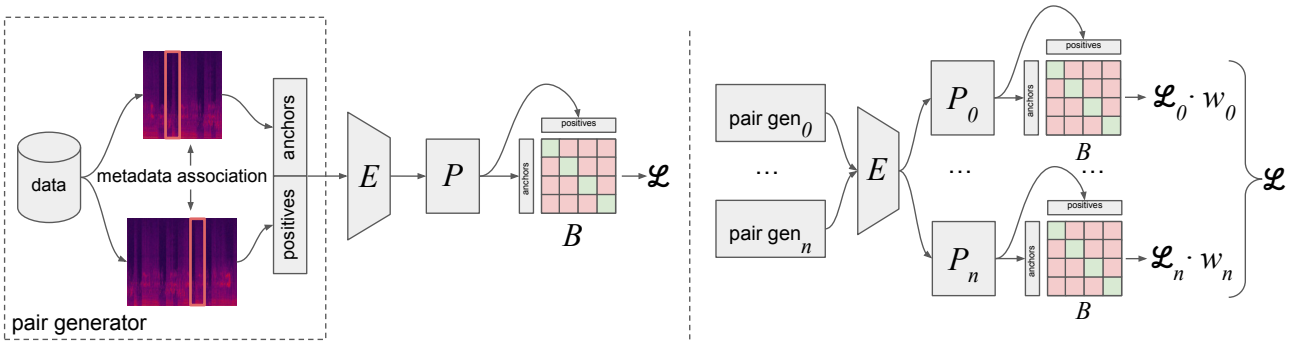


Figure 1. Overview of the proposed pipelines for a single (left) and multiple tasks (right). *E* is the encoder, *P* the projection head, *B* are the weights for the bilinear similarity, and $\mathcal{L}$ is the loss term. In the multi-task setup, tasks go from *0* to *n*.

showed a positive correlation with the downstream metrics, we will only perform full-size experiments and focus on unexplored aspects.

In particular, we are interested in imposing tight constraints on the pair generation process to emphasize the differences between the different associations. To formalize our problem, we consider a collection of music where each song has a track ID (*t*) and appears in one or more releases (*R*). Each release is produced by one or more artists (*A*) for one or more record labels (*L*). We define one self-supervised and three metadata-based associations:

- *track association*, the anchor and positive samples come from the same track (self-supervised),

$$t_a = t_p$$

- *release association*, t_a and t_p have at least a release in common,

$$|R(t_a) \cap R(t_p)| > 0$$

- *artist association*, t_a and t_p have at least an artist in common, but they do not appear in the same release,

$$|A(t_a) \cap A(t_p)| > 0, \text{ and } |R(t_a) \cap R(t_p)| = 0$$

- *label association*, t_a and t_p have at least a record label in common, but they do not share any artist,

$$|L(t_a) \cap L(t_p)| > 0, \text{ and } |A(t_a) \cap A(t_p)| = 0$$

Note that this approach considerably limits the number of pairs that can be matches and presumably leads to sub-optimal representations. However, our goal is to limit the number of overlapping pairs to emphasize the differences between associations. For instance, the release associations would be broadly a subset of the artist ones without the proposed constraints.

For each association, we generate a dataset of anchor/positive pairs. We initialize a pool with all the songs in the music collection. For each song, we pick a random pair that complies with the association's condition and remove both from the pool. While this approach does not exploit every possible association, it gives each track one association attempt, which provides us with sufficient training data for the scope of this work. We also considered a balanced version of the algorithm (e.g., same number of tracks per artist) that we discarded as the performance dropped due to the reduced dataset size without providing additional insights.

Following our contrastive paradigm, we do not need to create explicit anchor/negative pairs. For a given anchor, the rest of the positive samples in the batch are considered negatives (see Figure 1). While this naive approach implies that two pairs of tracks associated with the same artist will be respectively used as negative examples, there is a small probability of having repeated artists in a batch, and we ignored this issue. [2]

4. EXPERIMENTS

We pre-trained different contrastive systems following the proposed similarity notions. To evaluate the quality of the learned representations, we trained shallow classifiers on different multi-class and multi-label classification tasks. Additionally, we conducted experiments to understand the complementarity of such representations.

4.1 Contrastive targets based on Discogs' metadata

The Discogs database provides publicly available dumps of their *release* data that we used to create our training targets. According to Discogs, a release is "a broad term for any audio product that is made for general public consumption"; albums, singles, or compilations are examples of releases. Each release entry contains lists of the artists and record labels involved, the year and country of release, and a list of music genres and styles according to the Discogs' taxonomy. We matched our in-house audio collection to releases and master releases (groupings of different versions of a release) in the Discogs metadata dump resulting in 4 million tracks, with 58.2% of the tracks belonging to more than one release. A track may be linked to many releases for multiple reasons, including remasters, reeditions, or compilations containing it. For each track, we generated lists of artists and record labels by pooling the metadata on the

[2] For example, considering a dataset of one million tracks, a batch size of 200 pairs, and an average of 6 tracks per release, the probability of having two pairs from the same release in the batch is $6e-4$, which we consider an affordable false-negative rate.

Association	Pairs	Diversity	Time	Acc.
track	3.3M	3.3M	63h	88.7
release	846K	2.0M	21h	35.7
artist	1.2M	257K	33h	41.1
label	1.1M	142K	48h	24.7

Table 1. Statistics for the metadata association and their respective models. We show the number of pairs used, association diversity (number of different tracks, releases, artists, and record labels, respectively), training time (hours), and validation accuracy (%).

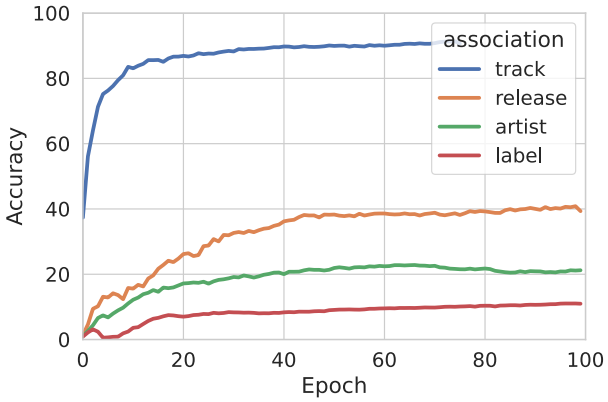


Figure 2. Training accuracies over the epochs for the different associations.

releases linked to it. We observed that different versions of a release could contain very different information, so for us, this was a simple way of maximizing the amount of information available per track.

We selected the subset with the top-400 most popular music styles resulting in 3.3 million tracks to train a baseline model with a multi-label classification objective (Style tags model). We reserved subsets of 50,000 tracks without artist overlap and a minimum frequency of 50 releases per music style tag for testing and validation. These sets were used as data pools to generate training, validation, and testing sets for the considered metadata associations following the methodology presented in Section 3. Note that we did not apply any data cleaning or deduplication, meaning that there may be room for improving the proposed representation models. Table 1 contains the resulting number of pairs and association diversities for each association, as well as the training time, and the accuracy obtained by their respective models. [3]

4.2 Pre-training the embedding models

We used an EfficientNet v1 on its B0 configuration as a backbone CNN to learn the embedding representations [37]. Our models operate on 2-second patches of mel-spectrograms with 96 bands extracted with the same parametrization as for MusiCNN [38] using the Essentia library [39]. [4]

Due to the massive dataset size, we stored the features as half-precision (16-bit) floats, and split the data into three machines with two GeForce RTX 2080 Ti GPUs each to parallelize the training. Every epoch, we fed the model with a random 2-second crop of the mel-spectrogram from each track in the training set. We also used a random patch per track for validation, but in this case, we used the same offset on every epoch to get more stable metrics. We relied on the Adam optimizer with an initial learning rate of $1e{-}3$ and a scheduler that reduced the learning rate by a factor of 10 if the validation loss had not decreased in 10 epochs. We trained the models for 100 epochs and considered two versions. The first one had the weights from the epoch where the lowest validation loss was achieved. The second model was obtained with stochastic weight averaging [40]. For the last 25 epochs, we imposed a learning rate of $1e{-}3$ and kept a moving average of the weights. As the latter version only reported minimum improvements in specific tasks, we decided to present the results of the former only.

As a baseline, we trained a model targeting the top-400 Discogs music styles with the multi-label soft-margin loss by connecting a fully-connected layer to the flattened output of the last convolutional block with 1280 units (embedding layer). As discussed on the Discogs website, [5] music styles usually go beyond purely stylistic descriptions and encode cultural, temporal, or geographical information, so we hypothesize that the learned representations are valuable beyond the task of genre recognition.

For the contrastive objectives, we used a fully-connected projection head with 512 dimensions on top of the same embedding layer, followed by a normalization layer and a *tanh* activation. We used the bilinear similarity ($a^T Bp$) and the cross-entropy loss as in the original implementation. While the authors of COLA showed that larger batch sizes improved the performance, we could only afford a batch size of 200 pairs due to the memory size of our GPUs. We parallelized the training so that each optimizer step aggregated six batches computed by different GPUs. This setup was close to the optimal number of pairs per optimizer step found in the original publication but used a larger ratio of positive samples.

To get additional insights into the models, we computed the top-1 accuracies by taking the *arg max* of the similarity matrices. Table 1 shows the training times and validation accuracies obtained by the model of each metadata association, and Figure 2 shows the evolution of the training accuracies over the epochs. The accuracies show that the associations have different difficulty levels, which aligns with our expectations.

All the pre-trained models are publicly available for feature extraction within the Essentia library. [6]

[3] The high number of releases is partially due to albums with multiple reeditions. On average, each track in our dataset is linked to 5 releases.

[4] `https://essentia.upf.edu/reference/std_TensorflowInputMusiCNN.html`

[5] `https://blog.discogs.com/en/genres-and-styles`

[6] `https://essentia.upf.edu/models.html`

Dataset	Size	Classes	Type
Genre	55,215	87	Multi-label
Instrument	25,135	40	Multi-label
Mood	18,486	56	Multi-label
Top50	54,380	50	Multi-label
MTAT	25,860	50	Multi-label
FMA	8,000	8	Multi-class

Table 2. Considered downstream datasets.

4.3 Downstream datasets

We evaluated our models as frozen feature extractors following a transfer learning setup. We consider three well-known music auto-tagging and classification datasets: FMA-small [41] (*FMA*), MagnaTagATune [42] (*MTAT*), and the MTG-Jamendo Dataset [43]. The latter contains different subsets for *Genre*, *Instrument*, and mood/theme (*Mood*) tags, as well as the top-50 tags in the dataset (*Top50*) that we treated independently. For *FMA* and *MTAT*, we used the splits proposed by the authors [41] and the 12:1:3 partition [44] respectively. In the MTG-Jamendo tasks, we used the sets defined by its *split-0* similarly to previous works [17, 45]. Table 2 shows the size of the considered datasets.

4.4 Transfer learning evaluation setup

As for the pre-training stage, we trained the downstream models with 2-second patches randomly cropped on each epoch. For validation, we averaged over the activations from the half-overlapped patches of the entire tracks. We passed the patches through the frozen backbone and used the flattened output of the last convolution layer as the input to a multi-layer perceptron with a single hidden layer of 512 units with a *sigmoid* or *softmax* activation for the multi-label or multi-class tasks. We used the cross-entropy loss and the Adam optimizer with a starting learning rate of $1e-3$ and added a weight decay of $1e-5$. A scheduler divided the learning rate by half if the loss had not decreased in five epochs. We trained the models for 30 epochs and used the weights from the epoch achieving the lowest validation loss to evaluate the test set.

4.5 Stacks of embeddings

Apart from evaluating the embeddings obtained from every association individually, we were interested in understanding their complementarity. We wanted to understand if the individually-learned representations could be combined to boost the performance, and if so, which were the best combinations. To investigate this, we ran stacks of embeddings through the presented evaluation protocol. First, we evaluated the stack of the Track, Release, Artist, and Label features for all the datasets. Additionally, we performed a systematic evaluation considering all the possible combinations of the four contrastive models plus the model trained on tags on *MTAT* considering two, three, four, and five embeddings.

4.6 Multi-task model

We also considered training a multi-task model to learn the metadata associations jointly. The architecture of the proposed system is depicted in Figure 1 (right). For each association, we have a separate pair generator and projection head. To perform an optimization step, we ran a batch of pairs from each association through the shared encoder and its specific projection head and computed a weighted sum of the losses. The loss weights were empirically selected prioritizing associations with better single-model performances (*track*: 0.1, *release*: 0.15, *artist*: 0.6, and *label*: 0.15). Additionally, we initialized the encoder with the weights of the model based on artist associations on its 20th epoch to speed up the training. While we experimented with multi-task models based on two association types, we did not find additional insights and decided to omit those results. Due to the additional model size, we had to reduce the batch size to 50 pairs per association.

4.7 Results and discussion

Table 3 reports the metrics obtained by our models and selected works from the literature. In descending order, the five groups in the table contain SOTA models trained from scratch without additional data, SOTA embedding models, baseline embedding models trained by us, our proposed embedding models trained on metadata associations, and models combining metadata associations. In the first group, we include MusiCNN [38], Harmonic CNN [45], and the winning submission of the 2021 Emotion and Theme Recognition challenge (team Lileonardo) [46] [7] as the best models from the literature in *MTAT*, *Top50*, and *Mood*, respectively. Similarly, MuLaP [17] reports the best performance as a frozen embedding extractor in *Genre*, *Mood*, *Top50*, and *FMA*, and the same applies to CALM [10] in *MTAT*. Note that comparisons against these reported metrics may be unreliable due to differences in training and evaluation settings.

We computed three baselines based on audio embeddings from an EfficientNet architecture with random weights, an EfficientNet architecture trained on music style tags as described in Section 4.2, and the VGGish model with its pre-trained weights [47].

Concerning our contrastive models, we observe that the model based on track associations (Track model) achieves competitive performance in some tasks, especially in *Top50*. Nevertheless, the models using metadata associations show better or equivalent performance despite seeing fewer pairs of tracks in the pre-training stage. In particular, we find that model based on artist associations (Artist model) is the best-performing with a few exceptions, which aligns with previous studies in metadata-based music representation learning [13].

Instrument and *MTAT* are the tasks where our models are further from the SOTA. For *MTAT*, we attribute this to the fact that CALM has 1,000 times more parameters, and

[7] https://multimediaeval.github.io/2021-Emotion-and-Theme-Recognition-in-Music-Task/

	Genre		Instrument		Mood		Top50		MTAT		FMA
	ROC	PR	ROC	PR	ROC	PR	ROC	PR	ROC	PR	Acc.
Lileonardo	-	-	-	-	**77.5**	**15.1**	-	-	-	-	-
Harmoic CNN	-	-	-	-	-	-	83.2	29.8	*91.3	*45.9	-
MusiCNN	-	-	-	-	-	-	-	-	90.7	38.4	-
MuLaP	85.9	-	76.8	-	76.1	-	82.8	-	*89.3	*40.2	**61.1**
CALM	-	-	-	-	-	-	-	-	**91.5**	**41.4**	-
Random weights	50.7	3.1	49.9	6.4	50.4	3.4	48.3	6.5	50.0	5.3	12.5
Style tags	87.7	19.9	77.6	19.8	75.6	13.6	83.1	29.7	90.2	37.4	59.1
VGGish	86.3	17.2	**77.8**	**20.2**	76.3	14.1	83.2	28.2	90.2	37.2	53.0
Track associations	86.3	18.0	69.9	16.7	74.0	12.8	82.9	29.4	89.7	36.4	58.9
Release associations	86.9	18.9	71.9	17.2	72.8	11.7	83.2	29.8	90.3	37.1	60.9
Artist associations	**87.7**	**20.3**	69.7	16.9	76.3	14.3	**83.6**	**30.6**	90.7	38.0	59.1
Label associations	87.0	19.4	75.0	18.2	74.8	12.8	83.1	29.9	88.7	34.2	59.5
Stack	86.9	19.4	74.7	18.8	74.3	13.0	83.4	30.0	90.8	38.6	59.8
Multi-task	87.2	19.9	70.5	17.2	76.1	14.4	83.5	30.3	90.8	37.8	60.0

Table 3. ROC-AUC, PR-AUC, and accuracy metrics for the downstream datasets. The five horizontal groups represent SOTA models from the literature trained from scratch, SOTA feature extractors from the literature, baseline feature extractors trained by ourselves, the proposed feature extractors based on metadata associations, and the proposed feature extractors combining associations. (*) results were computed in a clean version of *MTAT* and are not directly comparable.

that the authors report the best metrics from a grid search over shallow classifiers and hyperparameters. Also, we observed that models operating in the full *MTAT* previews tend to report higher PR-AUC performances [8, 10] than models operating in short chunks [5, 38].

The Stack is obtained by concatenating the embeddings from the models based on single associations as input for the MLPs. Except for *MTAT*, we do not get improvements over the best single model in any dataset despite the additional input dimensionality (5,120). Table 4 shows the top-5 results of models trained on combinations of embeddings for *MTAT*. The representations from the Artist and Style tags models are the only ones present in all the top-5 combinations, suggesting that these are the representations with the most valuable information. This observation aligns with the results from Table 3.

Similarly, our Multi-task model is not superior to the best single model in any dataset, but generally it is close in performance to the Artist model. We observe that Multi-task only overcomes Artist in the datasets where other associations perform better (i.e., *Instrument* and *FMA*), which shows its capability to pick the best information from different association types.

Tr	Re	Ar	La	St	ROC-AUC	PR-AUC
✓		✓	✓	✓	90.93	38.75
✓	✓	✓		✓	90.92	38.69
	✓	✓	✓	✓	90.92	38.51
	✓	✓		✓	90.89	38.57
		✓	✓	✓	90.87	38.57

Table 4. Top-5 combinations of the **Tr**ack, **Re**lease, **Ar**tist, **La**bel and **St**yle Tags features in *MTAT*. The results are sorted according to the ROC-AUC values.

5. CONCLUSIONS

In this paper, we studied the usage of editorial metadata as a source of supervision for contrastive feature extraction models intended for music classification. We contributed to the previous work on metadata-based feature learning by scaling up the experiments in terms of dataset size, considering additional editorial metadata notions, and by experimenting with a new contrastive learning setup. We could validate that some of the observations from previous studies still hold for models trained on 10 times more data. Namely, we observed that some metadata-based representations are superior to their tag-based counterparts and that artist associations provide the best representations. Additionally, we found that the features learned from different metadata notions can be combined or jointly learned, showing slight performance improvements for particular tasks. To the best of our knowledge, this is the first study using Discogs' metadata to train music feature models. We did this by generating pairs of tracks according to metadata notions with simple matching rules, which leaves room for experimentation with the dataset.

In future research, we will explore more complex relationships in the metadata. For example, we could rely on metadata co-occurrences (e.g., record labels sharing many artists) to create more detailed associations. Another direction is to combine our approach with more sophisticated contrastive learning paradigms, which opens up a possibility for performance improvements. Finally, we will consider extending the evaluation to additional tasks such as music recommendation or arousal and valence regression.

6. ACKNOWLEDGEMENTS

This research was carried out under the project Musical AI - PID2019-111403GB-I00/AEI/10.13039/501100011033, funded by the Spanish Ministerio de Ciencia e Innovación and the Agencia Estatal de Investigación.

7. REFERENCES

[1] A. van den Oord, S. Dieleman, and B. Schrauwen, "Transfer learning by supervised pre-training for audio-based music classification," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014.

[2] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "Transfer learning for music classification and regression tasks," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[3] J. Lee, J. Park, K. Kim, and J. Nam, "Samplecnn: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, 2018.

[4] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *Proceedings of the 44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[5] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[6] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Byol for audio: Self-supervised learning for general-purpose audio representation," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[7] D. Yao, Z. Zhao, S. Zhang, J. Zhu, Y. Zhu, R. Zhang, and X. He, "Contrastive learning with positive-negative frame mask for music representation," in *Proceedings of the ACM Web Conference 2022*, 2022.

[8] H. Zhao, C. Zhang, B. Zhu, Z. Ma, and K. Zhang, "S3t: Self-supervised pre-training with swin transformer for music classification," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

[9] L. Wang, P. Luc, Y. Wu, A. Recasens, L. Smaira, A. Brock, A. Jaegle, J.-B. Alayrac, S. Dieleman, J. Carreira *et al.*, "Towards learning universal audio representations," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022.

[10] R. Castellon, C. Donahue, and P. Liang, "Codified audio language modeling learns useful representations for music information retrieval," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021.

[11] J. Park, J. Lee, J.-W. Ha, and J. Nam, "Representation learning of music using artist labels," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[12] J. Kim, M. Won, X. Serra, and C. C. S. Liem, "Transfer learning of artist group factors to musical genre classification," *Companion Proceedings of the Web Conference 2018*, 2018.

[13] J. Lee, J. Park, and J. Nam, "Representation learning of music using artist, album, and track information," in *International Conference on Machine Learning (ICML) 2019, Machine Learning for Music Discovery Workshop*, 2019.

[14] J. Kim, J. Urbano, C. C. S. Liem, and A. Hanjalic, "One deep music representation to rule them all? a comparative analysis of different representation learning strategies," *Neural Computing and Applications*, 2020.

[15] Q. Huang, A. Jansen, L. Zhang, D. P. Ellis, R. A. Saurous, and J. Anderson, "Large-scale weakly-supervised content embeddings for music recommendation and tagging," in *Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[16] A. Ferraro, X. Favory, K. Drossos, Y. Kim, and D. Bogdanov, "Enriched music representations with multiple cross-modal contrastive learning," *IEEE Signal Processing Letters*, 2021.

[17] I. Manco, E. Benetos, E. Quinton, and G. Fazekas, "Learning music audio representations via weak language supervision," *arXiv:2112.04214v1 [cs.SD]*, 2021.

[18] A. Joorabchi and A. E. Mahdi, "An unsupervised approach to automatic classification of scientific literature utilizing bibliographic metadata," *Journal of Information Science*, 2011.

[19] J. K. Leung, I. Griva, and W. G. Kennedy, "Making use of affective features from media content metadata for better movie recommendation making," *arXiv preprint arXiv:2007.00636*, 2020.

[20] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International conference on machine learning*. PMLR, 2020.

[21] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, 2021.

[22] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, no. Jan, 2003.

[23] D. Bogdanov and P. Herrera, "Taking advantage of editorial metadata to recommend music," in *International Symposium on Computer Music Modeling and Retrieval (CMMR'12)*, 2012.

[24] D. Bogdanov and X. Serra, "Quantifying music trends and facts using editorial metadata from the Discogs database," in *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[25] J. Burke, R. Rygaard, and Z. Yellin-Flaherty, "Clam!: Inferring genres in the discogs collaboration network," 2014.

[26] P. Budner and J. Grahl, "Collaboration networks in the music industry," *arXiv preprint arXiv:1611.00377*, 2016.

[27] N. Andrade and F. Figueiredo, "Exploring the latent structure of collaborations in music recordings: A case study in jazz." in *ISMIR*, 2016.

[28] L. Gienapp, C. Kruckenberg, and M. Burghardt, "Topological properties of music collaboration networks: The case of jazz and hip hop." *DHQ: Digital Humanities Quarterly*, 2021.

[29] D. Bogdanov, A. Porter, H. Schreiber, J. Urbano, and S. Oramas, "The acousticbrainz genre dataset: Multi-source, multi-level, multi-label, and large-scale," in *Proceedings of the 20th International Society for Music Information Retrieval (ISMIR)*, 2019.

[30] R. Hennequin, J. Royo-Letelier, and M. Moussallam, "Audio Based Disambiguation Of Music Genre Tags," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, 2018.

[31] X. Favory, K. Drossos, T. Virtanen, and X. Serra, "COALA: co-aligned autoencoders for learning semantically enriched audio representations," in *Workshop on Self-supervised learning in Audio and Speech, International Conference on Machine Learning (ICML)*, 2020.

[32] ——, "Learning contextual tag embeddings for cross-modal alignment of audio and tags," in *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021.

[33] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar *et al.*, "Bootstrap your own latent-a new approach to self-supervised learning," *Advances in Neural Information Processing Systems*, 2020.

[34] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020.

[35] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

[36] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "Slowfast networks for video recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019.

[37] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

[38] J. Pons and X. Serra, "musicnn: Pre-trained convolutional neural networks for music audio tagging," *Late-Braeking/Demo, International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

[39] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. R. Zapata, and X. Serra, "ESSENTIA: an audio analysis library for music information retrieval," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 2013.

[40] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[41] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," *arXiv:1612.01840v3 [cs.SD]*, 2016.

[42] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *Proceedings of the 10th International Society for Music Information Retrieval Conference, (ISMIR)*, 2009.

[43] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, "The MTG-Jamendo dataset for automatic music tagging," in *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML)*, 2019.

[44] A. Van Den Oord, S. Dieleman, and B. Schrauwen, "Transfer learning by supervised pre-training for audio-based music classification," in *Conference of the International Society for Music Information Retrieval (ISMIR 2014)*, 2014.

[45] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of cnn-based automatic music tagging models," in *Proceedings of 17th Sound and Music Computing Conference (SMC)*, 2020.

[46] P. Tovstogan, D. Bogdanov, and A. Porter, "Mediaeval 2021: Emotion and theme recognition in music using jamendo," 2021.

[47] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "CNN architectures for large-scale audio classification," in *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017.

MELODY INFILLING WITH USER-PROVIDED STRUCTURAL CONTEXT

Chih-Pin Tan[1,2] **Alvin W.Y. Su**[1] **Yi-Hsuan Yang**[2,3]

[1] National Cheng Kung University, [2] Academia Sinica, [3] Taiwan AI Labs

p76091551@gs.ncku.edu.tw, alvinsu@mail.ncku.edu.tw, yang@citi.sinica.edu.tw

ABSTRACT

This paper proposes a novel Transformer-based model for music score infilling, to generate a music passage that fills in the gap between given past and future contexts. While existing infilling approaches can generate a passage that connects smoothly locally with the given contexts, they do not take into account the musical form or structure of the music and may therefore generate overly smooth results. To address this issue, we propose a structure-aware conditioning approach that employs a novel attention-selecting module to supply user-provided structure-related information to the Transformer for infilling. With both objective and subjective evaluations, we show that the proposed model can harness the structural information effectively and generate melodies in the style of pop of higher quality than the two existing structure-agnostic infilling models.

1. INTRODUCTION

In recent years, machine learning techniques have been widely applied to symbolic music generation. A large number of models attain *sequential generation* by accounting for only the *past context*, i.e., the generated music depends on only the preceding musical content [1–14]. While sequential generation can find useful use cases, it does not align with typical human compositional practices which can be non-sequential in nature. Musicians often write motifs or small pieces to get inspiration first, before working on the middle parts to connect them.

Hence, we focus on the scenario when both the past and *future* contexts are given, which is called *music score infilling* or inpainting [15]. As shown in Figure 1(a), the task is to let models fill in the missing part between the two given segments. Prompt-based conditioning approaches [15–23] have been applied to such a task in recent years, treating the two given segments as the "prompt." Among them, the variable-length infilling model (VLI) [20] obtains promising results by adding special positional encodings to XLNet [24], a permutation-based language model that is naturally suitable for generative tasks with given bi-

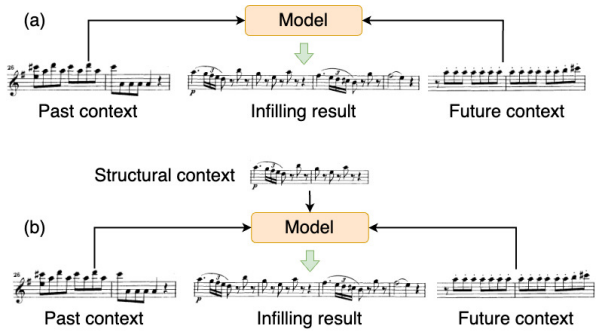


Figure 1: Comparison between (a) structure-agnostic and (b) structure-aware approaches for music score infilling.

directional contexts. The experiment of VLI shows that their model is capable of connecting the past and future contexts smoothly locally for infilling solo piano passages of up to 4 bars (measures).

Considering composers usually write musical pieces in a hierarchical manner [25], we note that prompt-based conditioning approaches have a strong limitation: they generate results with only consideration of local smoothness among the past context, future context, and result, without taking care of the overall musical *form* or *structure* of the music. For instance, a composer may like to write a song in a musical form of `ABA′B′`. If we consider the concatenation of the segments corresponding to `A` and `B` (i.e., `AB`) as the past context, and the segment corresponding to `B′` as the future context, and feed them to an existing infilling model, the model may generate a sequence that consists of similar melody and chord progression as the segments corresponding to `B` and `B′`, not the intended repetition or variation of the segment corresponding to `A`.

To address this issue, we propose in this paper a novel **structure-aware** setting for music infilling. As shown in Figure 1(b), besides the past and future contexts exploited by conventional structure-agnostic, prompt-based models, out approach additionally capitalizes for the infilling task the *structural context*, a music segment corresponding to a certain part of the whole music that is supposed to share the same *structure label* (such as `A` or `B`) with the missing segment. Accordingly, besides local smoothness, the model also needs to consider the similarity between the infilled segment and the structural context. Here, we assume the structural context is provided by a user, not generated by a model. For example, the user may designate the segment corresponding to `A` as the structural context, thereby

© Chih-Pin Tan, Alvin W.Y. Su, and Yi-Hsuan Yang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Chih-Pin Tan, Alvin W.Y. Su, and Yi-Hsuan Yang, "Melody Infilling with User-Provided Structural Context", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

inform the model with the intended musical form.

We improve upon the VLI model [20] in the following ways to realize structure-aware infilling. First, we use the classic Transformer [26–28] instead of the more sophisticated XLNet [24] as the model backbone, to make it easier to add a conditioning module to exploit the structural context. To improve the capability of the Transformer to account for bi-directional contexts, we propose two novel components, the *bar-count-down technique* (Section 3.2) and *order embeddings* (Section 3.3), which respectively give the model an explicit control of the length of the generated music, and a convenient way to attend to the future context. Second, being inspired by the Theme Transformer [29], we use not a Transformer decoder-only architecture but a sequence-to-sequence (seq2seq) Transformer encoder/decoder architecture, using the cross-attention between the encoder and decoder as the conditioning module to account for the structural context. Moreover, we propose an *attention-selecting module* that allows the Transformer to access multiple structural contexts while infilling different parts of a music piece, which can be useful both in the training and inference time (Section 3.4) .

For evaluation, we compare our model with two strong baselines, the VLI [20] and the work of Hsu & Chang [21], on the task of symbolic-domain melody infilling of 4-bar content using the POP909 dataset [30] and the associated structural labels from Dai *et al.* [31]. With objective and subjective analyses, we show that our model greatly outperforms the baselines in the structure completeness of the generated pieces, without degrading local smoothness.

We set up a webpage for demos [1] and open source our code at a public GitHub repository. [2]

2. RELATED WORK

Generating missing parts with given surrounding contexts has been attempted by early works. DeepBach [17] predicts missing notes based on the notes around them. They use two recurrent neural networks (RNNs) to capture the past and future contexts, and a feedforward neural network to capture the current context from notes with the same temporal position as the target note. COCONET [16] trains a convolutional neural network (CNN) to complete partial musical scores and explores the use of blocked Gibbs sampling as an analog to rewriting. They encode the music data with the piano roll representation and treat that as a fixed-size image, so the model can only perform fixed-length music infilling. Inpainting Net [15] uses an RNN to integrate the temporal information from a variational auto-encoder (VAE) [32] for bar-wise generation, Wei *et al.* [23] build the model with a similar concept as Inpainting Net and use the contrastive loss [33, 34] for training to improve the infilling quality. Some Transformer-based models have also been proposed to achieve music infilling. Ippolito *et al.* [18] concatenate the past and future context with a special separator token. They keep the original positional encoding of the contexts and the missing segment, which again limits the length of given contexts and generated sequence to be fixed. We see that these infilling models impose some data assumptions and thereby have certain restrictions, e.g., the length of the input sequence cannot be arbitrary, or the missing segment needs to be complete bars. The work of Hsu & Chang [21] is free of these restrictions. They use two Transformer encoders to capture the past and future context respectively and generate results with a Transformer decoder. The VLI model [20] can also realize variable-length infilling. However, to our best knowledge, no existing models have explicitly considered structure-related information for infilling.

Structure-based conditioning has been explored only recently by Shi *et al*. [29] in their Theme Transformer model for sequential music generation. They use a seq2seq Transformer to account for not only the past context but also an additional pre-given theme segment that is supposed to manifest itself multiple times in the model's generation result. The present work can be considered as an extension of their work to the scenario of music infilling.

3. METHODOLOGY

Given a past context C_{past} and a future context C_{future}, the general, **structure-agnostic music infilling** task entails generating an *infilled segment* T that interconnects C_{past} and C_{future} smoothly, preferably in a musically meaningful way. When using an autoregressive generative model such as the Transformer as the model backbone, the training object is to maximize the following likelihood function:

$$\prod_{0<k\leq|T|} P(t_k|t_{<k}, C_{\text{past}}, C_{\text{future}}) \,, \tag{1}$$

where t_k denotes the element of T at timestep k, $t_{<k}$ the subsequence consisting of all the previously generated elements, and $|\cdot|$ the length of a sequence.

Extending from Eq. (1), we propose and study in this paper a special case, called **structure-aware music infilling**, where an additional segment G representing the *structural context* is given, leading to the new objective:

$$\prod_{0<k\leq|T|} P(t_k|t_{<k}, C_{\text{past}}, C_{\text{future}}; G) \,. \tag{2}$$

As depicted in Figure 2(a), our model is based on Transformer with the encoder-decoder architecture. It uses the decoder to self-attend to the prompt (i.e., C_{past} and C_{future}) and the previously-generated elements (i.e., $t_{<k}$), and the encoder to cross-attend to the structural context G. We provide details of the proposed model below.

Note that we do not require the length of all the involved segments to be fixed; namely $|T|$, $|C_{\text{past}}|$, $|C_{\text{future}}|$ and $|G|$ are all variables in our setting.

3.1 REMI-based Token Representation

To incorporate structure-related information to our representation of the music data, we devise an extension of the REMI-based representation [8] that comprises five types

[1] https://tanchihpin0517.github.io/structure-aware_infilling
[2] https://github.com/tanchihpin0517/structure-aware_infilling

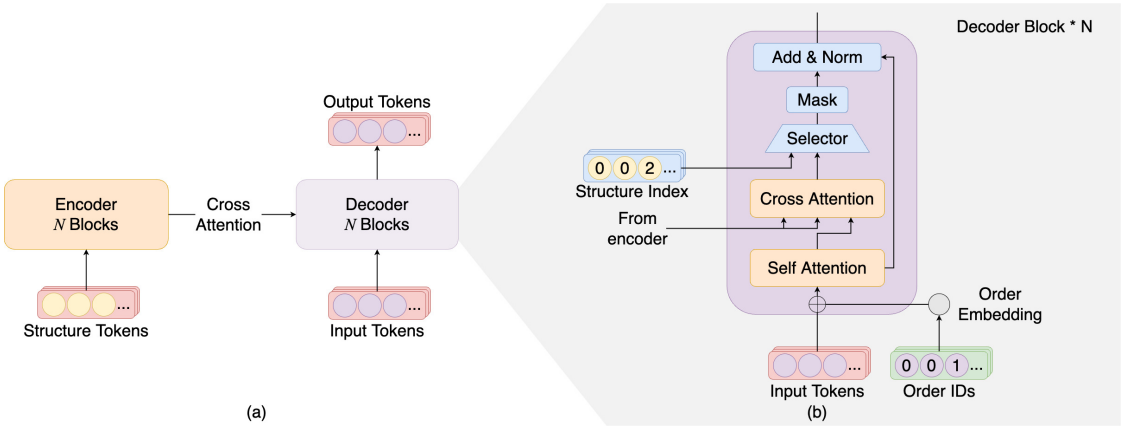


Figure 2: Schematic diagram of (a) the proposed seq2seq Transformer model for structure-aware infilling, and (b) a zoom-in of the decoder highlighting how the decoder utilizes the *order embedding* and *structure index*.

Token type	Voc. size	Values
`Bar`	32	1, 2, ... , 32
`Struct`	16	0, 1, ... , 15
`Tempo`	47	28, 32, ... , 212
`Position`	16	0, 1, ... , 15
`Pitch`	86	22, 23, ... , 107
`Duration`	16	1, 2, ... , 16

Table 1: The vocabulary of the token representation.

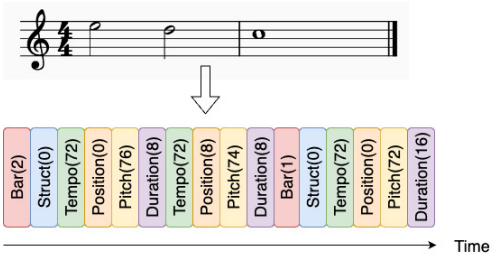


Figure 3: An example of representing a two-bar segment with our adapted REMI-based token representation. With the bar-count-down technique, `Bar(2)` and `Bar(1)` are used for the first and second `Bar` tokens, respectively.

of tokens: `Bar`, `Struct`, `Tempo`, `Position`, `Pitch` and `Duration`. Table 1 lists the vocabulary of our token representation. `Bar` consists of numbers from 1 to 32, standing for the number of remaining bars on the generation process. A music form, e.g., `ABA′B′`, consists of multiple groups of similar phrases or sections (each of multiple bars), e.g., {`A`, `A′`} and {`B`, `B′`}, where each group can be said to be associated with the same *structure label*. We use `Struct` to indicate the structure label for each bar. `Tempo` and `Position` are related to the musical metre. `Tempo` is the current tempo of beats per minute (BPM), and `Position` is the temporal distance between the onset of a musical note and the beginning of its bar, denoted by the number of 16-th notes. `Pitch` and `Duration` are related to musical notes, which are the MIDI pitch number and duration in 16-th notes, respectively. We show an example of how we encode the musical content in Figure 3.

3.2 Bar-Count-Down Technique

The work of Hsu & Chang [21] uses special `BOS` and `EOS` tokens as the "signal" to start or stop the generation process. At inference time, the infilled segment generated by their model comes to an end when the model generates an `EOS` token. While this may work fine in certain cases, doing so cannot give us an explicit control of the number of bars to be generated for the infilled segment. Such a control is preferable when we want to make sure that the previous context and the future context are a certain number of bars apart, which is highly needed for structure-aware infilling. For example, a user may want to specify the music form as `A8B8A′8B′8`, meaning that all the four sections `A`, `B`, `A′`, `B′` are eight-bar long each.

To have such a control, we employ a special token representation technique called "bar-count-down." For each infilled sequence T, we adjust the suffix number of `Bar` tokens to match the length of T. Take Figure 3 as an example: The number in `Bar` tokens are counted down from two because the sequence in Figure 3 is 2-bars long. Once training a model with this setup, the length of T can be controlled effectively by the number of remaining bars associated with the first `Bar` token given on generation.

3.3 Order Embedding

Transformers with causal masking are mainly designed for sequential generation. To apply the model to infilling tasks where the missing part is in the middle of the input sequence, the model proposed by Hsu & Chang [21] attends to bi-directional context from two encoders with cross-attention. As we want to instead use only the self-attention of the decoder to exploit bi-directional information, we reorder the sequence $\{C_{\text{past}}, T, C_{\text{future}}\}$ to $\{C_{\text{past}}, C_{\text{future}}, T\}$. Doing so would however change the original positional relationship among C_{past}, T, and C_{future}.[3] As depicted

[3] The principal idea of the attention mechanism [26] is to use the token embeddings of two tokens to compute their correlation, leading to the so-

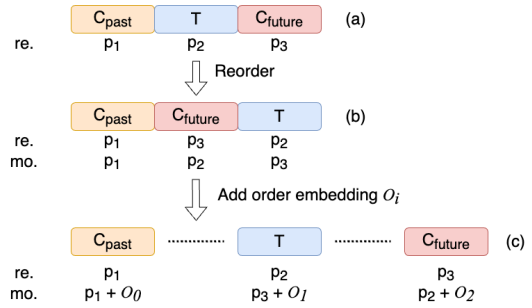


Figure 4: Illustration of the order embedding, where $re.$ and $mo.$ stand for the *real position* and *model-viewed position*. Conceptually, we shift the model-viewed position by O_i to make it consistent with the real position.

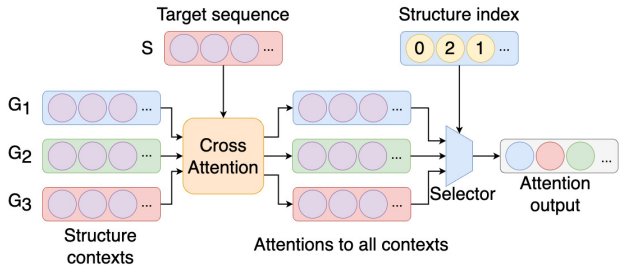


Figure 5: The schematic of attention selecting. Multiple structural contexts G_1, G_2, G_3 are passed to the cross-attention block simultaneously, and the output is filtered by the *selector* with the *structure index*.

in Figure 4(b), the *real* positional relationships among the segments entails associating the tokens in T with positional embeddings corresponding to a set of positions p_2 that signifies the model the segment T is after C_{past} (i.e., $p_1 \prec p_2$) and is before C_{future} ($p_2 \prec p_3$).[4] After reordering, however, using the typical way of computing the positional embeddings from left to right by the Transformers, T would be assigned with positional embeddings corresponding to p_3, meaning that by the *model-viewed* positional relationships, T is after both C_{past} and C_{future}. The real and the model-viewed are mismatched.

We introduce a new position-related *segment embedding* called "order embedding" to tackle this issue. Specifically, for the positional embeddings for all the tokens in C_{past}, we add to them the same additional embedding corresponding to a positional "offset" or "order," denoted as O_0. We similarly incorporate O_1 and O_2 to the positional embeddings for the tokens in T and C_{future}. As long as the "offset" is big enough, we can have $p_1 + O_0 \prec p_3 + O_1 \prec p_2 + O_2$, ensuring that the real and model-viewed positional relationships are matched, as depicted in Figure 4(c).

We note that, as the same order embedding is added to the positional embeddings of all the tokens in a segment, the proposed idea works nicely regardless of whether the segment lengths $|C_{\text{past}}|$, $|T|$, $|C_{\text{future}}|$ are fixed or not.

3.4 Attention-Selecting Module

While the formulation of Eq. (2) considers only one structural context G, in practice, we may want to designate multiple structural contexts $\{G_1, G_2, \ldots, G_N\}$ for infilling, with each segment G_n corresponding to a certain structure label such as `A` and `B`. This is useful, for example, when the first part of the infilled segment T is meant to be similar to phrase `A`, while the latter part similar to phrase `B`. To indicate which G_n a specific token t_k of T should refer to, we define the *structure index* $y_k \in \{0, \ldots, N\}$, for $k = \{1, \ldots, |T|\}$. The structure indices are also given by the user when the user specifies the intended musical form. With all these, we extend Eq. (2) to:

$$\prod_{0<k\leq|T|} P(t_k|t_{<k}, C_{\text{past}}, C_{\text{future}}; G_1, ..., G_N, y_k)\,. \quad (3)$$

We use $y_k = 0$ to indicate the case where t_k is supposed to follow none of the structural contexts (e.g., when t_k is part of the bridge). When $y_k \neq 0$, t_k follows only one of the structural context G_{y_k}. In our implementation, for tokens whose $y_k = 0$, we only use the self-attention to attend to the prompt and $t_{<k}$, leading to a formulation akin to Eq. (1). When $y_k \neq 0$, we have a "selector" that picks by y_k the structural context G_{y_k} to be attended to via cross-attention, resulting in a formulation akin to Eq. (2). This is depicted in Figure 2(b) and Figure 5.

Figure 5 also shows that, in our implementation, instead of computing the cross attention between the target sequence with a specific structural context G_{y_k}, we actually compute the cross attention between the target sequence with every structural contexts $\{G_1, G_2, \ldots, G_N\}$ and let the selector pick the right one. While this seems a waste of computing, we find doing so faster when GPUs are used.[5]

4. EXPERIMENTAL SETUP

We collect the melody data[6] from the POP909 dataset compiled by Wang *et al.* [30]. POP909 contains 909 MIDI files of pop piano performances, though we discard 8 of them due to errors encountered in the preprocessing stage. We split all remaining songs into 811 (90%) songs for training and 91 (10%) songs for testing. The 16-th note is set as the minimal temporal resolution to quantize the tempo, beat, and duration for reducing the vocabulary size.

Dai *et al.* [31] publicly share the *structural information* of songs in POP909 with letters and integers to indicate the structure labels and their lengths in bars, such as

`i4 A8 B8 x4 A8 B8 B8 X2 c4 c4 X2 B9 o2`, (4)

called attention score. However, using the token embeddings alone fails to consider the position-related relations of the two tokens (e.g., whether they are neighbors or distant apart). Accordingly, in practice, people add token-wise positional embeddings to the token embeddings before computing their attention [35–37].

[4] We use $p \prec q$ to denote that any elements in the set p is smaller, or "temporally before," any elements in the set q.

[5] There is another implementation detail: actually, not only the tokens in T but also those in C_{past} and C_{future} would go through the Transformer's self- and cross-attention blocks. This is to get the latent vectors for the tokens in C_{past} and C_{future}. In doing so, we calculate and employ the structure indices for C_{past} and C_{future} as well.

[6] Please note that the "melody" data in this paper is not exact monophonic music. We merge two midi tracks, "MELODY" and "BRIDGE" of the MIDI files of POP909 [30] to generate the melody data, which is monophonic most of the time but not always.

where the music phrases labeled with the same letter are considered to share the same structural context.[7] [8] Besides, the musical phrases with the labels `i`, `x`, and `o` are respectively the introduction, bridge, and ending, which do not have structural context in our setting. For each song, we use the phrases corresponding to the first occurrence of the structure labels such as `A` and `B` as the structural contexts G_1, G_2, etc. For example, we choose bars 5–12 (i.e., the `A8` phrase after `i4`) and bars 13–20 (i.e., the `B8` phrase before `x4`) in the example shown in Eq. (4) as the structural contexts for `A` and `B` since they are the first music phrases in the song with the corresponding labels.

The training data is generated with the following steps: (i) iterating through all labels except for the first and last ones in the structural information, (ii) choosing their corresponding music phrases as the infilled sequences T, and (iii) concatenating T with their preceding and following 6-bars music segments to get $\{C_{\text{past}}, T, C_{\text{future}}\}$, yielding 8,607 data in total. Before feeding the training data into the model, we reorder the input sequences and insert additional special tokens, `BOS`, `SEP`, and `EOS`, to change them into $\{\texttt{BOS}, C_{\text{past}}, \texttt{SEP}, C_{\text{future}}, \texttt{SEP}, T, \texttt{EOS}\}$. The reordered input sequence and structural contexts are transformed to the embeddings with size 512. The model consists of an encoder and a decoder, where both of them consist of six 8-head self-attention layers with intermediate layers of dimension 2,048. Each layer of the encoder and decoder is connected with an 8-head cross-attention, as shown in Figure 2. The output from the model is transformed back to a probability distribution of the vocabulary with the softmax function. At inference time, we use nucleus sampling [38] to sample the output tokens with the threshold value of 0.9.

For evaluation, we create the testing data by: (i) searching from the testing songs all the 4-bar phrases that correspond to only a structure label, (ii) keeping only the phrases that share the same structure label with one of its two neighboring phrases but a different structure label with the other neighbor, and (iii) setting the 4-bar phrase as the target T and concatenate them with their preceding and succeeding 6-bar music segments, which are set as the past context C_{past} and future context C_{future}, respectively. We get 156 test cases of $\{C_{\text{past}}, T, C_{\text{future}}\}$, each with 16 bars (i.e., $6+4+6$). All the cases have the form of, e.g., `AA'B` or `ABB'`, and the target lengths are **4 bars** with arbitrary number of notes (hence *variable* sequence lengths). In this setting, the attention selection mechanism is used only for training, since the target sequence T in our testing data only have one structural context to refer to.

We consider VLI [20] and the model from Hsu & Chang [21] as the baselines. By design, only our model has access to an external structural context. However, we consider the comparison as valid, since C_{past} and C_{future} are presumably long enough to provide sufficient context, and at least one of them has the same structure label as T. Besides, in our implementation, we found the model Hsu & Chang [21] rarely generates infilled segments with the desirable number of bars. Therefore, we slightly improve their model by incorporating the bar-count-down technique.

[7] The lower-case and capital letters indicate respectively *non-melodic* and *melodic* phrases (i.e., where a clear melody is present, mostly a vocal line or an instrument solo), but we do not use this information.

[8] Depending on the underlying musical form of a song, different songs may have different numbers of phrase groups. For example, a song with a simpler form may only have phrase groups `A` and `B`, while other songs have much more. In the POP909 dataset, a song can contain up to 9 unique phrase groups, which are labeled as `A`, `B`, `C`, `D`, etc.

	$H \downarrow$	$GS \uparrow$	$D \downarrow$
Ours	**2.75**$\pm$0.80	**0.70**$\pm$0.08	**25.73**$\pm$19.45
VLI [20]	3.47$\pm$1.57	0.67$\pm$0.09	49.40$\pm$25.12
Hsu [21]	9.87$\pm$4.64	0.64$\pm$0.09	65.41$\pm$38.00
Original	2.78$\pm$0.89	0.70$\pm$0.09	0.00$\pm$ 0.00

Table 2: Objective evaluation results. H: pitch class histogram cross entropy, GS: grooving pattern similarity, D: melody distance (↑/↓: the higher/lower the better).

5. OBJECTIVE EVALUATION RESULTS

We propose three new metrics for objective evaluation, all of which have not been used in the literature of music infilling. The first two metrics, **pitch class histogram cross entropy** (H) and **grooving pattern similarity** (GS), are extensions of the ones proposed by Wu & Yang for sequential generation [10] to our infilling task, evaluating respectively the consistency in terms of pitch class distribution (which is related to tonality) and rhythmic pattern. For H, we compute per test case the pitch class histogram of T, and that histogram of the concatenation of C_{past} and C_{future}, and then report the cross entropy between these two histograms. For GS, we use per bar a 16-dim binary vector indicating where there is at least a note onset for every position in a bar, calculate one minus the normalized XOR difference between every pair of bars [10], one from T and the other from either C_{past} and C_{future}, and then report the average per test case. The third metric, **melody distance** (D) measures the melody distance (dissimilarity) between the infilled segment T and the ground truth one (denoted as $T\#$) using the algorithm proposed by Hu *et al.* [39]. Lower H and higher GS may imply that T connects C_{past} and C_{future} smoothly, while lower D indicates that the generation result is similar to a human-made one.

Table 2 shows that our model achieves the best result in all the three metrics, followed by VLI and then the model of Hsu & Chang. Besides being consistent with the contexts, the infilling result of our model is closest to the ground truth one $T\#$, demonstrating the effectiveness of exploiting the structural context. Figure 6 exemplifies how the infilled bars by our model fit the desired musical form.

6. SUBJECTIVE EVALUATION RESULTS

We conduct additionally an online user study for subjective evaluation. We have 91 anonymous volunteers, where 12 of them are marked as professionals according to the question about their musical background. Each subject is pre-

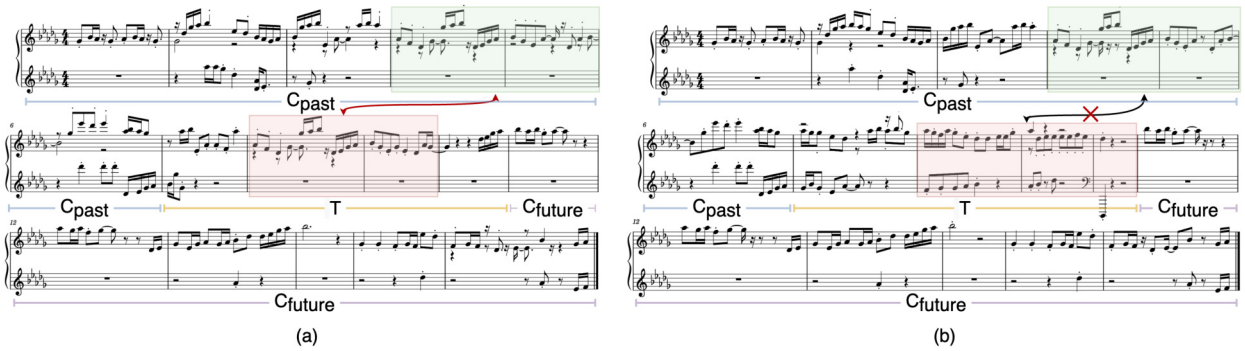


Figure 6: Results generated by (a) our model and (b) VLI. The structural relations of bars 4-5 (green area) and bars 8-9 (red area) are preserved well in (a) but not in (b).

		M	R	S	O
all	Ours	**3.46**	**3.51**	**3.40**	**3.42**
	VLI [20]	2.96	3.14	3.12	2.97
	Hsu [21]	2.60	2.95	2.75	2.64
	Real	**3.77**	**3.77**	**3.62**	**3.66**
pro	Ours	**3.58**	**3.28**	**3.28**	**3.42**
	VLI [20]	2.67	2.86	2.78	2.72
	Hsu [21]	2.36	2.75	2.39	2.44
	Real	**3.61**	**3.56**	**3.42**	**3.42**

Table 3: Results of the user study: mean opinion scores in 1–5 in **M** (melodic fluency), **R** (rhythmic fluency), **S** (Structureness), and **O** (Overall), from 'all' the participants or only the music 'pro'-fessionals.

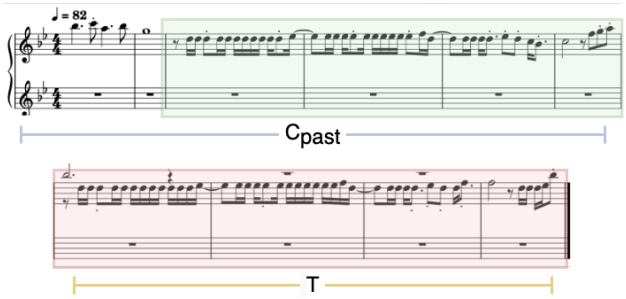


Figure 7: A result generated by our model, where C_{future} is not shown in this figure. Bars 3–6 (green area) and bars 7–10 (red area) look identical since the model overly imitates the given structural context. (Best viewed in color.)

sented with 3 out of 15 sets of music segments randomly sampled from the testing data. We inform them that the first and last 6 bars are the given prompts, and the middle 4 bars are the music generated by a model. Each set of music contains in random order 4 music including 3 generated by the models and 1 from the real data. The subjects rate each of the 4 music in a 5-point Likert scale (the higher the better) according to their (i) **melodic fluency**: do the pitches of notes go in the right tonality and connect the contexts fluently? (ii) **rhythmic fluency**: are the notes played on the right beats? (iii) **structureness**: how is the generated part of the music similar to its contexts? (ix) **overall**: how much do they like the music?

Table 3 shows the mean opinion scores (MOS) of the user study. Echoing the result of the objective evaluation, the proposed model outscores the baselines by a large margin in all the four subjective metrics, and is close to the real music with a small gap. The same observations can be made from either the average result of all the subjects, or only that from the 12 professionals.

However, we notice that our model may overly imitate the given structural contexts in some cases, as exemplified in Figure 7. When this happens, the generated music sounds rigid and non-creative. We conjecture that this can be attributed to the limited diversity of our training data—the melodies corresponding to the same structure label in POP909 appear to be too similar to each other, which may not be uncommon for pop songs. To study this, we implement additionally a 'Copy' baseline that simply copies the structural contexts as the result, and include its infilling result to the demo website. Our own subjective listening of its result confirms that the proposed method still outperforms the 'Copy' baseline most of the time, as the connections between the targets and their contexts are considered by our model, but not by the 'Copy' baseline.

7. CONCLUSION

In this paper, we have proposed a new structure-aware conditioning approach for music score infilling. To help Transformers exploit bi-directional contexts, we employ the order embedding to shift the position viewed by the model. Besides, we introduce a new attention-selecting mechanism to account for multiple structural contexts. Evaluations on 4-bar melody infilling validate the superiority of the proposed model over two existing Transformer-based structure-agnostic infilling methods [20, 21].

We can extend this work in three ways. First, instead of relying on user inputs, we may build models that generate the musical form automatically and predict the structural context for a specific infilled segment to refer to. Second, we like to expand our work to other music genres and polyphonic music. Finally, we like to study how the attention-selecting mechanism can be applied to sequential generative tasks such as theme-based generation [29].

8. ACKNOWLEDGEMENT

We are grateful to Shih-Lun Wu for sharing the unpublished idea of the bar-count-down technique, Ping-Yi Chen and Chin-Jui Chang for the assistance with experiments and discussion, Shan Lee for the help of figure drawing, and Chun-Wei Lai and Sophia Lin for the help in finding volunteers for the user study. We also thank the anonymous reviewers for their valuable feedbacks. Our research is funded by grants NSTC 109-2628-E-001-002-MY2 and NSTC 110-2221-E-006-137-MY3 from the National Science and Technology Council of Taiwan.

9. REFERENCES

[1] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription," in *Proc. International Conference on Machine Learning*, 2012, pp. 1881–1888.

[2] F. Colombo, "Algorithmic composition of melodies with deep recurrent neural networks," *Poster Session of International Conference on Artificial Intelligence and Statistics*, 2016.

[3] B. L. Sturm, J. F. Santos, O. Ben-Tal, and I. Korshunova, "Music transcription modelling and composition using deep learning," in *Proc. Conference on Computer Simulation of Musical Creativity*, 2016.

[4] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck, "A hierarchical latent vector model for learning long-term structure in music," in *Proc. International Conference on Machine Learning*, 2018, pp. 4364–4373.

[5] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang, "MidiNet: A convolutional generative adversarial network for symbolic-domain music generation," in *Proc. International Conference on Music Information Retrieval*, 2017.

[6] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music Transformer," in *Proc. International Conference on Learning Representations*, 2019.

[7] C. Donahue, H. H. Mao, Y. E. Li, G. W. Cottrell, and J. McAuley, "LakhNES: Improving multi-instrumental music generation with cross-domain pre-training," in *Proc. International Conference on Music Information Retrieval*, 2019.

[8] Y.-S. Huang and Y.-H. Yang, "Pop Music Transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proc. ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[9] Y. Ren, J. He, X. Tan, T. Qin, Z. Zhao, and T.-Y. Liu, "PopMAG: Pop music accompaniment generation," in *Proc. ACM Multimedia*, 2020.

[10] S.-L. Wu and Y.-H. Yang, "The Jazz Transformerm on the front line: Exploring the shortcomings of ai-composed music through quantitative measures," in *Proc. International Conference on Music Information Retrieval*, 2020.

[11] S. Dai, Z. Jin, C. Gomes, and R. B. Dannenberg, "Controllable deep melody generation via hierarchical music structure representation," in *Proc. International Conference on Music Information Retrieval*, 2021.

[12] S.-L. Wu and Y.-H. Yang, "MuseMorphose: Full-song and fine-grained music style transfer with one Transformer VAE," *arXiv preprint arXiv:2105.04090*, 2021.

[13] J. Liu, Y. Dong, Z. Cheng, X. Zhang, X. Li, F. Yu, and M. Sun, "Symphony generation with permutation invariant language model," *arXiv preprint arXiv:2205.05448*, 2022.

[14] H.-W. Dong, K. Chen, S. Dubnov, J. McAuley, and T. Berg-Kirkpatrick, "Multitrack Music Transformer: Learning long-term dependencies in music with diverse instruments," *arXiv preprint arXiv:2207.06983*, 2022.

[15] A. Pati, A. Lerch, and G. Hadjeres, "Learning to traverse latent spaces for musical score inpainting," in *Proc. International Conference on Music Information Retrieval*, 2019.

[16] C.-Z. A. Huang, T. Cooijmans, A. Roberts, A. Courville, and D. Eck, "Counterpoint by convolution," in *Proc. International Society for Music Information Retrieval*, 2017.

[17] G. Hadjeres, F. Pachet, and F. Nielsen, "Deepbach: a steerable model for bach chorales generation," in *Proc. International Conference on Machine Learning*, 2017, pp. 1362–1371.

[18] D. Ippolito, A. Huang, C. Hawthorne, and D. Eck, "Infilling piano performances," in *Proc. NIPS Workshop on Machine Learning for Creativity and Design*, 2018.

[19] T. Bazin and G. Hadjeres, "Nonoto: A model-agnostic web interface for interactive music composition by inpainting," in *Proc. International Conference on Computational Creativity*, 2019.

[20] C.-J. Chang, C.-Y. Lee, and Y.-H. Yang, "Variable-length music score infilling via XLNet and musically specialized positional encoding," in *Proc. International Conference on Music Information Retrieval*, 2021.

[21] J.-L. Hsu and S.-J. Chang, "Generating music transition by using a Transformer-based model," *Electronics*, vol. 10, no. 18, 2021.

[22] C.-P. Tan, C.-J. Chang, A. W. Y. Su, and Y.-H. Yang, "Music score expansion with variable-length infilling," in *Proc. International Conference on Music Information Retrieval*, 2021, late-breaking and demo paper.

[23] S. Wei, G. Xia, Y. Zhang, L. Lin, and W. Gao, "Music phrase inpainting using long-term representation and contrastive loss," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2022, pp. 186–190.

[24] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," *Proc. Advances in Neural Information Processing Systems*, vol. 32, 2019.

[25] F. Lerdahl and R. S. Jackendoff, *A Generative Theory of Tonal Music, Reissue, with a New Preface*. MIT press, 1996.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Proc. Advances in Neural Information Processing Systems*, vol. 30, 2017.

[27] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proc. Annual Meeting of the Association for Computational*, 2019.

[28] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[29] Y.-J. Shih, S.-L. Wu, F. Zalkow, M. Muller, and Y.-H. Yang, "Theme Transformer: Symbolic music generation with theme-conditioned transformer," *IEEE Transactions on Multimedia*, 2022.

[30] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia, "POP909: A pop-song dataset for music arrangement generation," in *Proc. International Conference on Music Information Retrieval*, 2020.

[31] S. Dai, H. Zhang, and R. B. Dannenberg, "Automatic analysis and influence of hierarchical structure on melody, rhythm and harmony in popular music," in *Proc. Conference on AI Music Creativity*, 2020.

[32] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. International Conference on Learning Representations*, 2014.

[33] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. International conference on machine learning*, 2020, pp. 1597–1607.

[34] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 9729–9738.

[35] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proc. Conf. North American Chapter of the Association for Computational Linguistics*, 2018.

[36] G. Ke, D. He, and T.-Y. Liu, "Rethinking positional encoding in language pre-training," in *Proc. International Conference on Learning Representations*, 2021.

[37] A. Liutkus, O. Cífka, S.-L. Wu, U. Simsekli, Y.-H. Yang, and G. Richard, "Relative positional encoding for Transformers with linear complexity," in *Proc. International Conference on Machine Learning*, 2021.

[38] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *Proc. International Conference on Learning Representations*, 2020.

[39] N. Hu, R. B. Dannenberg, and A. L. Lewis, "A probabilistic model of melodic similarity," in *Proc. International Computer Music Conference*, 2002.

ROBUST MELODY TRACK IDENTIFICATION IN SYMBOLIC MUSIC

Xichu Ma **Xiao Liu** **Bowen Zhang** **Ye Wang**
National University of Singapore
{ma_xichu, liuxiao, zbowen}@u.nus.edu, wangye@comp.nus.edu.sg

ABSTRACT

Melody tracks are worthy of special attention in symbolic music information retrieval (MIR) because they contribute more towards music perception than many other musical components. However, many existing symbolic MIR systems neglect the preprocessing of melody track identification (MTI). Moreover, existing MTI methods often deal with MIDIs of the same genres and follow specific track configuration. This limits their generalization and robustness to unseen data. To address these challenges, we propose a CNN-Transformer-based MTI model designed to identify a single melody track for arbitrary MIDI files robustly. To accommodate longer inputs, We also employ a sparse Transformer, speeding up attention computation. Our experiments show that our proposed model outperforms state-of-the-art (SOTA) algorithms in accuracy and benefits downstream MIR tasks.

1. INTRODUCTION

Listeners' perception of musical works is greatly shaped by the melodic lines of those works [1]. Human listeners can usually easily distinguish the melody from other musical components such as harmonic lines [2]. It is, however, not as easy for machines. Melody is also a major focus of many music information retrieval (MIR) tasks. These tasks fall into three categories: 1) Melody as a target, where the system seeks to output a suitable melody. Examples of this task include melody segregation [3, 4] and melody generation [5, 6]. 2) Melody as a requirement, where the model takes in an explicitly-identified melodic line as an input. Such tasks include lyrics generation from music [7, 8] and singing synthesis [9, 10]. 3) Melody as a dependency, where the melody is not explicitly provided as input but still heavily influences the results. Examples include genre classification [11] and music similarity measurement [12].

However, for melody-as-a-target tasks, the data of annotated melody labels can be difficult to obtain. Thus, an automatic MTI model can create more data for their training. The melodic line is the decisive factor for these melody-as-a-dependency tasks [13, 14]. However, current studies on melody-as-a-dependency tasks often neglect to intelligently identify the melodic lines of music. For example, a music embedding learning model, PiRhDy [15], simply regards the track with the highest average pitch as the melody. Existing genre classifiers treat all tracks equally rather than give special emphasis to the melodic line [11]. On the one hand, melody plays fundamental role in music perception [16]; studies have shown the success of attention and self-attention mechanisms in sequence modeling [17, 18]. On this basis, we hypothesize that melody-as-a-dependency tasks' performance would likely improve if the data are available with identified melodic lines. This preprocessing of MTI would also help melody-as-a-requirement applications. For instance, some stylistic music generation systems can only accommodate specifically formatted melody note sequences as input, demanding musicians' manual annotation [19]. Equipped with a MTI module, it could eliminate the dependence on manual processing.

There are two main challenges in developing a robust Melody Track Identification (MTI) model for symbolic music. First, many existing methods either rely on normative composition theories [20]. They assume that the input MIDIs are highly-formatted (i.e., one channel has only one track) and usually in limited genres [21, 22]. This assumption limits their generalization and robustness in dealing with more diverse or "dirtier" samples. Second, while local and global musical features are both significant for MTI, architectures such as CNNs and RNNs are not ideal in capturing short-and-long-term features simultaneously.

To address the above challenges, we propose a CNN-Transformer model to identify one single track as the melody for a given input MIDI file. This architecture effectively captures local musical features at the level of a measure (or a frame) while also making predictions over the whole piece based on global aggregations of those local features. We also utilize sparse attention in the Transformer [23] to speed up computation and reduce memory requirements so longer musical works can be processed.

We conduct the MTI experiment on a manually annotated dataset of 11,625 MIDIs. The results show that the proposed model makes a breakthrough in the accuracy of MTI. We also evaluate our proposed model as a preprocessing plug-in in the three types of melody-sensitive tasks. The results indicate that our proposed model is easy to incorporate and effectively improves their performance. This work includes a dataset with 13,100 widely used MIDIs whose melody tracks have been labeled. This dataset will be an asset to other MIR studies in the future.

© Xichu Ma, Xiao Liu, Bowen Zhang and Ye Wang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Xichu Ma, Xiao Liu, Bowen Zhang and Ye Wang, "Robust Melody Track Identification in Symbolic Music", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

2. RELATED WORK

2.1 Melody-Sensitive Tasks

Many studies have been done on MIR tasks which are heavily influenced by melody. Whether any specific such task has melody as a target, melody as a requirement, or melody as a dependency, accurate identification and knowledge of melody can help solve that task.

2.1.1 Melody as a Target

Tasks which set the melody as a target seek to output a melody. These tasks include generating melodies, extracting melodies from existing music, and converting one melody into another. Most existing systems for these tasks require annotated melodic lines as labels. For instance, MSNet is an encoder-decoder network designed for melody extraction from audio [4]. It takes an audio spectrum as input, extracts a salience map via UNet-like networks, and then estimates the audio's melodic line. Another system treating melody as a target developed a LSTM-GAN model conditioned on lyrics to generate symbolic melody sequences [5].

2.1.2 Melody as a Requirement

Tasks which take melody as a requirement, by definition, require an explicitly labeled melodic line as an input. The aforementioned PiRhDy is a model designed for one such task: it needs an explicitly labeled melodic line so that it can predict a MIDI file's melodic notes by the contexts and thus learning the representation of the music [15]. Similarly, Melody2Vec [24] can only utilize MIDIs with labeled track names of "melody" or "vocal" as training data. Another example is AI-Lyrics, a system which automatically generates lyrics to parallel an input musical piece's style and syllable alignment [8]. This generator requires a syllable template which can only be extracted from the melody track of the input MIDI files. Other lyric generators and singing synthesis systems also require vocal melody to be explicitly identified as inputs [7, 9, 10]. Currently, the melodic lines in MIDI files must be manually labeled for those files to work with these systems, which hinders the systems' practical applications.

2.1.3 Melody as a Dependency

Tasks which use melody as a dependency do not require MIDIs with explicitly labeled melodic lines, but they are dependent on melody information nonetheless. For instance, MuSeReNet is a genre classifier for symbolic music which extracts latent features of MIDI files with CNNs and then feeds them into a Multilayer Perceptron (MLP) to identify their genre [11]. Genre is heavily influenced by melody, so better knowledge of melody would no doubt help improve this system's accuracy. Another example is MusicBERT, which is a large-scale pre-trained model for music understanding, resembling the BERT from the field of Natural Language Processing (NLP) [25, 26]. It pre-trains the Transformer encoder [18] by solving a pretext task where it learns a music embedding by reconstructing notes' eight music elements such as tempo, pitch, velocity, and many others. Considering these elements are influenced by melody, we believe the knowledge of melody would likely help the learning.

2.2 Melody Identification

Melody extraction and identification problems can also be subdivided based on the different source and target data forms. The aforementioned MSNet is designed to extract melodic lines from acoustic audio, but though it achieves SOTA performance, it is sensitive to pre-defined settings such as input frame length and is not robust to inputs which do not match with those settings. Other models attempt to perform melodic line extraction from symbolic music [2, 3]. Namely, they output a note sequence representing the melodic line of an input MIDI. Hsiao et al., for instance, uses 1D-CNN networks to extract contextual information and further predict the probability of every pair of notes belonging to the same track. However, this model requires a lot of computation and thus have difficulty handling scaled up data size in a short time.

Rather than generating a note sequence as the melodic line, MTI systems aim to simply label the track which contains the melody. The Skyline algorithm, for instance, picks the track with the highest average pitch as the melody track [27]. There also exist Bayesian approaches which estimate a track to be a melody track if that track maximizes an accumulated probability derived from features such as note velocity, pitch values, and so on [28]. There are few deep learning approaches seeking to identify a single track of a MIDI file as the melody track. However, the emergence of self-attention mechanism makes the deep-learning-based models more powerful to cope with short term and long term musical structures simultaneously. And this motivates us to develop our CNN-Transformer model for this problem.

3. METHOD

In this section, we formulate the MTI problem, then present the data representation and the architecture design.

3.1 Problem Formulation

First, we define robustness of a MTI model for symbolic music in four aspects. A robust MTI model should be capable of identifying the melody tracks from MIDI files that 1) do not follow pre-defined track configurations (i.e., the first channel contains the melody track, the melody track is named as "Melody", etc.), 2) have several melodic and non-melodic tracks within one channel, 3) consist of various musical parameters (instruments, tempos, velocities and time signatures), and 4) diverse in genres, styles, and sentiments.

An arbitrary piece of symbolic music in MIDI format is denoted as $\mathcal{S} = \{Tr_N\}$, where N denotes its number of tracks, each track $tr_i \in Tr$ is a sequence of MIDI events, which trigger control demands or music notes. Namely,

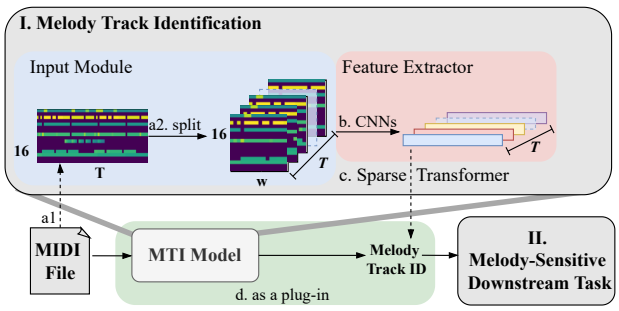


Figure 1. Dataflow of our method. (a) MIDI quantization and split. (b) CNN feature extractor. (c) Global information aggregator. (d) Plug in downstream tasks.

$tr_i = [e_0, e_1, ..., e_M]$ where $e_j \in [0, 127]$. Given an N-track MIDI file $\mathcal{S}$, our task is to predict the ID y^i of one single melody track, formally, $Y = \{y^i|\mathcal{S}\}$. Since a MIDI file has at most 16 sound output channels and one channel can contain several tracks, our MTI model can be seen as a supervised 16-way multirun classifier. When the melody track occupies an entire channel, the initial prediction result is the melody track ID; When the melody track shares a channel with tracks of other musical components, the model reruns a second prediction to further distinguish the melody track from the initially identified channel. Therefore, the prediction process is definitively convergent in finite time.

As shown in Figure 1, the data flow of our proposed robust MTI model is divided into several modules. First, an input MIDI file goes through the input module for preprocessing: it is quantized and serialized as a matrix with a resolution of sixteenth note (Figure 1-a1). Then the matrix is split into overlapped frames with step-wise scanning (Figure 1-a2). Second, a frame level CNN extracts the local features of a frame (Figure 1-b) and a sparse Transformer based classification module predicts the melody track ID based on the aggregated global features (Figure 1-c). Taking in a MIDI and outputting its melody track ID, the whole MTI module (Figure 1-I) can serve as a preprocessing plug-in for other melody-sensitive MIR tasks (Figure 1-II).

3.2 Input Module

Our input module converts an input MIDI file $\mathcal{S}$ into a 2D matrix $\boldsymbol{M} = [m_{ct}]_{C\times T}$, where m_{ct} denotes the pitch value in the c^{th} channel at the t^{th} time step, and T denotes the total length of the MIDI. One time step corresponds to a sixteenth-note length so that the encoding is tempo independent. If there are multiple tracks containing different pitch notes in the same channel c at time t, m_{ct} is temporarily set to the highest pitch value. The module also ensures that all input MIDI files have a full sixteen channels by padding any files with channels of zeros. This encoding provides following advantages against the piano roll representation: 1) It has a fixed dimension of channels; 2) It omits the velocity which can confuse the MTI model; 3) The tracks sharing the same channel can be temporarily fused into one representative pitch sequence and further distinguished in the rerun of the model later.

Inspired by Simonetta et al. [2], we apply step-wise scanning to split the 2D representation of a MIDI file into overlapping frames of fixed window size, w, equivalent to one bar. Each resultant frame is thus a $w \times w$ matrix $\boldsymbol{m}^{(i)}_{t-l:t+w-l}$, where $\boldsymbol{m}^{(i)}_{t-l:t+w-l}$ represents the frame centers around the t^{th} time step in the i^{th} MIDI file, and l represents the preceding l time steps.

3.3 Architecture Design

We propose a CNN + Sparse Transformer structure to learn local context information of each frame while also aggregating global information over the entire sequence. We also propose to overcome the memory bottleneck that a vanilla Transformer imposes. This enables the system to efficiently process longer music by reducing the density of self-attention connections. We adopt a hierarchical training strategy: First, we pretrain the CNN-based feature extractor by predicting the melody track ID at frame level; then, we fine-tune the whole model by predicting a single melody track ID of the input MIDI.

3.3.1 Local Context-Aware Feature Extraction

We use a CNN as our feature extractor to learn the local context information of given frames provided by the input module. The architecture for this portion of the system is shown in Figure 2. The feature extractor ϕ first goes through pretraining. Specifically, we add a temporary MLP layer with weights $\boldsymbol{W}$ and bias b to each frame to predict its fame-level melody track ID. The predicted probability for a certain frame $\boldsymbol{m}^{(i)}_{t-l:t+w-l}$ can be denoted as

$$\hat{y} = \text{softmax}(\boldsymbol{W}\phi(\boldsymbol{m}^{(i)}_{t-l:t+w-l}) + b) \tag{1}$$

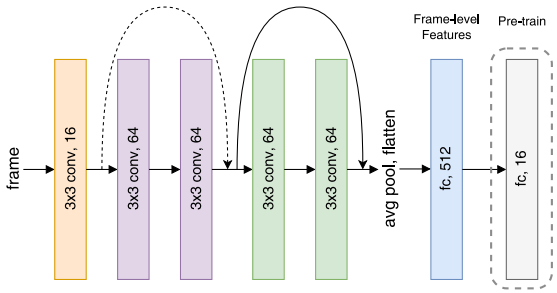


Figure 2. The architecture of our feature extractor. Shortcuts represent skipped paths. Dotted shortcut represents dimension increases. The last fully connected layer (grey) is only used in pre-training.

3.3.2 Global Classification Module

With the pre-trained feature extractor ϕ mentioned above, each MIDI file can now be transformed and concatenated into a context-aware embedding

$$\boldsymbol{M}^{(i)} = \mathop{||}_{t\in[1,T]} \phi(\boldsymbol{m}^{(i)}_{t-l:t+w-l}) \tag{2}$$

where $\boldsymbol{M}^{(i)} \in \mathbb{R}^{T\times d}$, $||$ denotes concatenation over time steps, and d denotes the dimension of feature extractor ϕ.

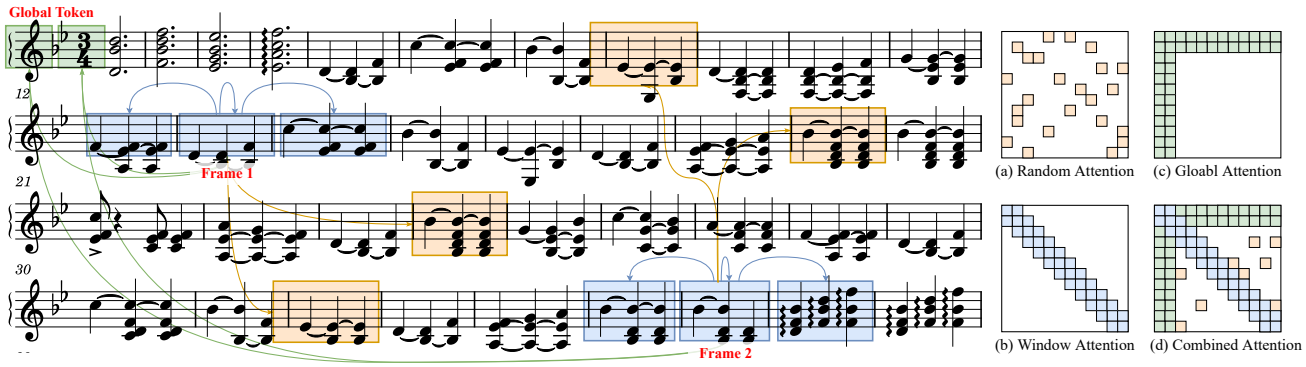


Figure 3. Left: an example of a sparse self-attention mechanism adapted in symbolic music, where frames 1 and 2 attend to three neighboring frames (blue), two random frames (orange) and a global frame (green) respectively. Right: an illustration of each sparse self-attention mechanism proposed by Big Bird, where $g = 2$, $w = 3$, $r = 2$.

We then apply and fine-tune the global feature aggregators (Sparse Transformer) upon the feature extractor by predicting a single melody track ID based on the aggregation of the frame-level features. Taking the sequence of the concatenated CNN features, the Sparse Transformer computes a sequence of embeddings with the same length and then aggregates to one single label, paying attentions to other positions in its computation. The computation of the multi-head self-attention is the same as the typical Transformer:

$$\mathcal{A}(x_i) = x_i + \phi_{attn}(\underset{h}{||}\ \delta(\frac{\mathcal{Q}_h(x_i)\mathcal{K}_h(x_i)^T}{\sqrt{d}})\mathcal{V}_h(x_i)) \quad (3)$$

where $\mathcal{Q}_h(\cdot)$, $\mathcal{K}_h(\cdot)$, $\mathcal{V}_h(\cdot)$ respectively denote the query, key and value function of x_i, $\delta(\cdot)$ denotes the scoring function, $\underset{h}{||}\cdot$ denotes concatenation over attention heads, and ϕ_{attn} denotes the projection applied after concatenation. Especially, the sparse self-attention reduces the density of self-attention connections to accommodate longer inputs.

3.3.3 Sparse Self-Attention In Symbolic Music

In a typical Transformer, the full self-attention mechanism requires a quadratic dependency on sequence length. However, the length of MIDI files is generally long; directly applying a full self-attention operation on it can lead to memory overflow. Rather than simply splitting the embedding $\boldsymbol{M}^{(i)}$ into shorter segments [29], which can cause a loss of global information, we leverage Big Bird [23], a transformer with sparse self-attention mechanism. Specifically, the following sparse self-attention mechanisms are proposed: 1) **Random Attention**, all frames attend to r random frames in the sequence, allowing the global information aggregator to have better generalization ability. 2) **Window Attention**, all frames attend to w neighboring frames in the sequence, functioning as an information gateway of a music bar. 3) **Global Attention**, g newly introduced global frames (e.g., a classification token [CLS] preceding the sequence) attend to all frames in the sequence, collecting the global information. By combining these three sparse attention mechanisms, we manage to approximate the effectiveness of a full self-attention meanwhile reducing the time and space complexity from $O(n^2)$ to $O(n)$. Figure 3 gives an example of how frames sparsely attend in symbolic music.

4. EXPERIMENTS AND RESULTS

This section contains the experiment design and the analysis of the results. First, we detail the construction of the datasets. Then we compare our proposed model with several baselines and alternative architectures so as to evaluate if our proposed MTI model can attain SOTA accuracy. Finally, we evaluate our model on three melody-sensitive downstream tasks, including melody segregation, music embedding learning, and genre classification.

4.1 Dataset

We create four different datasets, one per experiment, to train and evaluate our models. Each dataset is split into a training set and a validation set with an 8:2 ratio unless otherwise noted.

4.1.1 For MTI Model

We collect MIDI songs from three widely used MIDI datasets: LMD [30], Reddit MIDI dataset [1], and Free-MIDI [2]. We then eliminate samples with no obvious melody track, such as MIDIs of percussion instruments and chord progression patterns, and we also eliminate MIDI files where the melody changes tracks. Next, we manually annotate the melody track of the remaining songs and shuffle their melody track IDs for label distribution balance. The resultant dataset contains 11,625 samples for model training and validation (**D-MTI**). Additionally, the trained MTI model automatically annotates another 1,475 files in the Free-MIDI dataset, providing an overall dataset of 13,100 MIDI files (**D-Full**) [3] after manual checking. The proposed datasets are desirable for training and evaluating a robust MTI model because the data covers a variety of genres (17 in total), track configurations (track orders and names), as well as MIDIs with multi-track channels.

4.1.2 For Downstream Task Evaluations

To evaluate our MTI model's utility for downstream MIR tasks, we build three more datasets based on Free-MIDI: a dataset of 1,100 randomly selected MIDIs, including audio

[1] https://www.reddit.com/r/datasets/
[2] https://github.com/josephding23/Free-Midi-Library
[3] https://github.com/maxichu/MelodyTrackIdentification

renderings of the MIDIs and of their automatically identified melody tracks (denoted as **D-Audio**), a dataset of 5,000 genre-labeled MIDI files of 17 genres (**D-Genre**), and a dataset of 500 MIDI files of pop songs (**D-POP**).

4.2 Melody Track Identification

To evaluate the effectiveness, efficiency and robustness of our proposed MTI model, we select the following three baseline models: the Skyline algorithm, a Bayesian probability model with dynamic programming [28], and a 1D-CNN model with clustering [3]. The CNN-based melody segregation model [2] is not selected for comparison because it seeks to directly extract the melodic line rather than predict the melody track. We also selected three alternative architectures: a BiLSTM architecture, a CNN + MLP architecture and a CRNN architecture. We then compare our proposed CNN + Sparse Transformer (CNN-ST) architecture with all the above models.

Model	Accuracy(%)	Running Time(s)
Skyline	14.7	252.52
Bayesian	40.98	1094.15
1D-CNN	5.51	14818.89
BiLSTM	10.23	12.37
CNN-MLP	83.65	36.12
CRNN	81.60	32.74
CNN-ST	**84.40**	29.71

Table 1. Accuracy and efficiency of MTI models. The running time is the total time of infering 1,000 samples. (The frame-level accuracy of the basic CNN model is 49.72%)

As shown in Table 1, our proposed model achieves SOTA results without incurring extra computational costs. Both Skyline and the 1D-CNN model suffer sharp reductions in accuracy when dealing with samples which are not highly normative or formative. We believe that Skyline's dependency on music composition convention hinders its generalization, while the clustering phase of 1D-CNN model might be the bottleneck of the capability to accommodate complicated data sources. The Bayesian approaches obtains a lower accuracy of 40.98% on the proposed dataset than the reported 89% accuracy on a standard MIDI dataset. Its definitions of scoring algorithms targeting at only classical music could account for the accuracy drop. These issues can cause their respective algorithms to have low accuracy when processing long music with many different arrangements and genres. Worse still, the high time complexity of the 1D-CNN model can make it too costly to be incorporated into other models or applications.

As shown in Table 1, the comparison between BiLSTM and our proposed CNN-ST shows the CNN's effectiveness in capturing local musical features. The accuracy improvement over frame-level CNN indicates the ability of aggregating global information from frames of the proposed self-attention module. In the case study, our model handle the following occasions better than the baselines: 1) There are other components having a higher average pitch than the melody; 2) The music notes are dense and in large quantity; 3) There are tracks echoing the high-pitch part of the melody. Therefore, our proposed model leads in accuracy, efficiency, and robustness for the MTI task.

4.3 Downstream Tasks

We test our MTI model in three downstream tasks of different types to evaluate its benefits to the MIR community: 1) Improving melody segregation of audio music by providing more easily-obtained training samples; 2) Improving musical embedding learning by allowing more rigorous assumptions on melody; 3) Improving genre classification by emphasizing the impacts of the melody.

4.3.1 Melody Segregation

As a typical melody-as-a-target task, melody segregation requires pairs of {audio music, melody} for training. However, such data is laborious to obtain–labeling large numbers of melodic lines requires a significant amount of time and effort from trained music specialists. Training with small datasets may lead to underfitting issues, such as a recent melody segregation work whose dataset only contained 108 samples [4]. Furthermore, existing paired datasets are often collected from just a few genres, or even one genre, of music. Models trained by those datasets thus often fail to generalize to music from other genres.

This experiment aims to test whether our MTI model can contribute to the melody segregation task by both automatically labeling the melody tracks of MIDI files in order to rapidly expand the amount of audio, including audio from various genres, formats and compositional rules, which can be used to train these systems.

ID	Pretrain Data	Train Data	OA (%)
1	/	MedleyDB	37.7
2	/	D-Audio-1K	39.91
3	/	MedleyDB ∪ D-Audio-1K	**41.1**
4	D-Audio-1K	MedleyDB	**40.23**

Table 2. Accuracy of different training data settings for the melody segregation task. All results are computed on an independent 100-sample validation subset of D-Audio. OA is short for overall accuracy.

We trained four melody segregation models with the same architecture [4] but different datasets. Our control model was trained with the 108 samples of the original MedleyDB [31]. Our three other models were trained with D-Audio-1K (1,000 samples in D-Audio), a combined dataset of MedleyDB and D-Audio-1K, and D-Audio-1K (pretraining) followed by MedleyDB (fine-tuning), respectively. Validation was done with the set of the remaining 100 samples from D-Audio, denoted as D-Audio-100.

As shown in Table 2, not only does an increased amount of training data improve melody segregation accuracy but pretraining with the D-Audio-1K dataset provides superior results to using either of those datasets on their own. The

pretraining, however, does worse than the setting where all the data is used for training. This can be attributed to the bias of data distribution between the two datasets.

4.3.2 *Music Embedding Learning*

Music embedding learning is a melody-as-a-requirement task. Many music embedding learning models are trained by predicting the melody notes according to other music components. However, these models generally rely on the Skyline algorithm or track names for data preparation, which can result in errors if they do not find the melody correctly [15, 24].

We conduct an experiment with the PiRhDy embedding model [15] to evaluate how well our proposed MTI can improve music embedding learning models. We run all of PiRhDy's three subtasks for performing music embedding (token modeling, next context modeling, and accompaniment context modeling) on both the D-POP and D-Gen datasets. Before executing those subtasks, we run the Skyline and the proposed CNN-ST+ model to identify melodic lines for the tracks being analyzed.

As shown in Table 3, all three subtasks achieve higher accuracy when using melody tracks identified by our proposed model as compared to Skyline. The best improvement occurs in the subtask of token modeling on the D-Gen dataset, which indicates that our system better handles unusual genres than the existing Skyline algorithm too.

/	Skyline+ D-POP(%)	CNN-ST+ D-POP(%)	Skyline+ D-Gen(%)	CNN-ST+ D-Gen(%)
Token Modeling	60.49	**74.61**	74.77	**96.74**
Context Modeling-Next	91.43	**92.05**	91.89	**94.58**
Context Modeling-Acc	67.45	**78.74**	55.37	**56.53**

Table 3. Accuracy of models using different methods for PiRhDy embedding learning.

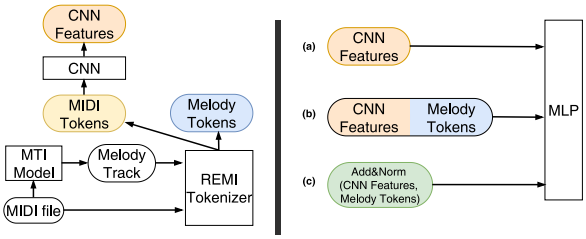


Figure 4. The three compared architectures for music genre classification with different inputs of the MLP layer. The inputs of the MLP are: (a) only the CNN features; (b) concatenation of CNN features and the melody tokens; (c) CNN features Add-Norm with the melody tokens.

4.3.3 *Genre Classification*

As a melody-as-a-dependency task, genre classification can be enhanced by explicitly emphasizing the melody over other musical components. This experiment is conducted on D-Gen. As shown in Figure 4, the full-track MIDIs and the identified melody track are first both tokenized into REMI [32] sequences with MidiTok toolkit [33]. Next, genre prediction is attempted with three different architectures. All of the architectures are CNN + MLP, and their input layers are just CNN features, CNN features appended with tokenized melody note sequences, and the CNN features Add-Norm with the tokenized melody note sequence respectively.

The results in Table 4.3.3 show that the accuracy increases when feeding the MLP layer with extra information about the melody. Therefore, highlighting the melody by either appending or blending it with the CNN features makes the music genre more distinguishable.

Inputs to MLP	Accuracy(%)
CNN features	41.51
CNN features ‖ Melody Tokens	**42.75**
Add&Norm (CNN features, Melody Tokens)	**45.76**

Table 4. Accuracy of models with different inputs to MLP layers for genre classification.

5. FUTURE WORK

We notice that emphasizing the melody track could dilute models for certain tasks where melody is not decisive, or at least is not the only decisive factor. For instance, we trained systems for key recognition [34, 35] with full MIDI files, just the melody tracks of MIDI files, and just the non-melody tracks of MIDI files. Whether using a rule-based algorithm [34] or a deep learning-based model [35], the systems trained with full MIDI files outperformed the others, and the systems trained with non-melody tracks outperformed those trained with just melody tracks. We thus estimate that accompaniments may matter more than melody for this task and are looking into ways to tweak our algorithm to account for this.

Also, the samples in our collected dataset contain only one melody track, and so for simplicity, we neglect scenarios such as multi-track melodies and melodies which switch tracks. In the future, we will develop a stronger model which can account for these scenarios even though they make the global musical structure of the song more challenging, and will also create datasets with more detailed annotations that can account for the position of the melody on a frame-by-frame basis.

6. CONCLUSION

Melody track identification is the first step towards music understanding and benefits melody-sensitive MIR tasks. This paper addresses the challenges of identifying the melody track accurately, efficiently, and robustly for any input MIDI. Our experiments show that our proposed model achieves SOTA performance and increases accuracy for many downstream tasks. We are optimistic that further research in this direction will not just enhance the ability of computers to identify melody tracks but will also improve many other aspects of MIR.

7. ACKNOWLEDGEMENTS

This project is funded in part by a grant (R-252-000-B75-112) from Singapore Ministry of Education.

8. REFERENCES

[1] E. Selfridge-Field, "Conceptual and representational issues in melodic comparison," *Computing in musicology: a directory of research*, no. 11, pp. 3–64, 1998.

[2] F. Simonetta, C. Cancino Chacón, S. Ntalampiras, and G. Widmer, "A convolutional approach to melody line identification in symbolic scores," pp. 924–931, 2019.

[3] Y.-W. Hsiao and L. Su, "Learning note-to-note affinity for voice segregation and melody line identification of symbolic music data," in *22nd International Society for Music Information Retrieval Conference*, 2021.

[4] T.-H. Hsieh, L. Su, and Y.-H. Yang, "A streamlined encoder/decoder architecture for melody extraction," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 156–160.

[5] Y. Yu, A. Srivastava, and S. Canales, "Conditional lstm-gan for melody generation from lyrics," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 17, no. 1, pp. 1–20, 2021.

[6] J. Wu, C. Hu, Y. Wang, X. Hu, and J. Zhu, "A hierarchical recurrent neural network for symbolic melody generation," *IEEE transactions on cybernetics*, vol. 50, no. 6, pp. 2749–2757, 2019.

[7] Y. Chen and A. Lerch, "Melody-conditioned lyrics generation with seqgans," in *2020 IEEE International Symposium on Multimedia (ISM)*. IEEE, 2020, pp. 189–196.

[8] X. Ma, Y. Wang, M.-Y. Kan, and W. S. Lee, "Ailyricist: Generating music and vocabulary constrained lyrics," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 1002–1011.

[9] O. Angelini, A. Moinet, K. Yanagisawa, and T. Drugman, "Singing synthesis: With a little help from my attention," in *Interspeech 2020*, 2020. [Online]. Available: https://www.amazon.science/publications/singing-synthesis-with-a-little-help-from-my-attention

[10] P. Lu, J. Wu, J. Luan, X. T. 0003, and L. Zhou, "Xiaoicesing: A high-quality and integrated singing voice synthesis system," in *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, 2020, pp. 1306–1310.

[11] E. Dervakos, N. Kotsani, and G. Stamou, "Genre recognition from symbolic music with cnns," in *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*. Springer, 2021, pp. 98–114.

[12] F. Simonetta, F. Carnovalini, N. Orio, and A. Rodà, "Symbolic music similarity through a graph-based representation," in *Proceedings of the Audio Mostly 2018 on Sound in Immersion and Emotion*, 2018, pp. 1–7.

[13] T. Endo, S.-i. Ito, Y. Mitsukura, and M. Fukumi, "The music analysis method based on melody analysis," in *2008 International Conference on Control, Automation and Systems*. IEEE, 2008, pp. 2559–2562.

[14] J. Salamon, B. Rocha, and E. Gómez, "Musical genre classification using melody features extracted from polyphonic music signals," in *2012 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE, 2012, pp. 81–84.

[15] H. Liang, W. Lei, P. Y. Chan, Z. Yang, M. Sun, and T.-S. Chua, "Pirhdy: Learning pitch-, rhythm-, and dynamics-aware embeddings for symbolic music," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 574–582.

[16] M. R. Jones, "Music perception: Current research and future directions," *Music perception*, pp. 1–12, 2010.

[17] D. Bahdanau, K. Cho *et al.*, "Neural machine translation by jointly learning to align and translate. arxiv preprint arxiv: 1409.0473," 2014.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[19] S. Dai, X. Ma, Y. Wang, and R. B. Dannenberg, "Personalized popular music generation using imitation and structure," *arXiv preprint arXiv:2105.04709*, 2021.

[20] D. Rizo, P. J. P. De Leon, A. Pertusa, C. Pérez-Sancho, and J. M. I. Quereda, "Melody track identification in music symbolic files." in *FLAIRS Conference*, 2006, pp. 254–259.

[21] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, X. Gu, and G. Xia, "Pop909: A pop-song dataset for music arrangement generation," in *ISMIR*, 2020.

[22] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and J. P. Bello, "Guitarset: A dataset for guitar transcription." in *ISMIR*, 2018, pp. 453–460.

[23] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 283–17 297, 2020.

[24] T. Hirai and S. Sawada, "Melody2vec: Distributed representations of melodic phrases based on melody segmentation," *Journal of Information Processing*, vol. 27, pp. 278–286, 2019.

[25] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "Musicbert: Symbolic music understanding with large-scale pre-training," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 791–800.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, 2019, pp. 4171–4186.

[27] A. Uitdenbogerd and J. Zobel, "Melodic matching techniques for large music databases," in *Proceedings of the Seventh ACM International Conference on Multimedia (Part 1)*, ser. MULTIMEDIA '99, New York, NY, USA, 1999, p. 57–66. [Online]. Available: https://doi.org/10.1145/319463.319470

[28] Z. Jiang and R. B. Dannenberg, "Melody identification in standard midi files," in *16th Sound & Music Computing Conference*, 2019, pp. 65–71.

[29] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. H. Engel, "Sequence-to-sequence piano transcription with transformers," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, 2021, pp. 246–253.

[30] C. Raffel, *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

[31] R. M. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam, and J. P. Bello, "Medleydb: A multitrack dataset for annotation-intensive mir research." in *ISMIR*, vol. 14, 2014, pp. 155–160.

[32] Y.-S. Huang and Y.-H. Yang, "Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions," in *Proceedings of the 28th ACM International Conference on Multimedia*, 2020, pp. 1180–1188.

[33] N. Fradet, J.-P. Briot, F. Chhel, A. El Fallah-Seghrouchni, and N. Gutowski, "Miditok: A python package for midi file tokenization," in *22nd International Society for Music Information Retrieval Conference*, 2021.

[34] D. Temperley, "What's key for key? the krumhansl-schmuckler key-finding algorithm reconsidered," *Music Perception*, vol. 17, no. 1, pp. 65–100, 1999.

[35] F. Foscarin, N. Audebert, and R. Fournier-S'Niehotta, "Pkspell: Data-driven pitch spelling and key signature estimation," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, 2021, pp. 197–204.

TRACKING THE EVOLUTION OF A BAND'S LIVE PERFORMANCES OVER DECADES

Florian Thalmann, Eita Nakamura, Kazuyoshi Yoshii
Graduate School of Informatics, Kyoto University, Japan
thalm007@umn.edu, {eita.nakamura, yoshii}@i.kyoto-u.ac.jp

ABSTRACT

Evolutionary studies have become a dominant thread in the analysis of large audio collections. Such corpora usually consist of musical pieces by various composers or bands and the studies usually focus on identifying general historical trends in harmonic content or music production techniques. In this paper we present a comparable study that examines the music of a single band whose publicly available live recordings span three decades. We first discuss the opportunities and challenges faced when working with single-artist and live-music datasets and introduce solutions for audio feature validation and outlier detection. We then investigate how individual songs vary over time and identify general performance trends using a new approach based on relative feature values, which improves accuracy for features with a large variance. Finally, we validate our findings by juxtaposing them with descriptions posted in online forums by experienced listeners of the band's large following.

1. INTRODUCTION

Music information retrieval has proven essential for the *analysis of large audio corpora*, especially ones for which traditional music analysis methods are limited. Such cases include large audio collections for which there is no appropriate symbolic transcription, such as ones containing non-western music or improvised music [1–5].

In recent years numerous studies have characterized the temporal evolution of musical characteristics in such corpora. Serra et al identified a restriction of pitch transitions, homogenization of timbral palette, and growing loudness levels in Western popular music [6]. On a similar corpus, Mauch et al discovered three stylistic revolutions between 1960 and 2010, based on topics identified via latent Dirichlet allocation of harmonic and timbral features [7]. Deruty and Pachet determined the 'loudness war' in popular music production to have peaked in 2007 [8]. Weiss et al found harmonic complexity to be gradually increasing in Jazz solos between the 1920s and the 2000s [4]. Weiss et al also confirmed common hypotheses concerning the evolution of chord transitions, intervals, and tonal complexity in Western classical music [9]. Parmer and Ahn measured information-theoretic complexity of pitch, loudness, timbre, and rhythm in a popular music dataset and identified trends over decades [10]. [1]

All of these studies look at music at a social or cultural level and use audio corpora that consist of material by various composers or musicians, as well as of different genres, subgenres, instrumentations, etc. In this paper we present an analogous study which however focuses on the *music of a single band*, the Grateful Dead, who are well-known for their ever-evolving performances. We use a dataset that consists of audio recordings of 2617 performances of 15 songs spanning three decades. Although this may be a sizable dataset for a band, we find that musical characteristics show a large variance over the relatively short timespan. For example, the tempos of the performances in the set range from 50 to 160, which over the whole time span results in a relatively sparse cloud of data points.

Previous approaches are relatively limited when dealing with such diverse data. They all consider the corpus as a whole and simply plot the evolution of audio features against time. This may work reasonably well for some musical characteristics such as tonal complexity, timbre, or loudness. However, we show how subdividing the corpus into subsets, in our case songs, and considering feature data relative to these subsets before integrating them into a whole can improve accuracy and confidence for the detection of overall trends. We identify such trends in various performance characteristics and juxtapose them with observations by experienced listeners from the band's large following. In particular, despite working with a relatively small subset of only 15 of the band's songs, we are able to identify generally perceived trends in tempo, song duration, and dynamic, spectral and harmonic content with promising accuracy.

2. DATASET

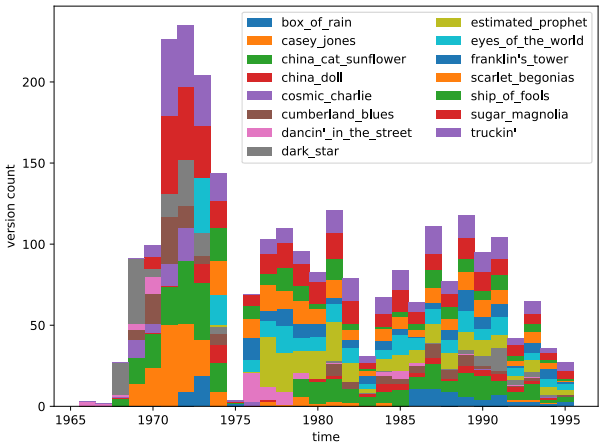
The cultural impact and decades long performance history of the Grateful Dead has led to continued interest in the band's music by both fans and scholars [14]. The band is especially known for their free and inclusive approach to music, their unwillingness to bow to the conventions of popular music, and their aspiration to provide their fans

[1] Similar trend analyses were recently done with non-audio music collections, such as with lyrics [11] or symbolic data [12, 13].

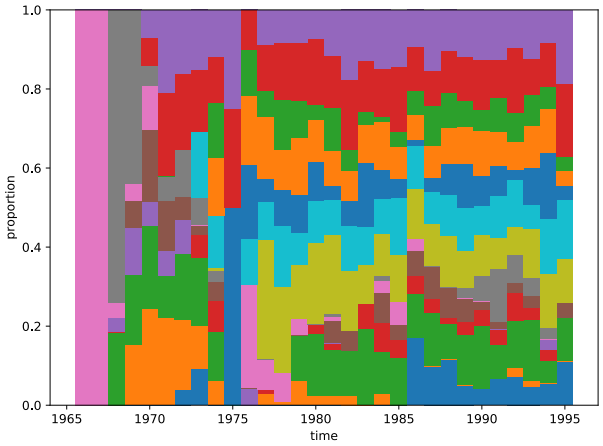
© F. Thalmann, E. Nakamura, and K. Yoshii. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** F. Thalmann, E. Nakamura, and K. Yoshii, "Tracking the Evolution of a Band's Live Performances over Decades", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

with a new experience at every concert. The performances of their songs often vary greatly, even from day to day, and they often engage in long improvisational parts or jams. Almost all of their over 2000 concerts between 1965 and 1995 have been recorded, often multiple times, and most of these recordings are public domain and included in the Live Music Archive's (LMA) largest subcollection. [2] To musicologists, this catalog may be intimidating for these reasons [15], but this make it all the more interesting for studies using MIR methods.

We use a dataset introduced by [5] [3] which includes 2617 performed versions of a set of 15 songs from the Grateful Dead collection of the Live Music Archive. According to the authors, the songs were selected based on two criteria: a large number of versions with soundboard recordings across the whole time span, as well as a studio recording as a potential reference. Many of these recordings contain crowd noise, which may affect the quality of features, and many are out of tune due to varying tape speed during recording. The dataset comes with a script that downloads the files from the LMA and automatically resamples them based on tuning ratios determined from chroma vectors. Figure 1 (a) shows a chronological distribution of the files. We observe very low counts for the first two years which may be due to a lower number of available recordings, and 1975 when the band retired for a year. A relative distribution of songs across the years (Figure 1 (b)) shows that the first two years of the dataset only contain one song, which may be problematic for an evolutionary analysis. A more systematic generation of such a dataset may prove useful in the future, but we chose to use it here without modifications.



(a)



(b)

Figure 1: (a) Chronological distribution of the recordings for each song in the dataset. (b) Relative distribution of songs by year, order and colors correspond to (a).

3. METHOD

The fact that we have subsets of identical or similar musical pieces or recordings can be leveraged at different points in the process. First, a large number of audio features are extracted for each recording in the corpus, now tuned as described above. We then validate these features relatively for each song, which allows us to detect outliers, i.e. wrongly classified songs, as well as adjust wrongly extracted features such as double-time beats. Our statistical analysis includes two steps. We first analyze the feature distribution and evolution for each song independently, which allows us to characterize the relative evolution of normalized feature values for each song. These relative evolutions are then collated into a global evolution curve for each feature, which we validate using bootstrapping, i.e. by alternately leaving out each song.

3.1 Feature Extraction

The set of audio features used in our study were inspired by previous work on other corpuses referenced in Section 1 and extracted using madmom [4] (*beats*, from which we derived tempo), librosa [5] (*chroma* and *mfcc*, summarized to the madmom beats), as well as the essentia freesound extractor [6] (for *dynamic* and *spectral* summary features). We used standard settings for all extractors, except for with madmom's DBNBeatTrackingProcessor where we used a much higher transition lambda of 2000 rather than the suggested maximum of 300 in the documentation. This was to reduce the probability of the processor jumping between tempos within one audio file, which may have been occurring due to the many long versions (median over 10min, sometimes over 30min), as well as the amount of crowd noise.

We also calculated a few additional features, inspired by other studies. *Tonal complexity* as defined by Weiss et al as the angular deviation or spread of pitch-classes on a circular chroma vector, permuted to correspond to the circle of fifths [4,9,16]. To measure pitch content agnostic of tonality, we devised an analogous *pitch complexity* feature based

[2] https://archive.org/details/GratefulDead

[3] https://github.com/grateful-dead-live/fifteen-songs-dataset

[4] https://madmom.readthedocs.io

[5] https://librosa.org

[6] https://essentia.upf.edu/freesound_extractor.html

on the reverse-sorted normalized summarized chroma vector, which results in a right-skewed distribution, of which we took the mean. From the 36-bin harmonic pitch class profiles (HPCP) extracted by essentia, we derived a *tuning complexity* feature, which expresses the average deviation from 440Hz in both directions. [7]

3.2 Beat Correction and Outlier Detection

The extracted features allow us to address two frequently occurring problems. The first problem is that beat tracking is often inconsistent between different versions of a song, due to the large variance in tempo and the high amount of improvisation in the dataset. The second problem is that some song sets may contain incomplete fragments or recordings of other songs, due to mis-annotations in the Live Music Archive, as noted by Page et al [17].

Our *beat correction* method consists of identifying instances of beat features that should rather be double-time, half-time, or two-thirds-time. [8] We start with the original beat pairings $B^i_b = (R^i, b_{R^i})$ of recordings R^i, $i = 1 \dots K$, and their corresponding beat sequences b_{R^i}, which are simply sequences of time points, as extracted by a feature extractor. For each beat sequence b_{R^k} we create three variants, a linearly interpolated *double-time* version d_{R^k}, [9] a *half-time* version h_{R^k} which only contain every other beat, and a *two-thirds-time* version t_{R^k} which contain every third beat of d_{R^k}. We end up with four sets of K beat sequence pairings $B^i_b, B^i_d, B^i_h, B^i_t$, defined analogously to B^i_b. For each of these pairings we extract a set of features $f_1, \dots f_F$ which all depend on the beat sequence in the pairing. For each feature f with a corresponding distance function δ_f, we then create four $K * K$ distance matrices $D^{x,f}_{ij} = \delta_f(B^i_x, B^j_b)$, one for each $x \in \{b, d, h, t\}$. These matrices express how well the four types of beat pairings match with the original pairings b_{R^i}. We normalize these matrices using min-max normalizations and calculate for each pairing B^k_x its average distance from the original pairings over all features

$$\Delta^k_x = \mu_{f=f_1 \dots f_F, j=1 \dots K}(D^{x,f}_{kj})$$

Finally, we choose for each recording R^k its best pairing B^k_x with minimum Δ^k_x.

The set of features that worked well for the dataset used in this paper are chord sequence distributions, onset distributions, and tempo. We generate a chord sequence distribution from a summarized chord sequence, where for every beat interval we take the chord that occupies the longest duration. From this sequence we gather all subsequences of length L at step size 1 and count the occurrences of each possible pattern. Onset distributions are generated by calculating the position of each onset o relative to its neighboring beats $b_<, b_>$, e.g $(o - b_<)/(b_> - b_<)$ and quantizing the resulting relative onsets to a grid of G values by multiplying them by G rounding to the nearest integer. Both kinds of distributions are normalized. Tempo is simply calculated as the average distance between successive beats. For beat correction, we chose the parameters $L = 4, G = 64$.

For distance functions δ_f we used the chi-square distance $\chi^2(x, y) = \frac{1}{2} \sum_i \frac{(x_i - y_i)^2}{x_i + y_i}$ for distributions, and the analogous $\frac{(x-y)^2}{x+y}$ for two single values x, y, such as for tempo.

Next, we identify *outlier recordings* using an extension of the feature set above, by adding chord sequence distributions with $L = 2$ as well as overall beat count, i.e. the length of the beat feature vector. Similar to the beat correction method, we calculate a distance matrix $D^f_{ij} = \delta_f(R^i, R^j)$ for each feature f and normalize it. We then calculate a normalized distance $d_{f,k}$ for each feature, consisting of the difference between the average deviation of R^k from the overall average deviation $\mu(D^f)$, normalized by overall standard deviation $\sigma(D^f)$.

$$d_{f,k} = \frac{\mu_{j=1 \dots K}(D^f_{kj}) - \mu(D^f)}{\sigma(D^f)}$$

Finally, we consider recordings as outliers if they deviate by 2.5 standard deviations in at least two features ($d_{f,k} > 2.5$) or by four standard deviations in at least one ($d_{f,k} > 4$). The latter condition particularly ensures that recordings of extreme length, such as incomplete fragments or improperly segmented files that include more than one song, get excluded due to an extreme deviation of overall beat count.

We obtained the best results by applying beat correction and outlier detection *successively and iteratively*, i.e. one after another in a loop, until both the beat positions and the set of included recordings are stable. Especially for songs with less regular structure, the process only terminates after several iterations, due to the reference beat sequences b_{R^j} changing at each step. Furthermore, with iterative application we can catch cases of quadruple time etc.

When applying this method to the dataset, a total of 211 outliers were removed (around 8%), including for a case where we identified a set consisting of recordings of two songs with similar titles, possibly due to wrong annotations. Of the beat features of the remaining versions, 99 were identified as half-time, 8 as two-thirds-time, and 21 as double-time.

3.3 Statistical Analysis

We are now ready to start our diachronic analysis of the feature data, which is structured as follows. For each feature we have one data point per version of every song. Each version is associated with a specific day on which it was played. For feature f we thus have the data points $P^f = \bigcup P^{f,1} \dots P^{f,15}$ where $P^{f,i} = (p^{f,i}_{k_i})_{k_i=1 \dots K_i}$ is the sequence of data points for song i with K_i versions.

Although our dataset consists of material from only one band, we face a relatively high variation in features, due

[7] These measures are also related to the information-theoretic complexity measures used by Serra et al or Parmer and Ahn [6, 10].

[8] Two-thirds-time has proven to be useful in situations where we have a ternary meter, such as $6/8$, which might be interpreted as either binary or ternary by the beat extractor.

[9] Each successive pair of beats is interspersed with an average of the two, and an additional beat is added at the end, at an interval corresponding to last interpolated one.

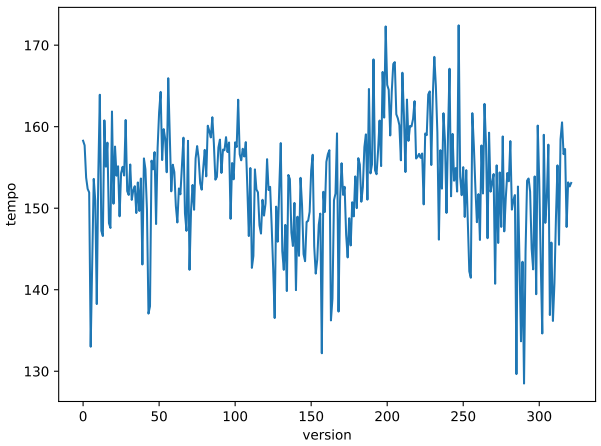


Figure 2: The chronological sequence of tempo data points of all 322 versions of *Sugar Magnolia*.

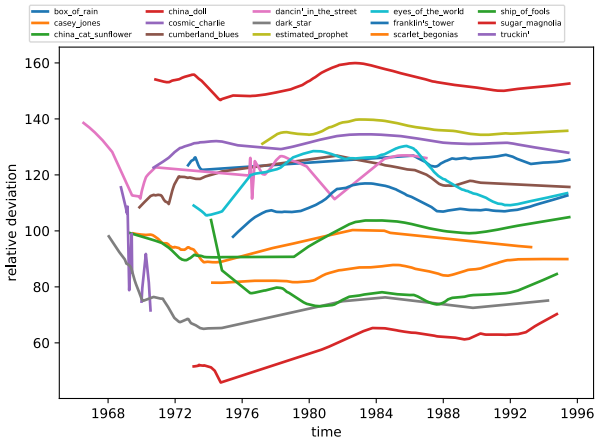


Figure 3: LOWESS plots of tempo for each individual song in the dataset.

to the constant experimentation and improvisation of the band. Figure 2 illustrates how much the tempo of the song *Sugar Magnolia* varies over time, sometimes by as much as 30% from one day to the next. We can identify a global trend in such data by creating a trend line using LOWESS [18], [10] which fits a polynomial regression line at each point, based on a local neighborhood. This neighborhood consists of a fixed proportion of all points, which we set to $f = .2$ or 20%. LOWESS works particularly well for data that is unevenly distributed along the x axis, which is the case here due to the uneven distribution of songs across time (Figure 1).

We can calculate such curves for each song individually, as shown in Figure 3. The top-most curve in this figure corresponds to the line shown in Figure 2 and we can identify two tempo peaks, one around 1973 and one around 1983. It is also apparent that the overall tempo range in the collection varies by more than a factor of 3, from around 50 to almost 160. In order to get a sense of how the tempo changes over all songs, we can create a plot for the whole set of songs, LOWESS(P^f). However, in order to make sure that the curve is not dominated by the values of a single song, we calculate a confidence interval using bootstrapping, leaving out each song once. [11] For each $j = 1 \ldots 15$ we calculate LOWESS($\bigcup_{i \neq j} P^{f,i}$), and determine at each time point the minimum and maximum value among these curves to get the confidence interval.

With absolute values, this confidence interval turns out to be quite large, leaving us unable to draw conclusions with certainty, as shown by the orange curve in Figure 4. For example, we cannot confidently say that the peak observed in 1984 is higher than the one in 1972, due to the overlapping confidence intervals. However, we can get a much better estimate of how the tempo changes by taking *relative feature values*, which we can simply calculate by

[10] As used by Moss et al in their analysis of the evolution of the distribution of pitch-class content on the line of fifths [13].

[11] An alternative method for this is bootstrap resampling, where we could ensure an equal number of points per song. However, due to limitations found in the dataset (after outlier detection one song only has 22 versions), we chose to use the present method.

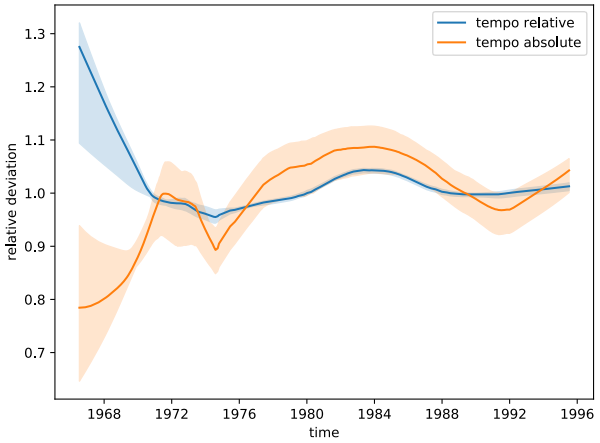


Figure 4: Bootstrapped LOWESS curves on the whole dataset for absolute and relative tempo values.

normalizing the sequences $(p_{k_i}^{f,i})$ as follows:

$$(p_{k_i}^{f,i})' = \frac{(p_{k_i}^{f,i})}{\mu(P^{f,i})}$$

In the case of tempo, each version of every song now has a relative tempo feature, which expresses how its tempo relates to all other performances of the song. The blue curve in Figure 4 is a bootstrapped LOWESS calculated on relative tempo values. We can observe a much narrower confidence interval, now enabling us to confidently claim that in 1984 the tempo was higher than any time in the 70s or 90s. We can also observe that in the beginning of the timeline the curves digress dramatically and with a much larger bootstrapping interval, which is due to the dataset containing few versions of few songs during that time, as observed in Section 2. All subsequent plots are of relative features and calculated as just described.

4. RESULTS

There has been very little musicological work on the Grateful Dead [19, 20], perhaps due to the intimidating size of

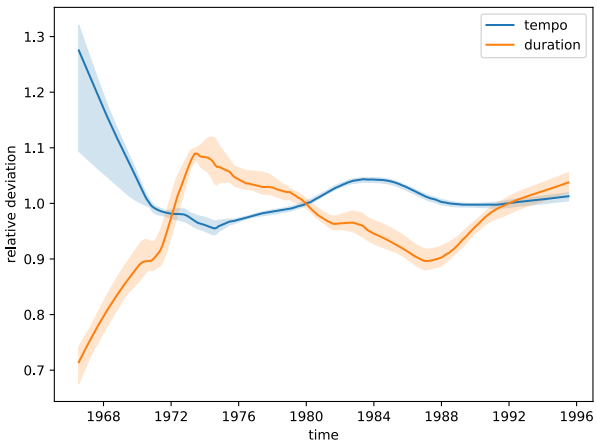


Figure 5: Bootstrapped LOWESS curves for tempo and song duration across all songs.

the band's catalog as well as the intentionally fleeting nature of the performances, as hypothesized by Tift [15]. However, when it comes to in-depth knowledge of the band's music, the best source are undoubtedly the band's fans themselves, the Deadheads. Many of them have spent decades listening to the music and debating it online, and some of them are so knowledgeable that they have an acute sense of how the music changes every single year throughout the band's history. In order to find such general yet detailed descriptions of the evolution of the band's music we searched the Grateful Dead subreddits, [12] the largest forums of their kind with a total of 174k members at the time of writing. We manually read through all pages with discussions on the differences between the band's phases and collected all general descriptions that include the musical characteristics we study, including tempo, timbre, song lengths, etc, with references to specific years. The pages were found using combinations of the search keywords *description, year, decade, 60s, 70s, 80s, 90s, progress, tempo, jam*. Some of the most detailed year by year descriptions found are by Wolfman92097, WesternEstatesHOA, MrCompletely, and an anonymous deleted user [21–24], and some people, such as maximinus-thrax have listened to, rated, and briefly described every of the band's more than 2300 shows [25, 26]. There are also many pages discussing the evolution of individual songs, such as [27]. We will now discuss our results while referring to relevant statements from the community where we can observe a consensus.

Figure 5 juxtaposes different temporal aspects of the music. We can identify an overall increase in tempo in the 1980s of more than 10%, and a subsequent decrease in the middle of the decade. In terms of duration, the differences are even more dramatic, a sharp increase in the early 70s, followed by a gradual decrease by more than 20% until the late 80s, when duration starts increasing again. Note that the early curve segments in all plots, until about 1969, can be considered biased for the reasons stated in Section 3.3. These observations coincide well with a notion in the community that, compared to the 70s, the early 80s are perceived as more energetic, faster paced, and containing fewer and less extensive jams. Wolfman92097 notes that in 1980 "the tempo [starts] to speed up a little", in 1984 "musically the tempo has really sped up", and in 1985 "the band slows the tempo a little." [22] It is striking that this increase and decrease directly corresponds to the tempo curve in Figure 5. Although a bit more abstractly, leedye similarly highlights that exact time period: "79-84 stands out to me as the disco/cocaine era ... post 84 just seems to be a little more of the slow churned vanilla as opposed to that sweet mint chocolate chip." [28] The same perception is true for individual songs. Discussing a recording of the song Eyes of the World, melwarren says "my 4 year old asked why EOTW was going so fast" and whenthattrain-rollsby replies "They played it too fast for my taste in the 80's." [29] On a different page the forum members observe how the song went back to a refreshingly new slow tempo in 1990 [27]. These observations can also be verified in Figure 3.

In terms of song duration, braney86 notes that the "late 70s was full of monster extended jams, and Jerry was absolutely on fire", while MrCompletely says that "82 through 85 is a long uneven slide down ... 87 is the comeback, shows are very different, most are tightly executed, very light on jams ... 88 starting to stretch back out a little." [23] Wolfman92097 also observes that in late 1986 "the setlists get much longer and way more experimental than they had been all year" and that in 1988 "jazz and extreme psychedelia gets added in." [22] Many deadheads' favorite years are 73/74, "the peak of their spacey, jazzy, psychedelic extended jams" according to devlinontheweb [30]. These observations again directly correspond to the plot in Figure 5. Song duration peaks in 1973, decreases throughout the late 70s and the early 80s with a turnaround point around 87/88, as perceived by MrCompletely.

Figure 6 shows plots for dynamics features. We can observe a long peak in overall loudness of the soundboard recordings, spanning the entire 1980s. With increasing loudness we see a decrease in dynamic range, which may hint at an increased use of compression in the live mix. However, when comparing with Figure 5, we can also see a striking correlation between loudness and tempo, as well as between dynamic complexity and duration. Dynamic complexity may be another indicator of the amount of improvisation in the recordings, similar to song duration. The latter two, however, seem to be somewhat independent nonetheless. We can particularly observe a bulge in dynamic complexity in the late 70s which does not occur in the duration curve, and in the 90s duration increases while dynamic complexity decreases. On the other hand increased loudness may be a direct consequence of higher tempo and energy. EvilLinux admits that "If I am alone in the car, I usually will choose the 80's. There is just more energy, more off the rails." [31] An anonymous deleted user also says that "the Dead did play some incredible

[12] https://www.reddit.com/r/gratefuldead/, https://www.reddit.com/r/grateful_dead/

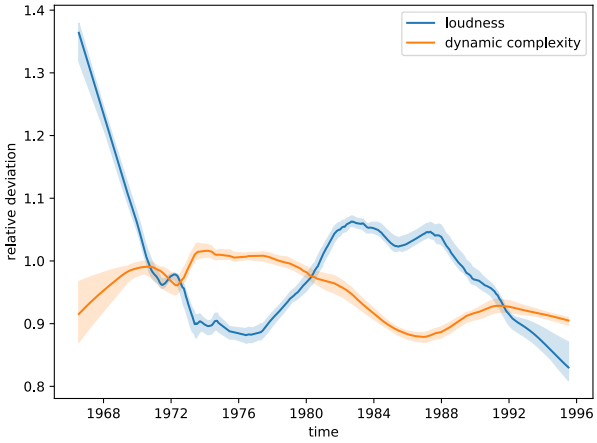


Figure 6: Bootstrapped LOWESS curves for dynamic essentia features (*loudness_ebu128.short_term.median* and *dynamic_complexity*) across all songs.

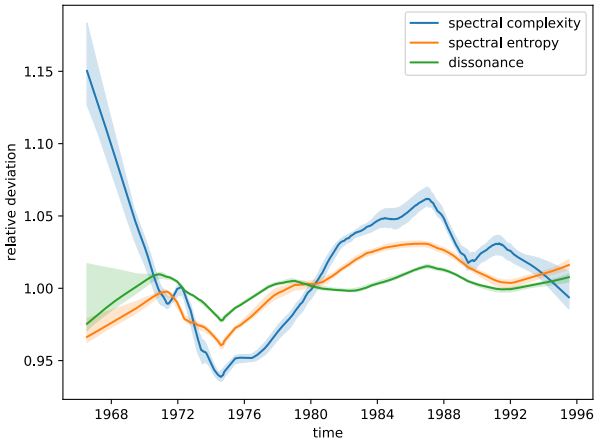


Figure 7: Bootstrapped LOWESS curves for spectral essentia features (*spectral_complexity.median*, *spectral_entropy.median* and *dissonance.median*).

shows in the early to mid-80s. They had a "fatter" sound, especially Jerry" [24].

A similar observation can be made for spectral features (Figure 7). Complexity and entropy increase for the entire duration of the 1980s, while dissonance has a shorter peak in the late 80s. Many listeners agree that the 80s have an entirely different sound, to a large part characterized by the keyboarder Brent Mydland who played with the band from 1979 to 1990 and who used a greater variety of keyboards and many synthesizers. According to BeaverMartin, Brent "really adds a whole different texture to the vocals and keys," [28] and WesternEstatesHOA says that "In 1983 the band truly takes off. The physical change of Brent's new keyboard is enough to change the band's sound alone. It has deep watery effects and adds so much depth to some of the more simple tunes." [22] Wolfman92097: "83 Garcia [guitar] is a little more distorted and Brent is using more fake keyboard sounds Phil [bass] is loud." [21]

As for tonal content, Figure 8 shows a selection of features calculated as described in Section 3.1. Tonal, pitch,

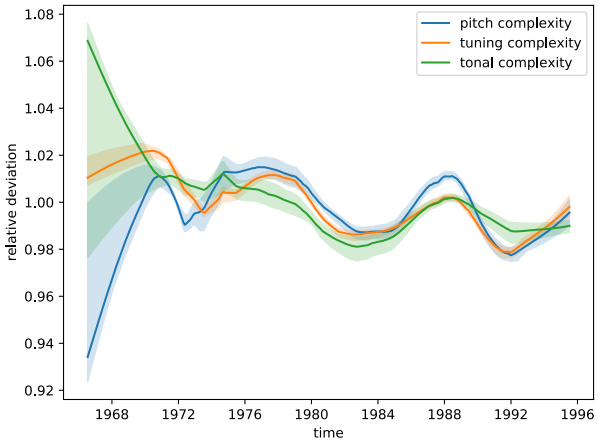


Figure 8: Bootstrapped LOWESS curves for pitch, tuning and tonal complexity features.

and tuning complexities run roughly in parallel and the period of 1980-84 is again demarcated, with a clear dip in all three features. This may again be due to the more streamlined faster performances and the lower degree of improvisation during this period. The late 1980s show parallels with the late 1970s, which may be related to the band starting to improvise more again. However, pitch and tuning complexities also seem highly correlated with dissonance (Figure 7) which may be related to timbre and the typical rich distorted and synthesizer-heavy 1980s Grateful Dead sound.

5. CONCLUSION

We have shown how evolutionary methods can not only be used for the study of general cultural trends in music, but also to investigate how the performances of a band change over time. We have also seen how we can improve the prediction accuracy of musical trends by considering the feature values relative to subsets of the data. It may be possible to apply a similar method in other situations, such as when studying the simultaneous evolution of the music of different composers, considering the music of each of them relative to their own work. We have also discovered limitations with the dataset used and suggest, in future work, to design a larger more systematic one in order to confirm our preliminary discoveries in this paper. Finally, while we have shown the potential reliability of the accounts of experienced listeners in the community, a more systematic collection and processing of online forum data may lead to more detailed results in the future.

6. ACKNOWLEDGMENTS

This work is supported in part by JST PRESTO No. JPMJPR20CB and JSPS KAKENHI Nos. 19H04137, 20K21813, 21K02846, 21K12187, 22H03661.

7. REFERENCES

[1] Y. Liu, Q. Xiang, Y. Wang, and L. Cai, "Cultural style based music classification of audio signals," in *2009*

IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 2009, pp. 57–60.

[2] D. Moelants, O. Cornelis, and M. Leman, "Exploring african tone scales," in *10th International Society for Music Information Retrieval Conference (ISMIR-2009)*. International Society for music Information Retrieval, 2009, pp. 489–494.

[3] M. Panteli, E. Benetos, and S. Dixon, "A computational study on outliers in world music," *Plos one*, vol. 12, no. 12, p. e0189399, 2017.

[4] C. Weiß, S. Balke, J. Abeßer, and M. Müller, "Computational corpus analysis: A case study on jazz solos." in *ISMIR*, 2018, pp. 416–423.

[5] F. Thalmann, K. Yoshii, T. Wilmering, G. A. Wiggins, and M. B. Sandler, "A method for analysis of shared structure in large music collections using techniques from genetic sequencing and graph theory," in *Proceedings of the 21st International Society for Music Information Retrieval Conference*, 2020.

[6] J. Serrà, Á. Corral, M. Boguñá, M. Haro, and J. L. Arcos, "Measuring the evolution of contemporary western popular music," *Scientific reports*, vol. 2, no. 1, pp. 1–6, 2012.

[7] M. Mauch, R. M. MacCallum, M. Levy, and A. M. Leroi, "The evolution of popular music: Usa 1960–2010," *Royal Society open science*, vol. 2, no. 5, p. 150081, 2015.

[8] E. Deruty and F. Pachet, "The mir perspective on the evolution of dynamics in mainstream music," in *Proceedings of the 16th ISMIR Conference*, vol. 723, 2015.

[9] C. Weiß, M. Mauch, S. Dixon, and M. Müller, "Investigating style evolution of western classical music: A computational approach," *Musicae Scientiae*, vol. 23, no. 4, pp. 486–507, 2019.

[10] T. Parmer and Y.-Y. Ahn, "Evolution of the informational complexity of contemporary western music," *arXiv preprint arXiv:1907.04292*, 2019.

[11] K. Choi and J. Stephen Downie, "A trend analysis on concreteness of popular song lyrics," in *6th International Conference on Digital Libraries for Musicology*, 2019, pp. 43–52.

[12] E. Nakamura and K. Kaneko, "Statistical evolutionary laws in music styles," *Scientific reports*, vol. 9, no. 1, pp. 1–11, 2019.

[13] F. C. Moss, M. Neuwirth, and M. Rohrmeier, "The line of fifths and the co-evolution of tonal pitch-classes," *Journal of Mathematics and Music*, pp. 1–25, 2022.

[14] M. Benson, *Why the Grateful Dead Matter*. University Press of New England, 2016.

[15] M. C. Tift, "Grateful dead musicking," in *All Graceful Instruments: The Contexts of the Grateful Dead Phenomenon*. Cambridge Scholars Publishing, 2021.

[16] C. Weiß and M. Müller, "Quantifying and visualizing tonal complexity," in *Proceedings of the Conference on Interdisciplinary Musicology (CIM)*, 2014, pp. 184–187.

[17] K. R. Page, S. Bechhofer, G. Fazekas, D. M. Weigl, and T. Wilmering, "Realising a layered digital library: exploration and analysis of the live music archive through linked data," in *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. IEEE, 2017, pp. 1–10.

[18] W. S. Cleveland and S. J. Devlin, "Locally weighted regression: an approach to regression analysis by local fitting," *Journal of the American statistical association*, vol. 83, no. 403, pp. 596–610, 1988.

[19] D. Malvinni, *Grateful Dead and the Art of Rock Improvisation*. Scarecrow Press, 2013.

[20] O. Longcroft-Wheaton, "The stylistic development of the grateful dead: 1965-1973." Ph.D. dissertation, University of Surrey, 2020.

[21] "Can you tell when a song was performed just by hearing it?" https://www.reddit.com/r/gratefuldead/comments/80ifqt/can_you_tell_when_a_song_was_performed_just_by/, accessed 2022-08-30.

[22] "Describing each year of the 80's," https://www.reddit.com/r/gratefuldead/comments/7x5zs9/describing_each_year_of_the_80s/, accessed 2022-08-30.

[23] "1980's Era Dead," https://www.reddit.com/r/gratefuldead/comments/1vf47n/1980s_era_dead/, accessed 2022-08-30.

[24] "Intro to listening to the Dead!" https://www.reddit.com/r/gratefuldead/comments/3crjek/intro_to_listening_to_the_dead/, accessed 2022-08-30.

[25] "Listened to all the shows. Listened to the best again. Here's the top 50 or so," https://www.reddit.com/r/gratefuldead/comments/sny6dx/listened_to_all_the_shows_listened_to_the_best/, accessed 2022-08-30.

[26] "Grateful Dead Reviews," https://raw.githubusercontent.com/maximinus/grateful-dead-reviews/master/dead_reviews.txt, accessed 2022-08-30.

[27] "When did Eyes get slowed down?" https://www.reddit.com/r/gratefuldead/comments/vax00p/when_did_eyes_get_slowed_down/, accessed 2022-08-30.

[28] "Fans of the 80s-90s - what do you like so much about this era?" https://www.reddit.com/r/gratefuldead/comments/sdufh5/fans_of_the_80s90s_what_do_you_like_so_much_about/, accessed 2022-08-30.

[29] "What are the fastest versions of Dead songs?" https://www.reddit.com/r/grateful_dead/comments/57btul/what_are_the_fastest_versions_of_dead_songs/, accessed 2022-08-30.

[30] "Best decade and year?" https://www.reddit.com/r/gratefuldead/comments/pdey6g/best_decade_and_year/, accessed 2022-08-30.

[31] "Can we just cut the shit and agree that the best Dead is from the 70s? Who's with me?" https://www.reddit.com/r/gratefuldead/comments/rblwiy/can_we_just_cut_the_shit_and_agree_that_the_best/, accessed 2022-08-30.

EVALUATING GENERATIVE AUDIO SYSTEMS AND THEIR METRICS

Ashvala Vinay
Center for Music Technology
Georgia Institute of Technology

Alexander Lerch
Center for Music Technology
Georgia Institute of Technology

ABSTRACT

Recent years have seen considerable advances in audio synthesis with deep generative models. However, the state-of-the-art is very difficult to quantify; different studies often use different evaluation methodologies and different metrics when reporting results, making a direct comparison to other systems difficult if not impossible. Furthermore, the perceptual relevance and meaning of the reported metrics in most cases unknown, prohibiting any conclusive insights with respect to practical usability and audio quality. This paper presents a study that investigates state-of-the-art approaches side-by-side with (i) a set of previously proposed objective metrics for audio reconstruction, and with (ii) a listening study. The results indicate that currently used objective metrics are insufficient to describe the perceptual quality of current systems.

1. INTRODUCTION

There has been growing research interest in building deep learning models that are capable of generating audio. Models such as WaveNet [1], NSynth [2], WaveGAN [3] and, most recently, DDSP [4] paved the way for data-driven Neural Audio Synthesis (NAS). Models like DDSP and NSynth have been advertised in YouTube videos prominently [5, 6], where artists make music using neural network generated audio, indicating that there is real world applicability to generative audio models.

Despite the many advances in generative modeling of sounds in the past couple of years, the evaluation of these systems lacks established methodology. Most importantly, systems are not evaluated with a consistent set of metrics. For instance, researchers have suggested the following to evaluate the output of such generative systems: the classification accuracies of neural networks trained to classify the sounds into predefined categories such as pitch and sound qualities [2,3,7,8], statistical methods [9,10], and subjective evaluation through listening studies [3, 9, 10].

This inconsistency in metrics and methodology makes a direct comparison of systems difficult at best, making it hard to understand the current state of the art and to measure the impact of new innovations in the field.

© Ashvala Vinay, Alexander Lerch. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Ashvala Vinay, Alexander Lerch, "Evaluating Generative Audio Systems and Their Metrics", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

In this study, we measure and compare the quality of the output of neural networks capable of producing short time-invariant samples. The goal is to evaluate state-of-the art systems comparatively with previously used evaluation metrics and to investigate the perceptual relevance of these metrics for measuring audio quality.

To pursue these goals, we trained three widely known neural networks, DDSP [4], NSynth [2], and Diffwave [9]. The evaluation results published with the introduction of these methods all imply that these models synthesize sounds at high quality. However, since different metrics are used for each system, no comparison is possible. To enable such a comparative analysis, we survey and implement a set of metrics and apply them to these systems. Furthermore, we conduct a listening study to measure the perceptual sound quality of the system outputs.

The core contributions of this paper are: (i) a review of currently used metrics for the evaluation of synthesis quality and a comparative analysis of 3 popular neural audio synthesizers, (ii) a listening study for assessing the perceptual audio quality of these synthesizers, and (iii) an investigation on the perceptual relevance of the objective metrics.

2. EVALUATION OF NAS SYSTEMS TODAY

Generative systems are notoriously difficult to evaluate [11] and new metrics are proposed frequently to understand whether generative networks are able to capture desirable characteristics required for a given task. In audio and music, the task of evaluation is difficult because (i) the ground truth is not well defined, as various outputs might be considered "correct," [12] (ii) the previously used set of objective metrics for NAS most likely misses or insufficiently models perceptual qualities of a sound [7, 13], (iii) and aesthetic preferences are, by definition, subjective. [14]

Some contemporary metrics for NAS derive from literature on evaluating Generative Adversarial Networks (GANs), where diversity of samples and modeling the distribution of data play a significant role [15]. Objective evaluation metrics typically rely on either mathematical formulations of a success measure, or —in the case of contemporary GAN literature— using a separate neural network to identify if the model is working appropriately. Accordingly, we categorize the metrics into the four groups (i) reconstruction metrics, (ii) sample diversity measures, (iii) distribution distance measures, (iv) and measures derived from subjective evaluation methods.

Reconstruction errors are computed as the difference between a given input sound S_i and a generated sound S_g.

The error is typically defined as either an ℓ_2 norm (squared error) or a ℓ_1 norm (absolute error). These differences can be computed on either the time-domain signal or a spectrogram. Typically, the MSE/MAE is computed on spectrograms, given their ubiquity as the input and generated representation for neural networks. MSE and MAE have a range between 0 and infinity, with a MSE/MAE of 0 indicating perfect reconstruction.

To give examples of practical use, Engel et al. use the multi-scale spectrogram loss, which compares spectrograms across a set of FFT sizes as a measure of reconstruction error and as a loss term for use in training [4]. This was recently used as a metric by Shan et al. to compare DDSP with their proposed Differential WaveTable Synthesis [16].

With respect to reconstruction metrics, there appears to be an ambiguity about the purpose of these metrics, as they seem to be used as both the training loss to be minimized and the evaluation metric itself. Also note that these errors are known to not have a clear perceptual meaning, as a high MSE between two sounds does not necessarily mean that such two sounds will be perceived as dissimilar (or vice versa). There is plenty of evidence in the field of (perceptual) audio coding verifying that measuring the power of the coding error is insufficient to capture the perceptual quality of the sound [17].

Sample diversity metrics: In GAN literature, a lot of emphasis is placed on the performance of the "generator" and to ensure that it is able to produce classifiable samples that capture the diversity of classes from the training dataset. The two metrics we will discuss in this section use machine learning and deep learning driven approaches to measure sample diversity.

- **Number of statistically Different Bins** (NDB/k) is a metric devised to identify mode collapse in GANs, a phenomena where a network produces a lot of outputs that look like alike, therefore lacking sample *diversity* [18]. NDB/k is computed on a Voronoi decomposition from the k-means centroids of the training samples. The clusters are computed directly in the sample space. [1] The k clusters are referred to as the "bins." To compute a score, test samples are assigned to the k clusters/bins using an $L2$ distance measure between the samples and the centroids of the clusters. A two-sample t-test on each bin identifies the statistically different bins. The final NDB score is given by counting the number of statistically different bins and dividing by the number of clusters.
  NDB/k scores are between the range of 0 and 1. According to the interpretation of the score provided by Richardson and Weiss [18], a score of 0 indicates that the network is producing a large diversity of samples that captures the training distribution well, whereas a score approaching 1 suggests that the generator has collapsed.
  This was used by Diffwave [9] and GANSynth [10] as part of their evaluation metrics.
  Richardson and Weiss [18] state in their definition of the metric that computing the distance for each pixel in an image using an $L2$ distance is perhaps not meaningful. As we have stated above, distances like $L2$ are not good at capturing perceptual qualities of sound, making this of particular concern when evaluating audio and the outputs of generative audio systems.
- **Inception scores (IS)** were proposed by Salimans et al. as a way to evaluate image generating GANs by using a classifier [15]. This classifier is used to automatically evaluate whether the output of a GAN was of reasonable quality and captured the *diversity* of samples in the dataset.
  The Inception classifier produces class label probabilities for a given input. Ideally, each input produces a high probability for one class label and our generative system is able to produce many of them. At the same time, the generator should be able to produce many such images, that can be classified uniquely into a large set of labels. If the difference between the probability distribution of predicted labels for the generated images and the marginal distribution of the labels from the generated data is small, it implies that the the generator is unable to produce a a diverse number of easily classifiable images. The mathematical formulation of IS can be found in [15]. The score itself has a lower bound of zero and an upper bound of infinity. The higher the score, the better.
  Within the context of audio, IS was used in evaluating WaveGAN [3], where an Inception network was trained on the SC09 dataset with spectrogram inputs. More recently, two inception scores have been proposed for NAS: the Pitch Inception Score (PIS) and the Instrument Inception Score (IIS) [8]. These measures tell us if generator has captured the true distribution of discrete MIDI pitch classes and the instrument classes in the NSynth dataset [2], and are thus focused on inherent sound properties that are not directly related to audio quality.
  Salimans et al. [15] stated that the Inception Score for an image generator was found to correlate well with human judgment of image quality [15]. However, no such work has been done in evaluating the perceptual meaningfulness of IS with audio.

Distribution distance metrics: Another important facet of evaluating GANs is identifying whether the distribution of data produced by the generator is close to the distribution of real data. Like the Inception score mentioned above, the metrics discussed here use neural networks. The difference here is that these metrics rely on using embeddings from a neural network.

As noted by Ananthabhotla et al. [13], matching distributions cannot guarantee a perceptually closer result.

- **Kernel Inception Distances (KID)** are scores that are generated by computing the distance between embeddings of input and generated data fed to inception networks. The distance is computed using Maximum Mean Discrepancy (MMD), a statistical

[1] in the case of images, the pixel distance

test that describes the difference between two distributions of data. They were first introduced in a paper by Binkowski et al. [19] where they used MMD to train a critic or discriminator and KID was shown as a metric to evaluate the convergence of the GAN. In order to compute KID, we compute a distribution of embeddings extracted from the Inception classifier for both the reference and generated output and compute the MMD between the distributions of reference and generated embeddings. The score is defined with a lower-bound of zero and an upper bound of infinity. This was used by Nistal et al. to compare input feature representations using GANs [7] and in a separate paper by Nistal et al. to measure the performance of an architecture they built called the VQCPC-GAN [8]. It was also used to evaluate DarkGAN [20].

- **Fréchet Audio Distance (FAD)** [21] is a metric originally developed to evaluate sound enhancement algorithms, but recently it has found use in evaluating NAS systems. The computation of the FAD relies on the VGGish embeddings for both the reference and the generated sounds. The VGGish embeddings are then fitted to multi-variate gaussians. The FAD itself is the Fréchet distance between the two distributions $\mathcal{N}_r$ and $\mathcal{N}_g$ representing the reference and generated gaussian distributions. The mathematical definition can be found in [21].
  This metric was used for evaluating Diffwave [9], Neural Waveshaping Synthesis [22], DarkGAN [20], and CRASH [23].

2.1 Subjective evaluation

Since the quality of the generated outputs is ultimately a perceptual property, listening studies have been previously used to evaluate a neural network's generated audio quality. A lot of listening studies use a popular method called "Mean Opinion Score" (MOS) [24]. Users participating in the listening study are asked to rate the sound they hear on a Likert scale between 1 and 5 across a set of questions. For example, WaveGAN asked participants to rate sounds on their "sound quality, ease of intelligibility, and speaker diversity" [3] where 1 indicates bad and 5 indicates excellent. These questions are asked for a collection of methods that the researcher seeks to evaluate.

The MOS survey strategy has been used for the evaluation of WaveGAN [3], Diffwave [9], and in neural speech generation literature [25, 26]. Kong et al. compared their results to WaveGAN using MOS and found that their network scored higher [9].

It should be noted that surveys that use MOS do not explicitly ask participants to compare the outputs against a reference and instead present participants with a single sound for every question. This means that MOS surveys give you an absolute rating for every sound, not a relative preference. The absence of reference precludes the ability to rank the methods that are being evaluated. For example, a person might like sound A and sound B in isolation and provide high ratings to both, however, it remains unclear whether the person has a relative preference for sound A or B.

A well known alternative to MOS surveys is to use a survey method called MUltiple Stimuli, Hidden Reference and Anchor (MUSHRA) [27] . It is an ITU recommendation for studies designed to evaluate differences in audio quality between audio codecs [28]. While it hasn't seen significant usage in evaluating NAS, it was recently used by Hayes et al. [22] in evaluating the performance of their Neural Waveshaping Synthesis (NeWT) model against the performance of DDSP. Their listening study results showed that their neural network outperformed DDSP on most instruments, with the exception of violins. This was shown to correlate well with the computed FAD scores for DDSP and NeWT.

3. EXPERIMENTAL SETUP

Aiming at our goal of comparing the output quality of popular generative systems and assessing the metrics commonly used for evaluation, we chose three neural networks and re-trained DiffWave and DDSP with the NSynth dataset and used the NSynth network directly. We report both objective metrics and subjective ratings of the outputs of these system.

We break our study into the three phases: (i) comparative analysis using objective metrics, (ii) comparative analysis using listening study results, and (iii) a brief investigation into the perceptual relevance of the objective metrics.

3.1 Dataset

The NSynth dataset is a publicly available dataset with approximately 300k sounds of 4 s length spanning acoustic and electronic timbres [2]. The sounds are all sampled at 16 kHz. It is comprised of 11 instrument families. There are ten unique "quality" descriptors that are attached to every sound in the dataset. The descriptors are timbral, for e.g "dark," "bright," "reverb," and "percussive." The NSynth dataset is frequently used in generative audio research and is therefore an obvious choice for this study.

The NSynth dataset's test set was used to generate all the samples that were used in both the listening study and to compute objective metrics.

3.2 Models

Model selection was based on the criteria of age, architecture, and public availability. There are a number of generative audio systems that have been published since 2017, but not all of them were available publicly or were difficult to train due to the lack of computational resources. Models selected were NSynth [2], DDSP [4], and DiffWave [9]. These models are widely known and represent different architectural designs.

NSynth [2] is a deep learning based generative audio system that uses a Variational Autoencoder architecture that learns to generate musical instrument timbres with the ability to be controllable. Its primary novelty is the fact that it can interpolate between multiple sounds and generate new timbres in the process. NSynth is the "oldest" of the

System	NDB/k (↓)	PKID(↓)	IKID(↓)	PIS(↑)	IIS(↑)	MSE(↓)	MAE (↓)	FAD (↓)
Diffwave	0.74	0.0093	0.0021	2.3814	**5.6477**	0.0291	0.1369	7.9488
DDSP	**0.20**	**0.0053**	0.0020	**3.3224**	5.3371	**0.0130**	**0.0666**	**1.1519**
NSynth	0.74	0.0101	0.0024	2.3238	4.6364	0.0329	0.1224	4.0590
Anchor	0.72	0.0123	**0.0006**	2.9356	5.3017	0.0257	0.0857	1.4952

Table 1. Table with objective results for each of the neural networks that we measured. ↓ indicates that a lower score is better and ↑ indicates that a higher score is considered better. Bold indicates best performance.

evaluated systems. The publicly available weights for the NSynth model were used for this study to generate the sounds.

DDSP [4] uses classical synthesis techniques like additive synthesis and noise filtering in the context of deep learning by treating them as differentiable blocks. It uses an encoder-decoder architecture that produces the fundamental frequency, loudness envelope and the necessary variables to control the additive synthesizer. DDSP is also known for it ability to perform "timbre-transfer," where it takes sounds produced from one instrument and outputs it on a different instrument. To train DDSP, we used publicly available code for training with the NSynth dataset and verified it worked by producing metrics similar to the metrics reported in the original paper.

Diffwave [9] is the most recent system that uses a new generative modeling technique called "Diffusion" on audio. Diffusion models start with white noise and through a fixed number of iterations, learn to generate audio. During training, the models use a forward and backward process, where the forward process takes the reference, corrupts it with white noise iteratively until the whole signal is noise. The backwards process learns how to iteratively remove the white noise to recover the reference. Diffwave was primarily evaluated on speech and produced results that seemed to outperform WaveGAN and Wavenet significantly.

DiffWave was trained with an implementation available online [2] . Since this algorithm was not originally trained with NSynth, we had to verify if our model worked properly. We trained the network for 2 million steps and evaluated the network's automatic metrics and found that the metrics aligned with the paper's reported results.

3.3 Objective metrics

To evaluate our neural networks, we used the set of metrics described above. As we discussed in Section 2, some of the objective metrics were designed with specific architectures like GANs in mind. However, since the metrics have found use in the evaluation of other types of architectures we chose to evaluate all of our networks with the same set of metrics.

The pre-trained pitch and instrument inception networks trained on the NSynth dataset from Nistal et al.'s paper on comparing audio representations [7] were used to compute neural network driven metrics such as as PKID and IIS and the "official" implementations of NDB/k [3] and FAD [4] were used. In order to compute NDB/k, we trained the K-means clustering on the NSynth dataset.

The rest of the metrics, like MSE and MAE were computed using SciKit-learn's built-in metrics [29]. We tried to include other metrics but could not select them due to high variance in scoring between implementations (such as PEAQ [17]).

To summarize, the following objective metrics are computed for the network outputs:

1. NDB/k, 2. PKID, 3. IKID, 4. PIS, 5. IIS, 6. MSE, 7. MAE, and 8. FAD.

3.4 Listening study

The listening study uses a variation of the aforementioned MUSHRA methodology. It is popularly used in evaluating "intermediate" differences in low bitrate audio codecs [28]. Given that we are measuring audio reconstruction, we believe that treating the neural network outputs similar to encoded audio is a suitable choice for evaluating audio quality differences. The rating scale used in the study is divided according to the MUSHRA specification. A score between 0 and 20 indicates that the sound is rated *bad*, 20 to 40 indicates that the sound is rated *poor*, 40 to 60 is considered *fair*, 60 to 80 is considered *good*, and 80 to 100 is considered *excellent*.

The 4096 sounds from the NSynth test set are reconstructed using the three generative systems. In addition, one anchor sound is generated for each test sample by low pass filtering the sound at a 1 kHz cutoff frequency and reducing its bit depth to 8 Bits.

Participants were recruited by emails sent to two large academic audio-focused communities. Participants were asked to use a good pair of headphones or speakers in order to participate in the survey. Prior to the listening study, the participants were asked to share their age bracket, experience with audio synthesis, and how much money they have spent on the audio equipment they used. This then led to a training phase where participants were presented with an example sound and necessary introduction to the survey.

When presenting the survey, a sound is randomly selected out of 3237 possible sounds with MIDI note numbers ranging from 22 to 84. [5] Five sliders are presented in random order, with three sliders referring to the generated audio output of the three neural networks to evaluate, and the other two sliders corresponding to the hidden reference and the anchor. The participants were asked to rate audio generated by each neural network on its audio quality

[2] https://github.com/lmntcom/diffwave

[3] https://github.com/eitanrich/gansngmms

[4] https://github.com/googleresearch/googleresearch/

[5] We used the note number range to remove sounds that were either inaudible or could potentially be uncomfortable to listen to.

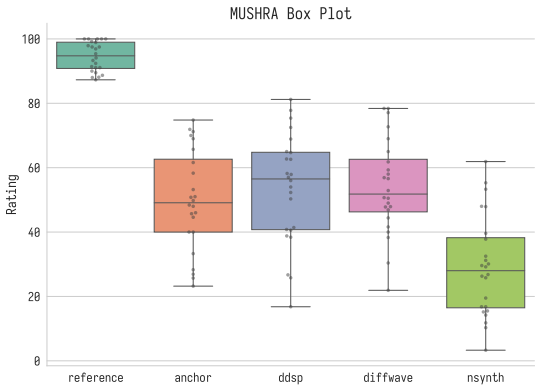


Figure 1. A boxplot of ratings from participants in the listening study. The size of the box represents the interquartile range of ratings from the listeners.

compared to the reference. This presentation is repeated ten times without collision (i.e., no two sounds are ever repeated for a participant).

To clean up the data, responses that are "incomplete" are removed entirely from data that can be used for analysis. MUSHRA analysis generally removes raters who rate the reference below a high rating threshold [30]; thus, participants who rate the reference below an average of 85 will be removed.

Statistical analysis of MUSHRA and MUSHRA-like data relies on running statistical tests such as ANOVA or Wilcoxon tests [30] to measure differences in results between the presented conditions. We will use the Wilcoxon tests for the ratings to measure statistical differences between presented conditions. The test indicates differences between the generative models. The MUSHRA results will also be broken down by demographic information collected above to measure if different demographic groupings rated the models differently.

In order to compare objective and subjective ratings, we will also compute rankings for both ratings. We are interested in knowing how frequently specific networks were considered the best and comparing it to the rankings for each metric. Rank driven correlations were used by Ycart et al. [31] in their paper validating perceptual metrics related to piano transcription.

4. RESULTS

4.1 Objective metric results

The results from the objective metrics as discussed above are shown in Table 1. Overall, these results seem to indicate that DDSP is producing higher quality samples that are more easily classifiable compared to DiffWave and NSynth.

DDSP has a 54% better PKID score over Diffwave and a 62% higher score than NSynth. This trend continues across most metrics, with the exceptions of the IIS, where Diffwave achieves a 5% higher score than DDSP, and the IKID where there is only a 4% difference between DDSP and Diffwave. The kernel distances mentioned here do not measure audio quality.

The MSE and MAE results tell us that DDSP is the "closest" to the reference sounds, with an MSE of 0.0130, a 76% and 79% improvement over Diffwave and NSynth, respectively.

DDSP significantly outperforms Diffwave and NSynth on the FrÃ©chet Audio Distance. FAD is also a category where NSynth outperforms Diffwave. This metric indicates that —according to the VGGish representation— DDSP and NSynth are generating samples that are closer to the reference than Diffwave.

Based on these metrics, we can state that DDSP outperforms Diffwave and NSynth and produces a diverse set of easily classifiable samples and produces samples that are much closer to the reference sounds.

4.2 Listening study results

Data was collected from 77 participants from our listening study. After data cleanup, a total of 24 submissions were considered trustworthy. 74% of our participants were between the age of 24 and 50, 22% of our participants between 18 and 24 and 3.7% or 1 participant over the age of 50. 37% of our participants reported that they were very familiar with music production tools and 18.5% reporting that they were extremely familiar. 29% of the participants reported that they were moderately knowledgeable about sound synthesis techniques, with an even distribution of familiarity ranging from "Not very knowledgeable" to "Extremely knowledgeable." 33% of our raters reported that they had spent over $750 on their audio equipment and 25% having spent between $250 and $500.

The overall visualization of the responses can be seen in Fig. 1. Every dot in the chart is an averaged rating from a participant. As expected, the reference scored highly with an average rating of 92. The analysis of the listening study results shows that DDSP and DiffWave scored similarly while NSynth performed considerably worse than the other networks. Both DDSP and Diffwave have average ratings around 53 while sounds generated by NSynth received an average rating of approx. 29.

The inter-rater Krippendorff α score [32] is 0.66 [6], suggesting that the subjects were largely in agreement.

The Wilcoxon test for statistical significance was applied to the data [30]. We found no statistically significant difference between DDSP/Diffwave ($p = 0.629$) and a statistically significant difference between DDSP/NSynth ($p = 8 \times 10^{-6}$) and Diffwave/NSynth ($p = 8 \times 10^{-6}$). There was a significant difference between the Reference and all the systems and the anchor ($p \leq 8 \times 10^{-6}$). There is a significant difference between Anchor/NSynth ($p = 8 \times 10^{-6}$), but no statistically significant difference between Anchor/Diffwave or Anchor/DDSP, with p-values of 0.144 and 0.4385, respectively.

Breaking down the results by the subjects' self-reported familiarity with audio synthesis technologies, we first investigate the inter-rater variation within different "expert levels."

[6] Krippendorff α ranges from -1 to 1, where -1 indicates significant disagreement between raters and 1 suggests maximal agreement

Participants who said they were the least knowledgeable had an Krippendorff α of 0.7, moderately knowledgeable raters had a Krippendorff α of 0.608 while participants who were the most knowledgeable had a Krippendorff α of 0.9. Participants who reported to be either very or extremely knowledgeable tended to rate DDSP and Diffwave lower than the overall scores (DDSP: 47, Diffwave: 43). Less knowledgeable participants tended to rate Diffwave and DDSP higher than the overall scores (DDSP: 56, Diffwave: 59). There was a statistically significant difference in how these two groups rated Diffwave ($p = 0.006$) but no statistically significant difference in how they rated DDSP, NSynth, or the reference and anchor point.

We found no statistically significant differences in ratings between the age categories or in ratings based on money spent on audio equipment.

Instrument results: To investigate whether specific instruments or instrument groups are consistently rated higher or lower than others, the listener ratings were broken down into ratings by instrument family (according to the NSynth dataset). We found no statistically significant differences in ratings between DDSP and Diffwave, but found statistically significant differences between DDSP/NSynth and Diffwave/NSynth. Additionally, there were no statistically significant differences between instrument sources and the ratings from the listeners.

The IKID metric tell us that all three networks are producing samples that are close to the reference and the IIS score tells us that Diffwave should be slightly better than DDSP and much better than NSynth. While we cannot state that this result is accurate for IKID, it is in line with the results from the IIS computation.

Comparing the listener ratings to objective metrics: In order to identify if our listening study results lined up with our objective metrics, we took the selection of sounds that were presented to participants and computed the correlation between their sample based metrics and our ratings using the Pearson correlation coefficient, which tells us how correlated two samples are from a range of -1 to 1. We have sample based results for MSE and MAE, since all the other metrics in our list rely on computing a difference across a distribution of samples. We also computed the Spearman R score and the R^2 from a linear regression between the ratings and the MSE and MAE. We found that there is no statistically significant correlation between the MSE/MAE errors and the listener ratings.

The listening responses were also ranked based on how frequently a specific network was ranked higher than the others. In the 240 responses, the listeners rated Diffwave higher than DDSP slightly more frequently (123 vs. 99). NSynth was rated the lowest most frequently (163 times). The ranking breakdown showing how frequently each of the 6 permutation of rankings between the 3 networks is shown in 2. We ran a Wilcoxon test on pairings rankings of each network by the listeners and found that the rankings were different with high statistical significance ($p < 0.05$).

When ranking the objective metrics, it is clear that DDSP is ranked the best because it has the best results in seven out of the eight metrics, with Diffwave and NSynth in second and third place. However, the rankings from our listening study tell us that the Diffwave is usually the preferred network. This tells us that the metrics are perhaps not entirely reflective of the perceptual audio quality of the networks.

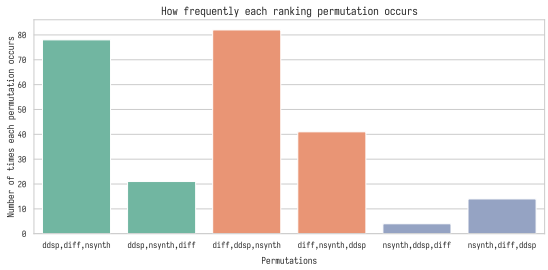


Figure 2. The plot here describes how frequently a unique permutation of rankings occurred. Diffwave is abbreivated to "diff".

5. CONCLUSION

We presented a systematic evaluation of three popular systems for neural audio synthesis, comparing them with a set of previously used objective metrics as well as a listening study. The results clearly show that (i) no previously used objective metric captures the perceptual quality of the synthesized sounds sufficiently well, (ii) any quality ranking based on these objective metrics is questionable, (iii) of the three evaluated audio generators, there is no clear listener preference between DDSP and Diffwave, but NSynth is rated lowly, and (iv) the subjects rate Nsynth worse than the 8 Bit anchor but rate DDSP and Diffwave similar to the anchor indicating that there is still considerable work to do on the quality of neural audio synthesis

These results should give pause to research in the field of neural audio synthesis. How can progress with respect to the audio quality be measured if all available metrics are unable to provide meaningful estimates of audio quality. Many of the objective metrics that we discussed in this paper were designed with the intent of measuring the performance of the sound generating component of the networks, i.e., can the generator produce (i) a diverse set of sounds, (ii) a distribution of samples that are close to the target samples in a relevant dimension, and (iii) accurate reconstructions.

Our results indicate that *measuring generator performance is insufficient to measure audio quality*. We believe that research in this space should not only include subjective results, it should also include greater efforts into critically evaluating the audio quality of network outputs with meaningful objective metrics.

In future work, we will investigate whether other objective metrics might be more meaningful than the ones evaluated here, or will start a research project on developing a more meaningful quality measure for audio synthesis.

6. REFERENCES

[1] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, "WaveNet: A generative model for raw audio," in *The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016*. ISCA, p. 125. [Online]. Available: http://www.isca-speech.org/archive/SSW_2016/abstracts/ssw9_DS-4_van_den_Oord.html

[2] J. Engel, C. Resnick, A. Roberts, S. Dieleman, M. Norouzi, D. Eck, and K. Simonyan, "Neural audio synthesis of musical notes with WaveNet autoencoders," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, pp. 1068–1077, ISSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v70/engel17a.html

[3] C. Donahue, J. J. McAuley, and M. S. Puckette, "Adversarial audio synthesis," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. [Online]. Available: https://openreview.net/forum?id=ByMVTsR5KQ

[4] J. H. Engel, L. Hantrakul, C. Gu, and A. Roberts, "DDSP: Differentiable digital signal processing," in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. [Online]. Available: https://openreview.net/forum?id=B1x1ma4tDr

[5] Adam Neely, "Turning BASS into violin (using AI)." [Online]. Available: https://www.youtube.com/watch?v=cIX4y22NWWc

[6] ANDREW HUANG, "Music with artificial intelligence." [Online]. Available: https://www.youtube.com/watch?v=AaALLWQmCdI

[7] J. Nistal, S. Lattner, and G. Richard, "Comparing representations for audio synthesis using generative adversarial networks," in *2020 28th European Signal Processing Conference (EUSIPCO)*, pp. 161–165, ISSN: 2076-1465.

[8] J. Nistal, C. Aouameur, S. Lattner, and G. Richard, "VQCPC-GAN: Variable-length adversarial audio synthesis using vector-quantized contrastive predictive coding," in *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, pp. 116–120, ISSN: 1947-1629.

[9] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro, "DiffWave: A versatile diffusion model for audio synthesis," in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. [Online]. Available: https://openreview.net/forum?id=a-xFK8Ymz5J

[10] J. H. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial neural audio synthesis," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. [Online]. Available: https://openreview.net/forum?id=H1xQVn09FX

[11] L. Theis, A. v. d. Oord, and M. Bethge, "A note on the evaluation of generative models," number: arXiv:1511.01844. [Online]. Available: http://arxiv.org/abs/1511.01844

[12] M. Caetano and N. Osaka, "A formal evaluation framework for sound morphing," in *ICMC*.

[13] I. Ananthabhotla, S. Ewert, and J. A. Paradiso, "Towards a perceptual loss: Using a neural network codec approximation as a loss for generative audio models," in *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, pp. 1518–1525. [Online]. Available: https://dl.acm.org/doi/10.1145/3343031.3351148

[14] H. Leder, B. Belke, A. Oeberst, and D. Augustin, "A model of aesthetic appreciation and aesthetic judgments," vol. 95, no. 4, pp. 489–508. [Online]. Available: https://bpspsychub.onlinelibrary.wiley.com/doi/abs/10.1348/0007126042369811

[15] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved techniques for training GANs," p. 9, QID: Q46993880.

[16] S. Shan, L. Hantrakul, J. Chen, M. Avent, and D. Trevelyan, "Differentiable wavetable synthesis," number: arXiv:2111.10003. [Online]. Available: http://arxiv.org/abs/2111.10003

[17] T. Thiede, "PEAQ–the ITU standardfor ObjectiveMeasuremenot f perceived audio quality," vol. 48, no. 1, p. 27.

[18] E. Richardson and Y. Weiss, "On GANs and GMMs," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/0172d289da48c48de8c5ebf3de9f7ee1-Paper.pdf

[19] M. Binkowski, D. J. Sutherland, M. Arbel, and A. Gretton, "Demystifying MMD GANs," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. [Online]. Available: https://openreview.net/forum?id=r1lUOzWCW

[20] J. Nistal, S. Lattner, and G. Richard, "DarkGAN: Exploiting knowledge distillation for comprehensible audio synthesis with GANs," in *Proceedings of the*

22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021, J. H. Lee, A. Lerch, Z. Duan, J. Nam, P. Rao, P. v. Kranenburg, and A. Srinivasamurthy, Eds., pp. 484–492. [Online]. Available: https://archives.ismir.net/ismir2021/paper/000060.pdf

[21] K. Kilgour, M. Zuluaga, D. Roblek, and M. Sharifi, "FrÃ©chet audio distance: A reference-free metric for evaluating music enhancement algorithms," in *Interspeech 2019*. ISCA, pp. 2350–2354. [Online]. Available: https://www.isca-speech.org/archive/interspeech_2019/kilgour19_interspeech.html

[22] B. Hayes, C. Saitis, and G. Fazekas, "Neural waveshaping synthesis," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, J. H. Lee, A. Lerch, Z. Duan, J. Nam, P. Rao, P. v. Kranenburg, and A. Srinivasamurthy, Eds., pp. 254–261. [Online]. Available: https://archives.ismir.net/ismir2021/paper/000031.pdf

[23] S. Rouard and G. Hadjeres, "CRASH: Raw audio score-based generative modeling for controllable high-resolution drum sound synthesis," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, J. H. Lee, A. Lerch, Z. Duan, J. Nam, P. Rao, P. v. Kranenburg, and A. Srinivasamurthy, Eds., pp. 579–585. [Online]. Available: https://archives.ismir.net/ismir2021/paper/000072.pdf

[24] F. Ribeiro, D. Florencio, C. Zhang, and M. Seltzer, "CROWDMOS: An approach for crowdsourcing mean opinion score studies," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, pp. 2416–2419. [Online]. Available: http://ieeexplore.ieee.org/document/5946971/

[25] R. Prenger, R. Valle, and B. Catanzaro, "Waveglow: A flow-based generative network for speech synthesis," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3617–3621, ISSN: 2379-190X.

[26] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, Q. Le, Y. Agiomyrgiannakis, R. Clark, and R. A. Saurous, "Tacotron: Towards end-to-end speech synthesis," in *Interspeech 2017*. ISCA, pp. 4006–4010. [Online]. Available: https://www.isca-speech.org/archive/interspeech_2017/wang17n_interspeech.html

[27] International Telecommunications Union (last). BS.1534Â :Â method for the subjective assessment of intermediate quality level of audio systems. [Online]. Available: https://www.itu.int/rec/R-REC-BS.1534/en

[28] S. Zielinski, P. Hardisty, C. Hummersone, and F. Rumsey, "Potential biases in MUSHRA listening tests." Audio Engineering Society. [Online]. Available: https://www.aes.org/e-lib/browse.cfm?elib=14237

[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and others, "Scikit-learn: Machine learning in python," vol. 12, pp. 2825–2830, QID: Q28365500.

[30] C. MendonÃ§a and S. Delikaris-Manias, "Statistical tests with MUSHRA data," in *Audio Engineering Society Convention 144*. [Online]. Available: http://www.aes.org/e-lib/browse.cfm?elib=19402

[31] A. Ycart, L. Liu, E. Benetos, and M. T. Pearce, "Investigating the perceptual validity of evaluation metrics for automatic piano music transcription," vol. 3, no. 1, pp. 68–81. [Online]. Available: http://transactions.ismir.net/articles/10.5334/tismir.57/

[32] K. Krippendorff, "Computing krippendorff's alpha-reliability." [Online]. Available: https://repository.upenn.edu/asc_papers/43

REPRESENTATION LEARNING FOR THE AUTOMATIC INDEXING OF SOUND EFFECTS LIBRARIES

Alison B. Ma
Music Informatics Group
Georgia Institute of Technology
ama67@gatech.edu

Alexander Lerch
Music Informatics Group
Georgia Institute of Technology
alexander.lerch@gatech.edu

ABSTRACT

Labeling and maintaining a commercial sound effects library is a time-consuming task exacerbated by databases that continually grow in size and undergo taxonomy updates. Moreover, sound search and taxonomy creation are complicated by non-uniform metadata, an unrelenting problem even with the introduction of a new industry standard, the Universal Category System. To address these problems and overcome dataset-dependent limitations that inhibit the successful training of deep learning models, we pursue representation learning to train generalized embeddings that can be used for a wide variety of sound effects libraries and are a taxonomy-agnostic representation of sound. We show that a task-specific but dataset-independent representation can successfully address data issues such as class imbalance, inconsistent class labels, and insufficient dataset size, outperforming established representations such as OpenL3. Detailed experimental results show the impact of metric learning approaches and different cross-dataset training methods on representational effectiveness.

1. INTRODUCTION

Sound effects libraries are collections of prerecorded audio assets curated and meant for use by sound designers and editors. In application, a user will search for sounds necessary for their project by querying the library with a computer or other commercial tools for sound search, which helps organize sounds or allow for varying modes of querying the database [1–4]. Examples of typical classes include *ambience*, *foley*, and *sci-fi*. Libraries are often used in game audio and audio post-production applications because access to raw audio assets facilitates efficiency in the creative process, accommodates lacking resources for recording equipment, and alleviates barriers to inconvenient recording locations necessary for particular sounds.

We identify two main problems with sound effects libraries in our work. First, interviews with dataset providers for this research revealed that too much time is spent labeling, re-labeling, and performing quality assurance on databases that continually grow in size and undergo taxonomy updates. One potential solution to this could be the application of machine learning and, more specifically, deep learning for automating sound classification. However, such machine learning models face several particular task-related challenges, such as (i) the size of available datasets can dramatically vary in terms of audio data and number of classes, (ii) data class distributions are often highly imbalanced, (iii) data labels are sometimes of poor quality or inconsistent, and (iv) training dedicated models on every existing sound effects library can be time-consuming.

Second, non-uniform metadata in sound effects libraries complicate the successful sound search for a user and the creation of a useful taxonomy for a vendor. The recent introduction of the Universal Category System (UCS) [1] attempts to address this problem. UCS is an industry-proposed solution to standardize taxonomies designed by and for sound designers and editors. It is designed to be complete and sufficient for sound effects library categorization. However, adopting this standard has been slow; while several sound effects libraries have already converted to this new industry standard, others continue using their own proprietary taxonomies for convenience or preference.

We address problems that arise when using a machine learning approach, training generalized embeddings that can represent any sound effects library and work on any taxonomy for sound classification. To build a powerful learned representation, we investigate two main methods for representation learning with a (i) metric-learning and (ii) cross-dataset training approach.

Our work has the following main contributions:

- the introduction of a powerful new representation for use in sound effects libraries trained on relevant, commercially available data,
- the investigation of UCS' generalizability (the industry's first approach to unifying metadata), and
- the presentation of extensive results emphasizing the positive impact of cross-dataset training, an underresearched aspect of representation learning.

The remainder is structured as follows. In Sect. 6.2, we analyze if metric learning models will learn a better-structured embedding space and produce higher classification results than a Cross-Entropy (CE) model. In Sect. 6.3, we see if a cross-dataset training scenario will outperform

© A.B. Ma and A. Lerch. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** A.B. Ma and A. Lerch, "Representation Learning for the Automatic Indexing of Sound Effects Libraries", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[1] universalcategorysystem.com, last accessed May 10, 2022.

all other training scenarios and discuss novel UCS-specific findings. In Sect. 6.4, we evaluate the impact of 3 different cross-dataset training methods. See Sect. 5.3.1 for data mixing methods, which explores accommodating for encoder bias with data training order, Sect. 5.3.2, which investigates accommodating for various dataset characteristics by 'Focal' dataset regularization, and Sect. 5.3.3, which experiments with using dataset-independent BatchNorm layers. We compare our best representations against OpenL3 State-of-The-Art (SoTA) deep audio embeddings in Sect. 6.5.

2. RELATED WORK

We present an in-depth literature review regarding sound effects libraries and two aspects of representation learning.

2.1 Sound Effects Libraries

While there is a plethora of work on sound event classification, there is little work that conducts extensive research for a sound effects library application. Peeters and Reiss conducted sound effects classification by focusing on the discrimination between two classes, ambience and sound effects; audio features were manually selected [5]. Audio tagging work has been done on Freesound [6], the BBC sound effects library [7], and other online collaborative sound collections [8, 9]. Some have designed features and taxonomies to improve sound classification. Moffat et al. tried to address the issue of a non-unified metadata labeling scheme with the Adobe Sound Effects Library. [2] They created a hierarchical taxonomy for sound effects based on the unsupervised learning of sonic attributes, using decision trees to assess audio and label semantic similarity [10]. Others that do not target a specific sound effects library application have experimented with psychoacoustic approaches to feature design using actions, material, and mood [3, 11–14]. Meanwhile, the rest of the work regarding sound effects libraries does not focus on classification but explores sound search, query, and retrieval. Pearce et al. investigated user search queries on Freesound [15], reducing these descriptors into timbral features to aid sound search [16]. Lafay et al. and Zhang et al. researched methods to effectively query databases without keywords or by vocal imitation [2, 4]. Yang et al. recently presented a new system for indexing sound effects libraries, evaluating their work on users [1].

2.2 Representation Learning

Representation learning aims to learn a highly discriminative embedding space that can generalize to various downstream tasks. Two examples of powerful deep audio embeddings are VGGish and OpenL3 [17–19], which have proven to be useful for many music information retrieval (MIR) and cross-modal tasks, e.g., classification, tagging, few-shot learning, and continual learning [20–23].

2.2.1 Metric Learning

A popular method for representation learning is metric learning, where methods include the Contrastive, Triplet, and Circle loss. Contrastive loss minimizes the distance between similar classes' feature vectors and maximizes the similarity between different classes' feature vectors, training on pairs of positive or negative input samples [24–26]. Triplet loss optimizes both positive and negative inputs simultaneously and adds an additional anchor input in its loss computation [27–29]. Circle loss tries to improve upon the Triplet loss, featuring more flexible optimization, more definite convergence, and tries to unify the goals of a metric learning and classification loss into one [30]. Although metric learning losses can be used in a self-supervised [25, 31–34] or un-supervised [24] manner, it is common to combine a metric learning loss with a classification loss to better fit a supervised learning problem. Khosla et al. have explored training methods to regularize and better structure the embedding space, introducing a Two-Step and Joint-Training method [26]. Furthermore, other works have explored the adoption of a contrastive approach to regression [35].

2.2.2 Cross-Dataset Training

Furthermore, training on large amounts of diverse data is a crucial aspect of representation learning [17–19]. This can be addressed with cross-dataset training; we note many similarities between cross-dataset training and multi-task learning. Though there are scattered works amongst various fields in the deep learning literature that combine multiple different datasets for training, only a few papers have explored proposing methodologies for this. Namely, most work has only been explored in the computer vision domain; there is little work in audio. Working on symbolic music generation, Dong et al. found that stratified sampling mixing alleviated the source imbalance problems that come with combining datasets of various sizes; this worked better than simply concatenating datasets for training [36]. Ranftl et al. explored a multi-task learning setup with Pareto-optimal mixing and a multi-objective loss, evaluating success in a zero-shot scenario for a monocular depth estimation task. Their multi-task learning setup and mixing strategy, which ensures that decreasing the loss on one dataset necessitates increasing the loss on another, produced results superior to a naive mixing strategy where they trained on minibatches of equally sampled datasets. They found that this method was better at leveraging the act of adding more datasets for training [37]. Wan et al. introduced a MultiReader method for a speaker verification task to support training with datasets of different keywords and languages. They regularized the datasets to address insufficient and imbalanced dataset sizes but only experimented with two heterogeneous data sources. Moreover, they do not explicitly illustrate a method of reweighing datasets, resorting to hyperparameter tuning [38]. Lastly, Wang et al. introduced a Dataset-Aware Block, which uses dataset-invariant convolutional layers and dataset-specific BatchNorm layers. They concluded that preserving heterogenous dataset characteristics improves performance [39]. To the best of our knowledge, cross-dataset training has not been explored extensively in audio-related tasks.

[2] goo.gl/TzQgsB, last accessed May 10, 2022.

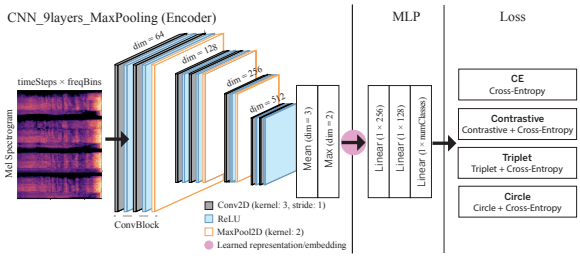


Figure 1. Network architecture: CNN9-max encoder, MLP classifier head, and loss functions.

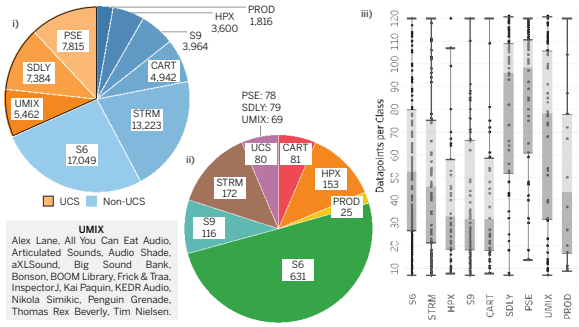


Figure 2. Imbalanced UCS & Non-UCS dataset statistics, (i) Number of audio-files per training dataset, (ii) Number of classes per taxonomy, (iii) Class distribution sorted from most to least classes among training datasets.

3. METHOD

Our best pre-trained model is freely available online. [3]

3.1 Model Architecture

Our encoder architecture adapts the Convolutional Neural Network CNN9-max architecture from a DCASE 2019 baseline system for multi-class classification and is displayed in Fig. 1 [40]. We pre-train the encoder with one Multi-Layer Perceptron (MLP) classifier head per dataset. After training, we freeze the encoder and pass the representations to a simple Nearest Neighbor classifier to train and test the learned embedding space. Models were trained with Python, PyTorch, and the PyTorch metric learning library. [4]

3.2 Input Representation

All audio files were re-sampled to 44.1 kHz, down-mixed, and normalized. Mel-spectrograms were computed with a block size of 46 ms, hop size of 23 ms, 96 Mel-Bins, and span the audible frequency range [41]. We used min-max normalization for spectrograms and z-score standardization for extracted embeddings. Spectrogram input dimensions to the network are (100, 96) and span approximately 2 s of audio. This length was determined in pilot experiments.

3.3 Training Procedure

We trained all models with a batch size of 64, the Adam optimizer, and early stopping. Model checkpoints used for inference correspond to the best validation loss. For cross-dataset training, we save the best average validation loss from all datasets. Hyper-parameters for pre-training were found via 20 trials of random search. During pre-training, 2 s frames of audio files are randomly sampled whenever a batch is retrieved. We re-shuffle the dataset per epoch. We compute embeddings with 50% overlap at inference time.

3.4 Evaluation Metrics

The macro F-1 score over classes in a dataset is our predominant metric for evaluating classification performance, as all datasets have imbalanced class distributions. We also monitor the Davies-Bouldin Index (DBI) to evaluate the quality of the embedding space structure, i.e., clustering. The DBI measures the average similarity between a cluster and its most similar cluster.

[3] github.com/alisonbma/aiSFX, last accessed August 11, 2022.

[4] kevinmusgrave.github.io/pytorch-metric-learning, last accessed May 10, 2022.

4. DATASETS & TAXONOMIES

We address our goal of training generalized embeddings by training and testing with 9 datasets and 7 taxonomies. We emphasize that we use a diverse assortment of datasets for this research and present corresponding data statistics in Fig. 2. We use 3 UCS-compliant datasets: Pro Sound Effects (*PSE*), Soundly (*SDLY*), UCS Mixed (*UMIX*), and 6 Non-UCS datasets from the Sound Ideas library: Cartoon Express (*CART*), HPX Digital (*HPX*), Production Elements (*PROD*), Series 6000 (*S6*), Series 9000 (*S9*), Soundstorm (*STRM*). For UCS-compliant datasets, we use UCSv8.1 *Category* labels as ground truth.

Preliminary experiments with UCS lead us to use experimental subsets of 150 or fewer datapoints per class. We pre-train and evaluate embeddings on the full imbalanced subsets (65k audio files) except in *Cross-Dataset* training scenarios, where we use class-balanced versions for Non-UCS data. A stratified train validation test split of 8:1:1 was conducted. For UCS-compliant datasets, the fine *CatID* labels determined stratification.

We select *PSE* as the base dataset for *UCS-Transfer* experiments because of its larger number of datapoints and classes, cleaner labels compared to Non-UCS datasets, better generalization when evaluated against other UCS datasets, and more even distribution among both levels of the UCS 2-level class hierarchy.

5. EXPERIMENTAL SETUP

We list our research questions and hypotheses below using the following experimental variables:

- *Metric learning loss functions (Fig. 1, Sect. 5.1)*,
- *Variable training scenarios (Sect. 5.2)*, and
- *Cross-dataset training methods* (Sect. 5.3).

5.1 RQ1: Impact of Metric Learning

We first ask whether metric learning loss functions that learn distances instead of absolute class labels will improve

classification results. We hypothesize that metric learning approaches will learn a more discriminative embedding space compared to CE models and that this better-structured embedding space will improve classification results.

5.1.1 Training Parameterization

We train all metric learning models with Khosla's Joint-Training scheme that optimizes a hybrid sum of a metric learning loss function and Cross-Entropy loss as it has been shown to be more effective than a Two-Step training method [26, 33, 35, 42, 43]. Fig. 1 illustrates our experiments with the joint metric learning losses (i) Contrastive, (ii) Triplet, and (iii) Circle.

The metric learning losses optimize cosine similarity. Positive and negative samples are computed from all possible pairs or triplets between samples in a batch. Accordingly, we find that our batch size is sufficient for training from initial experiments. Model training utilized online mining and the *regularface* embedding regularizer [44].

In addition to re-shuffling the dataset per epoch and randomly selecting 2 s frames, we re-sample the dataset so that it randomly selects new datapoints per class per epoch. Here, we equally sample 4 datapoints per class so that a single batch includes 16 different classes; we found minimal difference from weighted sampling.

5.2 RQ2: Impact of Cross-Dataset Training

Secondly, we ask whether cross-dataset training will improve our best Cross-Entropy and metric learning results in all training scenarios. We hypothesize that a *Cross-Dataset* training scenario will outperform all others, including *UCS-Transfer* (see definition below), improving upon both UCS & Non-UCS dataset results. Our rationale is that representations generated from *Cross-Dataset* models will have seen a more significant quantity and variety of data from different sound taxonomies, producing more generalized representations. Moreover, we hypothesize that *UCS-Transfer* will yield decent classification results as UCS is a standardized taxonomy for sound effects libraries. We denote *Cross-Dataset* models with the prefix, *X*.

5.2.1 Training Scenario Definitions

We conduct experiments with 3 training scenarios: (i) *Within-Dataset* training: Pre-train and evaluate the encoder on the same dataset, (ii) *UCS-Transfer*: Pre-train the encoder on a UCS-compliant dataset and evaluate on other datasets in a transfer learning scenario, and (iii) *Cross-Dataset* training: Pre-train the encoder on all datasets and taxonomies, evaluate the encoder on any dataset.

5.3 RQ3: Impact of Cross-Dataset Training Methods

Datasets used for training may have varying characteristics. For example, they may use taxonomies of dissimilar scopes and biases and have different dataset sizes, all of which an effective classifier must adapt to. Our final question is whether training scenarios accommodating dataset characteristics will improve classification results. We experiment with 3 methods as introduced in the following subsections, (i) Data mixing (*X-Sequential*, *X-Joint*), (ii) 'Focal' dataset regularization (*FDR*), and (iii) Dataset-independent BatchNorm layers (*BN*).

5.3.1 Data Mixing

We use two variations of data mixing. *Sequential* trains on all datapoints from a single dataset before training on the next, i.e., concatenate datasets, while *Joint* trains on datapoints from all datasets in mixed order, similar to stratified sampling in the literature [36]. We limit mixing for *Joint* so that all datapoints in a batch must correspond to the same dataset. We affirm that backpropagation is called per batch.

5.3.2 'Focal' Dataset Regularization

We regularize the datasets by 'difficulty' and reweigh them by a dataset's training convergence speed in epochs. Inspired by Focal Loss [38, 45, 46] and mining for hard pairs or triplets in metric learning [29, 47–49], easier datasets are down-weighted so that training focuses on difficult datasets.

$$\alpha_d = \frac{1 - \beta^{n_e}}{1 - \beta} \tag{1}$$

We modify the reweighting function shown in Eqn. 1 to re-balance the datasets and initialize these weights with n_e, the epoch at which a dataset's training set macro F-1 score crosses a threshold of 90% [50]. These weights may be adjusted with hyperparameter β to reduce or heighten the difference between dataset weights. a_d is the unnormalized reweighing factor per dataset, d. As previously done in the literature, we normalize α_d so that $\sum_{d=1}^{D} \alpha_d = D$, keeping the loss in roughly the same range. D represents the total number of datasets used for pre-training [45, 50].

5.3.3 Dataset-Independent BatchNorm Layers

Similar to selecting the correct classifier head per dataset when pre-training the encoder, we select a dataset's corresponding BatchNorm layers. This is equivalent to turning each ConvBlock in Fig. 1 into a Dataset-Aware Block [39].

5.4 RQ4: Comparison to SoTA

We select OpenL3 features as our reference to SoTA deep audio embeddings because L^3-Net is said to "consistently outperform VGGish and SoundNet on environmental sound classification" [18]. We use the default OpenL3 parameters as they yield the highest frame-level classification results and select the *Music* content type embeddings for comparison (6144 dimensionality, 0.1 s hop size, and 256 Mel Bins) [18]. We aggregate these features to represent 2 s of an audio file for consistency with our input representation.

6. RESULTS

All results are computed at the frame-level, i.e., aggregated over each 2 s frame extracted from audio files in the hold-out test set. Boxplot datapoints indicate results for a single dataset. Metric learning plots, Fig. 4 and 5, only show *X-Sequential* results for the *Cross-Dataset* training scenario.

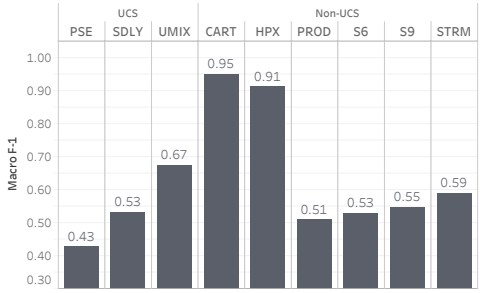


Figure 3. Baseline macro F-1 score classification results using Cross-Entropy loss, CE (*Within-Dataset*).

6.1 Baseline Results

Fig. 3 displays baseline classification results where each model is trained and tested on only one dataset, itself. Plots for the following experiments only show the change from this baseline plot.

6.2 RQ1: Impact of Metric Learning

Fig. 4 shows how metric learning models improve upon the baseline's embedding space structure to different degrees. A lower DBI indicates a better embedding space structure. Contrastive improves the space for most datasets in all training scenarios. Triplet results in a slightly higher average DBI in all but the *UCS-Transfer* training scenario. Circle performs significantly better in a *Within-Dataset* training scenario with an average DBI decrease of 0.43. However, it performs surprisingly poorly in *UCS-Transfer* & *Cross-Dataset*, suggesting that it is not ideal for generalization, a prime motivator of representation learning.

Contrary to our hypothesis, a better-structured embedding space (lower DBI) does not always improve classification results (higher macro F-1), shown in Fig. 5. We only see that metric learning models improve classification results in the *UCS-Transfer* training scenario. We specifically note that Circle significantly improves the embedding space structure compared to Contrastive in *Within-Dataset*. However, both embedding spaces still yield similar classification performance, an average of -1.75% and -1.50% from CE, respectively. We note other work by Lee et al. regarding the similarities between metric learning and classification [51].

Focusing on classification results, we identify Triplet as our 'best' metric learning loss with a slightly higher average performance than Contrastive. Triplet performs on par with CE in *Within-Dataset* and *Cross-Dataset* (less than 1% difference on average), and has an average of 4.25% improvement from CE in *UCS-Transfer*. Therefore, Contrastive and Circle loss results will be omitted in the following sections.

6.3 RQ2: Impact of Cross-Dataset Training

Results for cross-dataset training are shown in Fig. 6. Matching our hypothesis, we find that for CE models, *Cross-Dataset* outperforms *Within-Dataset* models for the majority of datasets by approximately 4-5%. We find that Triplet models also tend to improve by 1-3% except for *X-Joint-BN + Triplet*. This confirms that cross-dataset training indeed leads to better generalization across all sound taxonomies.

6.3.1 UCS Insights

We preface this section with preliminary results on UCS. We find that the UCS-compliant datasets in this study use the standardized UCS taxonomy in a non-uniform way. Without fine-tuning another classifier head, one cannot use a UCS taxonomy-trained model on other UCS-compliant datasets. This corroborates the need for generalized representations that can adapt to any taxonomy of sound.

With this said, Fig. 6 also shows that *UCS-Transfer* can achieve results considerably better than random. Although it under-performs *Within-Dataset* training by an average of 7.48% with CE models, we find that there is only an average of -1.62% difference from CE with Triplet models. Alongside, we also verify that other UCS-compliant libraries, *SDLY* and *UMIX*, can be used as base sets to achieve decent generalization in a transfer learning scenario.

We note some dataset-specific details for *UCS-Transfer + Triplet* compared to our best *Cross-Dataset* Triplet model, *X-Joint + Triplet*. Both perform similarly (around 1% worse) on Non-UCS *CART, HPX, PROD*, and *STRM* datasets. *UCS-Transfer* performs around 5-6% worse on UCS-compliant *SDLY* and *UMIX*, a substantial improvement from previous results that did not re-train a classifier head. Lastly, it performs around 7% worse on Non-UCS *S6*, likely because *S6* exhibits the highest number of classes amongst all datasets while *PSE* only has 78 classes.

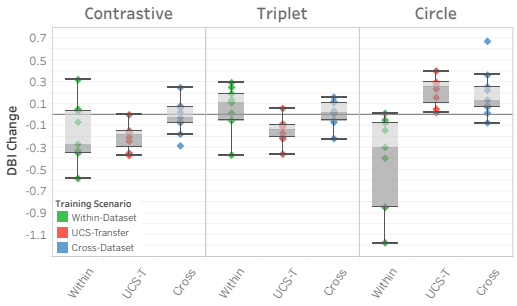


Figure 4. DBI change of metric learning models relative to CE in different training scenarios.

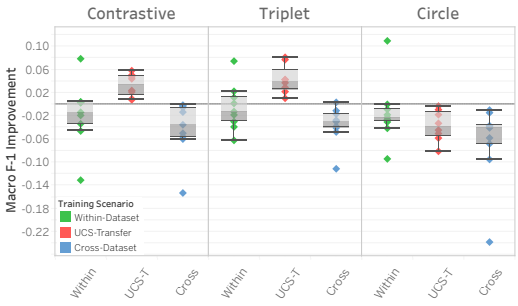


Figure 5. Macro F-1 score improvement of metric learning models relative to CE in different training scenarios.

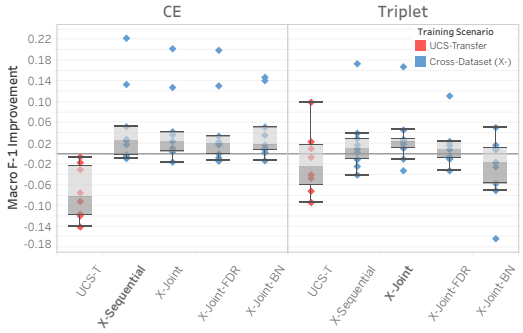


Figure 6. Macro F-1 score improvement of various training scenarios from *Within-Dataset*.

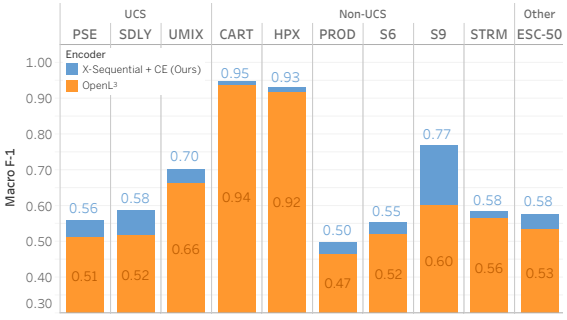


Figure 7. Our best model compared to OpenL3 audio embeddings. Blue bars illustrate the amount in which we outperform OpenL3.

6.4 RQ3: Impact of Cross-Dataset Training Methods

We experiment to see if accommodating different dataset characteristics with 3 cross-dataset training methods will further improve results. We compare all cross-dataset training methods to *X-Sequential* in our discussion.

First, sensitivity to encoder bias and training order does not severely impact results for our task. Data mixing results note an average 1% difference between *X-Sequential* and *X-Joint* in CE and Triplet.

Second, reweighing datasets as proposed in Sect. 5.3.2 does not lead to observable consistent trends. We show results that use $\beta = 0.999$. Looking at difficult datasets, the performance of *SDLY* worsens by 2.96%, while *STRM* improves by 3.27% despite having similar convergence speeds and being the second-ranked most 'difficult' datasets. We also observe an unfortunate worsening of results for faster-converging datasets, i.e., *HPX* decreases by 1.77% (Triplet), *S9* decreases by 2.32% (CE) and 6.20% (Triplet).

Third, we verify that independent BatchNorm layers do preserve dataset-specific characteristics. Unlike other work in the literature [39], we find that this may not always be beneficial as it may exacerbate dataset-specific problems. Shown in Fig. 6, we specifically note decreases in performance for Non-UCS *HPX, PROD, S6*, and *S9*, which we attribute to messier inconsistent labels and insufficient training data, apparent in Fig. 2. This decrease can be quite dramatic with *S9* dropping by 7.53% (CE) and 24.4% (Triplet). Alongside, we find that UCS-compliant *PSE* and *SDLY*, as well as Non-UCS *STRM* experience a 2-3$\%$ increase in performance, which we attribute to both the cleaner UCS taxonomy and datasets having sufficient training data.

6.5 RQ4: Comparison to SoTA

Overall, our work reveals that our best model is *X-Sequential + CE*. We compare *X-Sequential + CE* and OpenL3 in Fig. 7 and provide a commonly used benchmark dataset, ESC-50 [52], to extend the assessment of our work in a non-task-specific audio application. Unlike the rest of our datasets, we show the 5-fold cross-validation results for ESC-50 as commonly reported in the literature. Ultimately, we observe that our best model outperforms OpenL3 on all datasets. In addition, we note that our model requires fewer resources to achieve its results. While both models have a similar number of parameters (4.7M), our best model is trained on approximately 7x less data (39k audio-files vs. 296k videos) [18].

7. CONCLUSION

We introduce a new pre-trained representation for automatic sound effects library classification. Our task-specific representation outperforms OpenL3 on both UCS & Non-UCS taxonomies across diverse datasets. Moreover, we show the effectiveness of cross-dataset training with Cross-Entropy loss over metric learning for this task. Results suggest that the quality and diversity of datasets are key to pre-training robust representations, supportive of preliminary experimental results where pre-training on increasing quantities of similar data yielded improvements with diminishing returns. Our contributions include that we are the first to conduct extensive deep learning experiments on UCS, experiment on relevant data for a current problem, and investigate an under-researched aspect of representation learning with cross-dataset training in the audio domain.

We believe cross-dataset training is a prerequisite for future work. Our current research does not leverage all datasets available to us nor the abundance of taxonomies they are comprised of given that many datasets did not have sufficient labels for training. To leverage these resources, methods to effectively work with large amounts of diverse data become increasingly essential. Selecting optimal cross-dataset training methods is fundamental before introducing new techniques, such as semi-supervised or self-supervised learning to work with datasets of varying label quality [53,54]. Accordingly, we would like to (i) look into the connection between multi-task learning and cross-dataset training to understand the specific advantages and drawbacks of these approaches and (ii) conduct experiments to assess the impact of increasing the amount of training data, observing the relationship of this with more complex network architectures and other pre-training methods [55]. Finally, we hope to deepen our understanding of UCS by investigating the non-uniformity of class labels between UCS-compliant datasets in this study, looking at the interpretability and disentanglement of our representations [51], and exploring the concept of a generalized embedding from the perspective of taxonomy conversion [56].

8. REFERENCES

[1] J. Yang, Y. Zhang, and Y. Hai, "Retrieval and management system for layer sound effect library," *Cognitive Computation and Systems*, vol. 2, no. 4, pp. 247–253, 2020.

[2] G. Lafay, N. Misdariis, M. Lagrange, and M. Rossignol, "Semantic browsing of sound databases without keywords," *Journal of the Audio Engineering Society*, September 2016.

[3] S. Säger, B. Elizalde, D. Borth, C. Schulze, B. Raj, and I. Lane, "Audiopairbank: towards a large-scale tag-pair-based audio content analysis," *EURASIP Journal on Audio, Speech, and Music Processing*, p. 12, 2018.

[4] Y. Zhang, J. Hu, Y. Zhang, B. Pardo, and Z. Duan, "Vroom!: A search engine for sounds by vocal imitation queries," in *Proceedings of the 2020 Conference on Human Information Interaction and Retrieval (CHIIR)*. Association for Computing Machinery, 2020, pp. 23–32.

[5] G. G. Peeters and J. D. Reiss, "A deep learning approach to sound classification for film audio post-production," *Journal of the Audio Engineering Society*, May 2020.

[6] E. Fonseca, M. Plakal, F. Font, D. P. W. Ellis, X. Favory, J. Pons, and X. Serra, "General-purpose tagging of freesound audio with audioset labels: Task description, dataset, and baseline," *arXiv preprint arXiv:1807.09902*, 2018.

[7] D. Turnbull, L. Barrington, D. Torres, and G. Lanckriet, "Semantic annotation and retrieval of music and sound effects," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 2, pp. 467–476, 2008.

[8] X. Favory, F. Font, and X. Serra, "Search result clustering in collaborative sound collections," in *Proceedings of the 2020 International Conference on Multimedia Retrieval (ICMR '20)*, 2020, pp. 207–214.

[9] F. Font, G. Roma, and X. Serra, *Sound Sharing and Retrieval*. Cham: Springer International Publishing, 2018, pp. 279–301.

[10] D. Moffat, D. Ronan, and J. D. Reiss, "Unsupervised taxonomy of sound effects," in *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx)*, 2017.

[11] G. Lemaitre and L. M. Heller, "Evidence for a basic level in a taxonomy of everyday action sounds," *Experimental Brain Research*, vol. 226, no. 2, pp. 253–264, 2013.

[12] O. Bones, T. J. Cox, and W. J. Davies, "Sound categories: Category formation and evidence-based taxonomies," *Frontiers in Psychology*, vol. 9, pp. 1664–1078, 2018.

[13] B. Elizalde, R. Revutchi, S. Das, B. Raj, I. Lane, and L. M. Heller, "Identifying actions for sound event classification," in *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2021, pp. 26–30.

[14] B. M. Elizalde, "Never-ending learning of sounds," Ph.D. dissertation, Carnegie Mellon University, 2020.

[15] F. Font, G. Roma, and X. Serra, "Freesound technical demo," in *Proceedings of the 21st ACM International Conference on Multimedia (MM)*. Association for Computing Machinery, 2013, pp. 411–412.

[16] A. Pearce, T. Brookes, and R. Mason, "Timbral attributes for sound effect library searching," *Journal of the Audio Engineering Society*, June 2017.

[17] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. J. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2017, pp. 131–135.

[18] J. Cramer, H.-H. Wu, J. Salamon, and J. P. Bello, "Look, listen, and learn more: Design choices for deep audio embeddings," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2019, pp. 3852–3856.

[19] R. Arandjelovic and A. Zisserman, "Look, listen and learn," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017, pp. 609–617.

[20] Y. Wang, N. J. Bryan, M. Cartwright, J. P. Bello, and J. Salamon, "Few-shot continual learning for audio classification," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2021, pp. 321–325.

[21] Y. Wang, N. Bryan, J. Salamon, M. Cartwright, and J. Bello, "Who calls the shots? rethinking few-shot learning for audio," in *2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, 2021, pp. 36–40.

[22] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4367–4375.

[23] H. F. Garcia, A. Aguilar, E. Manilow, and B. Pardo, "Leveraging hierarchical structures for few-shot musical instrument recognition," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021, pp. 220–228.

[24] E. Fonseca, D. Ortego, K. McGuinness, N. E. O'Connor, and X. Serra, "Unsupervised contrastive learning of sound event representations," in *Proceedings of the*

IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), 2021, pp. 371–375.

[25] J. Spijkervet and J. A. Burgoyne, "Contrastive learning of musical representations," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021, pp. 673–681.

[26] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 18 661–18 673.

[27] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International Workshop on Similarity-Based Pattern Recognition*. Springer International Publishing, 2015, pp. 84–92.

[28] K. Q. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems (NeurIPS)*, Y. Weiss, B. Schölkopf, and J. Platt, Eds., vol. 18. MIT Press, 2005.

[29] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[30] Y. Sun, C. Cheng, Y. Zhang, C. Zhang, L. Zheng, Z. Wang, and Y. Wei, "Circle loss: A unified perspective of pair similarity optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 6398–6407.

[31] R. Zhang, H. Wu, W. Li, D. Jiang, W. Zou, and X. Li, "Transformer based unsupervised pre-training for acoustic representation learning," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2021, pp. 6933–6937.

[32] A. Saeed, D. Grangier, and N. Zeghidour, "Contrastive learning of general-purpose audio representations," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2021, pp. 3875–3879.

[33] D. Niizumi, D. Takeuchi, Y. Ohishi, N. Harada, and K. Kashino, "Byol for audio: Exploring pre-trained general-purpose audio representations," *arXiv preprint arXiv:2204.07402*, 2022.

[34] L. Wang, P. Luc, Y. Wu, A. Recasens, L. Smaira, A. Brock, A. Jaegle, J. Alayrac, S. Dieleman, J. Carreira, and A. van den Oord, "Towards learning universal audio representations," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2022, pp. 4593–4597.

[35] P. Seshadri and A. Lerch, "Improving music performance assessment with contrastive learning," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021, pp. 634–641.

[36] H.-W. Dong, K. Chen, J. McAuley, and T. Berg-Kirkpatrick, "Muspy: A toolkit for symbolic music generation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 101–108.

[37] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun, "Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 3, pp. 1623–1637, 2022.

[38] L. Wan, Q. Wang, A. Papir, and I. L. Moreno, "Generalized end-to-end loss for speaker verification," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2018, pp. 4879–4883.

[39] L. Wang, D. Li, Y. Zhu, L. Tian, and Y. Shan, "Cross-dataset collaborative learning for semantic segmentation in autonomous driving," *arXiv preprint arXiv:2103.11351*, 2021.

[40] Q. Kong, Y. Cao, T. Iqbal, Y. Xu, W. Wang, and M. D. Plumbley, "Cross-task learning for audio tagging, sound event detection and spatial localization: DCASE 2019 baseline systems," *arXiv preprint arXiv:1904.03476*, 2019.

[41] D. Bogdanov, N. Wack, E. Gómez, S. Gulati, P. Herrera, O. Mayor, G. Roma, J. Salamon, J. Zapata, and X. Serra, "Essentia: an audio analysis library for music information retrieval," in *Proceedings of the 14th International Society for Music Information Retrieval Conference (ISMIR)*, 2013, pp. 493–498.

[42] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 18 661–18 673.

[43] J. Fan, E. Nichols, D. Tompkins, A. E. M. Méndez, B. Elizalde, and P. Pasquier, "Multi-label sound event retrieval using a deep learning-based siamese structure with a pairwise presence matrix," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2020, pp. 3482–3486.

[44] K. Zhao, J. Xu, and M.-M. Cheng, "Regularface: Deep face recognition via exclusive regularization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1136–1144.

[45] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017, pp. 2999–3007.

[46] K. Nada, K. Imoto, R. Iwamae, and T. Tsuchiya, "Multitask learning of acoustic scenes and events using dynamic weight adaptation based on multi-focal loss," in *Proceedings of the 13th Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE, 2021, pp. 1156–1160.

[47] C.-Y. Wu, R. Manmatha, A. J. Smola, and P. Krahenbuhl, "Sampling matters in deep embedding learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2017, pp. 2840–2848.

[48] H. Xuan, A. Stylianou, and R. Pless, "Improved embeddings with easy positive triplet mining," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 2474–2482.

[49] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, "Multi-similarity loss with general pair weighting for deep metric learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5022–5030.

[50] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9260–9269.

[51] J. Lee, N. Bryan, J. Salamon, Z. Jin, and J. Nam, "Metric learning vs classification for disentangled music representation learning," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020, pp. 439–445.

[52] K. J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM International Conference on Multimedia (MM)*. New York, NY, USA: Association for Computing Machinery, 2015, pp. 1015–1018.

[53] K. Sohn, D. Berthelot, N. Carlini, Z. Zhang, H. Zhang, C. A. Raffel, E. D. Cubuk, A. Kurakin, and C.-L. Li, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 596–608.

[54] S. Gururani and A. Lerch, "Semi-supervised audio classification with partially labeled data," in *2021 IEEE International Symposium on Multimedia (ISM)*, 2021, pp. 111–114.

[55] T. Chen, Y. Xie, S. Zhang, S. Huang, H. Zhou, and J. Li, "Learning music sequence representation from text supervision," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2022, pp. 4583–4587.

[56] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mT5: A massively multilingual pre-trained text-to-text transformer," in *2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*. "Association for Computational Linguistics", 2021, pp. 483–498.

9. ACKNOWLEDGMENTS

We would like to thank those who provided the data required to conduct this research as well as those who took the time to share their insights and software licenses for tools regarding sound search, query, and retrieval.

Alex Lane [5]

All You Can Eat Audio [6] [7]

Articulated Sounds [8]

Audio Shade [9] [10]

aXLSound [11]

Big Sound Bank [12]

BaseHead [13]

Bonson [14] [15]

BOOM Library [16]

Frick & Traa [17]

[5] Alex Lane, *Resonant Rusty Door*. [Dataset]. Available: https://www.alex-lane.com. [Accessed: September 15, 2021].

[6] All You Can Eat Audio, *It's A Plain Phone*. [Dataset]. Available: https://allyoucaneataudio.com/library-packs/it-is-a-plain-phone-p04. [Accessed: September 15, 2021].

[7] All You Can Eat Audio, *Cuff 'Em*. [Dataset]. Available: https://allyoucaneataudio.com/library-packs/cuff-em. [Accessed: September 15, 2021].

[8] Articulated Sounds, *Starter Pack*. [Dataset] Available: https://articulatedsounds.com/audio-royalty-free-library/sfx/free-global-sample-pack. [Accessed: September 15, 2021].

[9] Audio Shade, *Bravo Dramatic Audiences*. [Dataset]. Available: https://audioshade.com/shop. [Accessed: September 15, 2021].

[10] Audio Shade, *Brutality - Gore and Combat FX Toolkit*. [Dataset]. Available: https://audioshade.com/shop. [Accessed: September 15, 2021].

[11] aXLSound, *Planes - Takeoffs & Landings*. [Dataset]. Available: https://axlsound.com. [Accessed: September 15, 2021].

[12] Big Sound Bank, *Big Sound Bank*. [Dataset]. Available: https://bigsoundbank.com. [Accessed: September 15, 2021].

[13] baseheadinc.com, last accessed September 15, 2021.

[14] Bonson, *Wings*. [Dataset]. Available: https://www.bonson.ca/product/wings. [Accessed: September 15, 2021].

[15] Bonson, *Moves*. [Dataset]. Available: https://www.bonson.ca/product/moves. [Accessed: September 15, 2021].

[16] BOOM Library, *Cyber Weapons*. [Dataset]. Available: https://www.boomlibrary.com/sound-effects/cyber-weapons. [Accessed: September 15, 2021].

[17] Frick & Traa, *City Bicycles*. [Dataset]. Available: https://www.frickandtraa.com/webshop/sound-libraries. [Accessed: September 15, 2021].

Hzandbits [18]

InspectorJ [19]

Kai Paquin [20]

KEDR Audio [21] [22] [23] [24] [25]

Krotos Audio [26] [27]

Nikola Simikic [28]

Penguin Grenade [29] [30] [31]

Pro Sound Effects [32]

Rick Allen Creative [33]

Sononym [34]

Sound Ideas [35]

Soundly [36]

Soundminer [37]

Storyblocks [38]

Tim Nielsen [39] [40] [41] [42]

Thomas Rex Beverly [43] [44]

ZapSplat [45]

[18] C. Hagelskjaer, "Interview with Christian Hagelskjaer," July 28, 2021.

[19] InspectorJ, *96 General Library*. [Dataset]. Available: https://inspectorj.sellfy.store/p/96-general-library-bundle. [Accessed: September 15, 2021].

[20] K. Paquin, *Kailibrary*. [Dataset].

[21] KEDR Audio, *Cinemaphone*. [Dataset]. Available: https://www.asoundeffect.com/sound-library/cinemaphone-ringtones-notifications. [Accessed: September 15, 2021].

[22] KEDR Audio, *Kinetics: Construction Kit*. [Dataset]. Available: https://www.asoundeffect.com/sound-library/kinetics-construction-kit. [Accessed: September 15, 2021].

[23] KEDR Audio, *Kinetics: Cinematic Tension*. [Dataset]. Available: https://www.asoundeffect.com/sound-library/kinetics-cinematic-tension. [Accessed: September 15, 2021].

[24] KEDR Audio, *Radio Nomad*. [Dataset]. Available: https://www.asoundeffect.com/sound-library/radio-nomad. [Accessed: September 15, 2021].

[25] KEDR Audio, *Vibrations*. [Dataset]. Available: https://www.asoundeffect.com/sound-library/vibrations. [Accessed: September 15, 2021].

[26] Krotos Audio, *Ammo and Reloads*. [Dataset]. Available: https://www.krotosaudio.com/products/ammo-reloads-sound-effects-library. [Accessed: September 15, 2021].

[27] Krotos Audio, *Mechanical*, 1. [Dataset]. Available: https://www.krotosaudio.com/products/mechanical-sound-effects-library-vol-1. [Accessed: September 15, 2021].

[28] N. Simikic, *Nikola Simikic Sound Library*. [Dataset].

[29] Penguin Grenade, *Arcs and Sparks*. [Dataset]. Available: https://www.asoundeffect.com/sound-library/arcs-and-sparks. [Accessed: September 15, 2021].

[30] Penguin Grenade, *Explosive Energy*. [Dataset]. Available: https://www.asoundeffect.com/sound-library/explosive-energy. [Accessed: September 15, 2021].

[31] Penguin Grenade, *Holograms*. [Dataset]. Available: https://www.asoundeffect.com/sound-library/holograms. [Accessed: September 15, 2021].

[32] Pro Sound Effects, *CORE 2*, Pro. [Dataset]. Available: https://www.prosoundeffects.com/core-2. [Accessed: September 15, 2021].

[33] R. Allen, "Interview with Rick Allen Creative," August 06, 2021.

[34] B. Næsby, "Interview with Bjørn Næsby from Sononym," September 10, 2021.

[35] Sound Ideas, *Sound Ideas Library*. [Dataset]. Available: https://www.sound-ideas.com. [Accessed: February 01, 2022].

[36] Soundly, *Soundly*. [Dataset]. Available: https://www.soundly.com. [Accessed: February 01, 2022].

[37] store.soundminer.com, last accessed September 15, 2021.

[38] storyblocks.com, last accessed September 15, 2021.

[39] T. Nielsen, *Early Mother Communicator*. [Dataset].

[40] T. Nielsen, *Ether*. [Dataset].

[41] T. Nielsen, *Yellowstone*. [Dataset].

[42] T. Nielsen, *Baltic*. [Dataset].

[43] T. R. Beverly, *Maine Bundle*. [Dataset]. Available: https://thomasrexbeverly.com/collections/sound-libraries/products/maine-bundle. [Accessed: September 15, 2021].

[44] T. R. Beverly, *Wild Animal Calls*. [Dataset]. Available: https://thomasrexbeverly.com/products/wild-animal-calls. [Accessed: September 15, 2021].

[45] A. McKinny, "Interview with Alan McKinny from ZapSplat," July 30, 2021.

CONCEPT-BASED TECHNIQUES FOR "MUSICOLOGIST-FRIENDLY" EXPLANATIONS IN A DEEP MUSIC CLASSIFIER

Francesco Foscarin[1*] **Katharina Hoedt**[1*] **Verena Praher**[1*]
Arthur Flexer[1] **Gerhard Widmer**[1,2]
[1] Institute of Computational Perception, Johannes Kepler University Linz, Austria
[2] LIT AI Lab, Linz Institute of Technology, Austria
{firstname}.{lastname}@jku.at

ABSTRACT

Current approaches for explaining deep learning systems applied to musical data provide results in a low-level feature space, e.g., by highlighting potentially relevant time-frequency bins in a spectrogram or time-pitch bins in a piano roll. This can be difficult to understand, particularly for musicologists without technical knowledge. To address this issue, we focus on more human-friendly explanations based on high-level musical concepts. Our research targets trained systems (post-hoc explanations) and explores two approaches: a supervised one, where the user can define a musical concept and test if it is relevant to the system; and an unsupervised one, where musical excerpts containing relevant concepts are automatically selected and given to the user for interpretation. We demonstrate both techniques on an existing symbolic composer classification system, showcase their potential, and highlight their intrinsic limitations.

1. INTRODUCTION

The mass adoption of deep learning methods in recent years has increased interest in the field of *explainability*, [1] i.e., the study of techniques that generate a human-understandable explanation of a model's decision [1]. As deep learning models are usually not intrinsically interpretable, techniques that can be applied to trained models (i.e., *post-hoc* methods) are of great interest. The resulting explanations cannot only reveal potential issues of the system itself and the data it uses, but can also provide insights into the problem we are targeting, thus helping us to gain knowledge about it [2].

Higher-level musical tasks such as chord transcription or composer classification may require explanations that can only be understood by persons with advanced musical expertise. However, the explanation techniques for musical systems that have been proposed in recent years [3–7] are *feature-based*, i.e., the explanation is given in terms of the input features the system considers. Typical input features for musical systems, e.g., spectrograms or piano roll representations, are high-dimensional, and, since the importance of a single time frequency / pitch bin does not convey much meaningful interpretation, feature-based explanations can be hard to understand. [2] Musicologists are therefore often unable to analyse the results of trained systems, let alone contribute to the development of new learning models. This motivates research on techniques that provide explanations that are as similar as possible to those that a human music domain expert would naturally use.

Concept-based explanations offer an interesting direction. They were first explored by Kim et al. [12] and later developed in several works (e.g., by Chen et al. [13]) for image systems, which also use high-dimensional input features. Instead of producing feature-level descriptors, the explanation is based on human-understandable concepts. For example, [12] tests whether the concept of "stripes" would increase the probability that an image classifier labels an image as "zebra". Music can also be described with *musical concepts*; terms such as "diatonic sequence", "alberti bass", "difficult-to-play music", "orchestral music", "rubato", "shuffle drum beat", "funky bass line", etc. are used to describe pieces or specific elements in a piece.

In this paper, we explore two concept-based techniques to explain deep learning systems that deal with musical data: one supervised and one unsupervised. The first (see Section 4) is based on Testing with Concept Activation Vectors (TCAV) [12]: the user defines a concept by providing examples and interrogates the system to find out if the concept is relevant or not for its decision. This can be applied to any kind of neural network and, even more generally, to any system that has a hidden layer for which we can compute directional derivatives. The second technique, described in Section 5, is an adaptation of [14] and works in an unsupervised fashion, where the most relevant concepts are automatically produced and given to the user for interpretation. Each concept is presented in the form of a set of musical excerpts. This approach requires networks whose hidden layers contain only non-negative values and

* Equal contribution.
[1] Considered synonymous to the term *interpretability* in this paper.

© F. Foscarin, K. Hoedt, V. Praher, A. Flexer, and G. Widmer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** F. Foscarin, K. Hoedt, V. Praher, A. Flexer, and G. Widmer, "Concept-Based Techniques for "Musicologist-friendly" Explanations in a Deep Music Classifier", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

[2] Moreover, their truthfulness has recently been debated [8–11].

have a spatial correlation with the input data, two conditions that are satisfied by most Convolutional Neural Networks (CNNs).

Our contributions are: the first application of concept-based post-hoc approaches to musical data, in particular, we target the composer classification system of Kim et al. [15] that uses piano roll representations of piano MIDI files as input; the definition and creation of musical concept datasets; a dedicated visualisation of unsupervised concepts for symbolic music data; and, finally, the exploration of the Non-negative Tucker Decomposition (NTD) for the factorisation of hidden layers. Our code and data are available on Github. [3]

2. RELATED WORK

Recent work in the field of Music Information Retrieval (MIR) that focus on explainability for deep models consists mainly of feature-based post hoc methods (e.g., [3–5, 7]). Chowdhury et al. introduce pre-defined "mid-level features" that could be considered concepts as intermediate targets in a two-level prediction model [16]. Related to this, approaches that consider instrinsic as opposed to post-hoc methods gain increasing attention in the audio domain as well (e.g., [17–19]). However, no prior studies have examined post-hoc concept-based explainability techniques on systems that work with musical data. To provide a technical context for our work, we focus on related approaches that work on audio or image data.

A recent approach [20] applies concept-based techniques to multimodal data (video, audio, and text) to explain an emotion classifier for video sequences of human conversations. For the audio signal, they only test the concept of "voice pitch", i.e., the averaged fundamental frequency of the speaker's voice. In another related study, Parekh et al. [21] learn a codebook of sounds (e.g., alarm sound) from input audio through Non-negative Matrix Factorisation (NMF), which is then used to obtain hidden network layer representations that indicate time activations of these pre-learnt components. This has some similarities with our unsupervised approach, as we also make use of non-negative factorisation techniques to disentangle concepts. However, while [21] performs the factorisation on the input and propagates the results to a hidden layer, we factorise the hidden layer activations and project the results back to the input data. The approach of [21] is promising if we assume that the underlying reason for a system decision can be extracted directly from the input with unsupervised separation approaches. However, since this might not always be the case, we factorise the layer activations to exploit the non-linear feature extraction a network does internally to obtain more meaningful explanations.

Our work uses techniques and results originally proposed for the image domain. The work of Kim et al. [12] provides the basis for the supervised explanation, although the creation of concept data sets is more challenging for music. We base the unsupervised explanation on the work of Zhang et al. [14], but propose a dedicated visualisation of piece excerpts and test different solutions for the tensor factorisation step by employing the NTD.

[3] https://github.com/CPJKU/composer_concept

3. EXPERIMENTAL SETUP

This section details the type of data and the system that we use to demonstrate our explainability techniques.

Data: We use MIDI representations of piano performances from the MAESTRO v2.0.0 dataset [22]. As proposed by Kim et al. [15], we pre-select data by composers with at least 16 pieces and remove files with more than one composer (e.g., Schubert/Liszt, "Der Mueller und der Bach"), so that pieces of 13 different composers remain (see Table 1). We randomly split the resulting 667 pieces in a training (462 pieces) and validation (205 pieces) set. For each piece, we sample 90 excerpts of 20 seconds randomly across time and different performances of the same piece (if available).

Composer Classifier: In this work, we investigate the composer classification system proposed by Kim et al. [15]. For more recent systems, the code was not available [23] or we were unable to reproduce their results [24]. During preprocessing, we transform MIDI excerpts into piano roll representations with a 50 ms time step, i.e., a matrix 88×400, which is used as input to a ResNet-50 [25]. Kim et al. [15] use an additional channel with onset information, which we omit because it does not improve the performance of our system. As proposed by [15], we train the network with Stochastic Gradient Descent with momentum (factor 0.9), L2 weight regularisation (factor 0.0001), and cross-entropy loss function. The initial learning rate is set to 0.01, and scheduled with cosine annealing [26]. Our attempt to retrain the system results in a F1 score of 0.93 compared to 0.83 in the original work [15]. The accuracy of our system is 0.93. The difference in performance could be attributed to a problem during preprocessing in the original code, which reduced the resolution of the piano roll.

4. SUPERVISED CONCEPT-BASED EXPLANATIONS

In this section, we use TCAV [12] to build a supervised concept-based explainer. We manually define musical concepts and interrogate a music classifier to find out how much a concept influences the results of the classifier.

4.1 Musical Concepts

Musical concepts describe the characteristics of a certain group of notes and are identified by musicologists with a specific name or with a small sentence (e.g., "staccato", "rubato", "melody with jumps", etc.). To define a musical concept in a way that can be used within our system, we construct *concept datasets*, i.e., sets of pieces that have one specific musical concept in common. In this paper, we build three different concept datasets, each consisting of 30 musical excerpts of ~25 seconds. Ideally, the bigger and more diverse the concept datasets is, the lower is the probability that it will also represent other unwanted concepts.

The first dataset describes the *"alberti bass"*: an accompaniment pattern first used during the classical period, where notes of chords are horizontally distributed in the left hand part [27]. For this dataset a semi-professional pianist composed ~25 second excerpts that contain this pattern, while trying to vary as much as possible other musical elements (e.g., key, tempo, content of the right hand).

The second concept is *"difficult-to-play music"*. A dataset of difficult musical excerpts was collected using the ranking produced by the musical score publisher G. Henle. [4] Excerpts were sampled from difficult pieces available in the MAESTRO dataset among different composers to avoid introducing biases toward some of them.

The third concept is *"contrapuntal texture"*, which denotes piano pieces composed of multiple monophonic voices that behave as separate instruments. This style is mostly present in pieces by some Baroque composers (e.g., Bach, Telemann, Händel, Buxtehude). For this dataset, we sampled Bach fugue performances, ensuring that they were not used during training the targeted composer classifier.

In addition to these three concept datasets, in this paper we use a collection of 10 different *random datasets*, which are built by randomly sampling 20 second excerpts from the MAESTRO dataset.

4.2 CAVs and Conceptual Sensitivity

A *Concept Activation Vector (CAV)* [12] $\mathbf{v}_l^k$ represents a concept k in the output space of a neural network layer l. To compute it, we need the corresponding concept dataset (containing e.g., pieces with alberti bass) and a random dataset [12]. For a specific network layer l, we compute the layer activations (i.e., the output of the layer) for every piano roll $\mathbf{x}$ in the concept dataset, as well as the random dataset (see Figure 1). These activations can be seen as points in a $(H \times W \times C)$-dimensional space, where H, W, C are the horizontal, vertical, and channel size in the layer activations tensor. We train a binary linear classifier (e.g., Support Vector Machine (SVM) or logistic regression) that separates the layer activations of the concept pieces from those of the random pieces. The vector of coefficients of this binary classifier, i.e., the vector orthogonal to the classification boundary, is the CAV $\mathbf{v}_l^k$ [12].

To measure whether a concept k is relevant for a piece being classified as a certain composer o, we use the *conceptual sensitivity* $S_{k,o,l}$ of the system [12], i.e., the directional derivative of the prediction in the direction of the CAV,

$$S_{k,o,l} = \nabla g_{l,o}(f_l(\mathbf{x})) \cdot \mathbf{v}_l^k. \quad (1)$$

Here, $g_{l,o}$ transforms the activation vector $f_l(\mathbf{x})$ to the logit for the output class o, that is, it represents the remaining computations after a layer l up to the output of the system. Intuitively, S is a scalar that measures how much the output logits change if we perturb the layer activations in the direction of the CAV. Positive values mean that a concept k encourages the classification of $\mathbf{x}$ as class o.

[4] https://www.henle.de/us/about-us/levels-of-difficulty-piano/

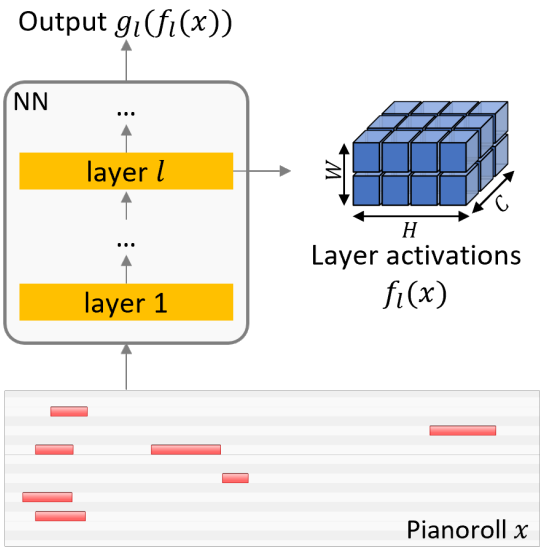


Figure 1. Layer l activations for one piece excerpt. H, W and C are the horizontal, vertical, and channel size. The function encoded by the neural network is represented in two parts: f_l from the NN input to the layer output, and g_l from the layer output to the NN output.

The conceptual sensitivity is a local explanation, i.e., it explains how a system behaves for a specific input. We produce a global explanation, the TCAV score, that no longer depends on a specific input, by taking multiple pieces that belong to one class and computing the ratio of pieces for which S is positive [12].

4.3 Experiments and Results

To investigate TCAV, we first compute a CAV for every one of our proposed concepts "alberti bass", "difficult-to-play music", and "contrapuntal texture". [5] As a linear classifier that separates the activations of the concept samples from those of the random samples, we use a linear SVM. This approach requires inputs of the same dimension, so we crop or pad all concepts to 20 seconds length (also used during training). Cropping is done by selecting the middle 20 seconds of a MIDI performance; padding adds silence until the appropriate length is reached.

In the next step, we examine the conceptual sensitivities of all the validation data and compute the TCAV score for all pieces by the same composer, i.e., the relative amount of positive conceptual sensitivities over the pieces. We again need to ensure inputs have the same length as our concepts, so we split every piece into non-overlapping 20 second segments and use all of these for subsequent computations. Although we can compute the TCAV score for any layer of the composer classifier, for brevity we show results of the penultimate layer subsequently, expecting this layer to encode the highest level features, similar to the image domain [28]. To compute TCAV scores, we perform ten runs with ten random datasets [12], and run a two-sided t-test and a Bonferroni correction for all concepts and composers

[5] Using https://captum.ai/api/concept.html

	Bach	*Scarlatti*	*Haydn*	*Mozart*	*Beethoven*	*Schubert*	*Chopin*	*Schumann*	*Liszt*	*Brahms*	*Debussy*	*Scriabin*	*Rachmn.*
alberti bass	+		+	+	+		-		-	+	-	-	-
diff.-to-play music	-		-	-				+	+			+	+
contrapuntal texture	+			+		-	-		-		-		

Table 1. Summary of TCAV scores for three concepts and the penultimate layer of a composer classifier. "+" indicate a positive influence of a concept on the classification of a composer, "-" negative influence. Empty cells show results that fail significance testing, i.e., the concept does not consistently en-/decourage the classification of a certain composer.

to validate our experiments. We use a significance threshold of $\alpha = 0.05/13$ (correcting for 13 hypothesis tests).

In our experiments, the SVM differentiating between concepts and random data has an accuracy greater than 0.9 for all concepts (in most cases, even 1). This means that the activations of the penultimate layer for the concept and the random samples are linearly separable, i.e., the CAV we produce represents the concept it is targeting. The TCAV score results are summarised in Table 1. All cells with "+" or "-" show results that pass our statistical significance test, and the remaining (empty) cells show results that fail. The symbol "+" means that a particular concept appears important for the classification of the corresponding composer (i.e., average TCAV score > 0.5); "-" that a concept discourages the classification of an input as a certain composer (i.e., average TCAV score < 0.5).

Table 1 shows both results that we would expect (e.g., alberti bass being relevant for Mozart, contrapuntal texture for Bach), and a few results which seem counter-intuitive (e.g., the model relying on alberti bass for Bach or Brahms – although one can find examples of such structures also in their works). It is possible that our model mixes the proposed concepts with other confounding ones, e.g., all pieces in the Alberti Bass dataset have also a quite simple harmonic structure. In general, there is no proof that the model *understands* our proposed concepts similarly to how a human listener would, yet we can make some interesting observations. Remarkably, even an extremely abstract concept such as "difficult-to-play music" might be grasped by our model: Liszt, Scriabin, Rachmaninoff are clear candidates for this attribute. The same applies to the "contrapuntal texture". Cases with negative impact ("-") should probably be interpreted with care: the fact that the classifier did not consider these concepts relevant for the classification does not necessarily mean that they are not present in the pieces. This could apply in particular to the subtle concept of contrapuntal texture. Also interesting is the case of Scarlatti, who is very much an outsider in classical music, style-wise ("a freakish if not downright incorrect composer" [29]) and could not be associated with any of our concepts.

5. UNSUPERVISED CONCEPT-BASED EXPLANATIONS

The supervised approach requires the user to pre-define concepts. This is very time-consuming, and the user could have to try a potentially infinite number of concepts if the network works differently than expected. In this section, we discuss an unsupervised approach instead: we build an explainer that identifies the relevant concepts and presents pieces where this concept is maximally activated (and some where the concept is not present). The musical expertise of the user is then used to translate these example pieces into a musical concept (with or without a name).

For the unsupervised approach described below, we introduce two limitations on the target neural network: we assume it to be convolutional and to use a non-negative activation function [30] (e.g., ReLU).

5.1 Tensor Factorisation for CAV Extraction

Consider a set of layer activations $\mathcal{X} = \{f_l(\mathbf{x}_1), \cdots, f_l(\mathbf{x}_N)\}$ generated from multiple pieces $\mathbf{x}_1, \ldots, \mathbf{x}_N$. Layer activations that are close together (in terms of Euclidean distance) correspond to perceptually similar inputs [31] and therefore might describe similar concepts within the inputs. We could cluster similar activations and consider the pieces that generate these activations as examples of the same concept [32].

This works best if only one concept is present in a piece excerpt. However, we expect an excerpt to contain a number of different musical concepts, e.g., an alberti bass and a legato melody, both following a certain chord progression. Since these concepts can be shared across the same notes, we need a way to disentangle their effects on the layer activations. Due to the restriction on the type of activations (only non-negative) that we introduced in this section, we can use the NTD for this objective (see Section 5.3).

5.2 Channel CAVs

Given layer activations $f_l(\mathbf{x})$, let us consider their *channel-mode tubes* [33], i.e., the vectors obtained by fixing an index h and w for the horizontal and vertical dimension (see left-hand side of Figure 2). In the case of CNNs, we can consider each of these vectors as a different representation of the same piece with a different receptive field [34]. As proposed in [14], we can analyse channel-mode tubes $\in \mathbb{R}^C$ instead of full layer activations $\in \mathbb{R}^{H \times W \times C}$. This increases the

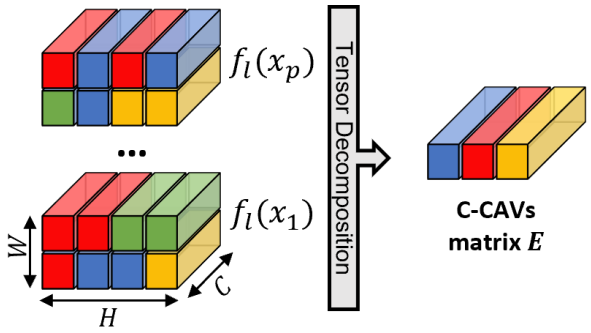


Figure 2. Left: Channel-mode tubes that result from fixing indices of activations in the W and H dimension of the activation space. Right: C-CAVs extraction from the layer activations of multiple pieces. Each channel tube is decomposed as a weighted sum of C' C-CAVs.

amount of data and reduces the dimension of each data point by a factor of $H \times W$, therefore, we expect that the tensor decomposition that we will run on these data will achieve better results. Since every channel-mode tube represents a piece, we can compute CAVs in this restricted channel space $\mathbb{R}^C$, and refer to them as *Channel-CAVs* (or C-CAVs). We can again compute the conceptual sensitivities for such C-CAVs, as explained in Section 4.2.

We compute C-CAVs by starting from a dataset of pieces (segmented in 20-second excerpts) of the composers we want to explain (any number of composers can be considered). We input each excerpt into the trained system and produce activations for a certain layer l (see Figure 1). The set of all activations can be seen as a tensor $\mathcal{X} \in \mathbb{R}^{N \times H \times W \times C}$, where N is the number of piece excerpts and H, W and C are the frequency, time, and channel size of the layer activation tensor. We then apply a NTD to $\mathcal{X}$ to obtain a set of C-CAVs.

Moving to a channel-based formulation also permits us to highlight in which part of a piece a certain concept is present: since layer activations have a spatial correlation with input data [35] we can project the position h, w of each C-CAV onto the input piano roll. This creates a *concept presence* heatmap showing the presence of a C-CAV [14] on a piano roll, which can be visualised to improve the user's understanding of the concept. Averaging the values in this heatmap gives a number that expresses how much a concept is activated in a certain excerpt and allows a ranking of the pieces according to their average concept presence.

5.3 Non-negative Tucker Decomposition

The NTD is a technique to decompose a tensor (e.g., $\mathcal{X}$) into a so-called non-negative *core* tensor $\mathcal{T}$, and multiple *factor* matrices $\mathbf{A} \in \mathbb{R}^{N \times N'}$, $\mathbf{B} \in \mathbb{R}^{H \times H'}$, $\mathbf{D} \in \mathbb{R}^{W \times W'}$ and $\mathbf{E} \in \mathbb{R}^{C \times C'}$ (one for every dimension) [33], such that

$$\mathcal{X} \approx \sum_{n=1}^{N'} \sum_{h=1}^{H'} \sum_{w=1}^{W'} \sum_{c=1}^{C'} t_{nhwc} \mathbf{a}_n \circ \mathbf{b}_h \circ \mathbf{d}_w \circ \mathbf{e}_c \quad (2)$$

Here, the core tensor $\mathcal{T}$ is in $\mathbb{R}^{N' \times H' \times W' \times C'}$, and one of its scalar elements is denoted by t_{nhwc}. The symbol "$\circ$" denotes the vector outer product of the column vectors of the (four) factor matrices $\mathbf{a}_n, \mathbf{b}_h, \mathbf{d}_w, \mathbf{e}_c$. The number of columns of the factor matrices (i.e., the NTD ranks), N', H', W' and C' are hyper-parameters that can be chosen by the user; if we set them to $N' = N$, $H' = H$, etc., an exact reproduction of the original tensor $\mathcal{X}$ is possible [33]. However, we mostly want to set them at lower values (i.e., $N' << N$, $H' << H$, etc.) to decrease the size of the matrices.

Equation 2 tells us that every channel-mode tube in $\mathcal{X}$ can be reconstructed as a weighted sum of the columns of matrix $\mathbf{E}$. As previously mentioned, each channel-mode tube represents a piece (in activation space), and each piece contains a sum of multiple concepts as C-CAVs. Then the C-CAVs we are looking for are disentangled as columns in $\mathbf{E}$ (see Figure 2), and their number is specified by rank C'.

The NTD also allows us to reconstruct an approximation of the original tensor (i.e., the original layer activations), and compute the output of the composer classifier by feeding it back into the network. We can then compute the ratio of the predictions that remain unchanged after the NTD step (i.e., the *fidelity*) [14] to evaluate its impact on the composer classifier. For more details on NTD, we refer to [33]; in this paper, we use the implementation provided in [36], with the Hierarchical non-negative Alternating Least Squares algorithm to update the factor matrices and the Fast Iterative Shrinkage-Thresholding Algorithm to update the core.

5.4 Experiments and Results

We compute the unsupervised explanations for the penultimate layer of our composer classification system and test multiple NTD ranks. As in [14], we present each concept to the user through the five piece excerpts with the highest average concept presence. The presentation of piece excerpts is more challenging for musical data than for images. Although symbolic performances can be visualised with piano rolls, some musical elements (e.g., harmonic elements) may be hard to understand in this format. We opt for a mixed audio-image visualisation where each excerpt is represented both with a piano roll (with a colour scale for velocity information) and with a listenable MIDI file. We create interactive piano roll visualisations (using Plotly [37]) in which the user can zoom in and out to explore different resolution levels. The concept presence heatmap is displayed as a semi-transparent mask over the piano roll. A heatmap with a fixed threshold, as proposed in [14], is hard to interpret for our data, so the user is presented with a slider that adapts the heatmap threshold (see Figure 3). We also provide "contrastive examples" for each concept, i.e., the 5 excerpts where the average concept presence is minimal. Although our explainer could find relevant concepts starting from a dataset that includes any number of composers, we focus on the results with only two composers. According to psychological studies [2], explanations are easier to understand when they involve only a small amount of information and when they target contrast cases [1], i.e., understanding why a composer is selected instead of another is easier than

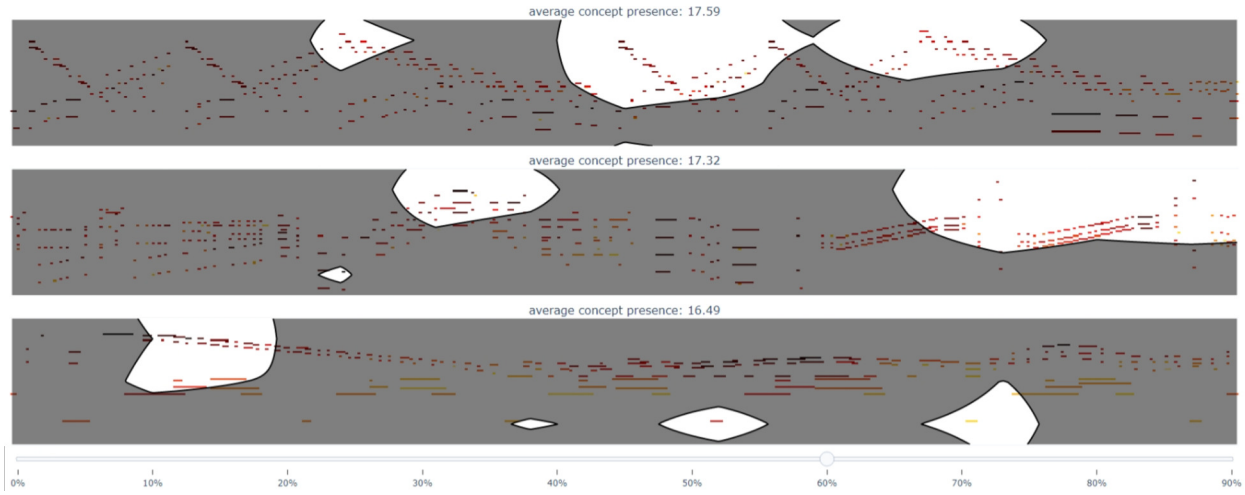


Figure 3. Visualisation of one concept (out of 4 produced by 4d NTD) through the masked piano rolls of the three dataset excerpts with the highest average concept presence. The concept heatmap threshold is set to 60%. This concept has positive conceptual sensitivity for Chopin and negative for Bach, therefore it is useful to distinguish between the two composers.

understanding why a composer is selected in general. For this reason, we also focus on C-CAVs with opposing conceptual sensitivities, i.e., negative for one class and positive for the other. We experimented with three different non-negative factorisation approaches: NTD applied to the 4d matrix (as explained in Section 5.3), NTD on a 3d matrix with concatenated horizontal and vertical dimensions, and NMF on a 2d matrix (as proposed by [14]) with concatenated horizontal, vertical, and piece dimension.

We found that the target classifier, when only two composers are considered, can be approximated with maximum fidelity by using only 3 to 5 C-CAVs, depending on the composers considered. For a fixed number of C-CAVs, we found no clear advantage for one of the three factorisation techniques with respect to the fidelity score. NTD allows for a much higher compression of $\mathcal{X}$, up to 15 times smaller, preserving the same fidelity; but it is also much slower to compute. From a manual analysis, we see that our unsupervised explainer finds, for each opposing concept, examples of typical composing styles that are useful for discriminating between two composers.

Figure 3 shows an example of what our model considers a typical Chopin-style pattern that is not present in Bach's music. In musical terms, it might be named "fast upward or downward movements (in the upper register) in parallel or broken thirds/sixths/octaves". Since our approach is based on non-negative factorisation techniques, some of their typical problems are also present in our results. For example, our system could produce one C-CAV that comprises what musical experts would typically interpret as two different concepts, or vice versa, produce two C-CAVs, both referring to the same concept. The former might have happened with the four small, seemingly unrelated blobs in the lower registers in the last two piano rolls of Figure 3. Furthermore, it is difficult to assign musically meaningful concept names to some C-CAVs, especially those with low average concept presence.

6. CONCLUSION AND FUTURE WORK

In this paper, we explored a supervised and an unsupervised approach with the aim of producing explanations of deep musical classifiers interpretable by musicologists. In the supervised approach, we define high-level musical concepts (e.g., alberti bass) by building concept datasets and interrogate a classifier to find the relevance of a concept for the classifier decisions. This approach is useful when the user wants to test a specific concept. However, the process of creating a concept dataset can be time-consuming, requires high-level music expertise, and it could be necessary to try many different concepts before finding a relevant one. A solution to these problems is the unsupervised approach, which selects the relevant concepts by itself. Each concept is presented as a set of piece excerpts where the concept is maximally present. The user can listen to those excerpts and visualise them in a piano roll representation with a heatmap highlighting the concept position.

Future work on the supervised explainer will integrate recent promising results on model non-linearity and stricter hypothesis testing [38]. The unsupervised part will benefit from a formal user-based evaluation by musicologists to see which number of C-CAVs produce the most interpretable musical concepts and if there is agreement on their naming. Sparsity constraints applied to the core tensor and matrices in the NTD may attenuate the non-negative factorisation problems. While both supervised and unsupervised approaches work on piece excerpts of fixed length, an extension to variable length pieces could enable the study of concepts that span a longer time frame (e.g., piece structure). Moreover, our two approaches could be applied to explain audio classifiers, although this would complicate the visualisation of the concept heatmap for the unsupervised explainer, and could make the creation of concept datasets more challenging. Finally, dedicated user interfaces enabling to define concepts and visualise results would be helpful for musicologists.

7. ACKNOWLEDGEMENTS

This work was supported by the European Research Council (ERC) under the EU's Horizon 2020 research & innovation programme, grant agreement No. 101019375 (*Whither Music?*), the Austrian Science Fund (FWF, project No. P31988), and the Federal State of Upper Austria (LIT AI Lab).

8. REFERENCES

[1] T. Miller, "Explanation in Artificial Intelligence: Insights from the Social Sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, 2019.

[2] C. Molnar, *Interpretable Machine Learning*, 2nd ed., 2022. [Online]. Available: https://christophm.github.io/interpretable-ml-book

[3] S. Mishra, B. L. Sturm, and S. Dixon, "Local Interpretable Model-agnostic Explanations for Music Content Analysis," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR*, 2017, pp. 537–543.

[4] S. Mishra, E. Benetos, B. L. Sturm, and S. Dixon, "Reliable Local Explanations for Machine Listening," in *Proceedings of the 2020 International Joint Conference on Neural Networks, IJCNN*. IEEE, 2020, pp. 1–8.

[5] V. Haunschmid, E. Manilow, and G. Widmer, "audioLIME: Listenable Explanations Using Source Separation," in *Proceedings of the 13th International Workshop on Machine Learning and Music, MML*, 2020, pp. 20–24.

[6] ——, "Towards Musically Meaningful Explanations Using Source Separation," *CoRR*, vol. abs/2009.02051, 2020.

[7] A. B. Melchiorre, V. Haunschmid, M. Schedl, and G. Widmer, "LEMONS: Listenable Explanations for Music recOmmeNder Systems," in *Advances in Information Retrieval: Proceedings of the 43rd European Conference on IR Research, ECIR*, vol. 12657. Springer, 2021, pp. 531–536.

[8] A. Ghorbani, A. Abid, and J. Y. Zou, "Interpretation of Neural Networks Is Fragile," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI*, 2019, pp. 3681–3688.

[9] P. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. T. Schütt, S. Dähne, D. Erhan, and B. Kim, "The (Un)reliability of Saliency Methods," in *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, ser. Lecture Notes in Computer Science. Springer, 2019, vol. 11700, pp. 267–280.

[10] J. Adebayo, J. Gilmer, M. Muelly, I. J. Goodfellow, M. Hardt, and B. Kim, "Sanity Checks for Saliency Maps," in *Advances in Neural Information Processing Systems 31: Proceedings of the 32nd Annual Conference on Neural Information Processing Systems, NeurIPS*, 2018, pp. 9525–9536.

[11] V. Praher, K. Prinz, A. Flexer, and G. Widmer, "On the Veracity of Local, Model-agnostic Explanations in Audio Classification: Targeted Investigations with Adversarial Examples," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 531–538.

[12] B. Kim, M. Wattenberg, J. Gilmer, C. J. Cai, J. Wexler, F. B. Viégas, and R. Sayres, "Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)," in *Proceedings of the 35th International Conference on Machine Learning, ICML*. PMLR, 2018, pp. 2673–2682.

[13] Z. Chen, Y. Bei, and C. Rudin, "Concept Whitening for Interpretable Image Recognition," *Nature Machine Intelligence*, vol. 2, no. 12, pp. 772–782, 2020.

[14] R. Zhang, P. Madumal, T. Miller, K. A. Ehinger, and B. I. P. Rubinstein, "Invertible Concept-based Explanations for CNN Models with Non-negative Concept Activation Vectors," in *Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI*, 2021, pp. 11 682–11 690.

[15] S. Kim, H. Lee, S. Park, J. Lee, and K. Choi, "Deep Composer Classification Using Symbolic Representation," *ISMIR Late Breaking and Demo Papers*, 2020.

[16] S. Chowdhury, A. Vall, V. Haunschmid, and G. Widmer, "Towards Explainable Music Emotion Recognition: The Route via Mid-level Features," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 237–243.

[17] P. Zinemanas, M. Rocamora, M. Miron, F. Font, and X. Serra, "An Interpretable Deep Learning Model for Automatic Sound Classification," *Electronics*, vol. 10, no. 7, p. 850, 2021.

[18] P. Zinemanas, M. Rocamora, E. Fonseca, F. Font, and X. Serra, "Toward Interpretable Polyphonic Sound Event Detection with Attention Maps Based on Local Prototypes," in *Proceedings of the 6th Workshop on Detection and Classification of Acoustic Scenes and Events, DCASE*, 2021, pp. 50–54.

[19] Z. Ren, T. T. Nguyen, and W. Nejdl, "Prototype Learning for Interpretable Respiratory Sound Analysis," in *Proceedings of the 2022 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2022, pp. 9087–9091.

[20] A. R. Asokan, N. Kumar, A. V. Ragam, and S. S. Sharath, "Interpretability for Multimodal Emotion Recognition using Concept Activation Vectors," *CoRR*, vol. abs/2202.01072, 2022.

[21] J. Parekh, S. Parekh, P. Mozharovskyi, F. d'Alché-Buc, and G. Richard, "Listen to Interpret: Post-hoc Interpretability for Audio Networks with NMF," *CoRR*, vol. abs/2202.11479, 2022.

[22] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C. A. Huang, S. Dieleman, E. Elsen, J. H. Engel, and D. Eck, "Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset," in *Proceedings of the 7th International Conference on Learning Representations, ICLR*, 2019.

[23] Q. Kong, K. Choi, and Y. Wang, "Large-Scale MIDI-based Composer Classification," *CoRR*, vol. abs/2010.14805, 2020.

[24] D. Yang and T. Tsai, "Composer Classification With Cross-Modal Transfer Learning and Musically-Informed Augmentation," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 802–809.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE Computer Society, 2016, pp. 770–778.

[26] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," in *Proceedings of the 5th International Conference on Learning Representations, ICLR*, 2017.

[27] C. Rosen, *The Classical Style: Haydn, Mozart, Beethoven*. WW Norton & Company, 1997, no. 653.

[28] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba, "Object Detectors Emerge in Deep Scene CNNs," in *Proceedings of the 3rd International Conference on Learning Representations, ICLR*, 2015.

[29] R. Kirkpatrick, *Domenico Scarlatti: Revised Edition*. Princeton University Press, 1983, vol. 200.

[30] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," *CoRR*, vol. abs/1811.03378, 2018.

[31] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric," in *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 586–595.

[32] A. Ghorbani, J. Wexler, J. Y. Zou, and B. Kim, "Towards Automatic Concept-based Explanations," in *Advances in Neural Information Processing Systems 32: Proceedings of the 33rd Annual Conference on Neural Information Processing Systems, NeurIPS*, 2019, pp. 9273–9282.

[33] T. G. Kolda and B. W. Bader, "Tensor Decompositions and Applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[34] W. Luo, Y. Li, R. Urtasun, and R. S. Zemel, "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 29: Proceedings of the 30th Annual Conference on Neural Information Processing Systems, NIPS*, 2016, pp. 4898–4906.

[35] J. Dai, K. He, and J. Sun, "Convolutional Feature Masking for Joint Object and Stuff Segmentation," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE Computer Society, 2015, pp. 3992–4000.

[36] J. Kossaifi, Y. Panagakis, A. Anandkumar, and M. Pantic, "TensorLy: Tensor Learning in Python," *Journal of Machine Learning Research*, vol. 20, pp. 26:1–26:6, 2019.

[37] P. T. Inc. (2015) Collaborative Data Science. Montreal, QC. [Online]. Available: https://plot.ly

[38] J. Pfau, A. T. Young, J. Wei, M. L. Wei, and M. J. Keiser, "Robust Semantic Interpretability: Revisiting Concept Activation Vectors," *CoRR*, vol. abs/2104.02768, 2021.

VERSE VERSUS CHORUS: STRUCTURE-AWARE FEATURE EXTRACTION FOR LYRICS-BASED GENRE RECOGNITION

Maximilian Mayerl[1] **Stefan Brandl**[2,3] **Günther Specht**[1]
Markus Schedl[2,3] **Eva Zangerle**[1]
[1] Department of Computer Science, Leopold-Franzens-Universität Innsbruck, Austria
[2] Institute of Computational Perception, Johannes Kepler Universität Linz, Austria
[3] Human-centered AI Group, AI Lab, Linz Institute of Technology (LIT), Austria
[1]{firstname.lastname}@uibk.ac.at [2,3]{firstname.lastname}@jku.at

ABSTRACT

Lyrics-based genre recognition aims to automatically determine the genre of a given song based on its lyrics. Previous approaches for this task have commonly used textual features extracted from the entirety of a song's lyrics, neglecting the inherent structure of lyrics consisting of, for instance, verses and choruses. Therefore, we pose the hypothesis that features extracted from different parts of the lyrics can have significantly different predictive power. To test this hypothesis, we perform a series of experiments to determine whether models trained on features taken from verses and choruses perform differently for genre recognition. Our experiments indeed confirm our hypothesis, showing that generally, using features extracted from verses leads to higher performance than features extracted from choruses. Digging deeper, we found that this is especially true for *pop* and *rap* songs. *Rock* songs show the opposite effect, with features extracted from choruses performing better than those taken from verses.

1. INTRODUCTION AND RELATED WORK

In music information retrieval, genre recognition (also known as genre prediction or genre classification) is the task of automatically classifying the genre of a given song by assigning it to one or more predefined genres. This has a variety of applications, including the organization of music collections into easily recognizable categories, or using genre information as a feature in recommender systems. Over the years, many approaches have been developed for this task, most of which focus on audio-based genre recognition, i.e., using information extracted from the song's audio signal [1]. Less, but still substantial, work has also been done on lyrics-based approaches, which use features extracted from a song's lyrics (e.g. [2–5]). Lastly, hybrid approaches combining both audio and lyrics information have also been proposed (e.g., [6,7]), and it has been shown that audio and lyrics features are complementary and that using both together can lead to increased model performance. The approaches making use of lyrics information cover a wide range of feature types as well as underlying machine learning models.

For instance, Neumayer and Rauber [6] explored using lyrics features in conjunction with low-level audio features to detect the genre of songs. For the lyrics, they extracted feature vectors using the popular bag-of-words model and weighted the words using *tf-idf* weighting. They then performed classification via support vector machines. Mayer et al. [2] relied only on lyrics and extracted *tf-idf* weighted bag-of-words features, rhyme, and part-of-speech features as well as general statistical text descriptors for their approach. These features were then used to perform genre recognition using different machine learning models. Ying et al. [3] used information of the parts-of-speech used in a song's lyrics to recognize both genre and mood of a song. They used these features to train and evaluate three different machine learning models, namely support vector machines, k-nearest neighbor, and naive Bayes.

What these, and most other, lyrics-based approaches have in common is that they treat a song's lyrics as a uniform document, extracting features from the entirety of the song's lyrics. However, in reality, song lyrics have a structure. They consist of different parts, which can be divided into categories such as *intro*, *verse*, *chorus*, *bridge*, or *outro*. Leveraging such structural information, Tsaptsinos [4] proposed using a hierarchical neural network model using an attention mechanism. To the best of our knowledge, this is the only work that directly seeks to exploit song structure for genre recognition, making use of the hierarchical structure of song lyrics (words forming lines, lines forming segments, and segments, in turn, forming the song). Their results show that their model, which due to its attention mechanism can automatically learn on which parts of a song it should focus, outperforms existing approaches that extract and use features uniformly across the whole song. This shows that the location of features within a song plays an important role in genre recognition. However, their model applies attention at the word, line, and segment level only, and does not incorporate higher structural elements like *verse* or *chorus*. Fell

© Maximilian Mayerl, Stefan Brandl, Günther Specht, Markus Schedl, Eva Zangerle. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Maximilian Mayerl, Stefan Brandl, Günther Specht, Markus Schedl, Eva Zangerle, "Verse Versus Chorus: Structure-aware Feature Extraction for Lyrics-based Genre Recognition", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

and Sporleder [5] used a variety of textual features for three distinct music classification tasks, including genre recognition. The features they employed include features describing information about a song's structure, like the number of repeated segments in a song and whether the song contains a chorus. Their results for genre recognition suggest that including such features can increase genre recognition performance, and therefore that song structure plays a role in determining a song's genre.

Inspired by those results, we formulate the following hypothesis: The structure of a song's lyrics plays a substantial role in genre recognition, and extracting features from different parts of the lyrics can lead to a significantly different performance of genre recognition models. To this end, we make the following contributions: (1) We construct a dataset of lyrics that contains the required information about lyrics' structure; (2) We perform a series of experiments, covering both feature sets as well as machine learning algorithms used in the past for genre recognition, and show that there indeed exists a significant difference in predictive performance between features extract from the verses of a song as opposed to the same features extracted from the choruses; and (3) We examine how that difference in performance depends on the concrete machine learning algorithm and feature set used, as well as on the genre.

The remainder of this paper is structured as follows. Section 2 explains how we created the dataset required for our experiments. In Section 3, we provide a detailed overview of our experiments. Following that, we discuss our results in Section 4 and finally provide a summary and outlook to future work in Section 5.

2. DATASET

To investigate the impact on classification performance of features from different structural elements of lyrics, we require a dataset that contains both lyrics data with structure information (i.e., information on which structural part of the song— verse, chorus, etc.—any particular line in the lyrics belong to) as well as genre tags. Since no such dataset is publicly available, we describe the creation of such a dataset in the following and subsequently, provide a statistical description of its contents.

2.1 Dataset Creation

We used the popular LFM-2b dataset [8] to obtain an initial list of songs. Using this list of songs, we then obtained lyrics data from *genius.com*. We chose this source because it not only provides lyrics for a large number of songs, but it also has an active community of users who annotate lyrics with structure information. In addition to lyrics, they also provide tags for many of the songs in their database. We use the primary tag as given by *genius.com* as our genre tag, and then filtered the list of songs such that only songs with one of the top five most occurring genre tags — *pop*, *rock*, *country*, *rap*, and *r&b* — remained. These steps provided us with an initial dataset of 2,135,504 songs with both genre and lyrics information.

Property	Value
Number of songs	295,416
Number of artists	39,357
Number of tokens in all choruses	193,696,032
Number of tokens in all verses	195,567,571
Average number of choruses per song	3.46
Average number of verses per song	2.37

Table 1: Summary statistics of our dataset.

We then performed a series of cleaning and transformation steps on this initial dataset. First, we expanded repeating parts of the lyrics that were given in short form; i.e., lyrics on *genius.com* frequently use annotations like *[x2]* to indicate that a given part (paragraph or line) in the lyrics should be repeated. We removed those repeating annotations and duplicated the corresponding parts in the lyrics text accordingly. Following that, we removed all other annotations that do not correspond to structural parts of the lyrics; for instance, instrument annotations or singer information. In the next step, we used `langdetect` version `1.0.9` as well as `polyglot` version `16.7.4` to remove all songs with non-English lyrics from the dataset. Finally, since the dataset at that point contained duplicates—i.e., songs with identical lyrics—we removed those. These duplicates exist because LFM-2b sometimes contains multiple versions of the same song, like covers by a different artist or versions recorded during live events. To decide which song among a given set of duplicates to keep, we used the number of listening events according to LFM-2b and kept the copy with the highest listening count. We chose this approach since the copy with the highest listening count is also most likely to have the most complete set of genre tags.

Following these cleaning and transformation steps, we split the lyrics of each song into its constituent structure parts. For this, we used the structure annotations provided by *genius.com*. These annotations specify which part of the song the subsequent text belongs to, with that part extending until the next structure annotation or until the end of the song. They often also give some additional information, like who sings the given part or which number the part has in the song. Examples of such annotations include *[Chorus]* or *[Verse 1: Austin Brown]*. For our splitting, we considered the following set of structural annotations: *verse*, *chorus*, *intro*, *outro*, *bridge*, *hook*, *refrain*, *interlude*, and *drop*. We also combined *chorus* and *refrain* and mapped both of those annotations to *chorus*. For our subsequent experiments, we considered only the *verse* and *chorus* parts of the songs in the dataset, but we retained the other parts for potential future work.

Subsequently, we removed all songs that did not consist of at least two parts. This gave us an intermediate dataset consisting of 416,945 songs. Finally, since our experiments focus on *verse* and *chorus*, we created a final dataset containing only those songs which have at least one verse and at least one chorus, yielding a total of 295,416 songs.

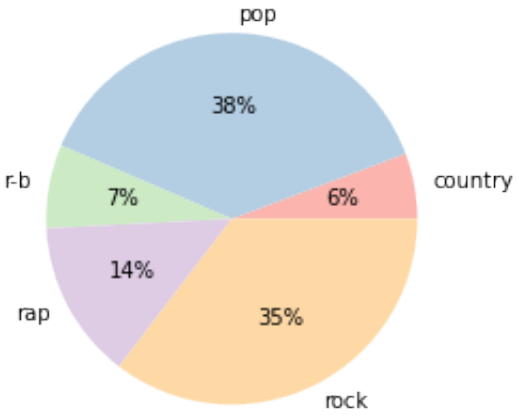


Figure 1: Genre distribution for primary genre.

2.2 Dataset Statistics

The final dataset consists of 295,416 songs created by 39,357 different artists. A summary of the dataset contents is given in Table 1. The total number of tokens (i.e., unigrams) contained in the choruses and verses for all songs is 193,696,032 and 195,567,571, respectively. This means that the amount of textual content for both chorus and verse is mostly balanced, ensuring that any difference in performance we may observe between using only chorus or verse does not stem from imbalanced training data. The distribution of primary genre labels in the dataset is given in Figure 1.

3. EXPERIMENTAL SETUP

The goal of this work is to determine whether there exists a difference in the predictive power between features extracted from different structural parts of a song's lyrics. For this, we carried out a series of experiments using different sets of textual features and different classification algorithms, each trained and evaluated once on only the *verse* and once only on the *chorus* parts of the songs in our dataset.

As explained in Section 2, we limited our final dataset to only songs which contain both at least one *chorus* and one *verse* part. This was done because, to get comparable results, we need to perform all our experiments on the same set of songs. As *chorus* and *verse* are the most common parts of a song, limiting our experiments to these two parts ensures that we have a dataset of sufficient size left to train and evaluate our models.

3.1 Feature Sets

Our experiments were performed using ten different sets of textual features commonly used in the literature (e.g., [2, 5–7, 9]). Those features together capture a wide variety of information about the lyrics, including content, grammatical structure, sound structure as well as complexity. Before using them in our experiments, we standardized them by scaling them to unit variance. We describe the feature sets used in the following:

- **Bag-of-words**: Classic bag-of-words features, counting the occurrences of word sequences of a given length, also known as n-grams, in the lyrics. In our experiments, we use unigrams, bigrams, trigrams, as well as (1,3)-grams, which is a combination of all word sequences of lengths one to three. To limit memory consumption of the resulting feature vectors, we only used the top 20,000 most common sequences appearing in the data. We also only consider sequences that appeared in at most 80% of all songs. This was done to filter out very common terms like function words. Such bag-of-words features have been frequently used in lyrics-based music classification, including for genre recognition, before (e.g. [2, 5–7, 9]).

- **Rhyme features**: A set of nine features describing the rhymes and alliterations present in the lyrics. The rhyme features (numbers of couplets, clerihews, alternating rhymes, and nested rhymes, percentage of rhymes in the text, and the number of unique rhyme words) were originally used by Mayer et al. [2] for lyrics-based genre recognition. Since alliterations play an important role in rap lyrics [10], we additionally added features describing alliterations (numbers of alliterations of length two, three, and four or longer).

- **Readability features**: A set of 13 readability metrics: Flesch reading easy, SMOG index, Flesch-Kincaid grade, automated readability index, Coleman-Liau index, Dale-Chall readability score, Linsear Write score, Gunning Fog index, Fernandez-Huerta score, Szigriszt-Pazos score, Gutierrez-Polini score, Crawford score, and the number of difficult words (i.e., words that are not on the Dale-Chall list of easy words). Readability features have been used for genre classification on normal texts, e.g. by Falkenjack et al. [11]. We included them in our experiments since we expect that they also carry useful information for lyrics.

- **Lexical features**: A set of 32 general lexical features, including token count, character count, repeated token ratio, number of unique tokens per line, average token length, average number of tokens per line, line count, unique line count, blank line count, blank line ratio, repeated line ratio, counts for specific symbols (exclamation mark, question mark, colon, etc.), count of digits, ratio of punctuation to the whole text, stop word count, stop word ratio, and hapax/dis/tris legomenon ratios. Different combinations of such lexical features have frequently been used for genre recognition (e.g., [2, 7, 9]).

- **Lexical diversity features**: A set of five lexical diversity metrics: measure of textual lexical diversity (MTLD), Herdan's C, Summer's S, Dugast's U and Maas' a.

- **Part-of-speech features**: A set of five features measuring the frequency of specific parts-of-speech within the lyrics: pronoun, adjective, adverb, noun, and verb frequency. Features describing the distribution of specific parts-of-speech within lyrics have been used for genre recognition for example by Mayer et al. [2], Mayer and Rauber [7], and Fell and Sporleder [5].

- **Morphological features**: A set of six features describing morphological properties of the lyrics: past tense ratio, -ing form ratio, comparative and superlative ratios for adjectives, and comparative and superlative ratios for adverbs. The ratio of verbs in past tense has been used for genre recognition before by Fell and Sporleder [5]. Since this was shown to be a valuable feature, we added the other five features to capture additional information about the morphological properties of a song's lyrics.

Since our goal is to determine potential differences between *verse* and *chorus* for individual feature types, we did not include a combination of all features in our experiments. It has already been shown before that different textual features carry orthogonal information, and combining them leads to increased performance [9].

3.2 Classification Algorithms

We repeat our experiments with different classification models to ensure that any difference in performance we measure between features extracted from different parts of a song is not only due to a specific property of any of the classification algorithms. Concretely, we chose the following algorithms: random forests, support vector machines, and feed-forward neural networks. All of these models are widely used in the machine learning community and have been shown to work well on many different tasks, including specifically genre recognition (e.g. [2–6, 9]).

For all three algorithms, we used the implementations provided by scikit-learn [1] version `1.0.2`. For support vector machines, we chose scikit-learn's `LinearSVC`, which uses LIBLINEAR [12], and followed the recommendation in the sklearn documentation to set `dual=False`, since in our case the number of training samples is significantly higher than the number of features. All other parameters for the used algorithms were left at their default values (100 estimators and Gini impurity for `RandomForstClassifier` and one hidden layer with 100 neurons and ReLU activation for `MLPClassifier`). Note that we did not perform a grid search to find the best hyperparameters, since our goal is only the determine the difference between features extracted from verses and choruses, not to get the best possible performance from the models.

[1] `https://scikit-learn.org/`

4. RESULTS AND DISCUSSION

We investigated every combination of song part (chorus or verse), feature set, and machine learning algorithm. Training and evaluation were done using 5-fold cross-validation. For the cross-validation, the data was shuffled before splitting it into folds. Feature extraction was done completely separately for the different song parts. For models trained and evaluated on the songs' verses, feature extraction only considered text located within the verses of the songs in the dataset, and equivalently for models trained and evaluated on the songs' choruses.

For evaluation, we chose the F_1 score to quantify the performance of each model. Since our dataset is imbalanced in terms of genres, we will focus our discussion on the macro-averaged F_1 scores, so as to not over-weigh the importance of the most common genres. However, for the sake of full transparency, we also report the micro-averaged F_1 scores for each model. Since the goal of our experiments is to determine whether there exists a performance difference between models trained on choruses and verses, we also performed statistical significance tests whenever our results indicated that the model trained on choruses performed better than the corresponding model trained on verses, or vice versa. For this, we employed a one-sided t-test ($p = .01$) on the scores of the individual cross-validation folds of both models, using the alternative hypothesis that the better overall score was indeed greater than the lower score.

We provide the results obtained in Table 2. We first take a look at the general performance of the models and feature sets. As is expected, we observe a substantial variance in performance for different machine learning models and features. The overall best performance in terms of the macro-averaged F_1 score is achieved by the support vector machine model trained on (1,3)-gram features extracted from the songs' verses, with an F_1 score of 0.5108. The best performance of the random forest and neural network model, respectively, was 0.4296 and 0.5082, both also for (1,3)-gram features trained on verses. The ten feature sets also exhibit substantial variability in performance across different machine learning models. This is especially true for part-of-speech features as well as morphological features. These two feature sets show a significantly lower performance when used with a support vector machine as opposed to random forests or neural networks. Looking at the higher score, obtained for the models trained and evaluated on the songs' verses, part-of-speech features achieved a score of 0.1872 against 0.2970 and 0.2888, and morphological features achieved a score of 0.1694 against 0.3154 and 0.2914.

Next, we examine the difference in performance between models trained and evaluated on choruses and verses, respectively. The results draw a clear picture: models trained and evaluated on verses consistently outperform the equivalent models trained and evaluated on choruses. We observe a difference in performance for every single combination of machine learning algorithm and feature set, at least in terms of the macro-averaged F_1 score. The

Feature Set	Random Forest Chorus	Random Forest Verse	Random Forest Diff.	Support Vector Machine Chorus	Support Vector Machine Verse	Support Vector Machine Diff.	Neural Network Chorus	Neural Network Verse	Neural Network Diff.
F_1 macro									
unigrams	.377 (±.002)	.428 (±.002)	.0506†	.414 (±.002)	.502 (±.002)	.0880†	.419 (±.003)	.505 (±.003)	.0858†
bigrams	.364 (±.000)	.416 (±.003)	.0522†	.396 (±.002)	.485 (±.001)	.0888†	.396 (±.003)	.479 (±.005)	.0824†
trigrams	.328 (±.002)	.385 (±.001)	.0562†	.333 (±.002)	.422 (±.001)	.0894†	.342 (±.003)	.419 (±.003)	.0772†
(1,3)-grams	.379 (±.002)	.430 (±.002)	.0502†	.432 (±.002)	.511 (±.002)	.0792†	.424 (±.002)	.508 (±.003)	.0846†
rhyme	.254 (±.002)	.318 (±.002)	.0638†	.200 (±.001)	.280 (±.001)	.0796†	.230 (±.004)	.307 (±.009)	.0776†
readability	.263 (±.001)	.371 (±.002)	.1078†	.224 (±.001)	.355 (±.001)	**.1308**†	.264 (±.006)	.360 (±.002)	.0964†
lexical	.359 (±.002)	.400 (±.002)	.0412†	.291 (±.002)	.363 (±.001)	.0720†	.360 (±.004)	.403 (±.001)	.0430†
lexical diversity	.253 (±.002)	.351 (±.001)	.0980†	.195 (±.000)	.319 (±.000)	.1242†	.245 (±.002)	.333 (±.001)	.0878†
part-of-speech	.223 (±.001)	.297 (±.001)	.0738†	.180 (±.000)	.187 (±.001)	.0068†	.207 (±.005)	.289 (±.002)	.0816†
morphological	.201 (±.002)	.315 (±.001)	**.1146**†	.164 (±.002)	.169 (±.000)	.0056†	.181 (±.003)	.291 (±.005)	**.1100**†
F_1 micro									
unigrams	.538 (±.002)	.577 (±.003)	.0396†	.526 (±.002)	.566 (±.002)	.0402†	.484 (±.003)	.548 (±.003)	.0644†
bigrams	.516 (±.000)	.559 (±.003)	.0432†	.504 (±.001)	.548 (±.000)	.0444†	.472 (±.003)	.532 (±.004)	.0602†
trigrams	.462 (±.001)	.516 (±.003)	.0534†	.460 (±.001)	.509 (±.001)	.0486†	.434 (±.004)	.490 (±.002)	.0566†
(1,3)-grams	.541 (±.002)	.581 (±.001)	.0402†	.525 (±.002)	.567 (±.002)	.0420†	.492 (±.003)	.552 (±.003)	.0598†
rhyme	.409 (±.002)	.449 (±.002)	.0404†	.425 (±.001)	.456 (±.002)	.0308†	.433 (±.002)	.461 (±.003)	.0286†
readability	.422 (±.001)	.499 (±.001)	**.0772**†	.439 (±.001)	.511 (±.002)	**.0716**†	.453 (±.001)	.513 (±.002)	.0596†
lexical	.486 (±.003)	.525 (±.001)	.0388†	.473 (±.003)	.519 (±.001)	.0464†	.493 (±.002)	.526 (±.001)	.0332†
lexical diversity	.363 (±.001)	.434 (±.002)	.0712†	.425 (±.002)	.478 (±.001)	.0522†	.376 (±.002)	.482 (±.002)	**.1068**†
part-of-speech	.392 (±.001)	.440 (±.001)	.0482†	.400 (±.001)	.406 (±.001)	.0054†	.408 (±.002)	.448 (±.002)	.0400†
morphological	.385 (±.001)	.435 (±.001)	.0502†	.388 (±.002)	.388 (±.001)	.0000	.396 (±.002)	.443 (±.002)	.0468†

Table 2: Summary of the results of our experiments. For all numbers, leading zeros were omitted for space reasons. *Diff.* is the change in F_1 score between verse and chorus. Numbers in parentheses are the standard deviation between the five cross-validation folds. Bold numbers indicate the highest and underlined numbers to lowest amount of change for each combination of feature set and machine learning algorithm. † indicates that the difference in performance is statistically significant.

micro-averaged F_1 scores show almost the same picture, with one notable exception: morphological features used with a support vector machine achieved the same score for *chorus* and *verse*. The exact difference in performance again varies depending on the machine learning algorithm and feature set. The biggest absolute difference in terms of macro-averaged scores is seen with support vector machines using readability features. Trained and evaluated on choruses, this model achieves an F_1 score of 0.2244, as opposed to a score of 0.3552 for verses. This constitutes an absolute change in score of 0.1308. The biggest relative difference is also shown by support vector machines, but for lexical diversity features. For this model, the score for verses is 0.3188 as opposed to 0.1946 for choruses, for a change of 0.1242, which is a relative change of 63.82%. The lowest difference in terms of macro-averaged F_1 scores is observed for morphological features used with support vector machines. The difference between *chorus* and *verse* for this model is 0.0056, which is also the lowest relative difference of 3.42%. For micro-averaged F_1 scores, the lowest change is 0, as mentioned before, also for morphological features using support vector machines.

To get a better sense of how the difference in performance depends on the used machine learning algorithm or feature set, we computed the average difference in macro-averaged F_1 scores between *chorus* and *verse* along both of these dimensions. The results for this are given in Table 3. We can observe from these that the performance difference

Average over ...	Average Difference
Machine Learning Algorithms	
Random Forest	0.0708
Support Vector Machine	0.0764
Neural Network	**0.0826**
Feature Sets	
unigrams	0.0748
bigrams	0.0745
trigrams	0.0743
(1,3)-grams	0.0713
rhyme	0.0737
readability	**0.1117**
lexical	0.0521
lexical diversity	0.1033
part-of-speech	0.0541
morphological	0.0767

Table 3: Average amount of change in the macro-averaged F_1 score between *chorus* and *verse*, averaged over machine learning algorithm and feature set. Bold numbers indicate the highest average differences.

	readability			lexical		
Genre	RF	SVM	NN	RF	SVM	NN
country	0.0054	0.0000	0.0011	-0.0448†	0.0001	-0.0076
pop	0.0298†	0.0367†	0.0360†	0.0110†	0.0271†	0.0193
r&b	0.0048	0.0000	-0.0021†	0.0271†	-0.0113†	-0.0583†
rap	0.4995†	0.6587†	0.4878†	0.3009†	0.3820†	0.2950†
rock	-0.0001	-0.0402†	-0.0412†	-0.0154†	-0.0383†	-0.0383

Table 4: Performance differences for the feature sets *readability* and *lexical* for individual genres in terms of F_1 scores. The values in the table are computed as the differences between chorus and verse, with positive numbers indicating that models trained on verses performed better, and negative numbers indicating that models trained on choruses performed better. † indicates that the difference in performance is statistically significant. Abbreviations: RF = random forest, SVM = support vector machine, NN = neural network.

does not seem to vary much with the machine learning algorithm. The biggest difference here is obtained by neural networks, with an average of 0.0826, while the smallest change is seen for random forests, with an average of 0.0708. In contrast, performance differences between feature sets are more substantial. The biggest average score difference there is produced by *readability*, with an average of 0.1117, while the smallest difference is seen for *lexical*, with an average of 0.0521.

Those results confirm our initial hypothesis: The predictive performance of genre recognition models is indeed significantly different depending on which parts of a song we extract the features from. Concretely, results show that features extracted from the verses of songs perform significantly better than those extracted from the choruses. As mentioned in Section 2, the amount of textual content in choruses and verses is almost identical for our dataset. We can therefore rule out that this difference is caused by different amounts of training data for the model. We also conducted our experiments with various machine learning algorithms and feature sets, showing that the changes in performance are also not due to any particular properties of specific features or algorithms. We can therefore conclude that these differences are caused by a more fundamental difference in information content between choruses and verses.

Finally, we aimed to investigate how this observed difference in performance depends on the concrete genre. To this end, we took the feature sets with the biggest and smallest difference — *readability* and *lexical* — and computed per-genre F_1 scores for them, again for the same three machine learning algorithms. We then, as before, computed the difference in performance between verse and chorus. These per-genre differences are given in Table 4. In this table, positive numbers indicate that the model trained on verses performed better, while negative numbers indicate that the models trained on choruses performed better.

Looking at these results, we can see a varied picture of the different genres. For *country*, we observe that the differences change between positive and negative, which indicates no clear tendency for whether verses or chorus perform better for this genre. The differences for countries are also not statistically significant, with one exception (random forests using *lexical* features). For *r&b*, there is also no clear tendency, with differences likewise varying between positive and negative. For *pop*, we consistently see better performance when using verse features, ranging from 0.011 to 0.0367, with five of the six observed differences being statistically significant. *Rap* is the genre for which we observe by far the largest difference in performance. Features extracted from verses clearly outperform those extracted from choruses for this genre, with differences between 0.295 and 0.6587, all of which are significant. Finally, for *rock*, we find the opposite behavior, with features extracted from choruses outperforming those extracted from verses. The statistically significant differences here range from -0.0154 to -0.0412.

5. CONCLUSION AND OUTLOOK

Building on earlier results that the structure of lyrics may play a role in genre recognition, we formulated the hypothesis that features extracted from different structural parts of a song lead to significant differences in predictive performance for genre recognition models. Through a series of experiments, we confirmed this hypothesis and showed that, depending on genre, features extracted from songs' verses can perform better than those extracted from choruses. Looking closer at how performance varies with the concrete genre, we found that this is especially true for the genres *pop* and *rap*. *Rock*, on the other hand, exhibits the opposite behavior, with classifiers using features extracted from choruses achieving better accuracy than those using features extracted from verses.

In future work, we will investigate whether we can find similar differences for other lyrics-based MIR tasks, like mood recognition or popularity prediction. Additionally, the observed differences might be used to construct better features or classification models which exploit the structure of song lyrics. Finally, we plan to incorporate audio-based features extracted from verses and choruses, respectively, and investigate whether those show similar behavior and whether they can complement lyrics-based features in a multi-modal classification setup.

6. REFERENCES

[1] B. L. Sturm, "A survey of evaluation in music genre recognition," in *International Workshop on Adaptive Multimedia Retrieval*. Springer, 2012, pp. 29–66.

[2] R. Mayer, R. Neumayer, and A. Rauber, "Rhyme and style features for musical genre classification by song lyrics." in *Proceedings of the International Conference on Music Information Retrieval*, 2008, pp. 337–342.

[3] T. C. Ying, S. Doraisamy, and L. N. Abdullah, "Genre and mood classification using lyric features," in *International Conference on Information Retrieval & Knowledge Management*. IEEE, 2012, pp. 260–263.

[4] A. Tsaptsinos, "Lyrics-Based Music Genre Classification Using a Hierarchical Attention Network." in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, Oct. 2017, pp. 694–701.

[5] M. Fell and C. Sporleder, "Lyrics-based analysis and classification of music," in *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 2014, pp. 620–631.

[6] R. Neumayer and A. Rauber, "Integration of text and audio features for genre classification in music information retrieval," in *European Conference on Information Retrieval*. Springer, 2007, pp. 724–727.

[7] R. Mayer and A. Rauber, "Musical genre classification by ensembles of audio and lyrics features," in *Proceedings of the International Conference on Music Information Retrieval*, 2011, pp. 675–680.

[8] A. B. Melchiorre, N. Rekabsaz, E. Parada-Cabaleiro, S. Brandl, O. Lesota, and M. Schedl, "Investigating gender fairness of recommendation algorithms in the music domain," *Information Processing & Management*, vol. 58, no. 5, p. 102666, 2021.

[9] M. Mayerl, M. Vötter, M. Moosleitner, and E. Zangerle, "Comparing lyrics features for genre recognition," in *Proceedings of the 1st Workshop on NLP for Music and Audio (NLP4MusA)*, 2020, pp. 73–77.

[10] C. Coscarello, "The word out: A stylistic analysis of rap music," Master's thesis, 2003.

[11] J. Falkenjack, M. Santini, and A. Jönsson, "An exploratory study on genre classification using readability features," in *Proceedings of the Sixth Swedish Language Technology Conference (SLTC 2016), Umeå, Sweden*, 2016.

[12] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.

TRANSFER LEARNING OF WAV2VEC 2.0 FOR AUTOMATIC LYRIC TRANSCRIPTION

Longshen Ou* **Xiangming Gu*** **Ye Wang**
School of Computing, National University of Singapore
{longshen, xiangming, wangye}@comp.nus.edu.sg
*Both authors contributed equally to this research.

ABSTRACT

Automatic speech recognition (ASR) has progressed significantly in recent years due to the emergence of large-scale datasets and the self-supervised learning (SSL) paradigm. However, as its counterpart problem in the singing domain, the development of automatic lyric transcription (ALT) suffers from limited data and degraded intelligibility of sung lyrics. To fill in the performance gap between ALT and ASR, we attempt to exploit the similarities between speech and singing. In this work, we propose a transfer-learning-based ALT solution that takes advantage of these similarities by adapting wav2vec 2.0, an SSL ASR model, to the singing domain. We maximize the effectiveness of transfer learning by exploring the influence of different transfer starting points. We further enhance the performance by extending the original CTC model to a hybrid CTC/attention model. Our method surpasses previous approaches by a large margin on various ALT benchmark datasets. Further experiments show that, with even a tiny proportion of training data, our method still achieves competitive performance.

1. INTRODUCTION

Automatic lyric transcription (ALT) systems allow for lyrics to be obtained from large musical datasets without requiring laborious manual transcription. These lyrics can then be used for many music information retrieval (MIR) tasks, including query by singing [1], audio indexing [2], etc. Besides, because lyric alignment systems are typically built upon ALT models [3, 4], a strong-performing ALT model can lay a solid foundation for better audio-to-text alignment performance. Consequently, ALT is becoming an increasingly active topic in the recent MIR community.

One option for improving ALT performance is to incorporate knowledge obtained from studies involving the transcription of speech. Indeed, ALT is usually treated as a separate problem from automatic speech recognition (ASR), e.g., [3,5–8], eschewing large-scale speech datasets and well-developed ASR systems. However, the absence of large-scale singing datasets has impeded the construction of high-performing ALT models. While there are distinctions between sung and spoken language, e.g., sung language being less intelligible and hence harder to recognize [5, 9], they share many similarities, such as having the same vocabularies and being produced by similar physical mechanisms. Therefore, we believe it is worth investigating whether we can use knowledge and datasets from the speech domain to compensate for the inadequacy of singing datasets and bolster the performance of ALT systems.

Transfer learning methods have been found to effectively alleviate the requirement for a large amount of training data for some low-resource tasks [10,11]. For example, speech recognition for non-native speakers [12], and machine translation for low-resource languages [13]. In such scenarios, transfer learning helps mitigate the problem of insufficient data by adapting data and knowledge from related high-resource tasks or domains.

In recent years, self-supervised learning (SSL) has become a new paradigm in ASR research. Several SSL methods can perform excellently with access to only a few hours or even a few minutes of labeled data [14–16]. Among them, wav2vec 2.0 [16] has been shown to be a particularly promising model for transfer learning [12]. wav2vec 2.0 is an effective few-shot learner that only requires a small amount of data from the target domain or problem to achieve impressive results [17, 18]. This property makes wav2vec 2.0 a promising candidate to help ALT systems overcome the issue of limited training data by transferring speech representation knowledge to the singing domain.

The contributions of this paper contain four aspects:

- We propose an ALT solution that takes advantage of the similarities between spoken and singing voices. This is achieved by performing transfer learning using wav2vec 2.0 on singing data after pretraining and finetuning on speech data.
- We maximize the effectiveness of transfer learning by exploring the influence of different transfer starting points. We show that both pretraining and finetuning on speech data contribute to the high performance of our ALT system.
- We further enhance the system's performance by

© Longshen Ou, Xiangming Gu, Ye Wang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Longshen Ou, Xiangming Gu, Ye Wang, "Transfer Learning of wav2vec 2.0 for Automatic Lyric Transcription", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

extending the original connectionist temporal classification (CTC) model to a hybrid CTC/attention model for better convergence and more accurate decoding.

- Our method surpasses previous ones on various benchmark ALT datasets, including DSing [5], DALI [19, 20], Jamendo [21], Hansen [22], and Mauch [23], by about 25% relative WER reduction on average. We further show that with less than one-tenth of labeled singing data, our method can still achieve state-of-the-art results on the test split of DSing, demonstrating its effectiveness in low-resource ALT setups.

2. RELATED WORK

2.1 Automatic Lyric Transcription

Recent progress in lyric transcription has been mainly driven by three factors. First, the construction and curation of datasets containing aligned audio and lyrics, including DAMP Sing! 300x30x2 [5, 24] and DALI [19, 20], lay the foundation for data-driven ALT models. Second, the design of ALT acoustic models benefits from architectures of automatic speech recognition (ASR) models and can be further improved by adopting singing domain knowledge as inductive bias. Representative work includes TDNN-F with its variants [3,5–7] and vanilla/convolution-augmented Transformers [8, 25, 26]. Additionally, [27] proposed to leverage the complementary information of additional modalities (video and wearable IMU sensors) for ALT systems. Third, through data augmentation methods such as adjusting speech data to make it more "song-like" [28] or synthesizing singing voice from speech voice [25,26], more training data can be created for ALT models, thus alleviating the data sparsity problem.

2.2 Self-supervised speech representation learning

The success of deep learning methods is highly related to the power of the learned representations. Although supervised learning still dominates the speech representation learning field, it has several drawbacks. For example, substantial amounts of labeled data are required to train supervised learning ASR models [15, 16]. Moreover, representations obtained through supervised learning tend to be biased to specific problems, thus are difficult to extend to other applications [29].

To mitigate the above problems, a series of self-supervised learning (SSL) frameworks for speech representation learning have emerged, e.g., Autoregressive Predictive Coding (APC) [30], Contrastive Predictive Coding (CPC) [15], and Masked Predictive Coding (MPC) [14, 31, 32]. Moreover, wav2vec 2.0 takes advantage of both CPC and MPC to conduct self-supervised learning and has become the new paradigm for the ASR task [16]. wav2vec 2.0 has been also widely adopted as the feature extractor for other speech-related applications, e.g. speech emotion recognition [33, 34], keyword spotting [35], speaker verification and language identification [36], demonstrating that speech representations learned from wav2vec 2.0 are robust and transferable for downstream tasks.

3. METHODOLOGY

In this section, we firstly recap the structure of wav2vec 2.0 [16]. Then we elaborate on the three training stages of the proposed methods, including pretraining on speech data, finetuning on speech data, and transferring to the singing domain.

3.1 Structure of wav2vec 2.0

As shown in Fig. 1, wav2vec 2.0 is built with a CNN-based feature encoder, a Transformer-based context network, and a quantization module. For raw audio inputs $\boldsymbol{x}$ with a sampling rate of 16 kHz, the feature encoder accepts $\boldsymbol{x}$ and obtains the latent speech representations $\boldsymbol{z}$. The feature encoder has seven blocks, each of which includes a 1D temporal convolution with 512 channels followed by layer normalization [37] and GELU activation [38]. Consequently, $\boldsymbol{z} \in \mathcal{R}^{T\times 1024}$ are 2D representations with a frequency of 49 Hz. To exploit the temporal relationship among different frames of latent representations, $\boldsymbol{z}$ are further fed into the context network, which is parameterized by 12 Transformer blocks [39]. Each block has a multi-head attention module with 16 attention heads and a Feed-Forward Network (FFN) with 4,096 hidden dimensions. Resulting context representations $\boldsymbol{c} \in \mathcal{R}^{T\times 1024}$ are the features extracted from the audio signal and used for downstream tasks.

In addition to being fed into the context network, $\boldsymbol{z}$ are also accepted by a quantization module, which learns quantized speech representations $\boldsymbol{q}$, thus facilitating the self-supervised training.

3.2 Stage I: Pretraining on speech data

wav2vec 2.0 is pretrained through an SSL method [16] on large-scale unlabeled speech data, as displayed in Fig. 1(a). Before latent representations $\boldsymbol{z}$ are fed into the context network, several consecutive frame sequences are randomly masked. The masked frames are replaced by a trainable vector. wav2vec 2.0 is trained by optimizing the combination of contrastive loss and diversity loss $\mathcal{L}_m + \alpha\mathcal{L}_d$ (α refers to a balancing hyper-parameter). The contrastive objective is defined as:

$$\mathcal{L}_m = -\log \frac{\exp(sim(\boldsymbol{c}_t, \boldsymbol{q}_t)/\kappa)}{\sum_{\widetilde{\boldsymbol{q}}\sim \boldsymbol{Q}_t} \exp(sim(\boldsymbol{c}_t, \widetilde{\boldsymbol{q}}_t)/\kappa)} \quad (1)$$

where $\boldsymbol{c}_t$ is t-th frame of context representations, $\boldsymbol{Q}_t$ represents all possible quantized representations, the temperature value κ is set as 0.1 and sim refers to the cosine similarity. The diversity loss $\mathcal{L}_d$ is designed to encourage the usage of all entries in codebooks. We refer readers to [16] for more details.

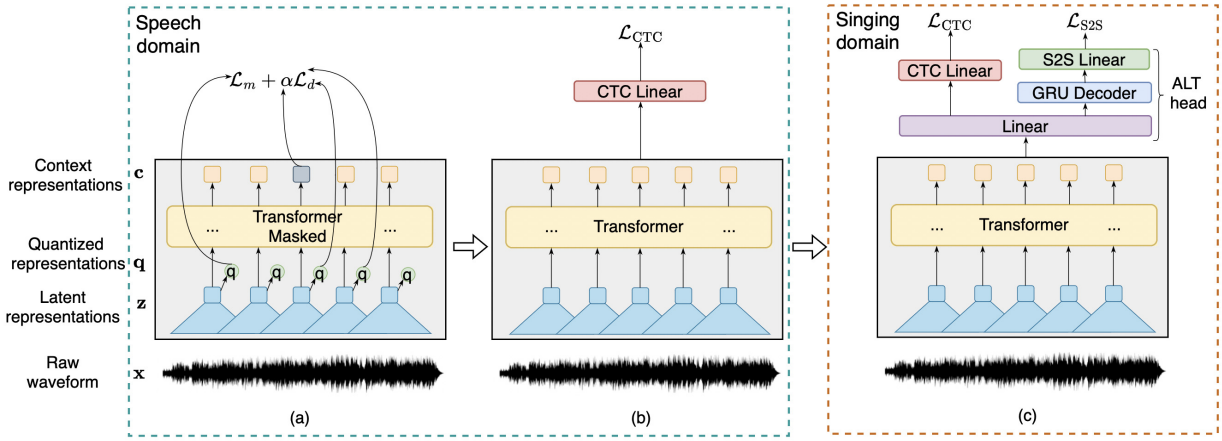


Figure 1. An overview of our training framework. (a) Stage I: Pretraining on speech data. (b) Stage II: Finetuning on speech data. (c) Stage III: Transferring on singing data.

3.3 Stage II: Finetuning on speech data

The process of finetuning requires labeled speech data. As shown in Fig. 1(b), the quantization module in wav2vec 2.0 is disabled since it is only used in Stage I, and a linear layer (CTC linear) is added on top of the Transformer. The whole model is trained by optimizing the connectionist temporal classification (CTC) loss $\mathcal{L}_{\text{CTC}}$ [40]. Suppose the ground-truth transcription is $\boldsymbol{w}^*$, which is a sequence of character tokens. The CTC loss is defined as:

$$\mathcal{L}_{\text{CTC}} = -\log \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{w}^*)} \prod_{t=1}^{T} p(\boldsymbol{\pi}_t | \boldsymbol{f}_t) \tag{2}$$

where $\boldsymbol{f}_t$ refers to $\boldsymbol{c}_t$ in this stage, T is the number of frames, $\mathcal{B}$ is a function to map an alignment sequence $\boldsymbol{\pi}_{1:T}$ to $\boldsymbol{w}^*_{1:N}$ (where N represents the number of character tokens) by removing duplicate characters and blanks while its inverse function $\mathcal{B}^{-1}(\boldsymbol{w}^*)$ refers to all the CTC paths mapped from $\boldsymbol{w}^*$. For speech data, there are 31 tokens for character targets, including 26 letters, the quotation mark, a word boundary token, $< bos >$, $< eos >$, and CTC blank token. The probability $p(\boldsymbol{\pi}_t | \boldsymbol{f}_t)$ is computed by the CTC linear layer followed by a softmax operation. Besides the supervised CTC loss, pseudo-labeling is also adopted during finetuning. Please refer to [16, 41] for more details.

3.4 Stage III: Transferring on singing data

3.4.1 From CTC to CTC/attention

To transfer the trained wav2vec 2.0 from the speech domain to the singing domain, we retain the weights of the feature encoder and the context network after Stages I and II. Furthermore, inspired by [42], we extend the original CTC system into a hybrid CTC/attention system through the addition of an ALT head on top of wav2vec 2.0 instead of a single CTC linear layer (Fig. 1(c)).

The context representations $\boldsymbol{c}$ are first fed into a linear layer followed by a leaky ReLU activation layer to obtain the features $\boldsymbol{f}$. Then $\boldsymbol{f}$ are sent to two network branches. One branch is a CTC linear layer, which aims to compute $p(\boldsymbol{\pi}_t | \boldsymbol{f}_t)$ as explained in sec. 3.2.2. Another branch is an attention-based GRU decoder [43] followed by a sequence-to-sequence (S2S) linear layer. The GRU decoder has a single layer with a hidden dimension of 1,024 and utilizes location-aware attention [43] with attention dimension 256. The decoder and S2S linear layer autoregressively compute the probability $p(\boldsymbol{w}_n | \boldsymbol{w}_{<n}, \boldsymbol{f}_{1:T})$, $n = 1, 2, ..., N$.

3.4.2 Training and Evaluation

During Stage III, wav2vec 2.0 and the ALT head are trained through the combination of CTC loss [40] and S2S loss [44]:

$$\mathcal{L}_w = \lambda_a \mathcal{L}_{\text{CTC}} + (1 - \lambda_a)\mathcal{L}_{\text{S2S}} \tag{3}$$

$$\mathcal{L}_{\text{S2S}} = -\log \prod_{n=1}^{N} p(\boldsymbol{w}^*_n | \boldsymbol{w}^*_{<n}, \boldsymbol{f}_{1:T}) \tag{4}$$

where λ_a is a hyper-parameter to balance the CTC loss term and S2S loss term. To overcome catastrophic forgetting, we adopt a smaller learning rate for wav2vec 2.0 compared to the ALT head.

To evaluate the performance of the trained model, the most likely lyrics are predicted using beam search:

$$\begin{aligned} \boldsymbol{w}' = \arg\max_{\boldsymbol{w}} \lambda_b \log \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\boldsymbol{w})} \prod_{t=1}^{T} p(\boldsymbol{\pi}_t | \boldsymbol{f}_t) \\ + (1 - \lambda_b) \log \prod_{s=1}^{S} p(\boldsymbol{w}_s | \boldsymbol{w}_{<s}, \boldsymbol{f}_{1:T}) \\ + \lambda_c \log p_{LM}(\boldsymbol{w}) \end{aligned} \tag{5}$$

where λ_b and λ_c are two hyper-parameters in the decoding process. The language model is implemented by a 3-layer LSTM. The characters are firstly projected to embeddings and then fed into the LSTM with a hidden dimension of 2,048 to obtain RNN features. Finally, the RNN features are accepted by a 3-layer MLP with a hidden dimension of 1,024 to output the probability $p_{LM}(\boldsymbol{w})$.

Split	Dataset	# Utt.	Total Dur.
Train	DSing1	8,794	15.1 h
	DSing3	25,526	44.7 h
	DSing30	81,092	149.1 h
	DALI$^{\text{train}}$	268,392	183.8 h
Dev	DSing$^{\text{dev}}$	482	41 min
	DALI$^{\text{dev}}$	1,313	55 min
Test	DSing$^{\text{test}}$	480	48 min
	DALI$^{\text{test}}$	12,471	9 h
	Jamendo	921	49 min
	Hansen	634	34 min
	Mauch	878	54 min

Table 1. Statistics of segmented utterance-level datasets.

4. EXPERIMENTS

4.1 Datasets and preprocessing

We use various accessible mainstream lyric transcription datasets for our experiments, including DALI [19, 20], Hansen [22], Mauch [23], Jamendo [21], and a curated version of DAMP Sing! 300x30x2 [24] called DSing [5]. The train/development/test splits in our experiments are defined as follows. For the DSing dataset, we use the same split configuration as in [5]. Specifically, there are three different sizes of training sets (DSing1, DSing3, DSing30), as well as a development set DSing$^{\text{dev}}$ and a test set DSing$^{\text{test}}$. As for the DALI dataset, we divide all publicly available audio in DALI v2 [20] into training and development subsets (DALI$^{\text{train}}$ and DALI$^{\text{dev}}$ respectively). [1] We use DALI$^{\text{test}}$ [7], a subset of DALI v1 [19], as a test set for experiments involving DALI. The full Hansen, Mauch, and Jamendo datasets are used as additional out-of-domain test sets. In the training and development splits of the DALI dataset, songs that overlapped with any of our test sets are removed for more objective testing.

The speech data used to pretrain and finetune the wav2vec 2.0 is initially monophonic. Therefore, we expect wav2vec 2.0 to extract better representation features for singing data if monophonic audios are given as the inputs. Thus, we extract vocal parts from all of the polyphonic recordings in DALI, Mauch, and Jamendo datasets using Demucs v3 *mdx_extra* [45], which is the state-of-the-art source separation model that achieved the first rank at the 2021 Sony Music DemiXing Challenge (MDX). This ensures that we are consistent with the input requirements of wav2vec 2.0 and minimize the interference of musical accompaniment. We adopt utterance-level input for both training and testing. To facilitate the experiments, we perform utterance-level segmentation on all audios according to their annotations. Utterances with obvious faulty annotations are removed (e.g., utterances labeled as several words but have shorter than 0.1 s duration). The statistics of all datasets after utterance-level segmentation are listed in Table 1. The "total duration" column refers to the sum of durations of all utterances in the datasets, excluding the instrumental-only parts between utterances, hence resulting in shorter durations than [7]. We notice that the average utterance duration of DSing is longer than DALI (6.52 s vs. 2.47 s). Finally, lyric texts in training sets are normalized by converting all letters to upper case, converting digits to words, discarding out-of-vocabulary characters, discarding meaningless lines (e.g., "**guitar solo**"), and removing redundant space.

4.2 Experiment setup

Our experiments are conducted through SpeechBrain toolkit [46] [2]. Before transferring on singing data, the wav2vec 2.0 has been pretrained on LibriVox (LV-60K) and finetuned on LibriSpeech (LS-960) [3] [16]. Then we randomly initialize the ALT head. During Stage III, we downsample all audios to 16 kHz and convert them to mono-channel by averaging the two channels of stereo audio signals. Then the singing data is augmented through SpecAugment [47]. wav2vec 2.0 and ALT head are trained using Adam optimizer [48]. The initial learning rates of ALT head and wav2vec 2.0 are 3×10^{-4} and 1×10^{-5} respectively. Learning rates are scheduled using the Newbob technique, with annealing factors of 0.8 and 0.9 respectively. The batch size is set as 4, and hyper-parameter λ_a is set as 0.2 during the experiments. We conduct our experiments on 4 RTX A5000 GPUs. Utterances whose duration is longer than 28 seconds are filtered out during training to prevent the out-of-memory issue. This filtering is not performed during the evaluation for a fair comparison with other existing methods.

We firstly utilize DSing30 to train the whole model for 10 epochs. We evaluate the performance of our model on DSing$^{\text{dev}}$ after each epoch. Finally, the best model is selected to be evaluated on the test split DSing$^{\text{test}}$. Word error rate (WER) is adopted as the evaluation metric. During the evaluation, WERs are averaged over all utterances in a test set. The RNNLM is trained for 20 epochs using an Adam optimizer [48] on texts of DSing30 split and validated on the DSing$^{\text{dev}}$ split after each epoch. The learning rate is 1×10^{-3} and the batch size is 20. During the decoding, the beam size is 512, and the hyper-parameters λ_b and λ_c are 0.4 and 0.5, respectively. The trained model is evaluated on the DSing$^{\text{test}}$ set.

For other test splits, we adopt both DSing$^{\text{train}}$ and DALI$^{\text{train}}$ to train the whole model. Since these two datasets are collected from different domains, we adopt a consecutive training strategy instead of training together in order to reduce the difficulty of training. Specifically, we continue training the whole model on DALI$^{\text{train}}$ split for 4 epochs. The weights of learnable parameters are initialized using the model trained on DSing30. After training, we evaluate the model on DALI$^{\text{test}}$ as well as the Hansen, Mauch, and Jamendo datasets. The configuration

[1] At the time of this research, some audios are not retrievable through their YouTube links in the public-available metadata. Although audios containing the same titles and artist names can be found online, we cannot guarantee they perfectly match the annotations for the original audio versions in DALI. Discarding invalid audio is only performed for the training and the development sets.

[2] Our code is released at https://github.com/guxm2021/ALT_SpeechBrain

[3] https://huggingface.co/facebook/wav2vec2-large-960h-lv60-self

Method	DSing^{dev}	DSing^{test}	DALI^{test}	Jamendo	Hansen	Mauch
TDNN-F [5]	23.33	19.60	67.12	76.37	77.59	76.98
CTDNN-SA [6]	17.70	14.96	76.72	66.96	78.53	78.50
Genre-informed AM [3]	-	56.90	-	50.64	39.00	40.43
MSTRE-Net [7]	-	15.38	42.11	34.94	36.78	37.33
DE2 - segmented [4]	-	-	-	44.52	49.92	-
Ours	**12.34**	**12.99**	**30.85**	**33.13**	**18.71**	**28.48**

Table 2. WERs (%) of various ALT systems on different singing datasets. "-" refers to "non-applicable". We use **bold face** to highlight the best results, and underline to mark the second-best results. Note that the results of [5–7] on DALI^{test}, Jamendo, Hansen, and Mauch datasets are obtained without utterance segmentation.

of RNNLM is the same as above, except that we utilize texts of both DSing30 and DALI^{train} to train the model. To decode the lyrics, we set the beam size as 512 and the hyper-parameters λ_b and λ_c as 0.3 and 0.2, respectively.

5. RESULTS

In this section, we firstly compare our method with state-of-the-art ALT systems on multiple benchmark datasets. Then we conduct extensive ablation studies to show the benefits of our design choices on the DSing dataset, which has more accurate manual annotations on its development and test splits. Finally, we show that our method is still effective with a limited amount of singing data.

5.1 Comparison with the state-of-the-art

We compare the performance of the proposed method with previous approaches, as shown in Table 2. Our method outperforms all previously published results on all the evaluation datasets. Our method achieves 5.36%, 1.97%, 11.26%, 1.81%, 18.07%, 8.85% absolute WER reduction on the DSing^{dev}, DSing^{test}, DALI^{test}, Jamendo, Hansen, and Mauch datasets, compared with the best results among all the previous state-of-the-art approaches respectively. Especially, on DALI^{test}, Hansen, and Mauch datasets, our method significantly exceeds MSTRE-Net [7] by an average of 12.73% absolute WER.

5.2 Effects of pretraining & finetuning on speech data

As shown in Fig. 1, wav2vec 2.0 is pretrained and finetuned on speech data before transferring to singing data. The feature representation knowledge learned from speech data is the key to the success of our method. To validate this statement, we conduct ablation studies by comparing the proposed method to two alternative configurations. The first alternative configuration is that we randomly initialize the weights of wav2vec 2.0 and ALT head and then train the whole model on the DSing dataset as per the experiment setup in section 4.2. Note that the first alternative has no transfer learning from the speech domain (without both Stages I and II). The second alternative configuration is that we only perform pretraining on speech data [4] without Stage II before transferring the wav2vec 2.0 to the singing domain.

Figure 2. Comparison of different training configurations. "w/o PT & FT" refers to without Stages I and II. "w/o FT" refers to without Stage II. (Left) Training loss of all configurations for the first 10 epochs; (Right) validation loss of all configurations for the first 10 epochs.

Method	DSing^{dev}	DSing^{test}
Ours	**12.34**	**12.99**
- Finetuning	12.64 (+ 0.30)	14.58 (+ 2.59)
- Pretraining	35.61 (+24.27)	39.13 (+26.14)

Table 3. WERs (%) of different training configurations on DSing dataset.

We evaluate the above training configurations on the DSing dataset. First, we show the curves of training loss and validation loss (loss of the development set) during the training for the first 10 epochs in Fig. 2. The losses are computed through Eq. 3. We observe that without Stages I and II, the training loss and validation loss are much higher than in the other two training configurations. In addition, its convergence is much slower. When we enable Stage I but disable Stage II, the behavior of the training loss is similar to the proposed configuration, except that the training loss is higher at the beginning. However, the validation loss in this setup is higher than that of the proposed configuration.

We continue training both alternatives until convergence and display the resultant performances in Table 3. We note that without Stage II, the performance drops by 0.30% higher WER on DSing^{dev} and 2.59% higher WER on DSing^{test}. Furthermore, without Stages I and II, the performance degrades severely as WERs on DSing^{dev} and

[4] https://huggingface.co/facebook/wav2vec2-large-lv60

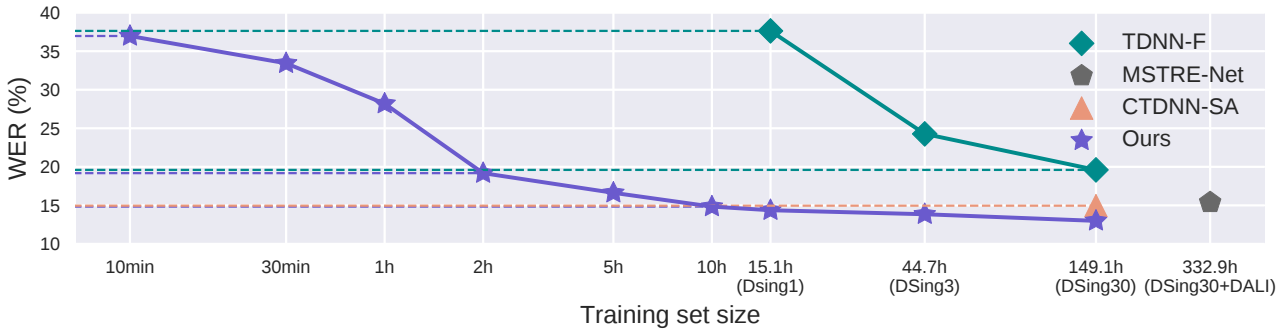


Figure 3. Transcription performance comparison when using different training set sizes, testing on the DSing$^{\text{test}}$ dataset.

Method	DSing$^{\text{dev}}$	DSing$^{\text{test}}$
CTC	19.86	20.99
+ S2S	15.63 (-4.23)	16.95 (-4.04)
+ LM	**12.34** (-7.52)	**12.99** (-8.00)

Table 4. WERs (%) of CTC model and hybrid CTC/attention model on DSing dataset.

DSing$^{\text{test}}$ increase by 24.27% and 26.14% respectively, compared to the proposed configuration. The results are consistent with our observations in Fig. 2. Therefore, we conclude that pretraining on speech data plays a significant role in transferring wav2vec 2.0 to the singing domain. Although finetuning on speech data is less crucial than pretraining, it also contributes to empirical performance gains.

5.3 Effects of extending CTC to CTC/attention model

To validate the effectiveness of changing from CTC to CTC/attention (as in Fig. 1), we compare the performance of our hybrid CTC/attention model with its CTC version, as shown in Table 4. We set $\lambda_b = 1$ in Eq. 5 to disable the branch of the GRU decoder and the S2S linear during the decoding. When $\lambda_c = 0$, the RNNLM is disabled.

We observe that the hybrid CTC/attention model achieves better performance by 4.23% and 4.04% WER on DSing$^{\text{dev}}$ and DSing$^{\text{test}}$ respectively than its CTC counterpart, which demonstrates the superiority of our ALT head design. Furthermore, we evaluate the benefits brought by the language model and find that the final model leads to 3.29% and 3.96% further absolute WER improvements compared to the hybrid CTC/attention model.

5.4 Effectiveness of transfer learning in low-resource scenarios

To explore the effectiveness of our transfer learning method in reducing the amount of required training data, we conduct an ablation study with different training set sizes. We first train our model on DSing30, DSing3, and DSing1, respectively, to observe the performance differences. Then, we further reduce the training set size to a minimum of 10 minutes to create more demanding low-resource setups. We report the WERs achieved on DSing$^{\text{test}}$ as the performance measure. When training on 10-minute and 30-minute datasets, the GRU decoder converges too slowly; hence, WERs are computed according to the CTC outputs.

As shown in Fig. 3, our method achieves 14.84% WER with only 10 hours (about 6.7% size of DSing30) of labeled singing data, which surpasses the state-of-the-art results of 14.96% WER achieved by CTDNN-SA [6] trained on DSing30. It also has better performance with only 2 hours of training data (about 1.3% size of DSing30) than TDNN-F [5] trained on DSing30 (19.18% vs. 19.60% WER). Further, with only 10 minutes of data (1.1% size of DSing1), our method achieves better results than TDNN-F trained on DSing1 (36.97% vs. 37.63% WER). These results demonstrate the feasibility of achieving competitive results with much less training data by adopting the representation knowledge from the speech domain.

6. CONCLUSION

We have introduced a transfer learning approach for the automatic lyric transcription (ALT) task by utilizing the representation knowledge learned by self-supervised learning models on speech data. By performing parameter transfer on wav2vec 2.0 towards the singing domain and extending the original CTC model to a hybrid CTC/attention version, we achieved significant improvement compared to previous state-of-the-art methods on various singing datasets. We demonstrated that both pretraining and finetuning on speech data contribute to the final ALT performance and that pretraining brings more performance gains than finetuning on speech data. Additionally, our method still showed competitive performance using only a tiny proportion of training data, indicating its potential in low-resource scenarios.

7. ACKNOWLEDGEMENT

We would like to thank anonymous reviewers for their valuable suggestions. We also appreciate the help from Emir Demirel, Gabriel Meseguer-Brocal, Jens Kofod Hansen, Chitralekha Gupta for providing datasets. This project is funded in part by a grant (R-252-000-B78-114) from Singapore Ministry of Education.

8. REFERENCES

[1] T. Hosoya, M. Suzuki, A. Ito, S. Makino, L. A. Smith, D. Bainbridge, and I. H. Witten, "Lyrics recognition from a singing voice based on finite state automaton for music information retrieval." in *ISMIR*, 2005, pp. 532–535.

[2] H. Fujihara, M. Goto, and J. Ogata, "Hyperlinking lyrics: A method for creating hyperlinks between phrases in song lyrics." in *ISMIR*, 2008, pp. 281–286.

[3] C. Gupta, E. Yılmaz, and H. Li, "Automatic lyrics alignment and transcription in polyphonic music: Does background music help?" in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 496–500.

[4] E. Demirel, S. Ahlbäck, and S. Dixon, "Low resource audio-to-lyrics alignment from polyphonic music recordings," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 586–590.

[5] G. R. Dabike and J. Barker, "Automatic lyric transcription from karaoke vocal tracks: Resources and a baseline system." in *Interspeech*, 2019, pp. 579–583.

[6] E. Demirel, S. Ahlbäck, and S. Dixon, "Automatic lyrics transcription using dilated convolutional neural networks with self-attention," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[7] E. Demirel, S. Ahlbäck, and S. Dixon, "Mstre-net: Multistreaming acoustic modeling for automatic lyrics transcription," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 7-12, 2021*, J. H. Lee, A. Lerch, Z. Duan, J. Nam, P. Rao, P. van Kranenburg, and A. Srinivasamurthy, Eds., 2021, pp. 151–158.

[8] X. Gao, C. Gupta, and H. Li, "Genre-conditioned acoustic models for automatic lyrics transcription of polyphonic music," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 791–795.

[9] B. Sharma and Y. Wang, "Automatic evaluation of song intelligibility using singing adapted stoi and vocal-specific features," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 319–331, 2019.

[10] L. Torrey and J. Shavlik, "Transfer learning in handbook of research on machine learning applications (eds. soria, e., martin, j., magdalena, r., martinez, m. & serrano, a.) 242–264," 2009.

[11] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[12] P. Sullivan, T. Shibano, and M. Abdul-Mageed, "Improving automatic speech recognition for non-native english with transfer learning and language model decoding," *arXiv preprint arXiv:2202.05209*, 2022.

[13] B. Zoph, D. Yuret, J. May, and K. Knight, "Transfer learning for low-resource neural machine translation," *arXiv preprint arXiv:1604.02201*, 2016.

[14] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[15] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised pre-training for speech recognition," *arXiv preprint arXiv:1904.05862*, 2019.

[16] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 449–12 460, 2020.

[17] M. Riviere, A. Joulin, P.-E. Mazaré, and E. Dupoux, "Unsupervised pretraining transfers well across languages," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 7414–7418.

[18] S. Khurana, A. Laurent, and J. Glass, "Magic dust for cross-lingual adaptation of monolingual wav2vec-2.0," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 6647–6651.

[19] G. Meseguer-Brocal, A. Cohen-Hadria, and G. Peeters, "Dali: A large dataset of synchronized audio, lyrics and notes, automatically created using teacher-student machine learning paradigm," *arXiv preprint arXiv:1906.10606*, 2019.

[20] ——, "Creating dali, a large dataset of synchronized audio, lyrics, and notes," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.

[21] D. Stoller, S. Durand, and S. Ewert, "End-to-end lyrics alignment for polyphonic music using an audio-to-character recognition model," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 181–185.

[22] J. K. Hansen and I. Fraunhofer, "Recognition of phonemes in a-cappella recordings using temporal patterns and mel frequency cepstral coefficients," in *9th Sound and Music Computing Conference (SMC)*, 2012, pp. 494–499.

[23] M. Mauch, H. Fujihara, and M. Goto, "Integrating additional chord information into hmm-based lyrics-to-audio alignment," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 200–210, 2011.

[24] Ccrma.Stanford.Edu, "Smule sing! 300x30x2 dataset," Sep. 2018. [Online]. Available: https://ccrma.stanford.edu/damp/

[25] S. Basak, S. Agarwal, S. Ganapathy, and N. Takahashi, "End-to-end lyrics recognition with voice to singing style transfer," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 266–270.

[26] C. Zhang, J. Yu, L. Chang, X. Tan, J. Chen, T. Qin, and K. Zhang, "Pdaugment: Data augmentation by pitch and duration adjustments for automatic lyrics transcription," *arXiv preprint arXiv:2109.07940*, 2021.

[27] X. Gu, L. Ou, D. Ong, and Y. Wang, "Mm-alt: A multimodal automatic lyric transcription system," *arXiv preprint arXiv:2207.06127*, 2022.

[28] A. M. Kruspe, "Training phoneme models for singing with "songified" speech data," in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, M. Müller and F. Wiering, Eds., 2015, pp. 336–342.

[29] S. Pascual, M. Ravanelli, J. Serra, A. Bonafonte, and Y. Bengio, "Learning problem-agnostic speech representations from multiple self-supervised tasks," *arXiv preprint arXiv:1904.03416*, 2019.

[30] Y.-A. Chung, W.-N. Hsu, H. Tang, and J. Glass, "An unsupervised autoregressive model for speech representation learning," *arXiv preprint arXiv:1904.03240*, 2019.

[31] D. Jiang, X. Lei, W. Li, N. Luo, Y. Hu, W. Zou, and X. Li, "Improving transformer-based speech recognition using unsupervised pre-training," *arXiv preprint arXiv:1910.09932*, 2019.

[32] R. Zhang, H. Wu, W. Li, D. Jiang, W. Zou, and X. Li, "Transformer based unsupervised pre-training for acoustic representation learning," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 6933–6937.

[33] L. Pepino, P. Riera, and L. Ferrer, "Emotion recognition from speech using wav2vec 2.0 embeddings," *arXiv preprint arXiv:2104.03502*, 2021.

[34] J. Zhao, R. Li, Q. Jin, X. Wang, and H. Li, "Memobert: Pre-training model with prompt-based learning for multimodal emotion recognition," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2022, pp. 4703–4707.

[35] D. Seo, H.-S. Oh, and Y. Jung, "Wav2kws: Transfer learning from speech representations for keyword spotting," *IEEE Access*, vol. 9, pp. 80 682–80 691, 2021.

[36] Z. Fan, M. Li, S. Zhou, and B. Xu, "Exploring wav2vec 2.0 on speaker verification and language identification," *arXiv preprint arXiv:2012.06185*, 2020.

[37] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[38] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

[39] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[40] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 369–376.

[41] Q. Xu, A. Baevski, T. Likhomanenko, P. Tomasello, A. Conneau, R. Collobert, G. Synnaeve, and M. Auli, "Self-training and pre-training are complementary for speech recognition," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 3030–3034.

[42] S. Watanabe, T. Hori, S. Kim, J. R. Hershey, and T. Hayashi, "Hybrid ctc/attention architecture for end-to-end speech recognition," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1240–1253, 2017.

[43] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," *Advances in neural information processing systems*, vol. 28, 2015.

[44] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *2016 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2016, pp. 4945–4949.

[45] A. Défossez, "Hybrid spectrogram and waveform source separation," in *Proceedings of the ISMIR 2021 Workshop on Music Source Separation*, 2021.

[46] M. Ravanelli, T. Parcollet, P. Plantinga, A. Rouhe, S. Cornell, L. Lugosch, C. Subakan, N. Dawalatabad, A. Heba, J. Zhong *et al.*, "Speechbrain: A general-purpose speech toolkit," *arXiv preprint arXiv:2106.04624*, 2021.

[47] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[48] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

A NOVEL DATASET AND DEEP LEARNING BENCHMARK FOR CLASSICAL MUSIC FORM RECOGNITION AND ANALYSIS

Daniel Szelogowski **Lopamudra Mukherjee** **Benjamin Whitcomb**
University of Wisconsin — Whitewater
{szelogowdj19,mukherjl,whitcomb}@uww.edu

ABSTRACT

Automated computational analysis schemes for Western classical music analysis based on form and hierarchical structure have not received much attention in the literature so far. One reason, of course, is the paucity of labeled datasets — which, if available, could be used to train machine learning approaches. Dataset curation cannot be crowdsourced; one needs trained musicians to devote sizable effort to carry out such annotations. Further, such an analysis is not simple for beginners; obtaining labeled data that can capture the nuances of a musician's reasoning acquired over years of practice is fraught with challenges. To this end, we provide a system for computational analysis of classical music, both for machine learning and music researchers. First, we introduce a labeled dataset containing 200 classical music pieces annotated by form and phrases. Then, by leveraging this dataset, we show that deep learning-based methods can be used to learn Form Classification as well as Phrase Analysis and Classification, for which few (if any) results have been reported yet. Taken together, we provide the community with a unique dataset as well as a toolkit needed to analyze classical music structure, which can be used or extended to drive applications in both commercial and educational settings.

1. INTRODUCTION

Musical form analysis is the process of analyzing classical music pieces based on structure, themes, harmonies, and the relationship between them. This includes the large form (i.e., the musical "template" defining the piece's overall structure) and the hierarchical breakdown of themes, phrases, and often other substructures, including cadences. It has applications in various areas of music, such as music education, music analysis, forensic musicology, and so on, which we will discuss shortly. Typically, this task is carried out by music theorists benefiting from formalizations that have evolved over the centuries. However, form analysis is considered challenging, both for human analysts and signal processing algorithms. For humans, it takes years of training to learn to *understand* (not just read) classical music well enough to classify its structure. On the other hand, developing an ML-based approach is hindered by the difficulty of formulating this in a way that can be successfully learned as a computational task. Indeed, the curation of labeled datasets, a central ingredient in the success of supervised machine learning models, is prohibitive in terms of both time and money. First, it would require highly skilled musicians as annotators. Second, the dataset size is open-ended. With little guidance from the literature, it is also not obvious what sample sizes may be meaningful to get a basic model operational. Thus, budgeting is risky, even for a feasibility study, with assured cost overruns.

Motivation: The above challenges notwithstanding, let us consider the potential value for some key applications, *if* such a resource were publicly available.

Audio Thumbnail and Fingerprint Generation: Such a system can enable audio thumbnail/fingerprint generation for classical music where the user can quickly grasp the key sections without listening to the entire piece. This can be beneficial in marketing a piece of music as a product in a streaming service or web store (iTunes, Amazon Music) to draw in revenue for content creators [1–3].

Copyright Detection and Forensic Musicology: Such a system can also facilitate Forensic Musicology, which compares numerous pieces for similar or exact replications of musical phrases, motives, or other structures [1, 4].

Data Mining for Anthologies: Such a system can drive the production of musical form/analysis-based anthologies, alongside other fields of musicology where significant computational research is still in its early stages [5–7].

Music Education and Pedagogy: So far, music education and evaluation are purely human-guided. The availability of such a tool can facilitate the development of music practice and analysis software, such as dividing a piece by themes for rehearsal, assignment generation, or a grading system for human-analyzed scores. During a transitional time where many educational formats are moving to a hybrid setup, these developments will be a net win [8, 9].

Other potential applications also include Audio classification and Generalized Audio Structure Analysis.

Limitations of existing methods: In spite of the clearly defined need, existing structural analysis methods have been investigated almost exclusively in the context of popular music and for such tasks as segmentation of a piece into intro, verse, chorus, bridge, and outro [10]. These ideas cannot be directly applied to classical music: the for-

© D. Szelogowski, L. Mukherjee, and B. Whitcomb. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** D. Szelogowski, L. Mukherjee, and B. Whitcomb, "A Novel Dataset and Deep Learning Benchmark for Classical Music Form Recognition and Analysis", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

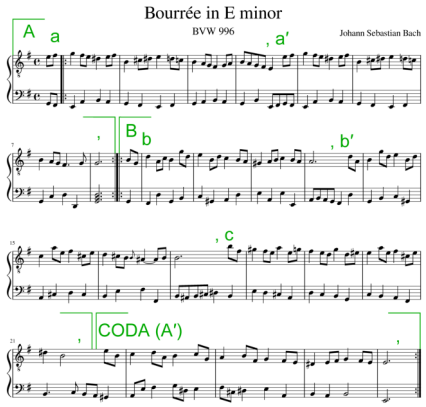


Figure 1. Example of phrase labeling from analysis of Bourrée from J.S. Bach's BWV 996 on a human-annotated score, where the relation between phrases and their respective part can be seen hierarchically. On an analyzed score, it is standard that only the first instance of a structure is labeled, although it may continue far beyond the initial instance.

mal structure in classical music is much more complicated and includes multiple forms which do not appear in popular music. Further, beyond form classification, one can also classify a musical piece into *phrases*, which typically correspond to smaller units within the piece. While classical music form analysis shares some similarities to poetic form at the *part/section* level(s) ($A\prime$ or stanza), which can be likened to popular music form (i.e., Verse-Chorus), the phrase-level analysis is drastically different due to the hierarchy of musical structures involved and a vast number of possible labels. In addition, classical music forms apply to the majority of pieces of classical music (given the large variety of large form structures). This means the form structure should be applicable to the audio (or the sheet music) by itself without the need for the segmentation of lyrical structure since a large majority of classical pieces are entirely instrumental or only partially lyrical.

Limitations of available datasets: The closest related dataset available, the Structural Analysis of Large Amounts of Music Information database, or SALAMI [11], lacks standardized conventions for the purpose of allowing for genre flexibility. It uses live recordings for time-relational analysis rather than basing timestamps on the sheet music or the score. This is not very useful for classical music since different conductors and performers often take drastically different tempi (which may be adjustable, but we seek the best overall system performance for this benchmark) and may omit or add repeats, therefore the audio cannot be compared to the sheet music directly. Hence, the use of this dataset for our goals is unfeasible.

Our Contributions: This paper provides a starting point for automated analysis of classical music forms and phrases. First, we introduce a new dataset, the **Standardized Musical Form and Structure Analysis (SMFSA) Database**, [1] consisting of 200 manually classified MIDIs, which use common analytical conventions and have categorical divisions by large musical forms. To demonstrate the use of the dataset, we also develop a deep learning-based framework to perform full form analysis, including form classification, segmentation, and part/phrase labeling. Together, this provides a comprehensive system for automated analysis of classical music, which is not otherwise available. This work provides the starting point for further development of computational techniques devoted to classical music as well as stimulating the development of numerous end-user applications.

[1] The database, research [12], and all code for this project can be found in [13]. The dataset was compiled from open sources, including [14–18].

2. RELATED WORK

While there are methods to perform genre classification, musical segmentation, and single-label segment classification in popular music, none have focused on the analytical process used by classical musicians specifically. These systems typically focus on popular music tasks including Verse-Chorus classification/segmentation [19] and genre classification [20], as well as segmenting a piece by phrases [21] — albeit without classifying. Next, we discuss a few related methods in chronological order.

Hörnel and Menzel [22] presented a melody and harmony generator using Feedforward Neural Networks to analyze musical and structural data in order to learn the characteristics of the writing style of a composer. However, their models were unable to learn higher-level musical structures occurring at multiple time scales simultaneously or recognize the melodic versus harmonic context of notes and intervals. Ponde de León and Iñesta [23] provided a framework for automatic musical style recognition of digital scores (MIDI) through the classification of rhythmic, harmonic, and melodic descriptors using k-Nearest Neighbors (k-NN), Bayesian classification, and Self-Organizing Maps (SOMs). Ullrich et. al. [24] discussed the importance of boundary recognition in structural music analysis using Convolutional Neural Networks (CNNs). Their model was trained on annotated Mel-scaled log-magnitude Spectrograms (MLS) [24] (from SALAMI) to peak-pick the onset boundaries of a given piece. Grill and Schlüter [25] further improved this model by assigning labels to digital audio using the MLS, Self-Similarity Lag Matrices (SSLMs), and human-annotated data (again from SALAMI). O'Brien [26] proposed an extension to the CNN architecture in [24] using the matrices derived from the Non-negative Matrix Factorization of a piece of music and identifying boundaries using segment association. O'Brien also noted that two major issues with their model were the lack of model memory in the CNN (i.e., the lack of LSTM or Recurrent cells) and the architecture had to be expanded to allow for a larger dataset [26]. De Berardinis et. al. [27] discussed the challenges of automatic musical structure detection and the issue of most current algorithms only being able to produce flat segmentations that cannot be applied to reveal the hierarchical structure of the piece. As such, they presented a new system for this task using multi-resolution community detection and graph theory to perform boundary detection and structural grouping, yielding a structural hierarchy. They noted that the method might also be used for structure visualization and finer-level musical structure analysis using tree representations to reflect additional structural relationships, and that CNNs will continue to lack improvement without recurrent layers or unsupervised methods.

3. METHODS

Next, we describe the key components of our model, starting with the dataset collection, the network architecture of the form analyzer, as well as a peak-picking scheme needed as input for the phrase analyzer.

3.1 Data Collection

We constructed a dataset — the **SMFSA Database** — consisting of 200 pieces of classical music in MIDI format [2] (for ease of score conversion, error correction, and signal processing). For each piece, we have an accompanying text document containing the form classification and part/phrase labels with their respective timestamps (obtained from the sheet music analysis performed by a human annotator), [3] written in a format similar to the SALAMI dataset. To represent the audio signals, the Mel Spectrogram was first generated by resampling the audio with a sample rate of 44.1 kHz, then computing the Short-Time Fourier Transform (STFT) over the entire signal with a hop length of 6144 (0.139 seconds per frame, see [28]) and 8192 samples per frame (a window size of 0.209 seconds per frame multiplied by the sampling rate). Further, to extract meaningful representations from the raw data, we obtain the 2D Self-Similarity Matrix (SSM) [25] of the Mel Spectrogram, constructed from various attributes and similarity metrics as well as the duration of the piece to create the set of features. The final tabulated dataset contains the mean and variance arrays for the following features: Mel Spectrogram SSM, Mel-Frequency Cepstral Coefficient (MFCC) Spectrogram SSLMs (Euclidean and Cosine distances), Chromagram SSLMs (Euclidean and Cosine distances) [24], as well as 15 other features. Since only 200 data samples may not be enough for many training tasks, we extend our dataset to 1200 by augmenting the dataset using speed shifting, pitch shifting, time shifting, and noise injection [12, pp. 41-46]. The features were extracted similarly for the augmented dataset as well.

3.2 Form Analyzer Architecture

Our goal is to identify a piece of classical music as one of the following 12 forms: Arch, Bar, Binary, Minuet & Trio, Ritornello, Rondo, Sonata (-allegro), Ternary, Theme & Variation, Through Composed, Unary/Strophic, and Unique. Note that not all of the forms are equally distributed in the dataset (reflecting real-world musical pieces) and this leads to a class-imbalanced multi-class classification problem (see Supplement Figure 3 for the relative proportion of each form in the dataset). To address this problem, we adopt a framework that implements an ensemble of decision trees using a neural network, which has been shown to work well for class-imbalanced multi-class datasets [29, 30]. We note that other alternatives such as focal loss [31] or imposing fairness in terms of model performance across each pair of classes can also be used [32].

In our implementation, we use the TreeGrad approach by [33], which implements Gradient Boosted Decision Trees as Neural Networks and has been shown to have a small computational footprint. TreeGrad is an extension of Deep Neural Decision Forests (DNDF), which treats the node split structure of a decision tree as a neural network architecture search problem. Similar to DNDF, they have three layers: a *decision node* layer, a *routing* layer, and a *prediction* layer [34]. The decision layer consists of decision stumps, each of which computes the probability of routing the data node x to (left/positive or right/negative) child nodes for a node n and is formulated as (shown here only for the positive route) $d_n^+(x;\theta_+) = \sigma(\theta_+^{\mathrm{T}} x + b)$, where θ_+ are the learnable parameters and σ is a temperature-controlled softmax function. These routing probabilities (both positive and negative) of all nodes are then concatenated to yield $D(x;\tilde{\theta})$, which is the output of the first layer. The parameter vector in $\tilde{\theta}$ forms the linear decision boundary that results in how each node is routed. This is followed by the routing layer, which computes the probability that each node uses to route a data point to a particular leaf. For this step, we use a binary preconstructed matrix Q of size $l \times 2n$, where l is the number of leaves, and n is the number of internal nodes. This allows for enumerating the relationship between nodes and leaves. This layer computes μ_l which indicates the probability of reaching a leaf l and can be written as

$$\begin{aligned}\mu_l(x|\theta) &= \textstyle\prod_{j=1}^{n}(D(x;\tilde{\theta}_j) \odot Q_l + (1 - Q_l)) \qquad (1)\\ &= \exp(Q_l^{\mathrm{T}} \log(D(x;\tilde{\theta}))\end{aligned}$$

where Q_l is the l-th row of Q and $\odot$ is the Hadamard Product [33]. We then pass it through a softmax activation function to produce the output of the second layer. The final output layer is the leaf layer which is a fully connected layer. The activation function used here is $\exp(x)$. See Figure 2 for an illustration of the network architecture.

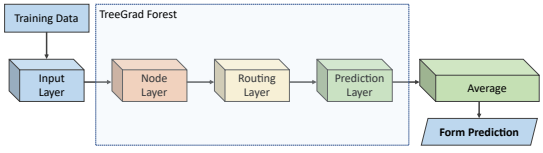


Figure 2. Architecture diagram of Form Analyzer

3.3 Phrase Analyzer Architecture

Recall that phrase analysis is a significantly more challenging task than form. To be able to learn phrase analysis similar to a musician, the model must be able to differentiate between musical events that can be labeled as having only the part label (*CODA*, *Episode 1*, *Middle Entry*, etc.), or only the phrase label (a, $a\prime$, b, *transition*, *melodic (variation)*, etc.) or both part and phrase labels. This presents several problems – the first deals with making the subtle distinction of when to assign both labels or just one, as well as choosing a suitable metric that captures the similarities correctly. Moreover, labels for varied phrases are also extremely difficult to grade, as a phrase (or part) may be repeated with a different harmonic goal (phrase a to a'

[2] While VSTs may differ across sequencing software, the Form Analyzer and Peak-Picking algorithm are intended to be instrument-neutral methods. As such, the timbral difference is negligible — including running the algorithm on different rendered sequences of the same pieces; we only seek to discover how the pitches are distributed structurally and temporally.

[3] We simply follow the human annotations from the original analyzed sheet music and copy these exactly at the time of their occurrence (in seconds, i.e., Part B with phrase d occurring at 15.27 seconds would be written as "15.270 B, d"), of course omitting the implicit labels as well.

or Parts A and A'). Variations of a can be either the same variation a' or a new variation a'' or a''' (these varied repetitions sometimes continue beyond 3, such as a^4 and so on). Since scoring such variations can be different based on the annotator, for experimentation on our dataset, we simply reduced the label set to remove prime marks and retain the phrase/part letter(s). Likewise, large and/or hybrid forms (such as Sonata-allegro form) in our dataset which have their own unique labels are simplified (during experimentation) to the letter label set with parts A, B, A'.

The task of Phrase Analysis requires labels to be provided with the form of the piece as well as the knowledge of where the musical phrase starts — a problem potentially solved using Onset or Peak Detection [25]. To do this, we implement a simple algorithm for peak-picking based on existing literature which is described next. The timestamps of the peaks, along with the output of the Form Analyzer (which is added as a feature), are provided as input to the Phrase Analyzer, where labels are classified sequentially (shown in Figure 4). We describe its components next.

3.3.1 *Onset Detection – Peak-Picking for Phrase Events*

To find the timestamps (in seconds) of the musical phrases, we use a reduced version of the peak-picking algorithm described in [35]. This algorithm computes the Mel Spectrogram and Self-Similarity Lag Matrix (SSLM) Chromagram (a group of pitch class distributions over time) to perform the onset detection. The Chromagram SSLM is computed using the Mel Spectrogram, which is clustered by pitch-class using k-Nearest Neighbors. The audio frames captured by the Short-Time Fourier Transform (STFT) are represented as a computed vector of peaks, which is returned as an array of timestamps. The choice of this scheme was based on preference for ideas that are common in the community, and we verified that it was comparable to other CNN architectures [36] and also greatly reduced system design time, given that training was not needed (since the approach was unsupervised).

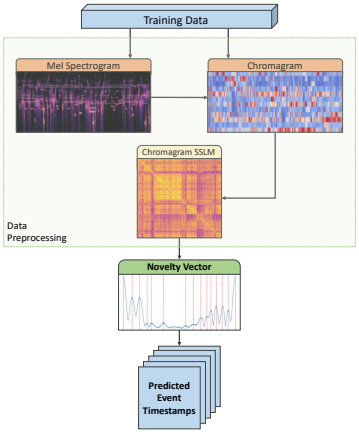


Figure 3. Schematic diagram of Peak-Picking algorithm

3.3.2 *Bidirectional LSTMs for Phrase Classification*

Using the onset detection method, the musical piece is essentially segmented into smaller phrases (i.e., the musical content between timestamps t and $t+1$ is denoted as feature vector x_t). Therefore, we think of a piece X as a collection of phrases $\{x_1, ..., x_T\}$, which occur in sequence. The task is to tag each phrase with a label for the phrase and/or the part. In all, we use 9 phrases, 9 parts, and 5 variations (phrase labels used for Theme & Variation pieces) for the labeling; specific names of these are provided in the Supplement. Clearly, the specific labeling associated with a phrase is dependent on what came before it, as well as what comes after. Therefore, we propose to use Bidirectional LSTMs to capture the dependence in such sequence data and replicate segment-segment similarity analysis.

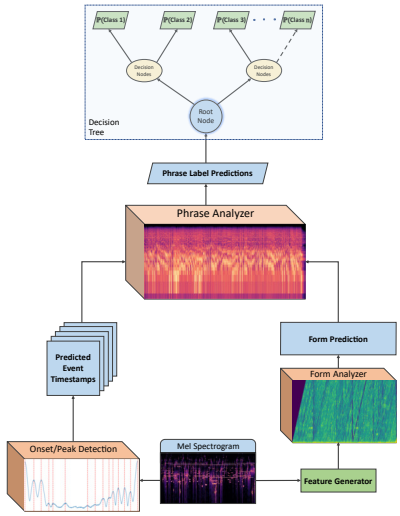


Figure 4. Architecture diagram of Phrase Analyzer

We now discuss the network architecture. X is passed as input to a Bidirectional Long Short-Term Memory (LSTM) [37] network. LSTM networks can capture long-term dependencies in temporal data and have been successfully used for a number of time series classification problems. LSTM (similar to Recurrent Neural Networks (RNNs) [38]) contains loops in its architecture that allow it to memorize previous states such that the network can effectively process temporal data. A typical LSTM layer consists of a set of recurrently connected blocks, known as memory blocks. Each block contains one or more recurrently connected memory cell (c^t) and three multiplicative units - the input (i^t), output (o^t), and forget gates (f^t) which regulate the extent to which data is propagated through the LSTM unit. The operations inside an LSTM block can be formulated using:

$$\begin{aligned} f_t &= \rho(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \rho(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \rho(W_o x_t + U_o h_{t-1} + b_o) \\ \hat{c}_t &= \phi(W_c x_t + U_c h_{t-1} + b_f) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \hat{c}_t \\ h_t &= o_t \circ \phi(c_t) \end{aligned} \tag{2}$$

Here ρ and ϕ are activation functions, $\circ$ denotes the element-wise product operation, x_t is the input vector, W and U are weights, and h_t is the hidden state vector — also known as the output vector of the LSTM unit. Because of the dependence on both past and future phrases, we use a Bidirectional LSTM (Bi-LSTM), which includes both a forward and backward layer of LSTMs. Both the forward and backward layer outputs are calculated by using the standard LSTM update equations (2). Then, Bi-LSTM

connects the two hidden layers to the same output layer. More details about Bi-LSTMs can be found in [39, 40].

To perform the final classification, the Bi-LSTM is connected to a Decision Tree to provide the label(s) for each timestamp. To accomplish this, we used the feature vector output from the (Time Distributed) Dense layer as the training set for the tree, which is fit along with the original set of labels. Once the tree is combined with the Bi-LSTM, it can be used to provide the final prediction for new timestamps. Using this approach also helped greatly decrease the training time of the Phrase Analyzer model and reduced the overfitting that the LSTM would suffer from by itself.

4. EXPERIMENTS

We evaluate each component of our system individually as well as a whole for the Phrase Analyzer, which utilizes all the components. For all the experiments, we utilize the entire augmented dataset, where 85% of the data was used for training, and the rest is used for testing.

4.1 Form Analyzer Evaluation

Setup: The Form Analyzer model takes in the features as described in Section 3.1 (the duration and Mel Spectrogram SSM) and outputs the predicted classification, which is compared against the "ground truth" label in the dataset. To turn the Mel Spectrogram SSM into a usable feature vector, we calculate the mean of the SSM to obtain a 1D array, which the model receives with the duration appended. Note that we can use the Form Analyzer architecture as a standalone model to identify the form of the musical piece. This can be useful to search a musical database by forms or for a musicologist to understand when and why a composer may choose a particular form over another. On the other hand, it can also be used to provide input to the Phrase Analyzer. Therefore, we evaluate its efficacy independently to evaluate its performance compared to other methods. The augmented dataset was split into 85% training and 15% testing for cross-validation, [4] and the analysis model was evaluated using the classification accuracy in addition to other metrics such as Precision, Recall, and F1 scores.

Parameters: The TreeGrad model was tuned to 31 leaves per tree, an unbound max depth, 100 estimators (trees in the forest) with refit splits enabled, and the learning parameters include $\alpha = 0.1$ and a batch size of 32.

Results: The final Form Analyzer model achieved a classification accuracy of 83.9% — a surprisingly good performance given the subjective nature of the form classification, as many of the "inaccurate" classifications may be subjectively true based on personal bias. We compared our model with a number of other classification methods — 2D CNN [41], 2D CNN Ensemble [42], 1D CNN [43], Autokeras [44], Neural-SVM [45], XBNet [46], DJINN [47], and an implementation of Neural Decision Trees [48]. In the plot in Figure 5 (left), our model outperforms other methods significantly. The next closest approach is the decision tree method, illustrating that the decision tree-based approaches indeed works better for this data. Note that the model can be further fine-tuned by using a different boosting method. Furthermore, our method reported a Precision value of 85%, whereas both Recall and F-Score were 84%.

In addition to numerical accuracy, we wanted to study the mistakes in our method and how others performed in identifying the forms, and whether some specific forms were commonly misclassified as others. For this, we compared the confusion matrices from each approach. To compare the confusion matrices numerically, we compute the confusion entropy [49] for each — this is shown in Figure 5 (center and right). Confusion entropy is computed by exploiting the class distribution information of misclassifications of all classes as other classes (off-diagonal elements of the confusion matrices) and is an appropriate measure for this purpose. The results show overall, our method has the lowest entropy. The confusion entropy for each class (Figure 5 (right)) shows which classes were harder to classify. Here we found that our method has an entropy of $.24$ even for the worst-performing class. In addition, by quantitative analysis, a common theme we found is that most models tended to confuse forms that are typically similar in length (binary/unary, theme & variation/sonata, etc.) or hybrid/compound forms and their derived form (e.g., sonata/binary) even though for our model, such misclassifications are minimal.

4.2 Peak-Picking Algorithm Evaluation

Results: While the onset detection algorithm was not evaluated by a formal metric, it was tested against the training data, and the output timestamps were found to be nearly identical or had a low enough difference to be subjectively true (similar to the bias of a human analyst). The output of the algorithm was compared to numerous hand-labeled pieces from the dataset, and the difference was found to be negligible for most pieces (around $\approx \pm 5.269$ seconds on average), with the greatest absolute time difference being $\approx \pm 14.823$ seconds for pieces much greater than 6 minutes in duration. This metric was based on the approach found in [25]. The algorithm was also found to be comparable to other machine learning approaches such as SOMs [23] and CNNs [24]. Our result was based on a comparison between the "ground truth" and predicted timestamps in the validation dataset by index pairs rather than by the pair of closest timestamps since the algorithm may report more or fewer events than our own analysis.

4.3 Phrase Analyzer Evaluation

This model is much more difficult to evaluate using a classification evaluation metric due to numerous factors that vary from conventional machine learning classification models. First, the model gets the timestamps from the Peak-Picking algorithm, which is difficult to compare to human-annotated scores (our "ground truth") due to integrative disagreement (i.e., a group of analysts would need to collectively agree on a ground truth as their analyses will likely vary). The labels are also often highly subjective and vary by the judgment of the annotator. In addition, some of the labels are implicit (for example, Part A continues until timestamp n but is only labeled at the first occurrence).

[4] The dataset is sorted alphabetically by form per augmentation; other permutations typically performed the same or worse.

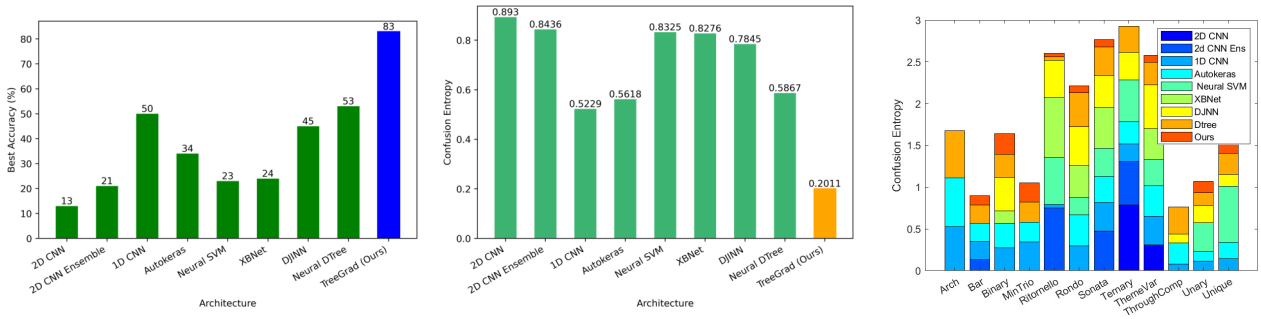


Figure 5. Form Analyzer Architecture Comparison with other methods (left), Confusion Entropy calculated for each method (center), and the class-wise Confusion Entropy for each method (right)

Therefore, we decided not to use a $0/1$ accuracy metric but a more realistic one based on how a music theory expert would grade the labeling of other musicians (such as in a music theory class) based on a rubric. Such rubric-based grading is commonly used in case of challenging problems such as automated essay and clinical note grading [50, 51].

Setup: We design a rubric as follows: we score each labeling in $[0, 1]$ giving a score of 1 if the part and/or phrase match perfectly, but also have a variety of other values in $[0, 1)$ for partially correct labeling. These partial score scenarios cover a range of possibilities, such as if either the part or phrase but not both has been guessed correctly, the degree of similarity of the predicted phrase to the actual phrase, and accounting for the fact that time stamping may be off by a few seconds. These partial scores are ordered from high to low depending on the extent and multiplicity of the issues mentioned above. The rubric is presented in the Supplement, and the code to do this grading on the output of the system will be provided with the dataset [13].

Parameters: The LSTM-Tree model has 4 LSTM units with a dropout rate of 0.2, a batch size of 10, sigmoid activation, and the learning parameters use the binary cross-entropy loss function, Adam optimizer, and 5 epochs.

Results: We evaluated the phrase analyzer using other state-of-the-art methods such as 1D CNN, RNN, and Feed-forward Neural Network and used the same rubric to come up with a weighted assessment score of the predicted phrases. We found that our LSTM-Tree architecture outperformed other NN architectures — our model reported a weighted score of 79.16%, whereas the best of the comparable methods reported a score of 77.75% (see Supplement Figure 1 for a plot related to this). A qualitative evaluation of the results reveals our LSTM-Decision Tree method produced labels that are more consistent with a human analyst (who is prone to some errors) even on one attempt, whereas for some of the other methods, the labelings are more random and less interpretable. These results show that the LSTM-Decision Trees not only substantially increased accuracy quantitatively for the Phrase Analyzer but also gave reasonably realistic phrase labels.

5. DISCUSSION

Here we briefly discuss some possible extensions and improvements to the dataset as well as methods presented in the paper. On the dataset front, we hope to expand the number of pieces as well as add additional annotators. The current dataset also features class imbalance; anthologies of classical music classified by form are lacking, [5] though our system could be employed to assist in the compilation of such a database. A more sophisticated system may also greatly benefit from restructuring the analysis labels in the database to the standards specified in [54]. On the method front, an obvious area of improvement is the Peak-Picking algorithm, which can be replaced by a deep learning approach such as CNN or LSTM so that the whole network can be considered as one large deep learning system and parameter weights learned jointly. The Phrase Analyzer model can also benefit from the inclusion of Curriculum Learning or a Human-in-the-Loop type of approach, much like that of a traditional Form and Analysis class — though a Sequence-to-Sequence or Autoencoder model may also be useful in developing a faster/more accurate system.

6. CONCLUSION AND FUTURE WORK

In this paper, we introduce a new dataset, the **SMFSA Database**, accompanied by deep learning benchmark methods to analyze classical music forms and phrases. To the best of our knowledge, this is the most comprehensive study of a computational approach used toward classical music structure analysis. While the current system is specific to classical music, it could be extended to classify numerous additional forms (e.g., through transfer learning or an extended architecture), such as those found in popular music, ethnomusicology, and more complex hybrid forms. Another difficult task lacking substantial research is Optical Music Recognition (OMR) — our methods could potentially be extended to perform both visual music analysis and the classification/segmentation on the score directly (i.e., in the style of a human analyst). The system may also be extendable for use in Forensic Musicology and Copyright Detection systems, using the output analysis from the system to compare multiple pieces of music for potentially similar or exact replications of musical phrases.

[5] Green's book on Form Analysis [52] was found to be the most useful and accurate resource resembling such an anthology, whereas other resources were presented as anthologies for analysis rather than classified works. As such, our dataset was built primarily on the pieces (and musical forms) selected in this book (including some from [53] as well) to serve as a common ground for analysis.

7. REFERENCES

[1] C. Burges, D. Plastina, J. Platt, E. Renshaw, and H. Malvar, "Using audio fingerprinting for duplicate detection and thumbnail generation," vol. 3, Apr 2005, pp. iii/9 – iii/12.

[2] J. Valinsky, "Facebook now lets users share music, listen to 30-second song snippets," *Digiday*, Nov 2015. [Online]. Available: https://digiday.com/?p=145138

[3] M. Müller, *Fundamentals of Music Processing: Using Python and Jupyter Notebooks*. Springer Nature, Apr 2021.

[4] R. Liao, "How china's acrcloud detects copyrighted music in short videos," *TechCrunch*, Aug 2020. [Online]. Available: https://techcrunch.com/2020/08/12/acrcloud-profile/

[5] T. Li, M. Ogihara, and G. Tzanetakis, *Music Data Mining*. CRC Press, Jul 2011.

[6] ——, "Special section on music data mining," *Multimedia, IEEE Transactions on*, vol. 16, pp. 1185–1187, Aug 2014.

[7] M. Gotham, "Moments musicaux," in *6th International Conference on Digital Libraries for Musicology*, ser. DLfM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 70–78. [Online]. Available: https://doi.org/10.1145/3358664.3358676

[8] R. Team, "Teaching the elements of music - form," Feb 2016. [Online]. Available: https://www.connollymusic.com/stringovation/teaching-the-elements-of-music-form

[9] M. L. Maher and D. Fisher, "Using ai to evaluate creative designs," in *Proceedings of the 2nd International Conference on Design Creativity (ICDC)*, vol. 1, Sep 2012, p. 45–54.

[10] H.-T. Cheng, Y.-H. Yang, Y.-C. Lin, and H. H. Chen, "Multimodal structure segmentation and analysis of music using audio and textual information," in *IEEE International Symposium on Circuits and Systems*, 2009, pp. 1677–1680.

[11] J. B. L. Smith, J. A. Burgoyne, I. Fujinaga, D. De Roure, and J. S. Downie, "Design and creation of a large-scale database of structural annotations." in *ISMIR*, vol. 11. Miami, FL, 2011, pp. 555–560.

[12] D. Szelogowski, "Deep learning for musical form: Recognition and analysis," Master's thesis, Apr 2022. [Online]. Available: https://doi.org/10.13140/RG.2.2.33554.12481

[13] ——, "Form-nn," Apr 2022. [Online]. Available: https://github.com/danielathome19/Form-NN

[14] C. M. Resource, "Welcome to the classical midi and mp3 resource." [Online]. Available: http://www.classicalmidiresource.com/

[15] B. Krueger, "Classical piano midi page - main page," 2018. [Online]. Available: http://www.piano-midi.de/midicoll.htm

[16] K. der Fuge Project, "kunstderfuge.com - the largest classical music midi collection." [Online]. Available: https://www.kunstderfuge.com/

[17] S. Ritchie, 2022. [Online]. Available: https://www.classicalmidi.co.uk/page7.htm

[18] T. C. A. Team, 2022. [Online]. Available: https://www.classicalarchives.com/midi.html

[19] E. Peiszer, T. Lidy, and A. Rauber, "Automatic audio segmentation: Segment boundary and structure detection in popular music," Tech. Rep., 2007.

[20] J. Yang, "Music genre classification with neural networks: An examination of several impactful variables," 2018.

[21] Y. Guan, J. Zhao, Y. Qiu, Z. Zhang, and G. Xia, "Melodic phrase segmentation by deep neural networks," 2018. [Online]. Available: https://arxiv.org/abs/1811.05688

[22] D. Hörnel and W. Menzel, "Learning musical structure and style with neural networks," *Computer Music Journal*, vol. 22, no. 4, pp. 44–62, 1998. [Online]. Available: http://www.jstor.org/stable/3680893

[23] P. J. Ponce de León and J. M. Iñesta, "Pattern recognition approach for music style identification using shallow statistical descriptors," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 2, pp. 248–257, 2007.

[24] K. Ullrich, J. Schlüter, and T. Grill, "Boundary detection in music structure analysis using convolutional neural networks," in *ISMIR*, 2014.

[25] T. Grill and J. Schlüter, "Structural segmentation with convolutional neural networks mirex submission," 2015.

[26] T. O'Brien, "Musical structure segmentation with convolutional neural networks," 2016.

[27] J. de Berardinis, M. Vamvakaris, A. Cangelosi, and E. Coutinho, "Unveiling the hierarchical structure of music by multi-resolution community detection," *Transactions of the International Society for Music Information Retrieval*, vol. 3, no. 1, 2020.

[28] carlosholivan, "Selfsimilaritymatrix-serra.ipynb," Sep 2020. [Online]. Available: https://github.com/carlosholivan/SelfSimilarityMatrices/blob/master/SelfSimilarityMatrix_Serra.ipynb

[29] T. R. Hoens, Q. Qian, N. V. Chawla, and Z.-H. Zhou, "Building decision trees for the multi-class imbalance problem," in *Advances in Knowledge Discovery and Data Mining*, P.-N. Tan, S. Chawla, C. K. Ho, and J. Bailey, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 122–134.

[30] X. Yuan, L. Xie, and M. Abouelenien, "A regularized ensemble framework of deep learning for cancer detection from multi-class, imbalanced training data," *Pattern Recognition*, vol. 77, pp. 160–172, 2018.

[31] K. Pasupa, S. Vatathanavaro, and S. Tungjitnob, "Convolutional neural networks based focal loss for class imbalance problem: A case study of canine red blood cells morphology classification," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–17, 2020.

[32] V. S. Lokhande, A. K. Akash, S. N. Ravi, and V. Singh, "Fairalm: Augmented lagrangian method for training fair models with little regret," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 365–381.

[33] C. Siu, "Transferring tree ensembles to neural networks," in *Neural Information Processing*, T. Gedeon, K. W. Wong, and M. Lee, Eds. Cham: Springer International Publishing, 2019, pp. 471–480.

[34] P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulò, "Deep neural decision forests," Dec 2015, pp. 1467–1475.

[35] J. Serrà, M. Müller, P. Grosche, and J. L. Arcos, "Unsupervised music structure annotation by time series structure features and segment similarity," *IEEE Transactions on Multimedia*, vol. 16, no. 5, pp. 1229–1240, 2014.

[36] C. Hernandez-Olivan, J. R. Beltran, and D. Diaz-Guerra, "Music boundary detection using convolutional neural networks: A comparative analysis of combined input features," Aug 2020. [Online]. Available: https://arxiv.org/abs/2008.07527

[37] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: Lstm cells and network architectures," *Neural computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[38] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, 2020.

[39] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm networks," in *Proceedings of the IEEE International Joint Conference on Neural Networks.*, vol. 4, 2005, pp. 2047–2052.

[40] Z. Cui, R. Ke, Z. Pu, and Y. Wang, "Stacked bidirectional and unidirectional lstm recurrent neural network for forecasting network-wide traffic state with missing values," *Transportation Research Part C: Emerging Technologies*, vol. 118, p. 102674, 2020.

[41] P. Nandi, "Cnns for audio classification," *Towards Data Science*, Dec 2021. [Online]. Available: https://towardsdatascience.com/cnns-for-audio-classification-6244954665ab

[42] L. Nanni, Y. M. G. Costa, R. L. Aguiar, R. B. Mangolin, S. Brahnam, and J. Silla, Carlos N., "Ensemble of convolutional neural networks to improve animal audio classification," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2020, no. 1, May 2020.

[43] S. Kiranyaz, T. Ince, O. Abdeljaber, O. Avci, and M. Gabbouj, "1-d convolutional neural networks for signal processing applications," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 8360–8364.

[44] J. Brownlee, "How to use autokeras for classification and regression," Sep 2020. [Online]. Available: https://machinelearningmastery.com/autokeras-for-classification-and-regression/

[45] M. A. Wiering, M. H. v. d. Ree, M. Embrechts, and L. Schomaker, "The neural support vector machine," Nov 2013. [Online]. Available: https://www.researchgate.net/publication/258435394_The_Neural_Support_Vector_Machine

[46] T. Sarkar, "Xbnet - an extremely boosted neural network," Jun 2021. [Online]. Available: https://arxiv.org/abs/2106.05239

[47] K. D. Humbird, L. Peterson, and R. McClarren, "Deep jointly-informed neural networks," Jul 2017. [Online]. Available: https://www.researchgate.net/publication/318205627_Deep_Jointly-Informed_Neural_Networks

[48] Y. Yang, I. G. Morillo, and T. M. Hospedales, "Deep neural decision trees," Jun 2018. [Online]. Available: https://arxiv.org/abs/1806.06988

[49] J.-M. Wei, X.-J. Yuan, Q.-H. Hu, and S.-Q. Wang, "A novel measure for evaluating classifiers," *Expert Systems with Applications*, vol. 37, no. 5, pp. 3799–3809, 2010.

[50] H. T. T. Nguyen, "Neural networks for automated essay grading," 2016.

[51] W.-w. Yim, A. Mills, H. Chun, T. Hashiguchi, J. Yew, and B. Lu, "Automatic rubric-based content grading for clinical notes," in *Proceedings of the Tenth International Workshop on Health Text Mining and Information Analysis (LOUHI 2019)*. Association for Computational Linguistics, 2019. [Online]. Available: http://dx.doi.org/10.18653/v1/d19-6216

[52] D. M. Green, *Form in Tonal Music: An Introduction to Analysis*. Cengage Learning, 1979.

[53] W. E. Caplin, *Classical Form: A Theory of Formal Functions for the Instrumental Music of Haydn, Mozart, and Beethoven*. Oxford University Press, Dec 2000.

[54] M. R. H. Gotham and M. Ireland, "Taking form: A representation standard, conversion code, and example corpora for recording, visualizing, and studying analyses of musical form," in *ISMIR*, 2019.

BAF: AN AUDIO FINGERPRINTING DATASET FOR BROADCAST MONITORING

Guillem Cortès [♩♪] **Alex Ciurana** [♩] **Emilio Molina** [♩] **Marius Miron** [♪]
Owen Meyers [♯] **Joren Six** [♭] **Xavier Serra** [♪]

[♩] BMAT Licensing S.L., Barcelona [♪] MTG, Universitat Pompeu Fabra, Barcelona
[♯] Epidemic Sound, Stockholm [♭] IPEM, Ghent University, Ghent

ABSTRACT

Audio Fingerprinting (AFP) is a well-studied problem in music information retrieval for various use-cases e.g. content-based copy detection, DJ-set monitoring, and music excerpt identification. However, AFP for continuous broadcast monitoring (e.g. for TV & Radio), where music is often in the background, has not received much attention despite its importance to the music industry. In this paper (1) we present BAF, the first public dataset for music monitoring in broadcast. It contains 74 hours of production music from Epidemic Sound and 57 hours of TV audio recordings. Furthermore, BAF provides cross-annotations with exact matching timestamps between Epidemic tracks and TV recordings. Approximately, 80% of the total annotated time is background music. (2) We benchmark BAF with public state-of-the-art AFP systems, together with our proposed baseline *PeakFP*: a simple, non-scalable AFP algorithm based on spectral peak matching. In this benchmark, none of the algorithms obtain a F1-score above 47%, pointing out that further research is needed to reach the AFP performance levels in other studied use cases. The dataset, baseline, and benchmark framework are open and available for research.

1. INTRODUCTION

Audio Fingerprinting (AFP) is the information retrieval task of identifying audio recordings in a given database of reference songs. The task is based on extracting content-based signatures that summarize an audio recording (*extraction*) [1], storing them in a database or in hash tables (*indexing*), and efficiently linking short snippets of unlabeled audio to the same content in the database (*matching*).

AFP has been successfully applied to different tasks such as query by example [2], advertisement tracking [3], integrity verification [4], and data deduplication [5]. A relevant AFP application is broadcast monitoring for its crucial role in royalties distribution. In 2021, US$ 2.9 billion were distributed among rights holders [6], representing 11.5% of the global recorded music industry revenues.

A great AFP system for broadcast monitoring must provide exact start and end timestamps within long audio recordings. It also must be robust against common distortions in broadcasting like music being in the background and with low SNR. To the best of our knowledge, the literature has not addressed the AFP use case of broadcasting monitoring with a heavy presence of background music. In this scenario, music may have a very low Signal-to-Noise ratio (SNR) that makes it difficult to identify [7]. Moreover, music may be masked by a large variety of non-musical sounds, such as speech, applause, laughter, urban and nature sounds, etc. [8].

To promote research in AFP for broadcast monitoring we propose BAF: a Broadcast Audio Fingerprinting dataset with TV recordings in which production music is played. The references are part of Epidemic Sound's private catalog [9], a collection of music for content creators (production music). BAF reflects a challenging (but realistic) scenario [7] for broadcast AFP systems: low sample rate monaural audio, background music of variable SNR, a large variety of contexts, long queries with true negative sections, and matches of multiple durations. More details about the dataset can be found in Section §3.

In order to show the challenges that BAF presents, in Section §4 we present a benchmark with 4 of the available AFP algorithms: Audfprint [10], Panako [11, 12], Olaf [13], and NeuralFP [14]. These are open-source implementations that represent different approaches to AFP. In addition, we propose an open implementation of a simple baseline based on spectral peak matching and the evaluation framework used in the benchmark.

2. RELATED DATASETS

In this section, we present public datasets that have been used in the AFP literature. We also gather information about the contents of private datasets and depict them in Table 1. We include information about the queries, references, and the goal of the dataset or publication. We have found in the literature 21 datasets that have been used for

Corresponding author: Guillem Cortès (cortes.sebastia@gmail.com)

© G. Cortès, A. Ciurana, E. Molina, M. Miron, O. Meyers, J. Six and X. Serra. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** G. Cortès, A. Ciurana, E. Molina, M. Miron, O. Meyers, J. Six and X. Serra, "BAF: an Audio Fingerprinting dataset for Broadcast Monitoring", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

Work	(Synthetic) Queries	References	Goal	Used in
2002 Haitsma, J. [15]	(✓) 4 excerpts	20,000	Scalable, robust to noise	
2003 Wang, A. [2]	250 excerpts	10,000	Scalable, robust to noise	
2005 Bartsch, M.A. [16]	93	93	Structural redundancy	
2008 Baluja, S. [17]	(✓) 1,000	10,000	Robust to pitch and time	
2008 Bellettini, C. [18, 19]	(✓) 15,000	15,000	Robust to pitch	
2011 Fenet, S. [20]	7d radio broad.	7,309 (60s)	Scalable, robust to pitch	
2011 Fenet, S. [20]	5d radio broad.	30,000	Scalable, robust to pitch	
2014 Malekesmaeili, M. [21]	(✓) 200	200	Robust to noise, pitch and time	
2015 Zhang, X. [22]	(✓) 10s excerpts	2,075	Robust to pitch and time	
2016 Sonnleitner, R. [23]	(✓) 300	20,000	Robust to noise, pitch and time	
2016 Sonnleitner, R [24]	8 DJ-mixes	296 $\approx$ 7h	Robust to pitch and time	
2016 Walter, T. [25]	10s excerpts	10M	Efficient and scalable AFP	
2017 Gfeller, B. [26]	12,000 excerpts	450h	Low-power music recognizer	
2020 Son H.-S. [27]	(✓) 100	100	Robust to pitch	
2020 Yu, Z. [28]	(✓) 5,000 (10s)	345,000	Robust to any degradation	
2009 MagnaTagATune [29]		25,863 $\approx$ 208h	Music Tagging	[30]
2006 TRECVID [31, 32]	(✓) 201	11,200 $\approx$ 400h	Content-based Copy Detection	[33–35]
2014 Panako [11]	(✓) 600 excerpts	30,000 $\approx$ 277h	Robust to pitch and time	[11]
2016 Mixotic [24]	10 DJ-mixes	723 $\approx$ 11h	Robust to pitch and time	[24]
2016 QuadFP [23]	(✓) 450,000	100,011 $\approx$ 6,899h	Robust to pitch and time	[23]
2021 NeuralFP [14]	(✓) short excerpts	100k $\approx$ 8,000h	High-specific audio retrieval	[14]

Table 1. Private (top ↑) and public (bottom ↓) datasets that have been used for AFP. For each dataset information about the queries, references and the goal of the original work is given. Synthetic (✓) queries have been created by applying transformations to reference audios. We also indicate if the queries are excerpts and the number of original audio pieces that have been transformed, not meaning the total number of queries after the transformation.

AFP. Even though the nature of the data varies from each dataset, most of them rely on the same principle: a private collection of tracks that constitute a reference set and a query set formed by applying transformations to some of the references. This is a good way to test the limits of the robustness to degradations like pitch-shifting or time-scaling. Still, it does not target the characteristics of realistic broadcast monitoring, where the music is in the background, masked by speech and a wide variety of sounds and noises (See Section §3.3.1 for a detailed analysis).

As Table 1 reflects, only 6 out of the 21 datasets are public, mainly due to the difficulty of legally publishing copyrighted music. In other cases, data remains private to protect intellectual property, as with private companies. To that extent, private datasets hinder reproducibility and slow-down scientific progress in AFP research. Of all private datasets, only the ones built by Fenet et al. [20] reflect the use case of broadcast monitoring: they use real radio broadcasted emissions as queries and a reference set of 459 songs. Moreover, only Fenet's [20], Sonnleitner et al., [24] and Walter and Gould [25] used unknown audios as queries. All other works used a version of the reference songs often modified with some degradation: echo, pitch and/or tempo alteration, reverb, etc. Besides these private datasets, there are some other public datasets that have been used in AFP works, even though none of them fits the broadcast monitoring task.

MagnaTagATune [29] is a public dataset created for Music Tagging. Each audio clip has associated a vector of binary annotations of 188 tags that describe the music piece. The dataset was used for AFP by Ramona and Peeters [30] in which they followed the experimental protocol of Haitsma and Kalker [15] applying a series of distortions like Amplitude dynamic compression, MP3 encoding, time-shifting, or equalization to 500 music clips from MagnaTagATune.

NIST-TRECVID [31, 32]. One of the tasks the TREC Video Retrieval Evaluation (TRECVID) proposed until 2011 was Content-Based Copy Detection (CCD) [36]. Various works [33–35] used the dataset provided with the task to evaluate AFP algorithms. The length of queries varies from 3 to 180 seconds and compromises multiple transformed fragments from 201 unique audio recordings.

Mixotic [24] was created by Sonnleitner et al. to test the robustness of QuadFP, Panako, and Audfprint in real DJ mixes. It was generated from free, CC-licensed DJ mixes that were published on mixotic netlabel.

Panako [11], **QuadFP** [23, 37] and **NeuralFP** [14] present public datasets that were curated to test their algorithms. Since these datasets are reproducible we list them as public AFP datasets, but in the case of Panako and QuadFP they share a script that downloads tracks from Jamendo instead of the audio files. It can happen that some audio works are not available anymore thus impeding the reconstruction of the dataset. NeuralFP shares the audio files since they come from the FMA dataset [38]. It was trained on 10k FMA songs and tested on a larger set of 100k songs ($\approx$ 8,000 hours).

3. BAF DATASET

This section describes the characteristics of BAF: Broadcast Audio Fingerprinting dataset. It is the only available dataset designed for broadcast monitoring. BAF contains TV recordings, reference tracks, and annotations done by 6 different annotators that cross-annotated matching queries and references. It is a self-contained dataset available upon user's access request. Open for non-commercial, research-only, with no adaptations or derivative works allowed and proper attribution. It must not be used for music generation or music synthesis research. Towards addressing ethical and sustainability concerns, we distribute a datasheet using the format proposed by Gebru et al. [39] with practical and detailed information about the dataset. Audio files, annotations, and the dataset datasheet are hosted in Zenodo [1] while the baseline code and evaluation scripts are in Github [2].

3.1 Methodology

The reference set contains 2,000 production music tracks (74 hours of audio) obtained directly from the Epidemic Sound [9] private catalog, described in Section 3.3.2. The queries are initially derived from TV stream monitoring on 478 TV channels from 43 countries, for a 2.5 months period at stereo maximum quality. We extract the audio with FFMPEG and we split the large audio files into 1-minute length queries.

As an automatic pre-annotation stage, we match queries with references relying on a proprietary stereo matching algorithm developed by BMAT. The algorithm is non-scalable and it relies on spectral peaks matching using stereo signals. It has been tailored to avoid false negatives, disregarding the presence of false positives and low computational efficiency. We then select all query segments that have at least one pre-annotation match. In addition, we discard queries matching more than 3 unique references since most of them contain false positives, which would be manually deleted during the later annotation process. Then, we shuffle all queries and select 3,425 of them, corresponding to 57 hours of TV broadcast audio from 203 TV channels across 23 countries. Finally, we convert all audios to publishing format: 8kHz mono, WAV `pcm_s16le`, a common specification in AFP [2, 14, 15, 34].

3.2 Annotation criteria

BAF has been annotated by six different annotators in a controlled environment. We built an in-house annotation web app in Django, secured by user credentials, in order to facilitate the annotation to remote users. The app displays all segments resulting from the automatic pre-annotation stage. Annotators were instructed to listen to the query and reference pairs, filter out false positives, and adjust the start and end times of the true positives with deciseconds precision. Queries with slight alterations with respect to the reference have also been annotated as true positives, such as versions with some missing stem (e.g. instrumental vs vocal), or 'edit' versions.

We ensured that each segment was annotated by three different annotators (sets of annotators created by random combination). We then created the cross-annotations with 3 different levels of agreement: *single, majority, unanimity*. These cross-annotations are the result of splitting annotations into segments and merging overlapping segments, assigning a tag depending on how many annotators marked a match in that time interval (1, 2, and all 3, respectively). Out of the 57 hours of queries, over 37 hours were marked as true positive by at least 1 annotator.

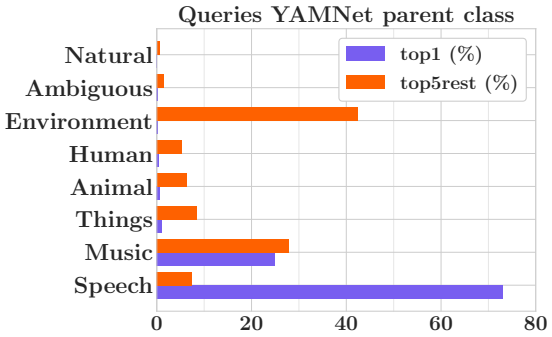


Figure 1. YAMNet classification of BAF queries. YAMNet uses a sliding window of length 0.96s and stride of 0.48s to generate one prediction for each step. Percentages are relative to each top classification.

3.3 Analysis

3.3.1 Queries

We have used the YAMNet sound event classifier [40] to study the most common sounds of BAF queries. YAMNet is a pretrained deep network that predicts 521 audio event classes based on the AudioSet-YouTube corpus [41]. Audioset ontology follows a tree structure so all classes are gathered under 7 different parent classes [42], from that, we extract *Speech* from *Human* as a separate class to better evaluate its presence. Figure 1 reflects that YAMNet's most predominant class is *Speech*, with 73% of the output predictions while only nearly 25% of them correspond to *Music*. Regarding the *top5rest* classes, *Environment* and *Music* are the most predominant, which means that noises and background music are common in the broadcast.

To obtain the YAMNet distributions we first run YAMNet for all queries and then select the outputs corresponding to the annotated segments. After that, we average the scores using a moving average window of length 2 to soften noisy scores. Lastly, we extract the top1 and top5 distributions and translate all labels to the corresponding parent class, following Audioset ontology. For each window, we remove from the top5 the parent class that matches that window's top1 parent class so with this, only the classes that are detected in the background remain. We name this distribution top5rest.

[1] `https://doi.org/10.5281/zenodo.6868083`
[2] `https://github.com/guillemcortes/baf-dataset`

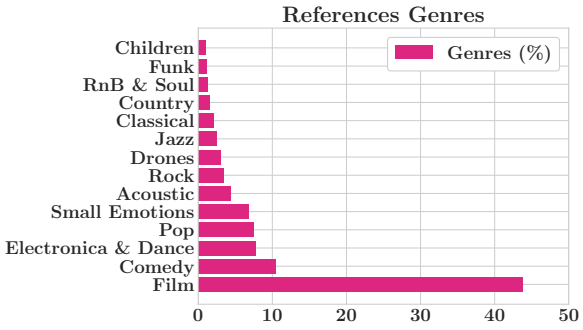


Figure 2. BAF references music genres. Only genres representing more than 1% are listed, the remainder represents 3% of the dataset.

In addition YAMNet predictions we used the MIREX 2019 Music Detection winner model [43, 44] to analyze BAF queries. The system is based on computing the Relative Music Loudness distribution [45] and classifies as *Foreground Music* only 18.07% of the total cross-annotated segments with the tag *unanimity* (segments all annotators agreed that there's music). Verifying the high presence of background music.

3.3.2 References

The BAF reference set is a selection of production music tracks from Epidemic Sound's private catalog of 35,000+ human-annotated tracks. In order to give valuable insights about instrumentalization, BPM, or genre, we have analyzed the tags and found that 7% of the selected tracks contain vocal elements like singing, while the remaining 93% are instrument-only versions. Figure 2 shows how the majority of the references are categorized as *Film music*, and cover styles/moods like Suspense, Drama, Build, Pulses, Small Emotions, or Solo Piano. Common keywords or tags found in this set of tracks include: comedic, piano, strings, driving, tension, corporate, guitar, and documentary. The references BPMs follow a Normal Distribution $\mathcal{N}(\mu, \sigma^2)$ with mean $\mu = 109$ and standard deviation $\sigma = 33$.

3.3.3 Matches / Annotations

Table 2 shows that 90.41% of the total annotated time (133,846 seconds) has been agreed by unanimity. Most of the differences between annotators come from divergences in start and end matching timestamps partially due to the difficulty to tell when a song starts or ends in a stream, especially with background music.

Additionally, to evaluate the reliability of annotators' agreement we compute the Fleiss' Kappa [46] indicator, a statistical measure of inter-rater reliability. It needs fixed-length elements to categorize them so we computed the Fleiss' Kappa indicator using 0.05 seconds length annotations. We obtained a factor of 0.9364 that can be interpreted as an almost perfect agreement [47].

3.4 Limitations

The size of the reference set is not large enough to mimic real production environments, where recordings are expected to be analyzed against tens of millions of tracks [25]. For future work, in order to study thoroughly the impact of False Positives (FP), a set of additional *noise* tracks should be added to the reference set. For this, public datasets mentioned in Section §2 could be used.

4. BENCHMARK

We benchmark the following available AFP algorithms: Audfprint [10], Panako [11, 12], Olaf [13], and NeuralFP [14] to study the performance of AFP in the broadcast monitoring use case. Additionally, we also propose a simple baseline that gives context to the results.

Audfprint is based on Shazam's algorithm [2]. It uses the locations of pairs of spectrogram peaks (local maxima points) as robust features for matching. **Panako** extends Shazam fingerprint by saving triplets of local maxima points in Constant-Q non-stationary Gabor transform. It uses time ratios to form a time-scale invariant fingerprint component. These components are hashed alongside a coarse value of the frequency position of the triplet, making it robust to time-scale and pitch modifications. **Olaf** is a lightweight AFP algorithm able to run in embedded systems. In the benchmarked version, it uses absolute exact frequencies and timestamps in the hash (like Shazam [2]) of triplets of peaks (like Panako [11]). It is not robust to pitch shifting or time distortions. **NeuralFP** is based on deep neural networks and created for high-specific audio retrieval using contrastive learning. It creates pairs of data applying distortions to short audio snippets so each batch of training data consists of randomly selected original samples and their augmented replicas. Then, it maximizes the inner product between pairs.

All algorithms except NeuralFP ran as they are published. NeuralFP, though, required changes in the indexing and matching modules to match the broadcast monitoring use case. The indexer now stores indexes on disk rather than on a memory map, and the matcher integrates the Maximum Inner Product Search used by the authors into PeakFP matcher pipeline, to be able to give start and end times of each match. The implementation [3] has been

[3] https://github.com/guillemcortes/neural-audio-fp

Class	%
single	3.69%
majority	5.90%
unanimity	90.41%

Table 2. Annotators agreement in percentage of annotation time length. *single* correspond to intervals where only 1 of the 3 annotators marked a match. In *majority* 2/3 annotators agreed while in *unanimity* there's full agreement.

validated by NeuralFP authors. Towards a fair comparison between systems, all algorithms return top1 matches. Audfprint and Olaf use the default configurations, Panako parameters are adjusted for 8kHz input signal and NeuralFP needs extra parameters for the custom matcher pipeline. Additionally, we study the impact of fingerprint density by increasing Audfprint (x2) and Panako (x1.5) peak density and also test two additional NeuralFP models *spcm1510* and *spc3000* that were trained with different levels of speech intensity [-15, 10] dB and [0, 10] dB, respectively. All configuration parameters used in this publication have been discussed with the authors of each respective algorithm and are available in the publication git repository.

Apart from the algorithms mentioned above, there are some others that we would have liked to benchmark, but no official public implementation of them was found. That is Fenet's et al. CQT approach [20], Google's Now Playing [26] lightweight, neural network-based, continuous monitoring system, and also Son's et al. FFMAP-based algorithm [27,48]. For QuadFP [23,49] and Waveprint [17] we tried to run third-party implementations but without success. We also plan to include Chromaprint [50], a public AFP implementation based on Ke et al. computer vision approach for music identification [51], and other algorithms to the benchmark.

4.1 Baseline: PeakFP

Available AFP systems aim at obtaining the best algorithm in terms of robustness, scalability, efficiency, etc. However, many existing systems are distributed as closed software packages or embedded into complicated frameworks which are difficult to adapt. Also, the literature lacks a simple and easy-to-use baseline that may be used as a starting point in AFP research. We address these issues by introducing PeakFP, a simple, open, non-data-driven, non-scalable, AFP algorithm based on spectral peak matching that it has been designed to be as simple as possible and not optimized for scalability, but at the same time, useful for detecting background music. While algorithms commonly use pairs or triplets of spectral peaks, PeakFP uses single-peak matching because pairs of spectral peaks are more prone to break when the SNR of the music signal is low due to music peaks being masked by other sounds. As a consequence, PeakFP is ineffective in front of pitch-shifting or time-scaling distortions.

PeakFP is divided into three modules: extractor, indexer, and matcher. They are designed to work independently following a simple pipeline we detail below. Note that the code of our implementation is available online (see Section 3). The extraction process involves finding peaks in a monaural audio magnitude spectrogram using a 2D max filter. Then, all the peaks (time-frequency tuples) are sorted by the time frame index and saved in a serialized binary file. The reference signature files are used to generate an inverted index on the peak frequency values. For a given frequency, the index contains the list of all occurrences of that frequency in every reference. The hash space comprises all the possible frequency values. This small hash space translates to a high quantity of hash matches that yield a high number of comparisons in the matching step. The matcher splits the peaks of the query recordings using a sliding temporal window defined by window length and hop size. Then, for all windows, it counts the common peaks for a specific query-reference time alignment similarly to Shazam [2]. After that, all matches go through a postprocessing stage in order to consolidate and resolve overlapping matches.

Figure 3. Proposed metric Match ratio. It defines the ratio between the number of identifications with respect to the number of annotations in the groundtruth.

4.2 Evaluation Metrics

A variety of metrics are used in the AFP literature for benchmarking and comparing algorithms, most of them based on classifying predictions into false positives/negatives or true labels [52], and using metrics derived from information retrieval such as True Positive Rate [11] Precision, Recall, and Specificity [2,23]. Other papers use Top-1 Hit Rate [14] or TRECVID's proposed evaluation metric Normalized Detection Cost Rate (NDCR) [33–35].

In broadcast monitoring, it is typically required to provide exact start and end matching timestamps for each reference identification. For this reason, we propose to use the percentage of identified seconds alongside to match classification into Precision, Recall, and F1-score.

Some algorithms are prone to give short overly-split matches, while others tend to generate longer matches that include gaps without annotations. Towards quantifying this we introduce a new metric *Match Ratio*, defined in equation 1 and depicted in Figure 3, that represents the ratio between the total correct identified segments (TP matches), and the total unique annotations identified, groundtruth (GT) segments.

$$\text{Match Ratio} = \frac{\#\text{ TP segments ID}}{\#\text{ TP segments GT}} \tag{1}$$

A Match Ratio value bigger than 1 means that some of the identifications belong to the same annotation. Conversely, a value lower than 1 indicates that the algorithm generates a single identification for a segment in which there is more than one unique identification according to the groundtruth. The value for an algorithm that perfectly matches the annotations is 1.

Algorithm	Match Ratio	# matches		seconds identified		
		Precision	GT Recall	Precision	Recall	F1-score
PeakFP (baseline)	1.64	.96	**.72**	.96	**.32**	**.47**
Panako2.0	1.85	**.98**	.21	**.98**	.06	.12
Panako2.0 (x1.5)	2.12	.70	.41	.69	.15	.25
Olaf	1.95	**.98**	.14	**.98**	.06	.11
NeuralFP	1.39	.22	.23	.37	.10	.15
NeuralFP-spcm1510	1.56	.23	.45	.38	.22	.28
NeuralFP-spc3000	1.40	.69	.31	.83	.13	.22
Audfprint	N/A*	.76	.05	.86	.02	.04
Audfprint (x2)	N/A*	.71	.10	.81	.04	.08

Table 3. Benchmark results on *unanimity* annotations. *Audfprint reports 1 match per query by default.

4.3 Results

Table 3 summarizes the performance of all benchmarked algorithms on *unanimity* annotations. All systems increase their Precision when considering the identified seconds because the False Positives identifications are shorter than the True Positives. At the same time, Recall decreases because the identifications are partial and do not cover the full annotation groundtruth. Hence the importance of studying also the performance in terms of identified seconds.

All algorithms except Audfprint obtain a Match Ratio > 1. This is caused because algorithms tend to detect small excerpts of the music (parts where the music SNR is higher) resulting in more than one identification per annotation. Audfprint only returns one identification per query by default, so its Match Ratio will always be 1 with this configuration. This also means that if a query has more than one annotation, Audfprint can't identify all of them.

The good Precision results (PeakFP, Panako, Olaf obtain over 0.96) should be analyzed taking into account that BAF is not challenging to False Positives since the reference set is limited to 2,000 references. The low Recall values show that there are a lot of identifications missed (False Negatives), manifesting that algorithms do not work well with background music or broadcast distortions. Increasing fingerprint density improves the F1-score in seconds identified: it helps to boost the Recall in both Panako (x1.5) and Audfprint (x2) but at expense of Precision.

Additionally, to frame the computational cost of each algorithm, we have benchmarked their extraction, indexing, and matching times as well as index size. Table 4 shows that Olaf is the fastest benchmarked system. On the other hand, PeakFP is the slowest even though the extraction process is quick due to its simplicity, but the reduced hash space yields a high collision of hashes that slows down the matching. NeuralFP extraction process takes a lot of time compared to others, mainly due to the deep neural network model complexity. This process can be sped up by running it on a GPU, where deep learning models operate more efficiently. For both Panako and Audfprint, the matching step takes x2.5 times longer than the extraction.

Algorithm	Extraction & Indexing	Matching	Index size
Olaf	53m	3h 30m	349 MB
Panako 2.0	2h 17m	5h 24m	273 MB
NeuralFP	49h 34m	9h 30m	37 MB
Audfprint	9h 50m	23h 01m	19 MB
PeakFP	50m	98h 39m	160 MB

Table 4. Computational cost benchmark.

Regarding the index sizes, Audfprint and NeuralFP generate the smallest indexes (19MB and 37 MB) to represent a set of 2,000 references (74 hours of audio). Olaf generates the biggest index with almost 350 MB, 18 times the size of the Audfprint index. Olaf would take around 1.75 TB for an industrial database of 10 million references while Audfprint would take 95 GB.

Experiments have been run in a reproduceable, isolated environment. All audios have been loaded on the RAM disk and ran on a 98GB RAM server with two 16-cores CPUs at 2.60GHz. We have executed each algorithm with multiprocessing (when it was possible) and cleared the cache before each run. The results in Table 4 are normalized to one single thread.

5. CONCLUSIONS AND FUTURE WORK

We present a new dataset for broadcast monitoring with 57 hours of TV broadcast recordings, 74 hours of production music, and over 37 hours of human cross-annotations. All audios are monaural sampled at 8kHz and more than 80% of the annotated music is in the background. The Benchmark of state-of-the-art public algorithms shows that AFP for broadcast monitoring with a high presence of background music is yet to be solved. For this task, in addition to other metrics used in the literature, we propose using the Match Ratio and analyzing Precision, Recall, and F1-score, especially in terms of seconds identified. We also provide a simple AFP baseline named PeakFP.

In future experiments, we plan to add noise tracks to the references set to study the evolution of False Positives and the scalability of each algorithm. We will get more insight into their behavior for different levels of background music, and we will benchmark more AFP algorithms.

6. ACKNOWLEDGEMENTS

The authors would like to thank Carl Thomé for making this collaboration possible. This research is part of *NextCore – New generation of music monitoring technology (RTC2019-007248-7)*, funded by the Spanish Ministerio de Ciencia e Innovación and the Agencia Estatal de Investigación. Also, has received support from Industrial Doctorates plan of the Secretaria d'universitats i Recerca, Departament d'Empresa i Coneixement de la Generalitat de Catalunya, grant agreement No. DI46-2020.

7. REFERENCES

[1] P. Cano, E. Batlle, E. Gómez, L. de C. T. Gomes, and M. Bonnet, "Audio fingerprinting: Concepts and applications," in *Computational Intelligence for Modelling and Prediction*, ser. Studies in Computational Intelligence, S. K. Halgamuge and L. Wang, Eds. Springer, 2005, vol. 2, pp. 233–245. [Online]. Available: https://doi.org/10.1007/10966518_17

[2] A. Wang, "An industrial strength audio search algorithm," in *ISMIR 2003, 4th International Conference on Music Information Retrieval, Baltimore, Maryland, USA, October 27-30, 2003, Proceedings*, 2003, pp. 7–13.

[3] J. R. Cerquides, "A real time audio fingerprinting system for advertisement tracking and reporting in fm radio," in *2007 17th International Conference Radioelektronika*. IEEE, 2007, pp. 1–4.

[4] E. Gomez, P. Cano, L. Gomes, E. Batlle, and M. Bonnet, "Mixed watermarking-fingerprinting approach for integrity verification of audio recordings," in *Proceedings of the International Telecommunications Symposium*, 2002.

[5] C. J. C. Burges, D. Plastina, J. C. Platt, E. Renshaw, and H. S. Malvar, "Using audio fingerprinting for duplicate detection and thumbnail generation," in *2005 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '05, Philadelphia, Pennsylvania, USA, March 18-23, 2005*. IEEE, 2005, pp. 9–12. [Online]. Available: https://doi.org/10.1109/ICASSP.2005.1415633

[6] IFPI, "Global music report 2022," https://www.ifpi.org/wp-content/uploads/2022/04/IFPI_Global_Music_Report_2022-State_of_the_Industry.pdf, 4 2022, [Accessed August 2022].

[7] B. Meléndez Catalán *et al.*, "Relative music loudness estimation in tv broadcast audio using deep learning: an industrial perspective," Ph.D. dissertation, Universitat Pompeu Fabra, 2021.

[8] A. G. Piotrowska, "Analyzing music in tv shows: some methodological considerations," *The science of television*, no. 14.3, pp. 10–27, 2018.

[9] "Epidemic sound," https://www.epidemicsound.com/, 2009, [Accessed August 2022].

[10] D. Ellis, "The 2014 labrosa audio fingerprint system," in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, TW, October 27-31, 2014*, 2014.

[11] J. Six and M. Leman, "Panako - A scalable acoustic fingerprinting system handling time-scale and pitch modification," in *Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR 2014, Taipei, Taiwan, October 27-31, 2014*, H. Wang, Y. Yang, and J. H. Lee, Eds., 2014, pp. 259–264. [Online]. Available: http://www.terasoft.com.tw/conf/ismir2014/proceedings/T048_122_Paper.pdf

[12] J. Six, "Panako 2.0-updates for an acoustic fingerprinting system," in *Demo / late-breaking abstracts of 22st International Society for Music Information Retrieval Conference, ISMIR 2021, Online, November 8-12, 2021*, 2021.

[13] ——, "Olaf: Overly lightweight acoustic fingerprinting," in *Demo / late-breaking abstracts of 21st International Society for Music Information Retrieval Conference, ISMIR 2020, Montréal, Canada, CA, October 11-16, 2020*, 2020.

[14] S. Chang, D. Lee, J. Park, H. Lim, K. Lee, K. Ko, and Y. Han, "Neural audio fingerprint for high-specific audio retrieval based on contrastive learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2021, Toronto, ON, Canada, June 6-11, 2021*. IEEE, 2021, pp. 3025–3029. [Online]. Available: https://doi.org/10.1109/ICASSP39728.2021.9414337

[15] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system," in *ISMIR 2002, 3rd International Conference on Music Information Retrieval, Paris, France, October 13-17, 2002, Proceedings*, 2002, pp. 107–115. [Online]. Available: http://ismir2002.ismir.net/proceedings/02-FP04-2.pdf

[16] M. A. Bartsch and G. H. Wakefield, "Audio thumbnailing of popular music using chroma-based representations," *IEEE Trans. Multim.*, vol. 7, no. 1, pp. 96–104, 2005. [Online]. Available: https://doi.org/10.1109/TMM.2004.840597

[17] S. Baluja and M. Covell, "Waveprint: Efficient wavelet-based audio fingerprinting," *Pattern Recognit.*, vol. 41, no. 11, pp. 3467–3480, 2008. [Online]. Available: https://doi.org/10.1016/j.patcog.2008.05.006

[18] C. Bellettini and G. Mazzini, "Reliable automatic recognition for pitch-shifted audio," in *Proceedings of the 17th International Conference on Computer Communications and Networks, IEEE ICCCN 2008, St. Thomas, U.S. Virgin Islands, August 3-7, 2008*.

IEEE, 2008, pp. 838–843. [Online]. Available: https://doi.org/10.1109/ICCCN.2008.ECP.157

[19] ——, "A framework for robust audio fingerprinting," *J. Commun.*, vol. 5, no. 5, pp. 409–424, 2010. [Online]. Available: https://doi.org/10.4304/jcm.5.5.409-424

[20] S. Fenet, G. Richard, and Y. Grenier, "A scalable audio fingerprint method with robustness to pitch-shifting," in *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011, Miami, Florida, USA, October 24-28, 2011*, A. Klapuri and C. Leider, Eds. University of Miami, 2011, pp. 121–126. [Online]. Available: http://ismir2011.ismir.net/papers/PS1-14.pdf

[21] M. Malekesmaeili and R. K. Ward, "A local fingerprinting approach for audio copy detection," *Signal Process.*, vol. 98, pp. 308–321, 2014. [Online]. Available: https://doi.org/10.1016/j.sigpro.2013.11.023

[22] X. Zhang, B. Zhu, L. Li, W. Li, X. Li, W. Wang, P. Lu, and W. Zhang, "Sift-based local spectrogram image descriptor: a novel feature for robust music identification," *EURASIP J. Audio Speech Music. Process.*, vol. 2015, p. 6, 2015. [Online]. Available: https://doi.org/10.1186/s13636-015-0050-0

[23] R. Sonnleitner and G. Widmer, "Robust quad-based audio fingerprinting," *IEEE ACM Trans. Audio Speech Lang. Process.*, vol. 24, no. 3, pp. 409–421, 2016. [Online]. Available: https://doi.org/10.1109/TASLP.2015.2509248

[24] R. Sonnleitner, A. Arzt, and G. Widmer, "Landmark-based audio fingerprinting for DJ mix monitoring," in *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, M. I. Mandel, J. Devaney, D. Turnbull, and G. Tzanetakis, Eds., 2016, pp. 185–191. [Online]. Available: https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/07/187_Paper.pdf

[25] T. Walther and M. D. Gould, "Audio identification method," Worldwide Patent WO/2016/189 307, 2016.

[26] B. A. y Arcas, B. Gfeller, R. Guo, K. Kilgour, S. Kumar, J. Lyon, J. Odell, M. Ritter, D. Roblek, M. Sharifi, and M. Velimirovic, "Now playing: Continuous low-power music recognition," *CoRR*, vol. abs/1711.10958, 2017. [Online]. Available: http://arxiv.org/abs/1711.10958

[27] H. Son, S. Byun, and S. Lee, "A robust audio fingerprinting using a new hashing method," *IEEE Access*, vol. 8, pp. 172 343–172 351, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3024951

[28] Z. Yu, X. Du, B. Zhu, and Z. Ma, "Contrastive unsupervised learning for audio fingerprinting," *CoRR*, vol. abs/2010.13540, 2020. [Online]. Available: https://arxiv.org/abs/2010.13540

[29] E. Law, K. West, M. I. Mandel, M. Bay, and J. S. Downie, "Evaluation of algorithms using games: The case of music tagging," in *Proceedings of the 10th International Society for Music Information Retrieval Conference, ISMIR 2009, Kobe International Conference Center, Kobe, Japan, October 26-30, 2009*, K. Hirata, G. Tzanetakis, and K. Yoshii, Eds. International Society for Music Information Retrieval, 2009, pp. 387–392. [Online]. Available: http://ismir2009.ismir.net/proceedings/OS5-5.pdf

[30] M. Ramona and G. Peeters, "Audio identification based on spectral modeling of bark-bands energy and synchronization through onset detection," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*. IEEE, 2011, pp. 477–480. [Online]. Available: https://doi.org/10.1109/ICASSP.2011.5946444

[31] A. F. Smeaton, P. Over, and W. Kraaij, "Evaluation campaigns and trecvid," in *Proceedings of the 8th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR 2006, October 26-27, 2006, Santa Barbara, California, USA*, J. Z. Wang, N. Boujemaa, and Y. Chen, Eds. ACM, 2006, pp. 321–330. [Online]. Available: https://doi.org/10.1145/1178677.1178722

[32] G. Awad, P. Over, and W. Kraaij, "Content-based video copy detection benchmarking at TRECVID," *ACM Trans. Inf. Syst.*, vol. 32, no. 3, pp. 14:1–14:40, 2014. [Online]. Available: https://doi.org/10.1145/2629531

[33] X. Anguera, A. Garzon, and T. Adamek, "MASK: robust local features for audio fingerprinting," in *Proceedings of the 2012 IEEE International Conference on Multimedia and Expo, ICME 2012, Melbourne, Australia, July 9-13, 2012*. IEEE Computer Society, 2012, pp. 455–460. [Online]. Available: https://doi.org/10.1109/ICME.2012.137

[34] C. Ouali, P. Dumouchel, and V. Gupta, "A robust audio fingerprinting method for content-based copy detection," in *12th International Workshop on Content-Based Multimedia Indexing, CBMI 2014, Klagenfurt, Austria, June 18-20, 2014*. IEEE, 2014, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CBMI.2014.6849814

[35] Z. J. Guzman-Zavaleta, C. F. Uribe, A. Menendez-Ortiz, and J. J. Garcia-Hernandez, "A robust audio fingerprinting method using spectrograms saliency maps," in *9th International Conference for Internet Technology and Secured Transactions, ICITST 2014, London, United Kingdom, December 8-10, 2014*. IEEE, 2014, pp. 47–52. [Online]. Available: https://doi.org/10.1109/ICITST.2014.7038773

[36] NIST-TRECVID, "Trecvid 2011 - content-based copy detection task," https://www-nlpir.nist.gov/projects/

tv2011/index.html#ccd, 2010, [Accessed August 2022].

[37] J. D. of Computational Perception, "Audio fingerprinting data sets," http://www.cp.jku.at/datasets/fingerprinting, 4 2017, [Accessed August 2022].

[38] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, S. J. Cunningham, Z. Duan, X. Hu, and D. Turnbull, Eds., 2017, pp. 316–323. [Online]. Available: https://ismir2017.smcnus.org/wp-content/uploads/2017/10/75_Paper.pdf

[39] T. Gebru, J. Morgenstern, B. Vecchione, J. W. Vaughan, H. M. Wallach, H. D. III, and K. Crawford, "Datasheets for datasets," *Commun. ACM*, vol. 64, no. 12, pp. 86–92, 2021. [Online]. Available: https://doi.org/10.1145/3458723

[40] Tensorflow, "Sound classification with yamnet," https://www.tensorflow.org/hub/tutorials/yamnet, 2022, [Accessed August 2022].

[41] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2017, New Orleans, LA, USA, March 5-9, 2017*. IEEE, 2017, pp. 776–780. [Online]. Available: https://doi.org/10.1109/ICASSP.2017.7952261

[42] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audioset ontology," https://research.google.com/audioset/ontology/index.html, 2017, [Accessed August 2022].

[43] MIREX, "2019: Music detection results," https://www.music-ir.org/mirex/wiki/2019:Music_Detection_Results, 2019, [Accessed August 2022].

[44] B. Meléndez-Catalán, "Relative music loudness estimation using temporal convolutional networks and a cnn feature extraction front-end," in *Proceedings of the 23rd International Conference on Digital Audio Effects (DAFx-20)*, vol. 5, 2020, pp. 273–280.

[45] B. Meléndez-Catalán, E. Molina, and E. Gómez, "Open broadcast media audio from TV: A dataset of TV broadcast audio with relative music loudness annotations," *Trans. Int. Soc. Music. Inf. Retr.*, vol. 2, no. 1, pp. 43–51, 2019. [Online]. Available: https://doi.org/10.5334/tismir.29

[46] J. L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

[47] A. J. Viera, J. M. Garrett *et al.*, "Understanding interobserver agreement: the kappa statistic," *Fam med*, vol. 37, no. 5, pp. 360–363, 2005.

[48] H. Son, S. W. Byun, and S. Lee, "Illegal audio copy detection using fundamental frequency map," in *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE 2019 - Volume 1: DCNET, ICE-B, OPTICS, SIGMAP and WINSYS, Prague, Czech Republic, July 26-28, 2019*, M. S. Obaidat, C. Callegari, M. van Sinderen, P. Novais, P. G. Sarigiannidis, S. Battiato, Á. S. S. de León, P. Lorenz, and F. Davoli, Eds. SciTePress, 2019, pp. 356–361. [Online]. Available: https://doi.org/10.5220/0008113403500355

[49] R. Sonnleitner and G. Widmer, "Quad-based audio fingerprinting robust to time and frequency scaling," in *Proceedings of the 17th International Conference on Digital Audio Effects, DAFx-14, Erlangen, Germany, September 1-5, 2014*, S. Disch, J. Herre, R. Rabenstein, B. Edler, M. Müller, and S. Turowski, Eds., 2014, pp. 173–180. [Online]. Available: http://www.dafx14.fau.de/papers/dafx14_reinhard_sonnleitner_quad_based_audio_fingerpr.pdf

[50] L. Lalinský, "Chromaprint," https://acoustid.org/chromaprint, [Accessed August 2022].

[51] Y. Ke, D. Hoiem, and R. Sukthankar, "Computer vision for music identification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005), 20-26 June 2005, San Diego, CA, USA*. IEEE Computer Society, 2005, pp. 597–604. [Online]. Available: https://doi.org/10.1109/CVPR.2005.105

[52] M. Ramona and G. Peeters, "Audioprint: An efficient audio fingerprint system based on a novel cost-less synchronization scheme," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*. IEEE, 2013, pp. 818–822. [Online]. Available: https://doi.org/10.1109/ICASSP.2013.6637762

CADENCE DETECTION IN SYMBOLIC CLASSICAL MUSIC USING GRAPH NEURAL NETWORKS

Emmanouil Karystinaios[1] **Gerhard Widmer**[1,2]
[1] Institute of Computational Perception, Johannes Kepler University Linz, Austria
[2] LIT AI Lab, Linz Institute of Technology, Austria
firstname.lastname@jku.at

ABSTRACT

Cadences are complex structures that have been driving music from the beginning of contrapuntal polyphony until today. Detecting such structures is vital for numerous MIR tasks such as musicological analysis, key detection, or music segmentation. However, automatic cadence detection remains challenging mainly because it involves a combination of high-level musical elements like harmony, voice leading, and rhythm. In this work, we present a graph representation of symbolic scores as an intermediate means to solve the cadence detection task. We approach cadence detection as an imbalanced node classification problem using a Graph Convolutional Network. We obtain results that are roughly on par with the state of the art, and we present a model capable of making predictions at multiple levels of granularity, from individual notes to beats, thanks to the fine-grained, note-by-note representation. Moreover, our experiments suggest that graph convolution can learn non-local features that assist in cadence detection, freeing us from the need of having to devise specialized features that encode non-local context. We argue that this general approach to modeling musical scores and classification tasks has a number of potential advantages, beyond the specific recognition task presented here.

1. INTRODUCTION

Graph Neural Networks (GNNs) have recently seen staggering successes in various fields. The MIR community has also experienced the influence of GNNs, principally in the field of recommender systems [1]. However, other sub-branches of MIR could potentially enjoy the graph representation and the benefits of graph deep learning.

Modeling musical scores in all their complexity has been challenging, with many approaches resorting to piano rolls [2], note arrays [3], or custom descriptors [4]. In this paper, we present a new representation of the score as a homogeneous graph with note-wise features to model aspects of the score. We use this representation to address the cadence detection task using graph neural networks, treating the task as a node classification problem. More specifically, our contribution is two-fold: a simple graph representation of scores extended with local features, and a Graph Convolutional Network (GCN) model to tackle heavily imbalanced classification tasks such as Cadence Detection. *Score modeling* itself has two aspects: (1) the construction of the graph, i.e., what are the nodes, and which connections do we define between them; and (2) the choice of score features, and how these relate to their respective graph nodes. The *classification model* is an adapted version of GraphSMOTE [5], a Graph Convolutional Network designed to deal with imbalanced classification problems, which we modified to deal with larger graphs and apply stochastic training. Henceforth, we call this model *Stochastic GraphSMOTE*. We employ this model on top of our score modeling with the intention of solving the Cadence Detection task.

The cadence detection setting is binary, i.e., there is a cadence (maybe of a specific type) or not. The current state of the art [4] uses an Support Vector Machine (SVM) classifier on a set of custom-designed cadence-specific features, based on three defined "cadence anchor points", and performs score/feature modeling and cadence classification at the level of beats. The model was tested on two annotated datasets: 24 Bach fugues and 42 Haydn string quartet expositions. Our new model proposed here will be shown to achieve comparable overall results; however, we will argue that it makes fewer task-related and musical assumptions, resulting in more general applicability. In particular, our empirical results suggest that by providing local features and applying a Graph Neural Network with neighbor convolution, we can learn nonlocal aspects that help improve prediction. This gives a more general approach for a variety of tasks where features are provided at the level of notes, but prediction may be note-wise, onset-wise, or beat-wise.

The rest of the paper is structured as follows. Section 2 discusses related work on cadence detection and music score modeling. Section 3 describes the score model and the graph construction from the score, section 4 introduces the corpora, and section 5 presents the proposed learning algorithm. Section 6 presents a series of three experiments and also takes a qualitative look at some examples. Finally, section 7 summarizes and concludes.

© E. Karystinaios, G. Widmer. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** E. Karystinaios, G. Widmer, "Cadence Detection in Symbolic Classical Music using Graph Neural Networks", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

2. RELATED WORK

Graphs have emerged as a natural representation of music since the development of Tonnetz by Euler. Since then, there have been various proposals to use graph representations for addressing music analysis and MIR tasks. For instance (to name just two), [6] introduced relational *Klumpenhouwer networks* for music analysis, and [7] used *Tonnetz trajectories* for composer classification. One can distinguish between *heterogeneous* and *homogeneous* graphs [8]. Heterogeneous graphs may have multiple types of edges and nodes, while homogeneous graphs are simpler, containing only a single edge and node type. Recently, the creators of VirtuosoNet, a computational model for generating piano performances, used a heterogeneous graph representation of the score and trained their system using a Graph Neural Network [9]. However, in later publications, they reverted to a model without using graphs which achieved better performance [10]. In the present paper, we wish to show that a simple, homogeneous graph representation can form a natural and general basis for modeling a non-trivial music analysis task.

Automatic *cadence detection* is a challenging task. Although cadences are well established concepts, their definition or annotation in music can cause disagreements among musicologists. Previous work on automatic cadence detection has been done by [11] on Bach fugues and by [12] for a generalized classical music analysis system. A feature-based approach using standard Machine Learning classifiers is presented in [4] which represents the current state of the art. Recently, Sears and Widmer [13] highlighted the difficulty of detecting textbook voice leading schemata that occur near cadences in written music. However, to our knowledge, there exists no method employing deep learning models to solve the task.

3. MODELING SCORES AS A GRAPH

We model a score as a graph with individual notes and rests as nodes and simple temporal relations as edges. In addition, each graph node is associated with a vector of feature values that represent some basic properties of a note and its immediate context. Formally, let $G = (V, E)$ be a graph, where V is the set of nodes and $E \subseteq V \times V$ the set of edges and let A be the adjacency matrix of G. Each note and each rest in a score are represented as a node in the graph. We create three types of undirected connections between notes/rests: edges E_{on} between notes that occur on the same onset; edges E_{cons} between consecutive notes, and edges E_{dur} between a note of longer duration and notes whose onsets occur during this time:

$$
\begin{aligned}
E_{on} &= \{(i,j) \mid on(n_i) = on(n_j)\} \\
E_{cons} &= \{(i,j) \mid on(n_i) + dur(n_i) = on(n_j)\} \\
E_{dur} &= \{[(i,j) \mid on(n_i) + dur(n_i) > on(n_j)] \wedge \\
&\qquad [on(n_i) < on(n_j)]\} \\
E &= E_{on} \cup E_{cons} \cup E_{dur}
\end{aligned}
$$

where n_i is the i^{th} note. on denotes the onset of a note, dur the duration. All edges in E are undirected.

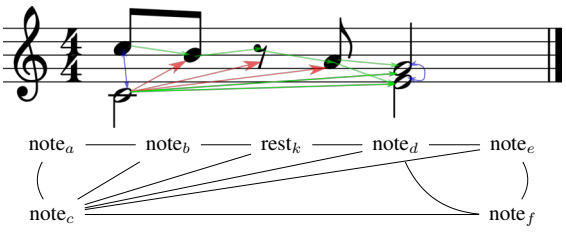


Figure 1. Example graph creation from a score following the process described in the text. E_{on} is denoted in blue, E_{cons} in green, and E_{dur} in red. Global attributes such as time and key signatures are added as node features.

3.1 Feature Overview

We use three types of features to further describe a note: [1] general-purpose note-level features to describe a note and its immediate rhythmic/melodic context; general graph topology features to capture aspects of local connectivity; and cadence-specific note features inspired by [4]. The third feature category is the only one that is designed with the specific classification target in mind; however, in contrast to [4], we restrict these to only consider the immediate local context of a note instead of using positional features relating to predefined past "cadence anchor points". In this way, we wish to demonstrate the generality of our representation and learning approach, which will hopefully learn more long-distance aspects automatically, as needed.

The first category, *general note-wise features*, is the largest one. For each note in the score, we extract onset time expressed in score-relative beats, duration in beats, and MIDI pitch, using the partitura package [14]. Furthermore, we translate global attributes such as time signature and assign them to each note. Also using partitura, we extract a set of generic note-wise features as defined in [15]. Finally, we extract features summarizing intervallic information at the time of onset of each note. These include *interval vectors* [16] and binary features activated when intervallic content is identical to the interval set corresponding to particular chord types, i.e., major, minor, diminished, etc.

Second, we add *graph-aware features* using the first 20 eigenvectors from the Laplacian of the adjacency matrix [17].

The final category contains *note-wise cadence-related features* similar to those in [4], such as voice leading information and voicing. However, our features are calculated at the note level only, considering the time of onset for each note and its immediate neighbors, such as adjacent past onsets or simultaneous onsets. In particular, we do not use any information about events that occur on previous beats. While these features are more restricted compared to [4] they are also more general, since we make no assumptions on and reference to "cadence anchor points" (e.g.,

[1] Code and a complete specification of all features is available on `https://github.com/manoskary/cadet`.

the occurrence of the preceding subdominant and dominant harmony), which in [4] are identified with specialized heuristics. In total, we store 135 features per node.

4. PROBLEM SETTING & CORPORA

In this work, we are interested in cadences of the Baroque and Classical periods. The main focus will be on detecting Perfect Authentic Cadences (PAC); where our annotated datasets permit, we will also consider root position Imperfect Authentic Cadences (rIAC) and Half Cadences (HC). The manual annotations in these datasets mark a cadence as occurring on the beat where the final I (i) arrives. Our precise task thus is to predict, for every note of the score, whether this note is contained in a cadence's arrival beat.

To benchmark our method, we used two datasets also used by Bigo et al. [4], and a third one annotated by Allegraud and al. [18]. The first set contains the 24 fugues from Bach's Well-tempered Clavier, Book I. The cadence annotations were presented in [11]. The second dataset contains 45 movement expositions from Haydn string quartets; the cadence annotations were produced by Sears and colleagues [19]. The last dataset contains 31 movements of Mozart string quartets with cadence annotations included. All the scores were retrieved from `http://kern.ccarh.org` and were parsed in python using the partitura package [14]. [2]

Cadences occur with low frequency in music. In particular, for the corpora we cover in this paper, cadences of all types combined account for less than 2% of the total notes in the score. Our produced score graphs range from approximately 25k nodes for the Bach fugues all the way to 70k nodes for the Mozart string quartets with more than 750k edges. Table 1 gives detailed dataset statistics.

5. MODEL

5.1 Graph Convolutional Network

The authors of [4] underline the importance of non-fixed positions for the cadence anchor points. We address this by employing a graph convolutional network. Graph Convolution Networks (GCNs) are based on the same principle as CNNs, but in the context of graphs we encounter the message passing concept, meaning convolution occurs only among nodes connected by edges. This theoretically allows local features to connect with distant features of their k-hop neighbors. Therefore, graph representation can learn, using local node information, higher lever information by sampling information from neighbors. Figure 2 illustrates the neighbor sampling concept.

For our model, we propose *Stochastic GraphSMOTE*, a Graph Convolutional Network with a built-in graph Auto-Encoder and Synthetic Minority Over-sampling for imbalanced node classification. The model consists of 4 parts, the encoder, a SMOTE layer in the encoder's latent space, the decoder, and the classifier. The structure of the model

[2] For reproducibility, we provide the generated graphs that were used for training on `https://github.com/manoskary/tonnetzcad`

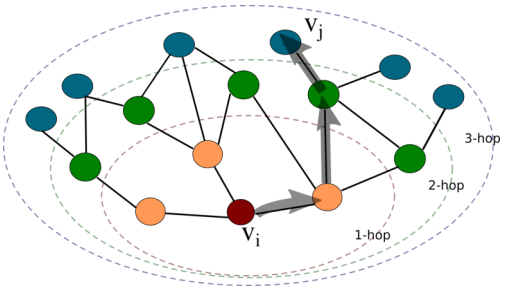


Figure 2. Multi-hop Neighborhood sampling. v_j is 3-hop neighbor of v_i. Color cues mark the k-hop neighborhoods occurring within the ellipses. The arrows demonstrate a random walk starting from v_i and ending at v_j.

follows GraphSMOTE [5] but with some major differences, mainly to adapt for stochastic training, which is needed because of the large size of our score graphs.

The encoder applied to a node i is defined as a standard GraphSAGE [20] stack given by:

$$\mathbf{h}_{\mathcal{N}(i)}^{(l+1)} = \text{mean}\left(\{\mathbf{W}_{pool}^{(l+1)} \cdot \mathbf{h}_j^l, \forall j \in \mathcal{N}(i)\}\right)$$
$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}_{enc}^{(l+1)} \cdot \text{concat}(\mathbf{h}_i^l, \mathbf{h}_{\mathcal{N}(i)}^{l+1})\right)$$
$$\mathbf{h}_i^{(l+1)} = \text{norm}(\mathbf{h}_i^l)$$

where $h_i^{(l)}$ is the hidden representation of node i on layer l, σ is an activation function, norm is a normalization function, $\mathbf{W}$ are learnable weights, and $\mathcal{N}(i) = \{j \mid (i,j) \in E\}$ are the neighbors of node i. Let $B \subseteq V$ a subset of nodes denoting a batch sample. Then, given L the total number of hidden layers, $\mathbf{H}_B^{(enc)} = \{\mathbf{h}_u^{(L+1)} \mid u \in B\}$.

5.2 Dealing with Extreme Class Imbalance: Stochastic GraphSMOTE

Since cadences are very sparse, we need to introduce a balancing technique in order to avoid gradient convergence that will result in predicting only the majority class, i.e., absence of cadence. To counter this effect, we introduce a SMOTE layer that is applied in the *latent space* of the encoder. SMOTE generates synthetic samples with the same label as the minority class (see [21] for details). The main novelty of our model is that the SMOTE is performed for each batch separately.

In each batch, we count the occurrence, μ_i, for each of the classes $i \in I$. In the binary setting, let μ_M be the number of samples with the same label as the majority class and μ_m be the number of samples with the same labels as the minority class. By generating $(\mu_M - \mu_m)$ samples with the same label as the minority class, we force a $1:1$ binary class distribution. To generate these samples, in each batch, we randomly select a sample instance of the minority class as an anchor point and gather the k nearest neighbor samples of the same class within the batch. Finally, μ samples are generated as random linear interpolations between a randomly selected neighbor out of the k, and the selected anchor point in the euclidean space. Performing

Dataset	Pieces	Nodes	Edges	PAC	rIAC	HC
Bach Fugues	24	24,567	229,107	237	78	15
Haydn String Quartets	45	38,661	441,491	434	24	340
Mozart String Quartets	31	68,190	762,796	1,089	-	1,930

Table 1. Cadence nodes constitute less than 2% of all nodes.

SMOTE in the latent space assumes that a more appropriate representation for the generation of the synthetic minority samples is learned.

If $\mathbf{H}_B^{(enc)}$ is the hidden representation of the batch sampled nodes after the encoder layer, then $\mathbf{H}_B^{(smote)}$ is the SMOTE upsampling algorithm applied on $\mathbf{H}_B^{(enc)}$. Our Decoder layer is responsible for generating edges within the original nodes of the graph and the synthetic ones, created by SMOTE. The decoder output is described by the following equation:

$$\mathbf{A}_B^{(dec)} = \sigma\left(\mathbf{H}_B^{(smote)} \cdot \mathbf{W}^{(dec)} \cdot \text{transpose}(\mathbf{H}_B^{(smote)})\right)$$
$$\mathbf{A}_B^{(thr)} = \text{hardshrink}\left(\mathbf{A}_B^{(dec)}, \tau\right)$$

where $W^{(dec)}$ are the decoder's learnable weights, σ is a sigmoid activation function, and hardshrink is the hard shrinkage function with threshold τ. $\mathbf{A}_B^{(dec)}$ is the generated adjacency from the decoder and $\mathbf{A}_B^{(thr)}$ is a thresholded adjacency by a factor τ.

We define a regularization loss that aims at constraining the generated adjacency close to the original, defined by:

$$\mathcal{L}_B^{(dec)} = \text{BCE}\left(\mathbf{A}_B^{(dec)}, \mathbf{A}_B\right)$$

where BCE is the binary cross entropy loss, $\mathbf{A}_B^{(dec)}$ is the generated adjacency of the decoder for batch sample B and A_B is the adjacency matrix for batch sample B. Since we learn an edge generator which is good at reconstructing the adjacency matrix using the encoder's latent representations, it should also give adequate edge predictions for synthetic nodes.

The GNN classifier is composed of a GraphSAGE layer [20] with a linear layer on top. By adding a graph convolution layer such as GraphSAGE in the classifier, we can benefit from learning information from the generated adjacency and the neighbors of nodes. The GraphSAGE layer of the classifier is slightly different from the encoder because it performs directly on the generated thresholded adjacency of each batch sample:

$$\mathbf{h}_{\mathcal{N}(i)}^{(clf)} = \text{mean}\left(\mathbf{W}^{(pool)} \cdot \mathbf{A}_B^{(thr)}[i,:] \cdot \mathbf{H}_B^{(enc)}\right)$$
$$\mathbf{h}_i^{(clf)} = \text{norm}\left(\sigma\left(\mathbf{W}^{(clf)} \cdot \text{concat}\left(\mathbf{h}_i^{(enc)}, \mathbf{h}_{\mathcal{N}(i)}^{(clf)}\right)\right)\right)$$
$$\mathbf{h}_i^{(clf)} = \text{softmax}(\mathbf{W}^{(proj)} \cdot \mathbf{h}_i^{(clf)})$$

where $\mathbf{h}_i^{(clf)}$ are the predicted class probabilities of node i, $\mathbf{W}$ are learnable weights, $\mathbf{A}_B^{(thr)}$ is the generated thresholded adjacency from the decoder, $\mathbf{H}_B^{(enc)}$ are the batch encodings of the encoder and $\mathbf{h}_i^{(enc)}$ is the encoder's output for node i. During training, we use $\mathbf{H}_B^{(smote)}$ and $\mathbf{h}_i^{(smote)}$ respectively instead of $\mathbf{H}_B^{(enc)}$ and $\mathbf{h}_i^{(enc)}$. We define the *total loss* of our model for batch samples B:

$$\mathcal{L}_B^{(tot)} = \mathcal{L}_B^{(CE)} + \gamma * \mathcal{L}_B^{(dec)}$$

where $\mathcal{L}_{CE}$ signifies the cross entropy loss and γ is a hyper parameter.

Our model is trained stochastically, meaning that to create each batch a subset B of nodes are sampled. From these sampled nodes, given a pre-defined depth k, we retrieve the immediate neighbors of every $v \in B$ up to their k-hop neighbors in the graph G. We use neighbor sampling to reduce the cost of retrieving all up to k-hop neighbors of v by defining a maximum number ϕ_l of neighbors per depth layer l.

6. EXPERIMENTS

We conduct three main experiments. The first compares our model to the state of the art results in [4], using the same data and train/test setup. The second experiment focuses on multi-class learning of the particular type of cadence using different sets of features, in order to investigate how the model generalizes to a more complex setting and inspect the relevance of different feature sets. The third experiment investigates how neighbor convolution contributes to the model's performance. [3]

We fix our model with a hidden dimension of 256, with $L = 2$ hidden layers with $\phi_1 = 10$ and $\phi_2 = 25$ sampled neighbors for hidden layers 1 and 2 of the encoder, respectively, and one hidden layer of the same dimension for the classifier. The learning rate is set at 0.007, the weight-decay at 0.007, with a batch size of 1024, $k = 3$ for SMOTE, the decoder regularization loss multiplier $\gamma = 0.5$, and adjacency threshold value $\tau = 0.5$.

6.1 Quantitative Results

Table 2 summarizes the results of the first experiment, comparing our model's performance to the state of the art. [4] The reference model [4] can only classify at the beat level; our representation and classification model are more flexible in this regard, as they have access to, and describe, individual notes. In particular, our model can provide predictions at three different levels, note-wise, onset-wise and beat-wise predictions (the latter two simply by aggregation). In Table 2 we present the results of these predictions at all levels,

[3] All results, experiments, and the trained models are available on `https://wandb.ai/melkisedeath/CadenceDetection`

[4] In accordance with [4], we ignore the HC in Bach and rIAC in Haydn, because of their low numbers.

Dataset	Model	F1 Note	F1 Onset	F1 Beat	Prec. Beat	Recall Beat
Bach Fugues (PAC) (12 fugues)	Bigo et al. model	-	-	**0.80**	**0.89**	0.72
	SGSMOTE	0.85	0.75	0.73	0.70	0.77
	Pretrained SGSMOTE	**0.90**	**0.83**	**0.80**	0.74	**0.89**
Bach Fugues (rIAC) (12 fugues)	Bigo et al. model	-	-	0.68	0.71	0.65
	SGSMOTE	**0.87**	**0.75**	**0.73**	**0.75**	0.72
	Pretrained SGSMOTE	0.87	0.73	0.71	0.62	**0.82**
Haydn String Quartets (PAC) (21 pieces)	Bigo et al. model	-	-	**0.69**	**0.60**	**0.82**
	SGSMOTE	0.77	0.56	0.59	0.47	0.78
	Pretrained SGSMOTE	**0.81**	**0.63**	0.64	0.54	0.78
Haydn String Quartets (HC) (21 pieces)	Bigo et al. model	-	-	0.29	0.19	**0.56**
	SGSMOTE	0.65	0.32	0.30	0.33	0.27
	Pretrained SGSMOTE	**0.69**	**0.44**	**0.41**	**0.41**	0.41

Table 2. Results using half of the dataset for training, half for testing. Bach: fugues no.1-12 were used for training, no.13-24 for testing; Haydn: random 21:21 split. The pretrained network was trained on the other dataset, i.e. *Pretrained SGSMOTE* for Bach Fugues was pre-trained on string quartets, etc. Classification is binary, the presented $F1$ scores are for the positive class, i.e., the cadence (PAC: Perfect Authentic Cadence; rIAC: root position Imperfect AC; HC: Half Cadence).

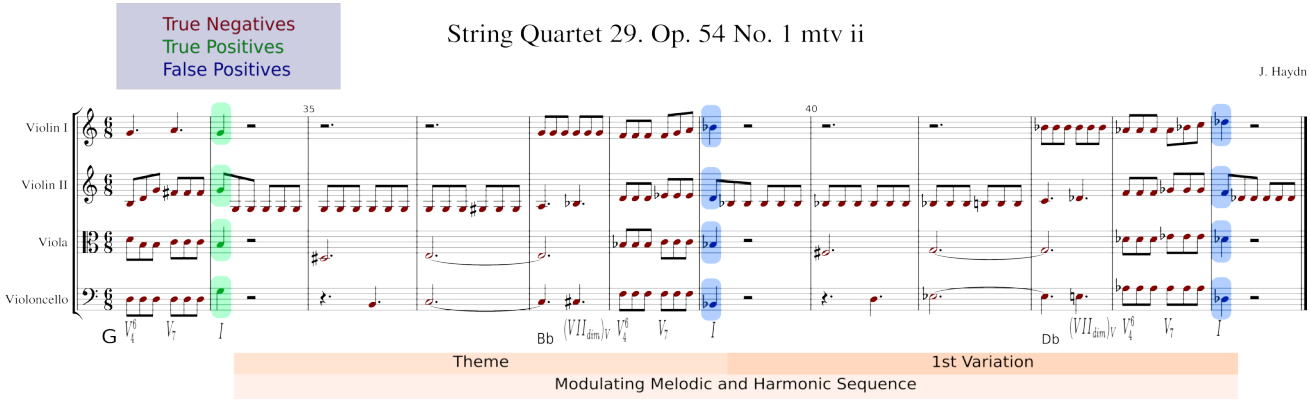


Figure 3. Haydn's String Quartet 29. Op.54 No.1 Mvt. II, mm. 33-45. Showing the output of the Stochastic GraphSMOTE Network for PAC prediction. True negatives are marked with red, true positives with green, false positives with blue. A partial analysis shows the chords towards the end of cadences and highlights a modulating sequence where every sequence ends with a cadential pattern, which counts as false positive predictions by the network.

in terms of F1 score. Only beat-wise scores are given for the reference model (taken from [4]). The last two columns of table 2 give the recall and precision for the beat-wise prediction. All metrics are presented for the positive, i.e. minority/cadence, class.

Our model matches or slightly surpasses the state of the art in rIAC detection in Bach fugues and on HCs in Haydn string quartets but does not reach the reference model's F1 results in PAC detection. We additionally present a pre-trained version of Stochastic GraphSMOTE, where the network was first trained on additional data and fine-tuned for the task. Specifically, the network for PAC prediction in Bach was pre-trained on the string quartets and vice versa. Pre-training, and thus the need for additional data, is the price we pay for the generality of the graph representation and the consequent size (number of parameters) of the deep network. Pre-training helps to (markedly) improve the results on HC, catch up with the reference on PAC in Bach, and narrow the gap on PAC in Haydn.

Generally, our results agree with [4] in implying that half cadences (HC) seem significantly harder to identify than authentic cadences, both perfect and imperfect. Another, more specific, observation concerns different ways in which the compared models achieve their overall F1 scores. In the PAC detection tasks, in particular, we observe comparable or higher recall of our model compared to the reference, but lower precision. This observation motivated us to check some of our model's false positive predictions; Section 6.2 below will show several instructive examples of 'almost correct' identifications.

The *second experiment* we conducted (Table 3) focuses on comparing the relevance of feature groups. For compactness we present here a multi-class classification scenario where we account not only for the existence of a cadence but also for the type of cadence present; that is, we have tree-class problems: no cadence, PAC, or rIAC (Bach) / HC (Haydn, Mozart). We compare two configurations: using all available features (as in the first experiment, feature set *all* in the table), or only feature sets 1 and 2, excluding the cadence-specific features (category 3 in Section 3.1; marked *general* in the table). Given this 3-class setting, we chose to report the macro averaged F1 score over all three classes.

Dataset	Features	F1 Note	F1 Beat
Bach Fugues	general	0.602	0.667
(PAC & rIAC)	all	0.653	0.702
Haydn String Quart.	general	0.542	0.610
(PAC & HC)	all	0.648	0.663
Mozart String Quart.	general	0.584	0.569
(PAC & HC)	all	0.588	0.606

Table 3. Three-class cadence classification with two different feature sets. Results were obtained by 5 fold cross validation (70% of pieces for training, 10% validation, 20% testing); no pre-training. Feature set *all* contains all features from Section 3.1; *general* excludes Category 3 cadence-specific engineered features.

(Macro averaging was chosen to counter the overwhelming effect of the majority class *no cadence*). The results (see Table 3) support the relevance of carefully devised cadence-related features à la [4]. However, also the general-purpose category 1 and 2 features alone support non-trivial cadence recognition and discrimination performance, which implies that the relational graph representation in combination with a convolutional approach manages to enrich highly local features with relevant non-local score context.

To investigate this latter aspect in more detail, we run a *third experiment*, to look at the effect of neighbor convolution depth on the obtainable classification score, again at three prediction granularity levels (note, onset, beat). Convolution depth refers to the number l of hidden layers of the encoder and the subsequent neighbor sampling up to l-hop neighbors. Our results (see Table 4) suggest that neighbor convolution clearly contributes to learning non-local features. Best results are achieved when using a convolution depth of 2. Increasing the receptive field beyond that level, we observed some instabilities emerging in the learning model, which could be attributed to the common vanishing gradient problem in deep GCNs [22].

Depth	F1 Note	F1 Onset	F1 Beat
None	0.833	0.671	0.667
1-hop	0.854	0.707	0.701
2-hop	**0.869**	**0.737**	**0.732**
3-hop	0.836	0.706	0.659

Table 4. Effect of neighbor convolution depth on PAC prediction in Bach fugues. The F1 Note/Onset/Beat scores presented are binary, i.e., for the PAC class. Depth refers to neighbor convolution depth. *None* means no graph convolution.

6.2 A Qualitative Look

Motivated by the fact that our model, while higher on recall, seems to be lower on precision than the model in [4], we take a closer look at some of the false positives in individual examples. Our findings suggest that many false positive predictions resemble cadences, in terms of tonal structure or implications, and could be considered and annotated as such, but lack some main components.

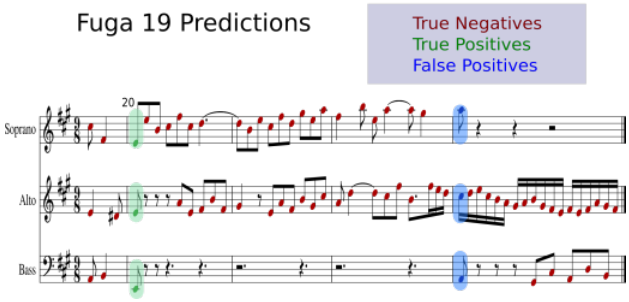


Figure 4. Predictions of Stochastic GraphSMOTE for fugue No.19, J.S.Bach, Well-tempered Clavier.

Figure 4 shows an example. The cadence prediction by our model on the downbeat of bar 23 is a false positive, according to the ground truth annotation. However, one could argue that the passage clearly has a cadence-like role, marking the end of the 2nd fugal episode and the return to the original tonality of A major [23].

Another example is the passage discussed in Fig.4 of [4], where a pattern occurs that has all the technical ingredients of a PAC, but was not annotated as such for (debatable) higher-level musicological considerations. Again, our model's PAC prediction there counts as a false positive.

As a final example, consider mm. 33-45 of Haydn's Op.54 No.1, 2nd mvt (Figure 3). We observe two false positive beat-wise predictions (8 if we count note-wise) in bars 39 and 44, respectively, following a true PAC on the beginning of bar 34. A harmonic analysis of these bars indicates a proper PAC preparation with text-book voice leading on the cadence arrival point in every occasion. These two false positive PACs form part of a modulating melodic and harmonic sequence; whether to classify them as cadences is a matter of higher-level musicological considerations.

We cite these few qualitative examples in an attempt to show that our prediction model can identify many more cadential patterns than the raw experimental figures suggest, but by design cannot consider high-level musical considerations such as, e.g., whether PAC-like patterns that occur in sequence should count as PACs or not.

7. CONCLUSION

We have presented a graph approach to effectively target the cadence detection task on symbolic classical scores. We demonstrated that our Graph Convolutional Network, Stochastic GraphSMOTE, can learn using only local note features, without the need for any musical assumptions about cadence anchor points. Furthermore, our network can produce fine-grained predictions at the level of individual notes.

Future work will address the performance of the model on different tasks, using the same graph representation. We hope to be able to show that this simple but general and natural representation of scores in terms of graphs can support a broad variety of symbolic music analysis and classification tasks.

8. ACKNOWLEDGEMENTS

This work is supported by the European Research Council (ERC) under the EU's Horizon 2020 research & innovation programme, grant agreement No. 101019375 ("Whither Music?"), and the Federal State of Upper Austria (LIT AI Lab). The authors would like to thank Dr. Hamid Eghbalzadeh for helpful discussions on Graph Neural Networks.

9. REFERENCES

[1] F. Korzeniowski, S. Oramas, and F. Gouyon, "Artist similarity with graph neural networks," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference*, 2021.

[2] C.-Z. A. Huang, C. Hawthorne, A. Roberts, M. Dinculescu, J. Wexler, L. Hong, and J. Howcroft, "The bach doodle: Approachable music composition with machine learning at scale," in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2019.

[3] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck, "Enabling factorized piano music modeling and generation with the maestro dataset," in *Proceedings of 7th International Conference on Learning Representations*, 2019.

[4] L. Bigo, L. Feisthauer, M. Giraud, and F. Levé, "Relevance of musical features for cadence detection," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018)*, 2018.

[5] T. Zhao, X. Zhang, and S. Wang, "Graphsmote: Imbalanced node classification on graphs with graph neural networks," in *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 2021.

[6] A. Popoff, M. Andreatta, and A. Ehresmann, "Relational poly-klumpenhouwer networks for transformational and voice-leading analysis," *Journal of Mathematics and Music*, vol. 12, no. 1, 2018.

[7] E. Karystinaios, C. Guichaoua, M. Andreatta, L. Bigo, and I. Bloch, "Music genre descriptor for classification based on tonnetz trajectories," in *Proceedings of Journées Informatiques Musicales*, 2021.

[8] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2016.

[9] D. Jeong, T. Kwon, Y. Kim, and J. Nam, "Graph neural network for music score data and modeling expressive piano performance," in *International Conference on Machine Learning*, 2019.

[10] D. Jeong, T. Kwon, Y. Kim, K. Lee, and J. Nam, "Virtuosonet: A hierarchical rnn-based system for modeling expressive piano performance." in *Proceedings of the 20th International Society of Music Information Retrieval Conference*, 2019.

[11] M. Giraud, R. Groult, E. Leguy, and F. Levé, "Computational fugue analysis," *Computer Music Journal*, vol. 39, no. 2, 2015.

[12] P. R. Illescas, D. Rizo, and J. M. I. Quereda, "Harmonic, melodic, and functional automatic analysis," in *Proceedings of the International Computer Music Conference*, 2007.

[13] D. R. Sears and G. Widmer, "Beneath (or beyond) the surface: Discovering voice-leading patterns with skip-grams," *Journal of Mathematics and Music*, vol. 15, no. 3, 2021.

[14] C. E. C. Chacón, P. Silvan, E. Karystinaios, F. Foscarin, M. Grachten, and G. Widmer, "Partitura: A python package for symbolic music processing," in *Proceedings of the Music Encoding Conference*, 2022.

[15] C. E. C. Chacón, "Computational modeling of expressive music performance with linear and non-linear basis function models," Ph.D. dissertation, Johannes Kepler University, Austria, 2018.

[16] M. Schuijer, *Analyzing atonal music: Pitch-class set theory and its contexts*. University Rochester Press, 2008.

[17] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," *arXiv preprint arXiv:2003.00982*, 2020.

[18] P. Allegraud, L. Bigo, L. Feisthauer, M. Giraud, R. Groult, E. Leguy, and F. Levé, "Learning sonata form structure on mozart's string quartets," *Transactions of the International Society for Music Information Retrieval (TISMIR)*, vol. 2, no. 1, 2019.

[19] D. R. Sears, M. T. Pearce, W. E. Caplin, and S. McAdams, "Simulating melodic and harmonic expectations for tonal cadences using probabilistic models," *Journal of New Music Research*, vol. 47, no. 1, pp. 29–52, 2018.

[20] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[22] G. Li, M. Muller, A. Thabet, and B. Ghanem, "Deepgcns: Can gcns go as deep as cnns?" in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019.

[23] "Bach: Prelude and fugue no.19 in a major, bwv 864 analysis," May 2018. [Online]. Available: https://tonic-chord.com/bach-prelude-and-fugue-no-19-in-a-major-bwv-864-analysis/

DOMAIN ADVERSARIAL TRAINING ON CONDITIONAL VARIATIONAL AUTO-ENCODER FOR CONTROLLABLE MUSIC GENERATION

Jingwei Zhao[2,4] **Gus Xia**[3,5] **Ye Wang**[1,2,4]
[1] School of Computing, NUS [2] Institute of Data Science, NUS [3] Music X Lab, NYU Shanghai
[4] Integrative Sciences and Engineering Programme, NUS Graduate School [5] MBZUAI
jzhao@u.nus.edu, gxia@nyu.edu, wangye@comp.nus.edu.sg

ABSTRACT

The variational auto-encoder has become a leading framework for symbolic music generation, and a popular research direction is to study how to effectively *control* the generation process. A straightforward way is to control a model using different conditions during inference. However, in music practice, conditions are usually sequential (rather than simple categorical labels), involving rich information that overlaps with the learned representation. Consequently, the decoder gets confused about whether to "listen to" the latent representation or the condition, and sometimes just ignores the condition. To solve this problem, we leverage *domain adversarial training* to *disentangle* the representation from condition cues for better control. Specifically, we propose a condition corruption objective that uses the representation to denoise a corrupted condition. Minimized by a discriminator and maximized by the VAE encoder, this objective adversarially induces a condition-invariant representation. In this paper, we focus on the task of melody harmonization [1] to illustrate our idea, while our methodology can be generalized to other controllable generative tasks. Demos and experiments show that our methodology facilitates not only condition-invariant representation learning but also higher-quality controllability compared to baselines.

1. INTRODUCTION

In deep music generation, improving *controllability* has been a major challenge that gains increasing research attention [1–6]. In practice, controllability is typically implemented under a conditional architecture, where the generation process is biased by external condition inputs. For example, EC^2-VAE [7] learns a representation z_x of 8-beat melody x while the underlying chords are given as condition c. The system is controllable if the generated melody can adapt to variable chords properly. For such representation-learning architectures, however, the decoder tends to find a shortcut from z_x to x without attending to c, leading to "condition collapse". The reason for this, as we argue, is that z_x is inevitably intertwined with condition c in the representation space, as c is often an innate property of x. In the case of EC^2-VAE, the condition of chords is very much implied by the melody.

To address this problem, the representation z_x must be disentangled from condition c. A popular way to achieve this goal is to use an adversarial objective that predicts c from z_x, as shown in Figure 1. On the one hand, this objective is optimized by a discriminator; on the other hand, the encoder is trained to "fool" the discriminator by detaching c-related cues out of z_x. In this way, the decoder cannot find a shortcut in z_x but is forced to seek c to reconstruct x. Such a technique stems from *domain adversarial training* (DAT) [8], where the "domain" is interpreted as "condition" that controls the generation.

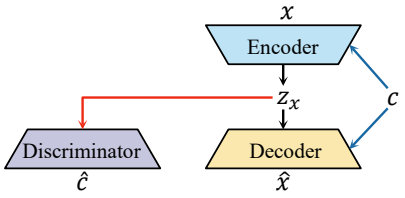


Figure 1: An illustration of domain adversarial training over a conditional generation architecture.

Apparently, DAT can be a powerful tool for controllable music generation. Previous studies [9, 10] have discussed simple scenarios where the condition is a global label (*e.g.*, note density). In music practice, however, local and sequential conditions [11] are more common. In such cases, c may not be fully implied by x, so the objective that simply predicts c from z_x does not necessarily hold.

In this paper, we focus on sequential conditions and develop a generalized form of DAT for controllable music generation. We illustrate our methodology with the task of *chord representation learning conditioned on melody*, where x stands for the chord progression, and c is the melody condition. In general, a chord progression can match many melodies, so we cannot directly predict c (melody) from z_x (chord) for the DAT objective. Instead, we leverage z_x to reconstruct c from a corrupted condition c^*. We rely on c^* to provide the melody context that cannot be hinted by chord x; on the other hand, the corrupted

[1] Demos and codes via https://zhaojw1998.github.io/DAT_CVAE.

© J. Zhao, G. Xia, and Y. Wang. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** J. Zhao, G. Xia, and Y. Wang, "Domain Adversarial Training on Conditional Variational Auto-Encoder for Controllable Music Generation", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.*, Bengaluru, India, 2022.

information reveals c's harmonic dependency on x, which we enforce the discriminator to learn. With proper corruption design, our DAT objective can be generalized to more scenarios with sequential conditions.

A well-trained model with good controllability can help us harmonize a new melody using the representation (style) of an existing chord progression. Experiments show that our model performs an excellent disentanglement of data representation from the condition, and the controllability outperforms the baselines. In summary, our contributions in this paper are as follows:

- **A general approach to controllability**: Based on a novel adversarial objective with condition corruption, we generalize domain adversarial training to music generation with sequential conditions;
- **A novel harmonization methodology**: We present a representation learning-based method for melody harmonization. Our current model harmonizes pop and folk melodies with the triad and seventh chords.

2. RELATED WORKS

2.1 Domain Adversarial Training

Domain adversarial training (DAT) is a representation learning approach initially proposed for domain adaptation tasks [12–14]. Through an adversarial process as described in Section 1, DAT enforces *domain invariance* to data representation so that it can be adapted to different domains flexibly. Such adaptability to new domains is analogous to controllability with new conditions. For generation tasks, DAT has been utilized to learn a condition-invariant data representation. Such invariance enforces the decoder to use condition information for reconstruction [15]. During inference, the decoder "listens to" new conditions as well and generates new data in a controllable way.

The first attempts that incorporate DAT with generation dealt with facial image generation conditioned on binary attributes (*e.g.*, male or female) [15, 16]. Such conditions cannot be explicitly supervised because we cannot find any pair of images that represents the same person both male and female. Fortunately, DAT enforces attribute invariance at encoding and learns attribute dependency at decoding, thus circumventing this problem. Recently, DAT has been extended to symbolic music generation conditioned on various attributes. Kawai *et al.* adopts DAT to a variational auto-encoder (VAE) for melody generation conditioned on statistical attributes (*e.g.*, note density) [9]. Later, Matsuoka *et al.* generalizes this methodology to generating polyphonic music with similar conditions [10].

For previous works, the conditions are particularly a global statistical label, which only represents a limited scenario of controllable generation. In our paper, we generalize the usage of DAT to sequential conditions. Conditioned on an 8-bar melody, we aim to learn a pitch-invariant representation of an 8-bar chord progression, which can later be adapted to varied melody conditions and to harmonize them. Our main novelty lies in a special design of the adversarial objective, which is to denoise corruption rather than make full prediction. This technique greatly helps us in dealing with the nuance of sequential conditions.

2.2 Controllable Music Generation

Controllable music generation takes various forms in terms of controlling technique and music representation [17]. For controlling technique, controllability can be achieved by sampling, interpolation, conditioning, and more ways [11]. For music representation, controls can be performed over statistical music properties (pitch variability, note density, etc.) [9, 10], compositional factors (chord progression, texture and rhythmic patterns, etc.) [7, 18–20], high-level semantics (emotion, cultural style, etc.) [21], and so on. With the development of representation learning, such properties can be abstracted and disentangled for flexible control.

In this paper, we are interested in chord representation learning conditioned on melodies, which falls into the category of controlling compositional factors via conditioning. Various conditional architectures, such as conditional VAE (C-VAE) [22], have been applied for similar purposes [7, 18–21]. However, as the condition is often easily implied by the representation, the decoder tends to skip the condition, and simply reconstruct the data for whatever conditions. To eradicate this problem, we introduce domain adversarial training and generalize it to sequential conditions (in our case, an 8-bar lead melody). Our model learns a pitch-invariant chord representation so that we can generate chord progressions harmoniously conditioned on varied melodies. Such control over compositional factors is common to broader music generation scenarios, and our methodology is generally applicable as well.

3. METHODOLOGY

In this section, we introduce our methodology with domain adversarial training on learning chord representation conditioned on the melody. An overview of our model is illustrated in Figure 2. We first describe our data representation and structure in Section 3.1. Then, we introduce our proposed model in Section 3.2. Finally, we elaborate on our novel design of condition corruption in Section 3.3.

3.1 Date Representation and Structure

3.1.1 Chord Representation

Our model generates an 8-bar chord progression conditioned on the melody. We quantize the chord progression at 1-beat unit and derive $T = 32$ timesteps. The maximum note count P for each chord is 4, which means we can flexibly represent any type of triad and seventh chords. Specifically, we treat chord progression as a piece of polyphony and follow [18] to represent it in both a surface structure (as model input) and a deep structure (for encoding).

The surface structure is a nested array of pitch attributes, denoted by $\{x_p^t | 1 \leq t \leq T, 1 \leq p \leq P\}$. Concretely, x_p^t is the p^{th} lowest pitch onset at time step t. We

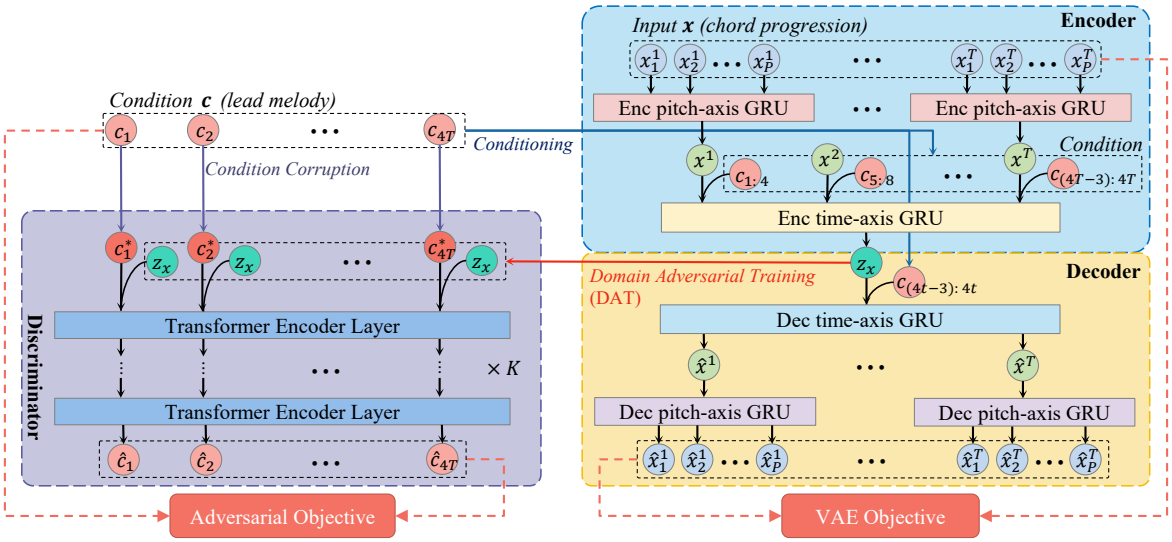


Figure 2: Chord representation learning with adversarial intervention for melody control.

represent x_p^t as a 13-D one-hot vector corresponding to 12 pitch classes plus a padding state. For most of our chord progression data, the offset of the last chord is precisely followed by the onset of the next one. Hence we do not explicitly consider the duration attributes.

For the deep structure, we build a syntax tree as in [18] to reveal the hierarchy from note via chord to chord progression. First, for $1 \leq t \leq T, 1 \leq p \leq P$, x_p^t itself constitutes the bottom layer of the tree. Then, for $1 \leq t \leq T$, we define x^t as the summary of $x_{1\leq p\leq P}^t$, which lies at the middle layer of the tree. Finally, we define z_x as the summary of $x^{1\leq t\leq T}$, which is the root of the tree. Such a deep structure is illustrated in Figure 3. Conceptually, while x^t is a compact representation of a single chord, z_x represents the complete chord progression.

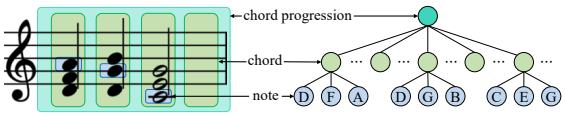


Figure 3: Tree-structure data representation of chord progression, reproduced from [18] with permission.

3.1.2 Melody Representation

Our model receives an 8-bar lead melody as the condition. we quantize the melody at $\frac{1}{4}$-beat unit and derive $4T = 128$ time steps. Following [7], we represent the melody as a sequence of note onsets plus a hold and a rest state. Each note onset consists of two one-hot vectors each representing 12 pitch classes and 10 octave ranges (registers). In our model, the melody pitch shares the same learnable embedding with the chord pitch.

3.2 Proposed Model

Our model applies a similar VAE architecture as PianoTree VAE [18], which learns representation for polyphonic music in a hierarchical manner. We use the surface structure of chord progression as the model input. The VAE architecture is built upon the deep tree-like structure.

We first illustrate the vanilla VAE design in the right half of Figure 2. Let x be the input chord progression and x_p^t be the p^{th} lowest pitch onset at time step t. The encoder first summarizes $x_{1\leq p\leq P}^t$ into an intermediate representation x^t (chord representation) for each time step t, and then encodes $x^{1\leq t\leq T}$ to the complete representation z_x. The decoder is basically a mirrored version of the encoder. The melody condition c, with its every four timesteps summed together, is concatenated to $x^{1\leq t\leq T}$ during encoding and to z_x during decoding. The loss function of our vanilla VAE architecture is:

$$\begin{aligned}\mathcal{L}(\theta_{\text{enc}}, \theta_{\text{dec}}) = &-\mathbb{E}_Q \left[\log P_{\theta_{\text{dec}}}\left(x \mid z_x, c\right)\right] \\ &+ \alpha \mathbb{KL}(Q_{\theta_{\text{enc}}}(z_x \mid x, c) \parallel \mathcal{N}(\mathbf{0}, \mathbf{1})),\end{aligned} \tag{1}$$

where $P_{\theta_{\text{dec}}}$ and $Q_{\theta_{\text{enc}}}$ refer to the VAE decoder and encoder. θ_{dec} and θ_{enc} are the learnable parameters. α is a balancing parameter for the regularization of KL loss [23].

Ideally, z_x should be a *relative* progression representation whose absolute pitch is controlled by melody c. However, as the input chord, x already has absolute pitch, this information is preserved in z_x as a redundant melody cue and confuses the decoder from attending to the condition.

To solve this problem, we assign a *discriminator* (left in Figure 2) to the VAE architecture. Instead of predicting c from z_x as conventional DAT objectives do, we bias the discriminator to denoise a *corrupted* melody condition. The corruption is done by transposing the melody to 12 keys with equal chance, which breaks the harmonic relation to the chord. In this way, we learn and extract the chord's dependency on its melody condition.

Formally, our discriminator leverages z_x to reconstruct melody condition c from a corrupted one c^*. Our DAT objective with condition corruption is trained in an adversarial manner. We optimize the discriminator by *minimizing* the reconstruction loss:

$$\mathcal{L}(\theta_{\text{dis}}) = -\mathbb{E}_Q \left[\log R_{\theta_{\text{dis}}}\left(c \mid z_x, c^*\right)\right], \tag{2}$$

where $R_{\theta_{\text{dis}}}$ is the discriminator with parameters θ_{dis}.

On the other hand, we optimize the VAE encoder by *maximizing* condition reconstruction error:

$$\begin{aligned}\mathcal{L}(\theta_{\text{enc}} \mid \theta_{\text{dis}}) = &-\mathbb{E}_Q \left[\log R_{\theta_{\text{dis}}} (\mathbf{1} - c \mid z_x, c^*)\right] \\ &+ \alpha \mathbb{KL}(Q_{\theta_{\text{enc}}}(z_x \mid x, c) \parallel \mathcal{N}(\mathbf{0}, \mathbf{1})),\end{aligned} \tag{3}$$

where $\mathbf{1} - c$ is a confusion criterion that encourages the encoder to "fool" the discriminator. $\mathcal{L}(\theta_i \mid \theta_j)$ means we optimize θ_i while fixing θ_j. The KL loss in Equation (3) and (1) ensures a consistent posterior regularization.

During domain adversarial training, Equation (2) and Equation (3) are iteratively optimized aside from the main VAE objective (1). In this way, the encoder is explicitly biased to disentangle z_x from c. The decoder learns to retrieve missing cues from c to reconstruct x, and thus guarantees controllability in the conditional architecture.

3.3 Condition Corruption

The main novelty of our architecture over previous applications of DAT [9,10,15] is that we incorporate a corrupted condition term to generalize this method to sequential conditions. The necessity of condition corruption is that, when c is not fully implied by x, the conventional DAT objective which predicts c from z_x no longer holds. In our case, x (chord) can be accompanied with various unique c (melodies), and a melody is largely independent of the chord in terms of sequential rhythmic patterns.

Condition corruption aims to reveal the dependency of c on x when a direct predictive inference from x to c cannot be established. The corrupted condition c^* serves as a *context* to fill in such prediction gap, and the *dependency* is highlighted when using z_x to denoise c^*. It may require field knowledge to design a proper corruption method for a specific scenario. Such corruption should keep the context part while blocking the dependency.

In our case, we corrupt the melody by transposing it to 12 keys with equal probability. The transposed melody c^* keeps the original rhythm and pitch curve shape while distorting the harmonic relation to the chord progression. Here the rhythm and the curve shape are the contexts, and the harmonic relation is the dependency. We compare our corruption method with a corruption-by-masking baseline in Section 4.6 to support the effectiveness of our design.

4. EXPERIMENTS

4.1 Dataset

We collect a total of 2K lead sheet pieces (melody with chord progression) for folk and pop songs from Nottingham [24] and POP909 [25] datasets. We only keep the pieces with $\frac{2}{4}$ and $\frac{4}{4}$ meters and slice them into 32-beat snippets at an 8-beat hop size, deriving a total of 35K samples. We quantize chords at 4^{th} note and melodies at 16^{th}. We randomly split the dataset (at song level) into training (95%) and validation (5%) sets. We further augment the training data by transposing each sample to all 12 keys.

4.2 Architecture Details

The VAE framework of our model is consistent with PianoTree VAE [18]. We implement the encoder with two bi-directional Gated Recurrent Unit (GRU) networks. The pitch-axis GRU and time-axis GRU each has a hidden dimension $d_{\text{p,enc}} = 256$ and $d_{\text{t,enc}} = 512$. The input embedding dimension d_{emb} and latent representation dimension d_{z} are both set to 128. The decoder mirrors the encoder with uni-directional GRUs, with hidden dimensions $d_{\text{t,dec}} = 1024$ and $d_{\text{p,dec}} = 512$. We set the KL balancing weight $\alpha = 0.1$ in Equation (1) and (3).

We implement the discriminator using BERT [26] with relative positional embedding [27–29], as our condition corruption is conceptually similar to language masking. For our model, we use 4 Transformer encoder layers with 4 heads [30] and 10% dropout [31]. The hidden dimensions of self-attention and feed-forward layers are $d_{\text{model}} = 256$ and $d_{\text{ff}} = 1024$. Our VAE and BERT discriminator each have 12.55M and 3.24M trainable parameters.

4.3 Training

Our model is trained using Adam optimizer [32], with a mini-batch of 256 samples and a learning rate from 1e-3 exponentially decayed to 1e-5. We use teacher forcing [33] for training the GRU-based decoder, with teacher forcing rate from 0.8 exponentially decayed to 0. We introduce domain adversarial training as an iterative process aside from the main VAE objective, as shown in Algorithm 1. We set $i = 10$, $j = 1$, $k = 5$, and $l = 5$. Our model is trained on a Geforce-2080Ti-12GB GPU. It takes 20 epochs (in around 15 hours) for our model to fully converge.

Algorithm 1: Domain Adversarial Training

```
while training do
    for i iterations do
        Optimize VAE with L(θ_enc, θ_dec),
    for j iterations do
        for k iteration do
            Optimize discriminator with L(θ_dis),
        for l iterations do
            Optimize encoder with L(θ_enc | θ_dis).
```

Figure 4 shows the trends of adversarial loss $\mathcal{L}(\theta_{\text{dis}})$ (in Equation (2)) and $\mathcal{L}(\theta_{\text{enc}} \mid \theta_{\text{dis}})$ (in Equation (3)). In the early stage, the discriminator learns to reconstruct c based on z_x, so the green curve decreases. However, as the adversarial procedure goes on, z_x is gradually disentangled from c-related cues. Consequently, the discriminator acquires less and less relevant information to reconstruct c well, and thus the green curve increases. The red curve exhibits an inverse trend, as it is supervised by $\mathbf{1} - c$. When each loss curve converges, we interpret it as an equilibrium that indicates a successful disentanglement of chord representation z_x from melody condition c.

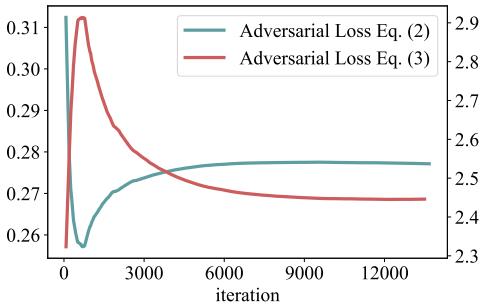
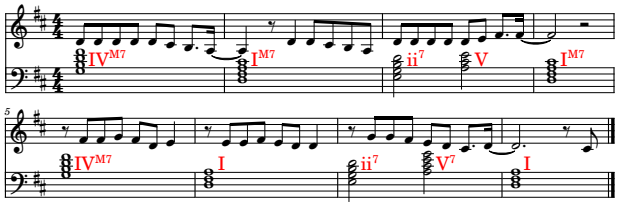


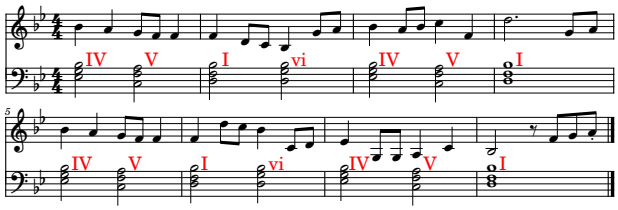
Figure 4: Adversarial loss curves with DAT. Such a trend is driven by the disentanglement of z_x from c.

4.4 Controllable Generation Results

Through domain adversarial training, our model gains reliable melody control over chord generation. Our model can harmonize a new melody using the representation of an existing chord progression. We hence develop a novel representation learning-based harmonization methodology. For example, Figure 5 presents two source lead sheets selected from our validation dataset. Both source samples are pop song phrases which share similar (but not exactly the same) chord progressions. However, the tonality and chromatic colours of these two pieces are quite different.



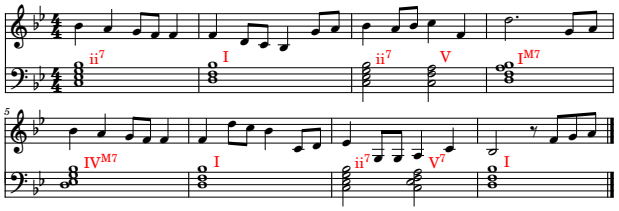
(a) Source A: a D major song accompanied by seventh chords.



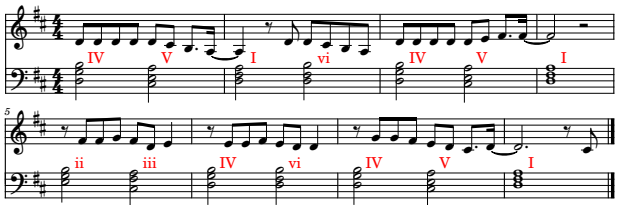
(b) Source B: a B♭ major song accompanied by triads.

Figure 5: Source lead sheets.

Figure 6a is the result where we reconstruct chord A conditioned on melody B, i.e., to harmonize melody B with the harmonic style in A. Here the "style" includes tensions with seventh chords and a typical cadence progression of ii-V-I. We see these features properly fitted to melody B in the correct tone. In other words, the generation of chord progression is controlled by the melody. Figure 6b is the result where we reconstruct chord B conditioned on melody A. For this case, the original seventh chords in A are replaced by triads with a IV-V-I cadence. These results suggest that our learned chord representation can well discern relative progression and chromatic colour, while our model is controllable in terms of tonality.



(a) Reconstruction of Chord A conditioned on Melody B.



(b) Reconstruction of Chord B conditioned on Melody A.

Figure 6: Chord generation conditioned on exchanged melody conditions. This process can also be viewed as melody harmonization using exchanged harmonic styles.

4.5 Subjective Evaluation

In this section, we evaluate our model's performance on the task of *harmonization*. We first derive the following three baseline models for an ablation study:

Non-DAT: Compared with our model, Non-DAT has the same VAE framework but does not have a discriminator. It does not explicitly try to disentangle z_x from c using domain adversarial training (DAT);

Mask-CR: Mask-CR has the same architecture as our model but uses a different condition corruption technique. Specifically, it applies *masking corruption* (as in [26]) rather than pitch transposition;

Non-CR: Compared with our model, Non-CR uses the conventional DAT objective *without condition corruption*. It predicts c directly from z_x with a GRU discriminator.

To compare our model with the baselines, we survey on rating the harmonization quality of all models. Our survey has 10 groups of harmonization results and each subject is required to listen to 4. In each group, the subjects first listen to an original lead sheet A and a single melody B. Both A and B are 8-bar long (16 seconds) and are randomly selected from different musical pieces from our validation set. As in Section 4.4, we harmonize melody B with the harmonic style of A using our model and the baseline models. Subjects are then required to evaluate each version of harmonization. The rating is based on a five-point scale from 1 (very poor) to 5 (very high) over three metrics: harmonicity, creativity, and musicality.

A total of 38 subjects with diverse music backgrounds participated in our survey and we obtain 142 effective ratings for each metric. As shown in Figure 7, the height of the bars represents the mean value of the ratings. The error bars represent the mean square errors (MSEs) computed by within-subject ANOVA [34]. We report a significantly better harmonization performance of our model than all three baselines in each metric (p-value $p < 0.05$). Specifically, we note that our model achieves such performance based

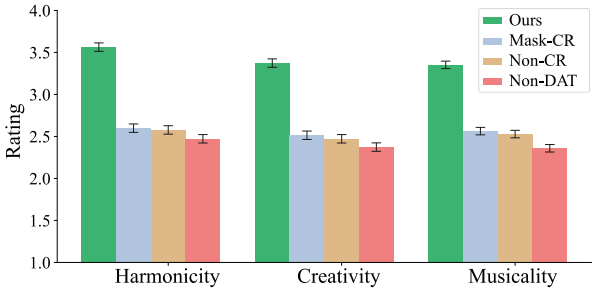


Figure 7: Subjective evaluation on the harmonization performance of our model and baseline models.

on a higher degree of representation disentanglement and controllability. We evaluate these methodological aspects with finer objective metrics in the following section.

4.6 Objective Evaluation

In this section, we objectively compare our model with the baselines in terms of *disentanglement* and *controllability*. The baseline models are as defined in Section 4.5.

4.6.1 Disentanglement

Our model disentangles chord representation z_x from melody condition c. In our case, the melody controls the absolute pitch of the chord progression. A satisfied disentanglement should derive a *pitch-invariant* representation. Following [7,35], we develop a similarity criterion to evaluate the performance on disentanglement.

Let $\mathrm{T}_i(\cdot)$ be a transposition operator with i semitones. We calculate cosine similarity $\cos(z_x, z_{\mathrm{T}_i(x)}), i = 1, 2, \cdots, 12$ for our model and for each baseline. In Figure 8, a higher similarity means representation z_x is less affected by the absolute pitch and thus is better disentangled. Our model outperforms all three baselines, including Mask-CR. This finding corroborates that a proper corruption strategy is crucial to applying domain adversarial training to concrete tasks. In our case, masking is not the best way to corrupt, as it is less aware of the harmonization context or dependency discussed in Section 3.3.

It is also worth noting that the similarity of z_x reflects human pitch perception. For each model, transposing a tritone ($\mathrm{T}_6(\cdot)$) derives the lowest similarity. Figure 8 shows that $z_{\mathrm{T}_6(x)}$ is literally orthogonal to z_x for Non-DAT and Non-CR. Interestingly, tritone is the most dissonant among all musical intervals in human perception. Such observation indicates that our model learns non-trivial music rules.

4.6.2 Controllability

A pitch-invariant representation helps us improve the model controllability by enforcing the decoder to rely on external conditions. In our case of harmonization, a good control generates harmonic chord progression conditioned on the lead melody. Aside from the subjective evaluation in Section 4.5, we introduce *harmony histogram* to objectively interpret the quality of control. Concretely, the harmony histogram is defined as the ratio of within-chord note positions on which the lead melody lies. For tonal music,

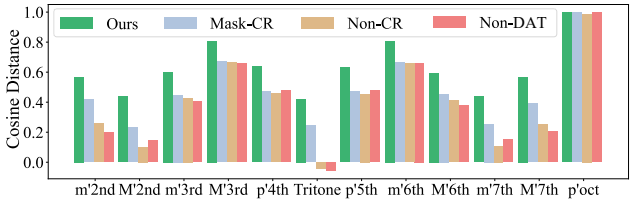


Figure 8: Object evaluation on representation similarity (invariance) against pitch transposition. A higher value denotes better disentanglement.

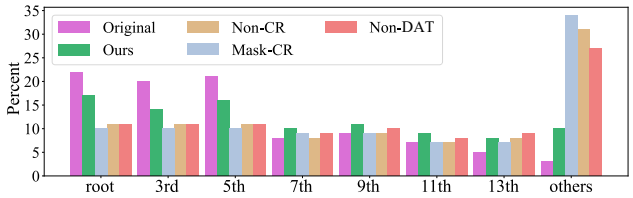


Figure 9: Objective evaluation on harmony histogram upon melody swapping. A higher ratio in root, 3rd, and 5th notes indicates a higher degree of controllability.

there should be more root, 3rd, and 5th notes appearing in the melody compared to 7th and higher, so that the music is considered harmonic.

In our experiment, we arrange our validation data into random pairs and reconstruct the chord progression with swapped melody conditions. We compare the harmony histogram of generated results from our model and all baselines. Additionally, we compute the histogram for the original (human-composed) data as ground truth. In Figure 9, we first observe that the histogram distribution has a larger portion in the root, 3rd, and 5th notes for the original data. For the baseline models, over 25% melody notes are beyond all chord notes and tensions (shown by "others" in Figure 9), which indicates excessive disharmony. Our proposed model, on the other hand, keeps a more consistent pattern with the ground truth.

5. CONCLUSION

In conclusion, we contribute a generalized form of domain adversarial training for controllable music generation, especially when complex sequential conditions are involved. The main novelty lies in the condition corruption objective, which contextualizes the exact dependency between representation z_x and condition c, and therefore assists disentanglement and control. Our method shows excellent performance in chord representation learning, where we learn a pitch-invariant representation conditioned on the melody and develop a novel harmonization strategy. Our improvement in disentanglement and controllability is elaborated with extensive subjective and objective evaluation. With the proposal of our methodology, we hope to bring a new perspective not only to music generation but also to more general scenarios of conditional representation learning.

6. REFERENCES

[1] A. Pati and A. Lerch, "Is disentanglement enough? on latent representations for controllable music generation," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 517–524.

[2] S. Dai, Z. Jin, C. Gomes, and R. B. Dannenberg, "Controllable deep melody generation via hierarchical music structure representation," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 143–150.

[3] K. Chen, C. Wang, T. Berg-Kirkpatrick, and S. Dubnov, "Music sketchnet: Controllable music generation via factorized representations of pitch and rhythm," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020, pp. 77–84.

[4] J. Jiang, G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa, "Transformer VAE: A hierarchical model for structure-aware and interpretable music representation learning," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2020, pp. 516–520.

[5] M. Xu, Z. Wang, and G. Xia, "Transferring piano performance control across environments," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2019, pp. 221–225.

[6] J. W. Kim, R. M. Bittner, A. Kumar, and J. P. Bello, "Neural music synthesis for flexible timbre control," in *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2019, pp. 176–180.

[7] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia, "Deep music analogy via latent representation disentanglement," in *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR*, 2019, pp. 596–603.

[8] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. S. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, pp. 59:1–59:35, 2016.

[9] L. Kawai, P. Esling, and T. Harada, "Attributes-aware deep music transformation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020, pp. 670–677.

[10] Y. Matsuoka and S. Sako, "Attribute-aware deep music transformation for polyphonic music," 2021, Late Breaking Demo in the 22nd International Society for Music Information Retrieval Conference, ISMIR. [Online]. Available: https://archives.ismir.net/ismir2021/latebreaking/000035.pdf

[11] J. Briot, G. Hadjeres, and F. Pachet, *Deep learning techniques for music generation*. Springer, 2020.

[12] G. Louppe, M. Kagan, and K. Cranmer, "Learning to pivot with adversarial networks," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, 2017, pp. 981–990.

[13] W. Wei, H. Zhu, E. Benetos, and Y. Wang, "A-CRNN: A domain adaptation model for sound event detection," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*. IEEE, 2020, pp. 276–280.

[14] F. J. Castellanos, A.-J. Gallego, and J. Calvo-Zaragoza, "Unsupervised domain adaptation for document analysis of music score images," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 81–87.

[15] G. Lample, N. Zeghidour, N. Usunier, A. Bordes, L. Denoyer, and M. Ranzato, "Fader networks: Manipulating images by sliding attributes," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, 2017, pp. 5967–5976.

[16] M. Li, W. Zuo, and D. Zhang, "Deep identity-aware transfer of facial attributes," *arXiv preprint arXiv:1610.05586*, 2016.

[17] Y. Zhang, "Representation learning for controllable music generation: A survey," 2020. [Online]. Available: https://doi.org/10.13140/RG.2.2.34458.11208

[18] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, G. Xia, and J. Zhao, "PIANOTREE VAE: structured representation learning for polyphonic music," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020, pp. 368–375.

[19] Z. Wang, D. Wang, Y. Zhang, and G. Xia, "Learning interpretable representation for controllable polyphonic music generation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020, pp. 662–669.

[20] Y. Chen, H. Lee, Y. Chen, and H. Wang, "Surprisenet: Melody harmonization conditioning on user-controlled surprise contours," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 105–112.

[21] Y. Zhang, Z. Wang, D. Wang, and G. Xia, "Butter: A representation learning framework for bi-directional music-sentence retrieval and generation," in *Proceedings of the 1st workshop on NLP for music and audio (NLP4MusA)*, 2020, pp. 54–58.

[22] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative

models," in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems*, 2015, pp. 3483–3491.

[23] I. Higgins, L. Matthey, A. Pal, C. P. Burgess, X. Glorot, M. M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *5th International Conference on Learning Representations, ICLR, Conference Track Proceedings*. OpenReview.net, 2017.

[24] E. Foxley, "Nottingham database," [EB/OL], https://ifdo.ca/~seymour/nottingham/nottingham.html Accessed May 25, 2021.

[25] Z. Wang*, K. Chen*, J. Jiang, Y. Zhang, M. Xu, S. Dai, and G. Xia, "POP909: A pop-song dataset for music arrangement generation," in *Proceedings of the 21th International Society for Music Information Retrieval Conference, ISMIR*, 2020, pp. 38–45.

[26] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, Volume 1*. Association for Computational Linguistics, 2019, pp. 4171–4186.

[27] P. Shaw, J. Uszkoreit, and A. Vaswani, "Self-attention with relative position representations," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, Volume 2*. Association for Computational Linguistics, 2018, pp. 464–468.

[28] C. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *7th International Conference on Learning Representations, ICLR*. OpenReview.net, 2019.

[29] Z. Wang and G. Xia, "Musebert: Pre-training music representation for music understanding and controllable generation," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 722–729.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, 2017, pp. 5998–6008.

[31] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR, Conference Track Proceedings*, 2015.

[33] N. B. Toomarian and J. Barhen, "Learning a trajectory using adjoint functions and teacher forcing," *Neural networks*, vol. 5, no. 3, pp. 473–484, 1992.

[34] H. Scheffe, *The analysis of variance*. John Wiley & Sons, 1999, vol. 72.

[35] S. Wei and G. Xia, "Learning long-term music representations via hierarchical contextual constraints," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference, ISMIR*, 2021, pp. 738–745.

MODELING PERCEPTUAL LOUDNESS OF PIANO TONE: THEORY AND APPLICATIONS

Yang Qu[1,2,*] **Yutian Qin**[1,3,*]
Lecheng Chao[1] **Hangkai Qian**[1] **Ziyu Wang**[1,4] **Gus Xia**[1,4]
[1] Music X Lab, NYU Shanghai [2] City University of Hong Kong [3] New York University
[4] Mohamed Bin Zayed University of Artificial Intelligence
yangqu7-c@my.cityu.edu.hk, {yq2120, lc4087, hq443, ziyu.wang, gxia}@nyu.edu

ABSTRACT

The relationship between perceptual loudness and physical attributes of sound is an important subject in both computer music and psychoacoustics. Early studies of "equal-loudness contour" can trace back to the 1920s and the measured loudness with respect to intensity and frequency has been revised many times since then. However, most studies merely focus on synthesized sound, and the induced theories on natural tones with complex timbre have rarely been justified. To this end, we investigate both theory and applications of natural-tone loudness perception in this paper via modeling piano tone. The theory part contains: 1) an accurate measurement of piano-tone equal-loudness contour of pitches, and 2) a machine-learning model capable of inferring loudness purely based on spectral features trained on human subject measurements. As for the application, we apply our theory to *piano control transfer*, in which we adjust the MIDI velocities on two different player pianos (in different acoustic environments) to achieve the same perceptual effect. Experiments show that both our theoretical loudness modeling and the corresponding performance control transfer algorithm significantly outperform their baselines. [1]

1. INTRODUCTION

Sound *intensity* and *loudness* are two relevant but very different terms. Intensity is the physical feature of sound derived from sound pressure. Loudness, however, is the perceptual measure dependent on our auditory systems, where other physical factors also contribute to human perception. For example, a 1 kHz tone is measured louder than a 100 Hz tone of equal intensity, but softer than an equal-intensity white noise. Previous psychoacoustic studies provide knowledge of loudness perception via human subject experiments and various computational models have been proposed to account for loudness estimation [1–6].

However, existing theories mainly focus on synthesized tones and thus fall short of explaining natural tone loudness such as the piano tone. The generation of piano tone involves a complicated physical process [7] and the sound contains rich timbral variations hard to be synthesized from both frequency and time domain perspectives [8, 9]. On the other hand, recently we see a growing number of music information retrieval tasks involving feature extraction of piano tone loudness, such as automatic music transcription [10–12] and performance rendering [13, 14]. In most of these studies, loudness is sometimes confused with intensity or even the MIDI velocity. This motivates us to investigate loudness perception specific to piano tone, beneficial for various downstream applications.

In this paper, we investigate both theory and applications on loudness perception specific to piano tone. The theory is studied in a hybrid method containing a *psychoacoustic* procedure and a *machine-learning* procedure. In the psychoacoustic procedure, we measure the "equal-loudness contour" on piano based on a human subject experiment similar to the pure-tone equal-loudness measurement [1]. Since piano tones cannot be directly controlled by frequency and intensity, we instead study pitch and velocity, two corresponding discrete performance controls. In our experiment, the controls are executed by (acoustic) player pianos for accuracy and reproducibility. We show that different pianos (in different acoustic environments) have distinctive equal-loudness contour patterns, and existing methods have limited power to explain the measured pattern.

In the machine-learning procedure, we extend the piano-tone equal-loudness contour to a computational model, capable of inferring loudness purely from spectral features. Traditionally, loudness models are based on mathematical approximation of our auditory systems [3–6]. We argue such an approach is laborious and even intractable for natural tones such as the piano tone to take into account all contributing facets of sound. Instead, we propose a machine-learning approach including a loudness model trained on the human subject measurement in the previous experiment. The model takes in the spectrogram and outputs the estimated loudness, calibrated to standard loudness unit *sone* in post-processing.

© Y. Qu, Y. Qin, L. Chao, H. Qian, Z. Wang, G. Xia. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** Y. Qu, Y. Qin, L. Chao, H. Qian, Z. Wang, G. Xia, "MODELING PERCEPTUAL LOUDNESS OF PIANO TONE: THEORY AND APPLICATIONS", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

*The two authors have equal contribution.

[1] Code and models can be accessed via https://github.com/yangqu2000/ModelingPerceptualLoudnessOfPianoTone

We show our theory of piano tone loudness provides a more reasonable explanation of piano sounds in downstream applications. Specifically, we apply our loudness model to *performance control transfer* [14], in which we adjust the MIDI velocities on two different player pianos (in different acoustic environments) to achieve the same perceptual effect. The original intensity-based loudness estimator is replaced with the proposed method, and experimental results show a significant improvement in terms of transfer quality.

2. RELATED WORK

The study of the equal-loudness contour (ELC) of pure tone is a serious subject in psychoacoustics, since it reveals fundamental facts of human loudness perception with respect to frequency spectrum and intensity. The experiment settings have been modified throughout the century, including the tuning of listening conditions [1, 15, 16], the improvement of experiment procedures [1,17,18], and the extension of measured frequency range [1,16,19,20]. Suzuki *et al.* [2] provides a detailed literature review of the existing experiments. Our piano-tone ELC experiment is modified from the original one, including careful adjustment to account for discrete frequency and intensity control.

The psychoacoustic listening tests of pure tone and many others [1, 16–18, 21, 22] provide fundamental understanding of our auditory systems. In return, the study of *loudness models* takes in the auditory system hypotheses and yields loudness estimation of the different types of tone, including [1, 16–18] for pure tone, [21] for complex tone, and [22, 23] for complex tone with amplitude modulations. So far, the state-of-the-art loudness model ISO 532-3 [24, 25] has the theoretical power to estimate the loudness of arbitrary waveforms. In our paper, we validate and compare with this model particularly on piano tone loudness estimation.

The majority of research on piano tone mainly focuses on the relationship between instrument control and the physical attributes of the generated sound, either statistically [26–28] or via physical modeling [7, 29, 30]. However, there is a lack of formal theory to cover the perception of piano tone. In this paper, we provide the first approach to study the relationship between instrument control and piano tone loudness in a psychoacoustic approach.

3. PIANO TONE EQUAL-LOUDNESS CONTOUR

Unlike pure tone which is determined by frequency and intensity, piano tone is largely dependent on the environment including the instrument itself that generates the sound (e.g., upright or grand piano) and the surrounding acoustic environment (e.g., concert hall or small room). Given a fixed environment, a piano tone can be controlled by four factors, namely *pitch*, *velocity*, *duration* and *pedal* [31]. Simple as they are, the four control parameters interact with the acoustic environment and produce a wide spectrum of piano timbre, resulting in an unexplored loudness perception pattern. In this paper, we study how perceptual loudness is affected by the first two factors, i.e., pitch and velocity, which can be roughly understood as the "frequency" and "intensity" of piano tone, respectively. We leave the others for future study.

The piano tones in our experiment are produced by player pianos, which can execute the control parameter in a more accurate manner than human pianists. Pitch is controlled by 88 MIDI pitches in $[21..108]$ and velocity is controlled by 128 velocity levels in $[0..127]$.

3.1 Method of Measurement

The goal of this experiment is to measure the *equal-loudness contours* (ELC) of piano tone on a specific piano and acoustic environment, that is, a sequence of piano keys under possibly different velocities that have equal perceptual loudness.

The experiment is modified from the original pure-tone ELC experiments [1]. Specifically, we first select a reference tone, denoted by $(p_{\text{ref}}, v_{\text{ref}})$, where p_{ref} is a reference pitch and v_{ref} is the velocity level that we are interested in. Then, we enumerate piano pitches and for each pitch p_{var}, we ask subjects to listen to $(p_{\text{var}}, v_{\text{var}}), v_{\text{var}} \in [0..127]$ and choose the louder note until they find a velocity v^*_{var} such that $(p_{\text{var}}, v^*_{\text{var}})$ and $(p_{\text{ref}}, v_{\text{ref}})$ have equal loudness.

In psychoacoustic terms, the reference tone $(p_{\text{ref}}, v_{\text{ref}})$ is called a *reference stimulus*, and given the pitch, the candidate tones under multiple velocities to compare (i.e., $(p_{\text{var}}, v_{\text{var}})$) are called *variable stimuli*. The solution $(p_{\text{var}}, v^*_{\text{var}})$ is called the *point of subjective equality* (PSE). The curve that connects the PSEs at multiple pitches is the equal-loudness contour at reference velocity v_0.

To adjust the velocity of reference stimuli and search for the PSE of each pitch, we adopt *randomized maximum likelihood sequential procedure* (RMLSP) [1], a common procedure used in pure-tone ELC experiments consisting of a series of test rounds. At each test round, the subject is asked to compare the loudness of a pair of piano tones, including the reference stimulus and a variable stimulus. Then we fit an online logistic regression model to predict the PSE according to the subject's response so far. The velocity of the variable stimulus in the next test round is randomly selected within a velocity range centered at this PSE velocity. The algorithm ensures the procedure will converge to the correct PSE.

3.2 Experiment Setting

In our experiment, we use A4 in the middle of the keyboard as the reference pitch (i.e., $p_{\text{ref}} = 69$). We measure equal loudness contours at four velocity levels: $v_{\text{ref}} \in \{32, 44, 60, 80\}$. The four velocities lie in the common range of velocity usage and are evenly distributed with respect to the statistical distribution [11].

We measure equal-loudness contours on 9 variable pitches: $p_{\text{var}} \in \{21, 33, 45, 57, 69, 81, 93, 105, 108\}$. The pitches are the A's in all octaves together with the highest note C8. The measurement on all 88 keys is ideal though not affordable, since RMLSP normally takes 20 minutes to find the PSE with respect to one single variable pitch.

ID	Environment		Number of subjects assigned to each velocity level				
	Instrument	Acoustic environment	$v_{\text{ref}} = 32$	$v_{\text{ref}} = 44$	$v_{\text{ref}} = 60$	$v_{\text{ref}} = 80$	Total
Env. I	Grand Disklavier	Anechoic chamber	6	6	6	5	23
Env. II	Upright Disklavier	Non-anechoic chamber	7	7	7	7	28

Table 1: Experiment settings and subjects assignment in two environments.

We address the loudness modeling on the other pitches in section 4.

We set the number of RMLSP test round to be 32. In the first two test rounds, the subject always listens to two piano tones of constant variable velocities $v_{\text{var}} = 90$ and $v_{\text{var}} = 30$, respectively. After that, the model yields the next variable velocity uniformly sampled from a $[-6..+6]$ interval centered at the current-step estimation of the PSE. The order of the piano tone pairs between the reference stimulus and the variable stimulus is random at each test round.

The experiment is conducted on two player pianos located in different acoustic environments (as shown in Table 1). Environment I contains a grand YAMAHA Disklavier piano in an anechoic chamber, and Environment II contains an upright YAMAHA Disklavier piano in a non-anechoic chamber. Subjects are seated half a meter in front of the pianos where the pianists usually sit and use a laptop to respond.

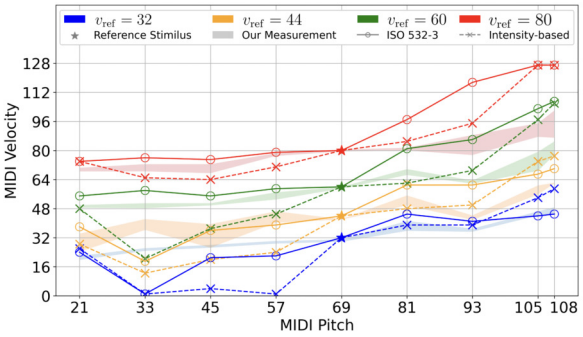
The subjects are 15 adults (18–35 years), 10 male and 5 female. All subjects have normal hearing sensitivity. 11 subjects participate in the experiment in Environment I and 9 subjects participate in the experiment in Environment II. Each subject is tested in a separate section.

The subject is tested at two or three of the four velocity levels randomly assigned (as shown in Table 1). For Environment I, 23 measurement results were collected. Each velocity level has a sample number of 6 except for velocity 80 with a sample number of 5. For Environment II, 28 measurement results were collected. Each velocity level has a sample number of 7.

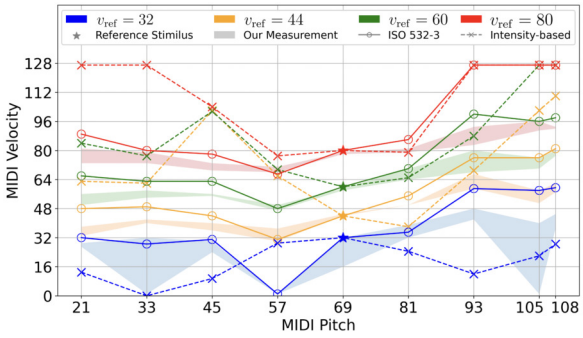
3.3 Result

Figure 1 shows our experiment results in Environment I and Environment II, respectively. Since MIDI velocity is an ordinal variable, it is improper to average the subjects' PSEs and show the mean equal-loudness curves. Instead, we present *equal-loudness ribbons*, where we replace the mean with the range of subjects' PSEs between the first and third quartiles among all data. Here, we use four different colors to indicate four reference velocities, and the diamond markers indicate the four reference stimuli on each graph. Note that in the lowest ($v_{\text{ref}} = 32$) ribbons, some of the variable pitches have PSEs equal to one, meaning that the variable pitches at the least audible velocity are still louder than or equal to the reference tone.

We see the patterns of equal-loudness ribbons are very different across the two environments. In Environment I, we see a gradual growth trend in all velocity levels, meaning higher pitches of the same velocity tend to be softer.



(a) Environment I.



(b) Environment II.

Figure 1: Measurement of equal-loudness ribbons of piano tone in two environments compared with ISO 532-3 [24,25] and intensity-based loudness computation [14,26].

In Environment II, the trend is less evident, and we see a valley at A3 (i.e., $p_{\text{var}} = 57$) at all reference velocity levels. Moreover, the range of equal-loudness ribbons varies, indicating the just noticeable difference in velocity change is uneven on different pianos, pitches and velocities.

3.4 Comparison with Existing Methods

We use the experiment result to validate two common loudness computation methods. The first method is ISO 532-3 by Moore *et al.* [24,25], the state-of-the-art loudness model based on the modeling of human auditory systems. The loudness value is computed as the maximum value of the predicted long-term loudness curve. The other method is used in [14,26] by naively computing the average intensity of the first 10 ms after the peak. For the two methods, we use the recordings of all the piano tones (discussed in section 4.3) and compute their corresponding loudness. The induced equal-loudness contours are shown in Figure 1 in solid lines with circles and dashed lines with crosses, respectively. Other popular methods such as applying A-

weighting is not considered since such methods are proved not generalizable to natural tone [32].

As shown in Figure 1(a) and Figure 1(b), in both environments, the results induced from ISO 532-3 are in general consistent with the equal-loudness measurement except for the pitch in the higher registers. The results induced from the intensity-based method fluctuate around the equal-loudness ribbons, failing to describe the perceptual effect.

4. LOUDNESS MODEL

In the previous section, we measure the equal-loudness contours (ELC) in two environments on 9 variable pitches and 4 reference velocities. In this section, we extend our findings to learn loudness models to account for loudness estimation of the remaining tones via a non-parametric approach as baseline (discussed in section 4.1) and a parametric approach (discussed in section 4.2). The parametric model is also capable of predicting loudness given the waveform of an arbitrary piano tone. We compare the prediction result with ISO 532-3 and the intensity-based method in section 4.3.

4.1 Non-parametric Method

We first propose a naive approach to predict the loudness of arbitrary piano tones in each environment via linear interpolation of the measured ELCs in section 3. First, we assign the loudness in *sone* to the tones of the reference pitch A4 under all the velocities using the calibration method discussed in section 4.2.3. Then, we linearly interpolate the ELCs measured on 9 variable pitches to all 88 pitches, and assign the same loudness level along each contour. Finally, for each pitch, we linearly interpolate between the ELCs. In section 5, we see such a simple method already yields satisfactory performance in downstream applications.

4.2 Parametric Method

Moreover, a parametric model is proposed to estimate the loudness using machine learning.

4.2.1 Model

We use $x = (p, v)$ to denote a piano tone, where p is the pitch and v is the velocity levels. We learn a parametric loudness model $\ell = f_\theta(x)$ to compute the loudness of a given piano tone x.

The loudness model is learned as a supervised classification problem (as shown in Figure 2). The input is a pair of piano tones (x_1, x_2), where $x_1 = (p_1, v_1)$ and $x_2 = (p_2, v_2)$. The target is the ground-truth label y, which is the indicator of whether x_1 is the louder one inferred from the ELCs. The difference between the estimated loudness ℓ_1 and ℓ_2 is used to predict the y. Specifically, the loss function is:

$$\mathcal{L}(\theta; x_1, x_2, y) = \text{BCE}(f_\theta(x_1) - f_\theta(x_2), y), \quad (1)$$

where $\text{BCE}(\cdot)$ is the binary cross entropy function.

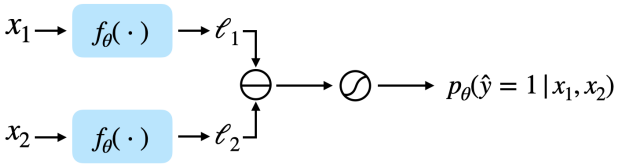


Figure 2: Illustration of the objective function.

The pairs (x_1, x_2) are sampled from all the tone pairs in each environment whose loudness comparisons are derivable based on the human subject results. Specifically, a pair must satisfy either of the two conditions:

(**C.1**) The two tones have the same pitch and thus can be compared based on velocity. The one with the larger velocity is assumed to be louder.

(**C.2**) The two tones have different variable pitches and the loudness comparison can be inferred from the equal-loudness ribbons in Figure 1.

4.2.2 Implementation

We define our loudness model $\ell = f_\theta(x)$ as the linear combination of the values on the mel-spectrogram of x (without an intercept term). The mel-spectrogram has 8 mel-frequency bins, a 2048 window size, and a 512 hop size under the sample rate of 22050 Hz. We select the 5 time frames (approx. 0.1s) right after the onset of the tone detected by [33]. In all, we have 80 input features of a pair of piano tones and the model can be optimized using logistic regression.

The small mel-frequency bin number in the current model can be understood as a constant convolution kernel to achieve better generalization on other non-variable pitches. In the future, the model can also be extended to neural networks given a larger amount of human subject results.

4.2.3 Calibration

The parametric method learns an unstandardized estimation of piano tone loudness based on classification, which needs to be further calibrated to the standard loudness unit in sone. We address the problem by matching the piano tones under reference pitch A4 to the corresponding equal-loudness pure tones and compute the corresponding pure tone loudness using ISO 532-3. The matching procedure follows from the *method of adjustment* [32].

4.3 Evaluation

4.3.1 Data Preparation

We record the piano tones of two Disklavier pianos in both environments for all pitches and velocities. The recorder is placed 0.5 meters in front of the piano (similar to the subjects' ear position) and is kept still during the recording procedure. For all piano tones, the MIDI duration is 0.3 seconds and each recording clip lasts 1.3 seconds.

To train the parametric model, we select all the piano tone pairs that satisfy the two conditions (defined in 4.2.1),

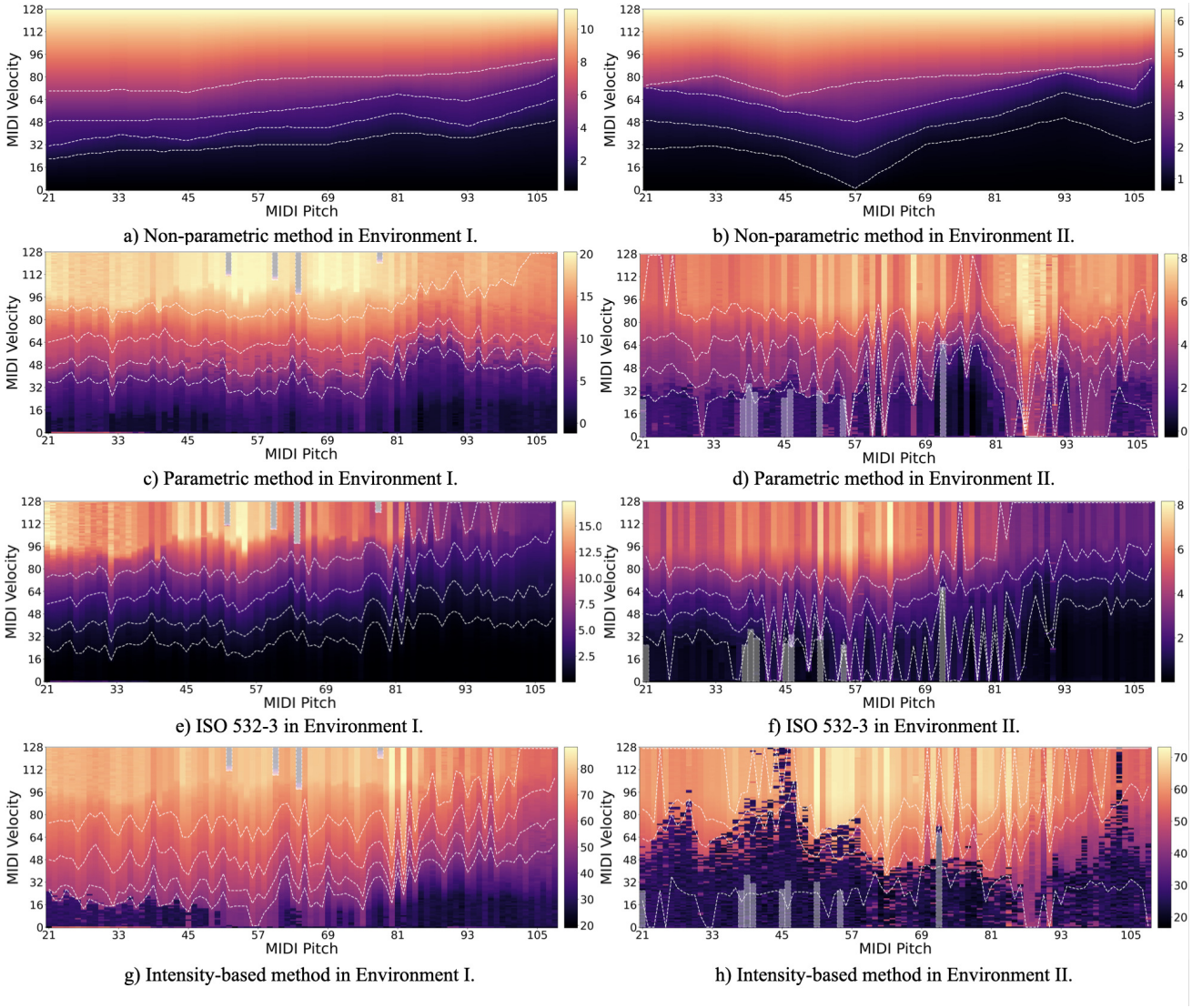


Figure 3: Visualization of calculated loudness value in Environment I and Environment II using our proposed loudness models, ISO 532-3 [24,25], and intensity-based method [14,26]. Points with a rectangular mask indicate mechanical failure of the player piano.

including 673,035 pairs satisfying the **C.1** and 427,633 pairs satisfying the **C.2** in Environment I, and 695,860 pairs satisfying the **C.1** and 422,386 pairs satisfying the **C.2** in Environment II. The dataset is randomly split into train set (80%) and test set (20%).

4.3.2 Results

Figure 3 shows the heatmaps of the predicted loudness of all piano tones in both environments by four methods: 1) our non-parametric method, 2) our parametric method, 3) ISO 532-3, and 4) intensity-based method. We show the ELCs derived from the heatmaps in dashed lines for better readability.

We see the non-parametric method presents a smooth interpolation of the measured ELCs. The result of the parametric method demonstrates a similar trend as ISO 532-3 except in the higher pitch range, which is more consistent with human subject measurement in Figure 1. Moreover, both parametric method and ISO 532-3 show more discontinuity than Figure 1, suggesting the MIDI velocity is not assigned in a uniform manner for different pitches. Finally, the result of intensity-based method are noisy and inconsistent with the human perception.

Table 2 shows the model accuracy on the test set of two conditions defined in 4.2.1 separately. The results demonstrate that our proposed method outperforms the two baselines in both environments, especially in the accuracy of **C.2**. We show the generalization ability in two ways. First, we apply the model trained in one environment to the other environment and achieves satisfactory accuracy (as indicated by the underlined numbers). Second, we combine the training data in both environments and train a hybrid model. The model outperforms the baselines and even achieves the best performance in environment II in both conditions. The evaluation result shows our proposed model capture certain features in the piano-tone mel-spectogram contributing to the perceptual loudness.

Methods	Acc. in Env. I		Acc. in Env. II	
	C.1	C.2	C.1	C.2
ISO 532-3	**0.9689**	0.9631	0.9037	0.9455
Intensity-based	0.9658	0.9290	0.8377	0.8555
PM - Env. I	**0.9689**	**0.9893**	0.8976	0.9614
PM - Env. II	0.9528	0.9607	0.9121	0.9793
Hybrid PM	0.9401	0.9785	**0.9370**	**0.9881**

Table 2: Evaluation of loudness comparison accuracy in Environment I and Environment II. The methods to compare are: 1) ISO 532-3: [24,25], 2) Intensity-based [14,26], 3) PM - Env. I: Parametric model trained on the data recorded in Environment I, 4) PM - Env. II: Parametric model trained on the data recorded in Environment II, and 5) Hybrid PM: Parametric model trained on the data recorded in both environments. Underlined data are tested by the parametric model training in the other environment.

5. PERFORMANCE CONTROL TRANSFER

In this section, we apply our theory of piano tone loudness to the downstream application of *performance control transfer*, in which we adjust the MIDI velocities on two different player pianos to achieve the same perceptual effect. An example scenario of the application is to reproduce a pianist's performance in the concert hall to one's living room.

5.1 Modification

Performance control transfer is originally proposed in [14], in which piano tone loudness is treated as a fundamental invariant property between the original performance and the transferred version. However, the study mistakenly uses the physical intensity as the loudness measure. Our approach replace the loudness estimator by the proposed loudness models (discussed in section 4) and keep the remaining algorithm the same. We also use ISO 532-3 as the loudness estimator as a baseline method.

5.2 Listening Test

We conduct a listening test to invite participants to evaluate the performance transferred by different algorithms, similar to the one conducted in [14]. Specifically, we prepare four pieces, including two monophonic pieces and two polyphonic pieces. The pieces are selected to cover classical and popular genres.

We invite two pianists to play the four pieces in Environment I and record both pianists' MIDI control and the performance audio. Then, we transfer the MIDI files recorded in Environment I to Environment II using different performance transfer methods, including 1) our non-parametric method, 2) our parametric method, 3) ISO 532-3, 4) intensity-based model (i.e., the original transfer method [14]), and 5) a raw transfer without any MIDI velocity editing. We play the transferred MIDI in Environment II and record them from the same microphone position discussed in section 4.3.

We invite people to subjectively rate the transfer quality through a double-blind online survey. During the survey, the subjects listen to four groups of samples. In each group, the original performance in Environment I is played, followed by the five transferred versions. Both the order of groups and the sample order within each group are randomized. After listening to each sample, the subjects rate them based on a 5-point scale from 1 (very low) to 5 (very high) according to three criteria: *faithfulness* (to the original performance), *naturalness* and *overall musicality*.

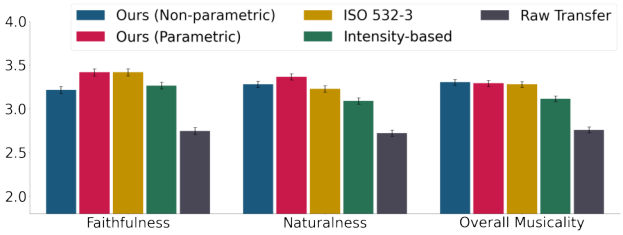


Figure 4: The subjective evaluation results of the five transfer methods.

5.3 Results

A total of 21 participants (14 male, 7 female) with different musical backgrounds have completed the survey. Figure 4 shows the result where the heights of the bars represent the means of the ratings and the error bars represent the confidence intervals computed via within-subject ANOVA [34].

The results show that all the loudness-based methods significantly outperforms the original intensity-based implementation and raw transfer (with p-value < 0.005). Moreover, our parametric method is marginally better than ISO 532-3 in terms of naturalness and overall musicality, and comparable with ISO 532-3 in faithfulness. Both our parametric and non-parametric methods demonstrate with only sparse human subject measurement data, we can achieve promising performance transfer results.

6. CONCLUSION

In this paper, we have contributed the first study to model the perceptual loudness of piano tones in a hybrid approach combining psychoacoustic experiments, and data-driven methods. Our theory include: 1) measurements of piano-tone equal-loudness contours on two player pianos, and 2) data-driven loudness models to estimate piano-tone loudness based on the measured contours. Experiments show our model provides more satisfactory estimation than existing loudness theories. Based on our findings, we validate the state-of-the-art loudness model ISO 532-3 on piano-specific tasks and argue that existing methods can be greatly improved if we have a more detailed human subject measurement and a deeper understanding of the timbral features of piano tone.

On the application side, we improve the existing performance control transfer method with the proposed loudness estimator. We believe the other downstream applications can also benefit from our study to propose a clearer problem definition and more accurate feature extraction.

7. REFERENCES

[1] H. Takeshima, Y. Suzuki, H. Fujii, M. Kumagai, K. Ashihara, T. Fujimori, and T. Sone, "Equal-loudness contours measured by the randomized maximum likelihood sequential procedure," *Acta Acustica united with Acustica*, vol. 87, no. 3, pp. 389–399, 2001.

[2] Y. Suzuki and H. Takeshima, "Equal-loudness-level contours for pure tones," *The Journal of the Acoustical Society of America*, vol. 116, no. 2, pp. 918–933, 2004. [Online]. Available: https://doi.org/10.1121/1.1763601

[3] B. C. Moore, B. R. Glasberg, and T. Baer, "A model for the prediction of thresholds, loudness, and partial loudness," *Journal of the Audio Engineering Society*, vol. 45, no. 4, pp. 224–240, 1997.

[4] E. Zwicker, G. Flottorp, and S. S. Stevens, "Critical band width in loudness summation," *The Journal of the Acoustical Society of America*, vol. 29, no. 5, pp. 548–557, 1957. [Online]. Available: https://doi.org/10.1121/1.1908963

[5] B. R. Glasberg and B. C. Moore, "A model of loudness applicable to time-varying sounds," *Journal of the Audio Engineering Society*, vol. 50, no. 5, pp. 331–342, 2002.

[6] H. Fletcher, "Loudness, pitch and the timbre of musical tones and their relation to the intensity, the frequency and the overtone structure," *The Journal of the Acoustical Society of America*, vol. 6, no. 2, pp. 59–69, 1934.

[7] D. E. Hall and A. Askenfelt, "Piano string excitation v: Spectra for real hammers and strings," *The Journal of the Acoustical Society of America*, vol. 83, no. 4, pp. 1627–1638, 1988.

[8] B. Bank, F. Avanzini, G. Borin, G. De Poli, F. Fontana, and D. Rocchesso, "Physically informed signal processing methods for piano sound synthesis: a research overview," *EURASIP Journal on Advances in Signal Processing*, vol. 2003, no. 10, pp. 1–12, 2003.

[9] B. Bank and L. Sujbert, "Generation of longitudinal vibrations in piano strings: From physics to sound synthesis," *The Journal of the Acoustical Society of America*, vol. 117, no. 4, pp. 2268–2278, 2005.

[10] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-resolution piano transcription with pedals by regressing onset and offset times," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3707–3717, 2021.

[11] D. Jeong, T. Kwon, and J. Nam, "A timbre-based approach to estimate key velocity from polyphonic piano recordings." in *ISMIR*, 2018, pp. 120–127.

[12] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: challenges and future directions," *Journal of Intelligent Information Systems*, vol. 41, no. 3, pp. 407–434, 2013.

[13] A. Maezawa, K. Yamamoto, and T. Fujishima, "Rendering music performance with interpretation variations using conditional variational RNN." in *ISMIR*, 2019, pp. 855–861.

[14] M. Xu, Z. Wang, and G. G. Xia, "Transferring piano performance control across environments," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 221–225.

[15] B. Kingsbury, "A direct comparison of the loudness of pure tones," *Physical Review*, vol. 29, no. 4, p. 588, 1927.

[16] H. Takeshima, Y. Suzuki, K. Ashihara, and T. Fujimori, "Equal-loudness contours between 1 khz and 12.5 khz for 60 and 80 phons," *Acoustical Science and Technology*, vol. 23, no. 2, pp. 106–109, 2002.

[17] K. Betke, "New measurements of equal-loudness level contours," in *Proc. Inter-noise*, vol. 89, 1989, pp. 793–796.

[18] J. Hall, "Maximum-likelihood sequential procedure for estimation of psychometric functions," *The Journal of the Acoustical Society of America*, vol. 44, no. 1, pp. 370–370, 1968.

[19] H. Møller and J. Andresen, "Loudness of pure tones at low and infrasonic frequencies," *Journal of Low Frequency Noise, Vibration and Active Control*, vol. 3, no. 2, pp. 78–87, 1984.

[20] H. Fasti, A. Jaroszewski, E. Schorer, and E. Zwicker, "Equal loudness contours between 100 and 1000 hz for 30, 50, and 70 phon," *Acta Acustica united with Acustica*, vol. 70, no. 3, pp. 197–201, 1990.

[21] L. L. Beranek, J. Marshall, A. Cudworth, and A. P. G. Peterson, "Calculation and measurement of the loudness of sounds," *The Journal of the Acoustical Society of America*, vol. 23, no. 3, pp. 261–269, 1951.

[22] B. Roß, C. Borgmann, R. Draganova, L. E. Roberts, and C. Pantev, "A high-precision magnetoencephalographic study of human auditory steady-state responses to amplitude-modulated tones," *The Journal of the Acoustical Society of America*, vol. 108, no. 2, pp. 679–691, 2000.

[23] S. Kuwada, R. Batra, and V. L. Maher, "Scalp potentials of normal and hearing-impaired subjects in response to sinusoidally amplitude-modulated tones," *Hearing research*, vol. 21, no. 2, pp. 179–192, 1986.

[24] "Acoustics — Methods for calculating loudness — Part 3: Moore-Glasberg-Schlittenlacher method," International Organization for Standardization, Geneva, CH, Standard, 2017.

[25] B. C. J. Moore and B. R. Glasberg, "Modeling binaural loudness," *The Journal of the Acoustical Society of America*, vol. 121, no. 3, pp. 1604–1612, 2007. [Online]. Available: https://doi.org/10.1121/1.2431331

[26] R. B. Dannenberg, "The interpretation of midi velocity," in *ICMC*, 2006.

[27] A. Adli, Z. Nakao, and Y. Nagata, "Calculating the expected sound intensity level of solo piano sound in midi file," vol. 2006, 2006, pp. 731–736.

[28] M. Keane, "Statistical analysis of classical piano recordings in midi format," in *Proceedings New Zealand Acoustical Society Conference*, 2004.

[29] B. Bank and J. Chabassier, "Model-based digital pianos: from physics to sound synthesis," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 103–114, 2018.

[30] S. Ewert and M. Müller, "Estimating note intensities in music recordings," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2011, pp. 385–388.

[31] O. Ortmann, *The Physical basis of piano touch and tone: An experimental investigation of the effect of the player's touch upon the tone of the piano*. K. Paul, Trench, Trubner & Company Limited, 1925.

[32] E. Zwicker and H. Fastl, *Psychoacoustics: Facts and models*. Springer Science & Business Media, 2013, vol. 22.

[33] B. McFee, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, D. Ellis, J. Mason, E. Battenberg, S. Seyfarth, R. Yamamoto, viktorandreevichmorozov, K. Choi, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, A. Weiss, D. Hereñú, F.-R. Stöter, P. Friesch, M. Vollrath, T. Kim, and Thassilo, "librosa/librosa: 0.9.1," Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6097378

[34] H. Scheffe, *The analysis of variance*. John Wiley & Sons, 1999, vol. 72.

ON THE IMPACT AND INTERPLAY OF INPUT REPRESENTATIONS AND NETWORK ARCHITECTURES FOR AUTOMATIC MUSIC TAGGING

Maximilian Damböck
Faculty of Informatics
TU Wien, Austria
maxi.damboeck@gmail.com

Richard Vogl
Faculty of Informatics
TU Wien, Austria
richard.vogl@tuwien.ac.at

Peter Knees
Faculty of Informatics, TU Wien, AT
School of Music, Georgia Tech, USA
peter.knees@tuwien.ac.at

ABSTRACT

Automatic music tagging systems have once more gained relevance over the last years, not least through their use in applications such as music recommender systems. State-of-the-art systems are based on a variant of convolutional neural networks (CNNs) and use some type of time-frequency audio representation as input, in a fitting combination, to predict semantic tags available through expert or crowd-based annotation. In this work we systematically compare five widely used audio input representations (STFT, CQT, Mel spectrograms, MFCCs, and raw audio waveform) using five established convolutional neural network architectures (musicnn, VGG-16, ResNet, a squeeze and excitation network (SeNet), as well as a newly proposed musicnn variant using dilated convolutions) for the task of music tag prediction. Performance of all factor combinations are measured on two distinct tagging datasets, namely MagnaTagATune and MTG Jamendo. A two-way ANOVA shows that both input representation and model architecture significantly impact the classification results. Despite differently sized input representations and practical impact on model training, we find that using STFT as input representations provides the best results overall. Furthermore, the proposed dilated convolutional architecture shows significant performance improvements for all input representations except raw waveform.

1. INTRODUCTION

Music tagging is a multi-label classification task to predict semantic high-level tags like "rock", "piano" or "fast" for music pieces. As for many music information retrieval (MIR) tasks, deep learning has become the state-of-the-art for music tagging [1]. Specifically, CNNs have provided the best results for this task, using either types of audio spectrograms or raw waveform as network input [2, 3].

The selection of audio input representations for neural networks is often decided heuristically or by falling back to standard spectrograms and usually not evaluated [4]. However, the choice of input representation can have a significant impact on performance, as shown in other tasks like natural language processing (NLP), justifying also an evaluation of different representations [5].

While existing work compares different aspects of neural network architectures and input representations (see section 2), an evaluation of their interplay is missing. In this work, as our first contribution, we systematically compare five different architectures and five different input representations in a full-factorial design. The baseline architectures are: *(i)* VGG-16 [6], *(ii)* ResNet [7], *(iii)* SENet [8], *(iv)* musicnn [9], and *(v)* a musicnn architecture with dilated convolutional layers. Five different input representations are tested on each of those architectures: *(i)* raw waveform, *(ii)* short time Fourier transformation (STFT), *(iii)* Mel spectrogram, *(iv)* Mel frequency cepstral coefficients (MFCCs), and *(v)* constant-Q transformation (CQT) [2, 3, 10]. All experiments are repeated on two different datasets and a statistical analysis is used to test if differences of results are significant.

Through this extensive evaluation, we can state that our second contribution is the introduction of dilated convolutional layers in a music tagging network architecture (musicnn). This observed improvement can be related to the fact that the dilated layers allow for a large receptive field when only using a limited amount of layers, cf. [10].

Next, in section 2, we review related work, followed by an outline of the five input representations, five neural network architectures, and datasets used in section 3. Section 4 describes the experimental setup, training, and evaluation strategy and presents the statistical evaluation and significance analysis of the results. Finally, conclusions on the results and findings are drawn in section 5.

2. RELATED WORK

In [2], the authors evaluate state-of-the-art CNN models for music auto-tagging either by using the audio waveform or Mel spectrogram as input representations. MagnaTagATune (MTAT), Million Song dataset (MSD), and MTG-Jamendo (MTGJ) serve as datasets for training and evaluation. Among others, the study considers the following model architectures: A fully convolutional network, *musicnn* (cf. section 3.2.4), a Sample-Level CNN to process raw waveforms using squeeze- and excitation blocks (cf. section 3.2.3), a convolutional recurrent neural network, as well as a self-attention based approach using an adaptation of the transformer architecture. Furthermore, the work fo-

© M. Damböck, R. Vogl, and P. Knees. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** M. Damböck, R. Vogl, and P. Knees, "On the Impact and Interplay of Input Representations and Network Architectures for Automatic Music Tagging", in *Proc. of the 23rd Int. Society for Music Information Retrieval Conf.,* Bengaluru, India, 2022.

cuses on effects of data augmentation like pitch shifting or time stretching to improve generalization abilities. Note that while we follow a similar strategy wrt. evaluated architectures in our work, we focus on the impact of different input representations on these different architectures and evaluate their interplay.

More recently, transformer networks have been applied for music tagging [11,12]. While these models improve the state-of-the-art performance, they are significantly more complex than plain convolutional models. Therefore, we consider them not a good match for the evaluation in this work. Other deep learning techniques like transfer-learning, data augmentation, and model aggregation which have been used in music tagging [13], are mainly model agnostic and can be applied on top of any input representation and model combination.

In [14], the authors tested the effect of training data size for music tagging. They trained models with dataset sizes ranging from 100k to 1.2 million training instances using the private 1.2M-Songs dataset. Two CNN architectures are used, one for raw waveforms and a second one for Mel spectrograms inputs. The Mel spectrogram-based model outperformed the raw waveform-based model in both datasets on all tested configurations, which encouraged the conclusion that domain knowledge can be beneficial for designing model architectures.

Using dilated convolutions has shown promising results in image segmentation- and classification over the last few years [15–17]. A similar trend can be observed for the audio domain [18–21]. Dilation can be used to increase the receptive field using only a few layers, which can be beneficial for tasks focusing on global properties of the input.

In [22], the authors use dilated convolutions for environmental sound classification. This task (similarly to music tagging) requires an architecture with a large receptive field. Previously this was achieved by using very deep CNNs. The evaluation shows performance improvements of up to 10% on different datasets compared to existing deep CNNs while reducing model size.

In [4], the authors compare the effects of several audio preprocessing methods for music auto-tagging when using a CNN. For this, the same neural network was trained on the MSD using either STFT or Mel spectrogram as input representations. The evaluation shows no significant differences in performance for the two input representations. However, using log-compressed magnitude spectrograms showed a significant performance increase for all tested configurations.

A more comprehensive comparison of spectrogram representations for audio was conducted in [10] in the context of environmental sound classification. Different configurations for STFT, Mel spectrogram, CQT, continuous wavelet transform, and MFCCs were evaluated in the experiments, using different CNN configurations for training. The evaluation shows that in nearly all configurations, shallower CNN models outperformed deeper ones. The authors suspect that overfitting of the deeper model is the cause of this. The results also show that Mel spectrograms performed consistently well, while STFT and CQT did not, while MFCCs performed worst.

The effects of the reduction of frequency and time resolution of Mel spectrograms for CNN architectures in the context of music auto-tagging were investigated in [23]. Two different state-of-the-art CNN architectures served as a reference for the comparison of Mel spectrograms with different frequency and time resolutions. The results show that while reducing the frequency bands from 128 to 48, the performance loss was negligible.

3. METHOD

This section covers the description of the input representations as well as the neural network architectures used in the evaluation. The experiments are implemented in Python, using librosa [1] for audio processing and TensorFlow as deep learning framework. The source code is available online, refer to the source code for all implementational details [2], there are also additional figures on the accompanying website [3].

3.1 Input Representations

In the following section, the used input representations are discussed and parameter settings are provided. In this work, standard parameter settings established in the literature are used. Figure 1 visualizes the 2D input representations using a 10s audio snippet.

3.1.1 Raw Waveform

In the context of this work, the audio material is resampled to 16kHz mono. This is used consistently also for the source of all spectrograms. As waveform input representation, 16kHz mono 32bit float pulse-code modulation (PCM) data is used.

3.1.2 STFT

A STFT is calculated by repeatedly applying a discrete Fourier transformation (DFT) to small frames of the audio signal. As a baseline spectrogram, in this work, the librosa STFT with default parameters is used: frame size of 2 048 at 16kHz, a hop length of 512, and a Hann window function.

3.1.3 Mel Spectrogram

A usual modification to plain spectrograms is to use psychoacoustically-motivated frequency scales. A common choice is the Mel scale, which has a logarithmic characteristic. In this work, 96-frequency-bin Mel spectrograms are used. A minimum frequency of 32Hz and 12 bins per octave are used. These or similar values are consistently used throughout the literature [23].

3.1.4 MFCC

MFCCs are a compact representation reducing the frequency dimension of the Mel spectrogram using the discrete cosine transform (DCT). MFCCs have been successfully applied in speech recognition and to some extend

[1] https://librosa.org
[2] https://gitlab.com/MaxDamb/dl-autotagging
[3] https://tinyurl.com/ismir22-317

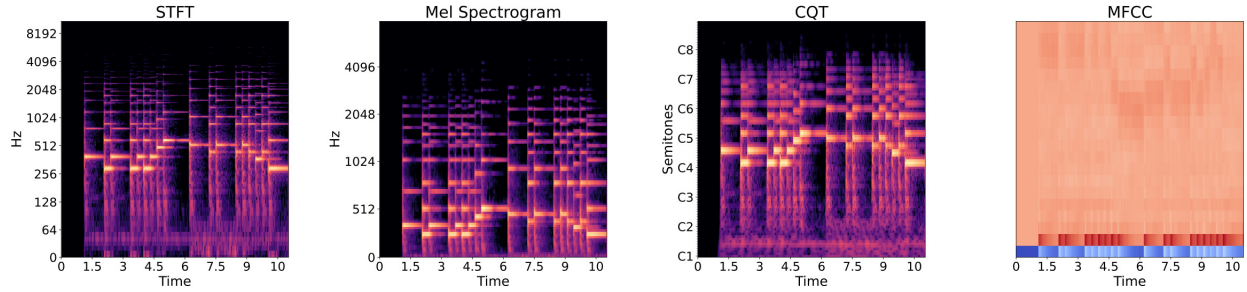


Figure 1. Different 2D input representations, from left to right: STFT, Mel spectrogram, CQT, and MFCC

also in MIR. The resulting coefficients describe the overall shape of the spectrogram columns [24]. Often the first 20 coefficients are used, which is also the default value in librosa, and therefore used in this work.

3.1.5 Constant-Q Transformation

Similar to the DFT, the CQT can be used to extract a time-frequency representation. The CQT results in a log-frequency scale and needs no further frequency transformation to match human audio perception. It is calculated similar to the STFT but enforces a constant ratio of the center frequency to resolution (constant-q) [25]. All parameters are chosen equal to comparable parameters of the Mel spectrogram: hop size 512, minimum frequency 32Hz, 12 bins per octave, and 96 frequency bins,

3.2 Network Architectures

In this section the five CNN architectures used for the experiments are discussed. Whenever kernel sizes are discussed, e.g. 3x3 filter kernels, these refer to two-dimensional inputs and imply 3(x1) filters for one-dimensional inputs. For more implementational details please refer to the original works, or the source code of the implementations.

3.2.1 VGG-16

The VGG-16 architecture introduced in [6] is used as a CNN baseline in this work. It consists of five convolutional blocks, each with 2-3 convolutional layers with 3x3 kernels and max-pooling with 2x2 kernels, followed by fully connected (FC) output layers. For each block the number of channels increases: 64, 128, 256, 512, and 512. For the FC layers (4 096 neurons each) the flattened output of the last convolutional block is used as input. Finally 50 sigmoid neurons form the output layer responsible for predicting the tags [6]. ReLU is used as an activation function for all hidden layers, and dropout- and batch-normalization layers are added after each block for regularization.

3.2.2 ResNet

Generally speaking, deeper networks have more processing power, but at the same time are harder to train because of the vanishing gradient problem. Residual networks implement so-called skip connections which are a solution to this problem and allow to train deeper networks [7]. A skip connection is a shortcut that adds the input to the output of a convolutional block or layer.

In this work, the ResNet-101 architecture (101 convolutional layers) [7] is used and reimplemented in TensorFlow. The basic building block is a bottleneck which consists of three convolutional layers (1x1, 3x3, 1x1) and a skip connection from the input to the output of the block. The 1x1 layers reduce the input dimensions by a quarter for the 3x3-layer and increase it again afterwards, which is done to reduces the memory footprint of the network. After the first convolutional layer (7x7) to downsize the input, the architecture consists of 3 blocks with 64 channels, 4 with 128, 23 with 256, and 3 blocks with 512 channels, followed by the sigmoid output layer.

3.2.3 Squeeze and Excitation Network

This architecture is based on the sample-level CNN architecture proposed in [26]. It uses one-dimensional raw audio-data as input. A basic block consists of a convolutional layer (3x3), followed by batch-normalization and max-pooling (2x2 for 2D and 3 for 1D). After that, the squeeze and excitation (SE) part is applied. The squeeze operation consists of a global average pooling to compress each channel. The excitation operation consists of two fully-connected layers (ReLU, sigmoid). Finally, the output of the basic convolutional part gets scaled with the output of the SE part by channel-wise multiplication. The squeeze and excitation network (SENet) consists of nine of these blocks, where the outputs of the last three blocks are max-pooled globally and concatenated to serve as input for two FC output layers, which use sigmoid outputs [26].

To make this architecture work for 2D input representations the frequency dimension is reduced gradually using 2D filters throughout the network.

3.2.4 Musicnn

Musicnn is a CNN designed specifically for music tagging and has been re-implemented and slightly adapted for this work. The original architecture uses Mel spectrograms with a length of three seconds and 96 Mel-bins as input [9]. The architecture is divided into three parts: *frontend*, *midend*, and *backend*. The *frontend* uses one convolutional layer with different kernel sizes motivated by music-domain knowledge. In the original work, 7x38, 7x67, 128x1, 64x1 and 32x1 filters are used. These filters are scaled relatively to the size of the frequency dimen-

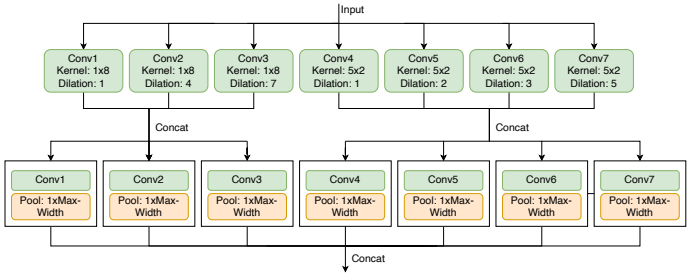


Figure 2. Dilated CNN frontend architecture for 2D-input

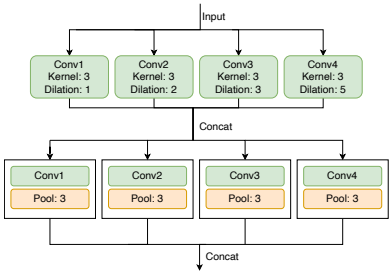


Figure 3. Dilated CNN block for frontend for 1D-input

sion for the other input representations used in this work. For the one-dimensional input, the frontend architecture from [14] is used, consisting of seven 3x1 convolutions, combined with max-pooling layers. The *midend* consists of three convolutional layers (7xN kernels) supposed to extract higher-level representations from the low-level features of the frontend. In the original work, midend and frontend layers are connected by summation of each input with the previous outputs (quasi-residual connections, inspired by dense connections [27]). The output layer concatenates all previous layers of the midend. The *backend* uses temporal pooling for the final output. It combines average- and max-pooling over the time dimension of the output of the midend, concatenates their outputs and then uses two dense-layers with sigmoid outputs for the final prediction [14].

3.2.5 Dilated CNN

This section presents an extension of the musicnn using dilated convolutions in the frontend. Dilated convolutions spread out the filters over a wider area, thus achieving a larger receptive field without increasing resource consumption [21, 28, 29]. As discussed in the introduction, a large receptive field is desirable since music tags depend on global properties of the audio signal. Figure 2 displays the architectural layout of the frontend for the dilated CNN. Three convolutional layers are used to extract temporal features, while four layers span the frequency axis. Then, all outputs are concatenated and serve as input for the midend. Convolutions in the first block have 26 channels and 51 in the second one. As described for the stacked dilated convolution block in [29], the outputs are concatenated after the first block of convolutions and are used as input for the second block.

For the one-dimensional input, the musicnn frontend was adapted differently. It consists of seven sequential stacked dilated convolutions to reduce the input size sufficiently. Each of those blocks consists of four parallel convolutions stacked in two blocks, as displayed in Figure 3.

3.3 Datasets

The two music tagging datasets used in this work are: MagnaTagATune ("MTAT") [30] and MTG-Jamendo ("MTGJ") [31]. Both these datasets contain tags belonging to the three categories "genre", "instrument" and "mood" identified relevant for the tag grouping in this work [32, 33].

3.3.1 MagnaTagATune

The dataset was created using the game-with-a-purpose called TagATune [30, 34]. MTAT consists of 5 223 songs, segmented into 25 863 song-snippets with a length of 29 seconds. Each snippet comes with associated tags from a set of 188 tags. Some of the tags in the dataset are synonyms for each other, therefore similar tags such as "choir" and "choral" or "horn" and "horns" are merged. Since some tags only yield an insufficient number of training examples, only the top 50 tags by occurrence are used, which is a common practice [26, 35, 36]. This results in 17 genre-, 22 instrument-, 9 mood- and two uncategorized tags. To be consistent on the two datasets, a 60:20:20 train/validation/test data split is used. When generating the splits, snippets of the same song are always put in the same split to counteract overfitting.

3.3.2 MTG-Jamendo

MTGJ consists of royalty-free music, and contains 55 709 full-length songs from 3 565 different artists. Every song is at least 30s in length and 224s on average, resulting in 3 777 hours of audio. In order to be consistent with the MTAT, only the first 29s of each audio segment are used, which reduces training- and test-data size. In total, there are 195 different tag annotations for each song, each belonging to one of the three categories "genre", "instrument", or "mood". The set of tags has already been cleaned up and does not contain tags with the same meaning [31]. The distribution of the number of examples per tag is comparable to MTAT. Again, only the top 50 most frequent tags are used for the experiments, which is consistent with the literature [31]. There are 31 genre-, 14 instrument- and 5 mood-tags. For train-, validation, and test-splits, the first split definition from the dataset repository is used [31]. The split ratios are approximately 60:20:20. The split definition ensures that tracks of the same artists are only present in one subset and that tags, tracks, and artists are equally distributed.

4. EVALUATION

4.1 Experiment Setup

For each combination of network architecture, input representation, and dataset, a model is trained and evaluated. For MTAT seven runs with random initialization, and for MTGJ five runs are performed and results are averaged over the runs. As evaluation criteria ROC-AUC was used

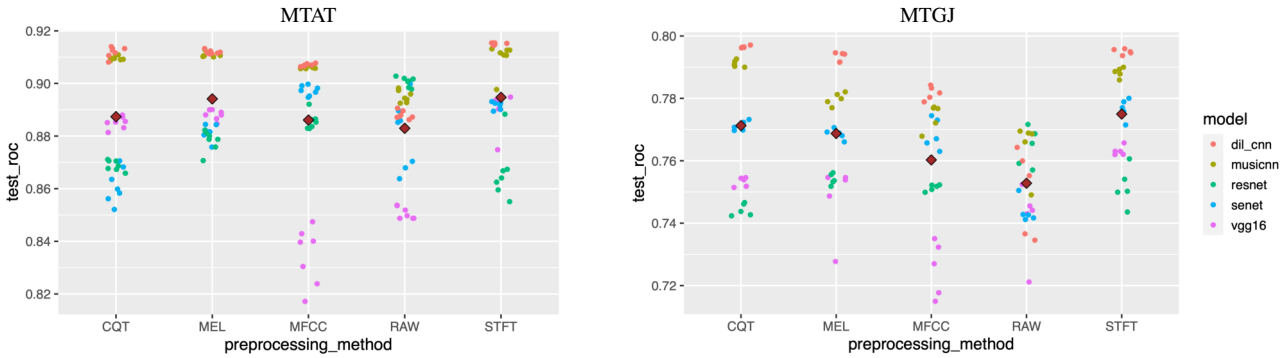


Figure 4. ROC-AUC scores for input representations and architectures on MTAT and MTGJ.

[35]. For training, early stopping was applied: training is stopped as soon as no further improvement on the validation split is achieved. As optimizer, ADAM in combination with a simple learning-rate schedule was used. Because of the smaller size of MTAT, additional refinement using stochastic gradient descent (SGD) with a reduced learning rate (1/5) was performed [37].

4.2 Results and Statistical Analysis

Figure 4 displays the ROC-AUC scores for the seven runs on MTAT (left) and the five runs on MTGJ (right) of input and architecture combinations. Different colors represent architectures, additionally, the average performance per input representation is shown in brown. The results for MTAT are on-par with state-of-the art results: dil_cnn+STFT: 0.915 v.s. 0.913 in [2]. For MTGJ, results are slightly below state-of-the-art (0.793 v.s. 0.832 in [2]), however, for compatibility reasons with MTAT in this work only 29s of each track are used v.s. full tracks in [2].

A two-way ANOVA is used to check if network architecture and/or input representation have a significant influence on the performance. Assumptions for ANOVA (independence and homoscedasticity of observations, normal distribution of residuals) are ensured: Since each run-configuration uses a different architecture and input representation combination, independence of observations is given. Homoscedasticity (equality of variances) of values is checked using the Levene test which indicates that this assumption is not met for MTAT. Therefore, a data transformation is applied to the result values on MTAT, which was not necessary for MTGJ. Residuals are found to be approximately normally distributed, which satisfies the last assumption [38].

Table 1 displays the results for the two-way ANOVA analysis. The p-values for the input representation and the architectures individually as well as combined indicate a strong influence on the performance. Due to the data transformation, the sum of squares (SS) and the mean squares (MS) for MTAT are not interpretable.

Tukey-HSD tests are used to check for significant differences of mean values between individual input representations as well as network architectures on both datasets. Table 2 shows the results with and without the data transformation to see absolute difference between groups while

MTAT	DF	SS (E+10)	MS (E+10)	F	P
input	4	2.83	70.8	57.64	<1E-5
arch.	4	31.6	7.89	642.13	<1E-5
input&arch.	16	15.1	94.5	76.86	<1E-5
residual	150	1.84	123		
MTGJ	DF	SS (E-5)	MS (E-5)	F	P
input	4	800	200	63.61	<1E-5
arch.	4	2600	650	205.51	<1E-5
input&arch.	16	1010	60	20.04	<1E-5
Residual	100	320	3.16		

Table 1. Two-way ANOVA results on MTAT and MTGJ. SS is sum of square, MS is mean square, DF is degrees of freedom, F is f-value, and P the p-value.

	diff (t)	p-value (t)	diff	p-value
MEL-CQT	13926.88	4.92E-06	0.0068	5.11E-07
MFCC-CQT	222.24	1.0000	-0.0013	0.8135
RAW-CQT	-17197.21	1.18E-08	-0.0044	0.0024
STFT-CQT	19305.97	1.69E-10	0.0074	4.22E-08
MFCC-MEL	-13704.64	7.21E-06	-0.0080	2.05E-09
RAW-MEL	-31124.09	0.0000	-0.0112	2.85E-14
STFT-MEL	5379.09	0.2569	0.0006	0.9869
RAW-MFCC	-17419.45	7.63E-09	-0.0031	0.0651
STFT-MFCC	19083.73	2.68E-10	0.0086	1.35E-10
STFT-RAW	36503.18	0.0000	0.0118	3.12E-14

Table 2. Tukey-HSD ROC-AUC results for input representations on MTAT with transformations (t) and without.

being able to validate or reject those results. The overall worst performing input representation is the raw waveform. It is 0.44% worse than the CQT transformation with a p-value of 0.0024 and significantly worse for the transformed input with a p-value of 1.18e-08. Compared to the MFCCs, the raw waveform performs 0.0031% worse, but with a p-value of 0.0651, so this difference is insignificant for the untransformed input. However, with the transformed input, there is a strong significance with a p-value of 7.63e-09 that the MFCCs outperform the raw waveform to an equal extent as the CQT transformation does. Comparing the MFCCs with the CQT transformation, neither of both outperforms the other for the transformed input and the untransformed with very high p-values of 1.00 and 0.8135. As displayed before in Figure 4, the former performs better in combination with all architec-

tures except for the VGG-16 model than the latter, where it shows a very poor performance compared to all other input representations. Mel spectrograms in average perform 0.8% better than MFCCs with a p-value of 2.05e-09 and 0.68% better than the CQT transformation with a p-value of 5.11e-07. These strong significance levels are also validated with the transformed inputs. There is no significant difference between the Mel spectrogram and the STFT with high p-values of 0.9869 and 0.2569 for untransformed and transformed inputs, respectively.

	diff (t)	p-val (t)	diff	p-value
musicnn-dil_cnn	-4834.96	0.3634	-0.0005	0.9938
resnet-dil_cnn	-84812.56	0.0000	-0.0269	0.0000
senet-dil_cnn	-78691.83	0.0000	-0.0243	0.0000
vgg16-dil_cnn	-100118.83	0.0000	-0.0370	0.0000
resnet-musicnn	-79977.60	0.0000	-0.0264	0.0000
senet-musicnn	-73856.87	0.0000	-0.0238	0.0000
vgg16-musicnn	-95283.87	0.0000	-0.0365	0.0000
senet-resnet	6120.73	0.1475	0.0026	1.93E-01
vgg16-resnet	-15306.27	4.24E-07	-0.0101	1.32E-13
vgg16-senet	-21427.00	1.95E-12	-0.0127	1.54E-14

Table 3. Tukey-HSD ROC-AUC results for network architectures on MTAT with transformations (t) and without.

Table 3 shows the results for the Tukey-HSD range test on the different architectures. VGG-16 performs significantly worse than all other models. On average, it performs 1.01% worse than ResNet and 1.27% worse than SENet with p-values of $< 1e-06$ for both transformed and untransformed scores. There is no significant difference between the latter two because the p-value is 14.75% for the transformed input. Musicnn and the dilated CNN significantly outperform all other models with $> 2.4\%$ difference. As already discussed previously, there is no significant difference between those two architectures with p-values of 99.38% and 36.34% for untransformed and transformed scores, respectively.

The results on MTGJ have been found to be consistent with those on MTAT, due to space restrictions, the tables are only available on the accompanying website. For audio representations, the raw waveform again performs worse than all other inputs, but now the significance for MFCCs outperforming the raw waveform is much stronger, with a p-value of 8.22e-05 for having a 0.75% higher ROC-score. On average, the Mel spectrogram significantly performs 0.85% better than the MFCCs, and the CQT transformation performs 1.1% better than the latter. In contrast to the previous dataset, the Mel spectrogram and the CQT transformation perform equally well. The STFT performs 0.62% better than the Mel spectrogram. However, the performance difference between STFT and CQT is not statistically significant, with a p-value of 0.1472.

Regarding architectures, there is no significant difference between the musicnn and the dilated CNN, which perform best overall. VGG-16 performs worse than other architectures, followed by ResNet and SeNet.

Additionally, the same evaluation is performed separately for the three tag categories "genre", "instrument" and "mood". The motivation for this is the assumption that tag categories may depend on different properties of music which might be captured differently by individual input representation and network architecture combinations. On MTAT the results for genre and instrument are comparable to the overall results, while for mood the MFCC input representation performed surprisingly well, however, this could not be reproduced on MTGJ. Due to space restriction the detailed results are not provided here, but can be obtained on the accompanying website.

4.3 Discussion

The evaluation shows that both architecture and input representation significantly impact the classification results during all experiments. The STFT provides the best overall results on almost all tested configurations. This comes to the price of higher resource consumption due to the larger size of this input representation. Raw waveforms show the worst results on all configurations which might be because the architectures are originally designed for two-dimensional inputs and are adapted. Despite their small size, MFCCs perform well across almost all architectures, so they might be a good choice for tasks requiring low resource consumption. Mel spectrograms show solid performances across all architectures but are still significantly worse than the STFT. However, due to their around ten times smaller size, they are much less resource-consuming which might be the reason why they are a commonly used input representation. The dilated CNN significantly outperforms the original musicnn on two-dimensional input representations and performs worse on the raw waveform.

On MTAT the CQT transformation has the second-worst results but on MTGJ it performs equally well as the STFT. Therefore, no general conclusion can be drawn about the performance of this input representation which is comparable in terms of size to the Mel spectrogram. The evaluation on individual tag categories was inconsistent between the two datasets making further conclusions difficult. For genre tags, similar results can be observed on MTAT and MTGJ, but there is no particular trend identifiable compared to the evaluation across all tags. The results suggest that mood classification differs the most from the other tag categories. For example, MFCCs perform the second-worst for genre- and instrument classification but the second-best on mood tags. However, this was not the case for MTGJ, where MFCCs only show slightly better results than the raw waveform.

5. CONCLUSIONS

In this work, five CNN architectures and five input representations have been evaluated on two commonly used music tagging datasets. A thorough statistical analysis of the results, identifies significant performance differences between all combinations, where the combination of STFT inputs and the newly introduced dilated variant of the musicnn performs best. The evaluation on tag categories does not yield any consistent trends, but points to some interesting hypothesis, e.g., that for mood tags compressed input representations like MFCC might be sufficient.

6. ACKNOWLEDGEMENTS

This research was funded in whole, or in part, by the Austrian Science Fund (FWF) [P33526]. For the purpose of open access, the author has applied a CC BY public copyright license to any Author Accepted Manuscript version arising from this submission.

7. REFERENCES

[1] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, and T. Sainath, "Deep learning for audio signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, 2019.

[2] M. Won, A. Ferraro, D. Bogdanov, and X. Serra, "Evaluation of CNN-based automatic music tagging models," in *Proceedings of the 17th Sound and Music Computing Conference (SMC)*, Toronto, Ontario, Canada, 2020.

[3] Y.-b. Yu, M.-h. Qi, Y.-f. Tang, Q.-x. Deng, F. Mai, and N. Zhaxi, "A sample-level DCNN for music autotagging," *Multimedia Tools and Applications*, vol. 80, pp. 11 459–11 469, 2021.

[4] K. Choi, G. Fazekas, M. Sandler, and K. Cho, "A comparison of audio signal preprocessing methods for deep neural networks on music tagging," in *Proceedings of the 26th European Signal Processing Conference (EUSIPCO)*, Rome, Italy, 2018.

[5] J. Camacho-Collados and M. T. Pilehvar, "On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis," in *Proceedings of the EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Brussels, Belgium, 2018.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, Nevada, USA, 2016.

[8] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the 2018 IEEE conference on computer vision and pattern recognition (CVPR)*, Salt Lake City, Utah, USA, 2018.

[9] J. Pons and X. Serra, "Musicnn: Pre-trained convolutional neural networks for music audio tagging," in *Late Breaking and Demos of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, Delft, The Netherlands, 2019.

[10] M. Huzaifah, "Comparison of time-frequency representations for environmental sound classification using convolutional neural networks," *arXiv preprint arXiv:1706.07156*, 2017.

[11] M. Won, K. Choi, and X. Serra, "Semi-supervised music tagging transformer," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, Online, 2021.

[12] W.-T. Lu, J.-C. Wang, M. Won, K. Choi, and X. Song, "SpecTNT: A time-frequency transformer for music audio," in *Proceedings of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, Online, 2021.

[13] Y. Gong, Y.-A. Chung, and J. Glass, "PSLA: Improving audio tagging with pretraining, sampling, labeling, and aggregation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3292–3306, 2021.

[14] J. Pons, O. Nieto, M. Prockup, E. Schmidt, A. Ehmann, and X. Serra, "End-to-end learning for music audio tagging at scale," in *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, Paris, France, 2018.

[15] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, San Juan, Puerto Rico, 2016.

[16] F. Yu, V. Koltun, and T. Funkhouser, "Dilated residual networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA, 2017.

[17] K. Muhammad, Mustaqeem, A. Ullah, A. S. Imran, M. Sajjad, M. S. Kiran, G. Sannino, and V. H. C. de Albuquerque, "Human action recognition using attention based LSTM network with dilated CNN features," *Future Generation Computer Systems*, vol. 125, pp. 820–830, 2021.

[18] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.

[19] S. Böck and M. E. Davies, "Deconstruct, analyse, reconstruct: How to improve tempo, beat, and downbeat estimation," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montreal, QC, Canada, 2020.

[20] B. Di Giorgi, M. Mauch, and M. Levy, "Downbeat tracking with tempo-invariant convolutional neural networks," in *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, Montreal, QC, Canada, 2020.

[21] M. E. Davies and S. Böck, "Temporal convolutional networks for musical audio beat tracking," in *Proceedings of the 27th European Signal Processing Conference (EUSIPCO)*, A Coruña, Spain, 2019.

[22] X. Zhang, Y. Zou, and W. Shi, "Dilated convolution neural network with LeakyReLU for environmental sound classification," in *Proceedings of the 22nd International Conference on Digital Signal Processing (DSP)*, London, UK, 2017.

[23] A. Ferraro, D. Bogdanov, X. Serra, J. H. Jeon, and J. Yoon, "How low can you go? Reducing frequency and time resolution in current CNN architectures for music auto-tagging," in *Proceedings of the 28th European Signal Processing Conference (EUSIPCO)*, Amsterdam, The Netherlands, 2021.

[24] B. Logan, "Mel frequency cepstral coefficients for music modeling," in *Proceedings of the International Symposium on Music Information Retrieval (ISMIR)*, Plymouth, USA, 2000.

[25] J. C. Brown, "Calculation of a constant Q spectral transform," *Journal of the Acoustical Society of America*, vol. 89, pp. 425–434, 1991.

[26] T. Kim, J. Lee, and J. Nam, "Sample-level CNN architectures for music auto-tagging using raw waveforms," in *Proceedings of the 43th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB, Canada, 2018.

[27] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, Hawaii, USA, 2017.

[28] X. Lei, H. Pan, and X. Huang, "A dilated CNN model for image classification," *IEEE Access*, vol. 7, pp. 124 087–124 095, 2019.

[29] R. Schuster, O. Wasenmuller, C. Unger, and D. Stricker, "SDC-stacked dilated convolution: A unified descriptor network for dense matching tasks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA, 2019.

[30] E. Law, K. West, M. Mandel, M. Bay, and J. Downie, "Evaluation of algorithms using games: The case of music tagging." in *Proceedings of the 10th International Conference on Music Information Retrieval (ISMIR)*, Kobe, Japan, 2009.

[31] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, "The MTG-Jamendo dataset for automatic music tagging," in *Machine Learning for Music Discovery Workshop, International Conference on Machine Learning (ICML)*, Long Beach, CA, United States, 2019.

[32] T. Bertin-Mahieux, D. Eck, and M. Mandel, "Automatic tagging of audio: The state-of-the-art," *Machine Audition: Principles, Algorithms and Systems*, pp. 334–352, 2010. [Online]. Available: https://www.igi-global.com/gateway/book/40288

[33] G. Marques, M. Domingues, T. Langlois, and F. Gouyon, "Three current issues in music autotagging." in *Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR)*, Miami, Florida, USA, 2011.

[34] E. L. Law, L. Von Ahn, R. B. Dannenberg, and M. Crawford, "Tagatune: A game for music and sound annotation." in *Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, Vienna, Austria, 2007.

[35] K. Choi, G. Fazekas, and M. Sandler, "Automatic tagging using deep convolutional neural networks," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, New York City, USA, 2016.

[36] J. Lee and J. Nam, "Multi-level and multi-scale feature aggregation using pretrained convolutional neural networks for music auto-tagging," *IEEE Signal Processing Letters*, vol. 24, no. 8, pp. 1208–1212, 2017.

[37] M. Won, S. Chun, and X. Serra, "Toward interpretable music tagging with self-attention," *arXiv preprint arXiv:1906.04972*, 2019.

[38] M. Kozak and H.-P. Piepho, "What's normal anyway? Residual plots are more telling than significance tests when checking ANOVA assumptions," *Journal of agronomy and crop science*, vol. 204, no. 1, 2018.

Author Index

Author Index