Lisa Kristina Kerle, BSc

SPEAKER INTERPOLATION BASED DATA AUGMENTATION FOR AUTOMATIC SPEECH RECOGNITION

MASTER'S THESIS

submitted to

Graz University of Technology

Supervisors

Doz. Mag.phil. Dipl.-Ing. Dr.techn. Michael Pucher
Ass.Prof. Mag.rer.nat. Dr. Barbara Schuppler

Signal Processing and Speech Communication Laboratory

Graz, September, 2022

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

date	(signature)

Acknowledgement

At this point I would like to thank all those who have supported me not only in writing my Master's thesis, but also throughout my studies.

First of all, I would like to thank my two supervisors Barbara Schuppler and Michael Pucher, who have always been available to answer my questions and have made a significant contribution to the success of this thesis. Furthermore, I thank Julian Linke, who provided me with his speech recognition recipe, and Lorenz Gutscher for his support in the area of speech synthesis.

I also have to thank the SPSC for the use of the DSP-Lab and Markus Köberl, who helped me a lot with his knowledge of computer technology.

To my fellow students and friends I also want to say thank you, for making my time as a student an unforgettable part of my life, not only by learning together, but also by doing things away from university.

Last but not least, I would like to thank my parents, who have always supported my decisions and helped me in all situations and thus made it possible for me to complete my studies. Likewise, my two sisters were a great support to me during my studies, for which I am very grateful.

Danksagung

An dieser Stelle möchte ich mich bei all jenen bedanken, die mich nicht nur beim Anfertigen meiner Masterarbeit, sondern auch während meiner gesamten Studienzeit unterstützt haben.

Zunächst möchte ich mich bei meiner Betreuerin Barbara Schuppler und meinem Betreuer Michael Pucher bedanken, welche mir bei Fragen stets zur Verfügung gestanden sind und einen wesentlichen Teil zum Gelingen dieser Arbeit beigetragen haben. Des Weiteren gilt auch Julian Linke ein großer Dank, der mir sein Rezept für das Spracherkennungssystems bereitgestellt hat sowie auch Lorenz Gutscher für seine Unterstützung im Bereich der Sprachsynthese.

Für die Benutzung des DSP-Labs am SPSC möchte ich mich ebenfalls bedanken. Markus Köberl, der mir vor allem mit seinem computertechnischen Wissen zu Beginn meiner Arbeit sehr weitergeholfen hat, danke ich ebenfalls.

Ein großer Dank gilt auch meinen StudienkollegInnen und FreundInnen, welche die Studienzeit nicht nur durch das gemeinsame Lernen, sondern auch durch Unternehmungen abseits der Universität zu einem unvergesslichen Lebensabschnitt gemacht haben.

Nicht zuletzt danke ich meinen Eltern, die meine Entscheidungen stets mitgetragen haben, mich in jeglichen Lebenslagen unterstützt und mir somit den Studienabschluss ermöglicht haben. Ebenso waren meine beiden Schwestern während des Studiums eine große Stütze für mich, wofür ich sehr dankbar bin.

Abstract (English)

The development of speech recognition systems in recent years has ensured their widespread use in a broad range of areas. This success is mainly due to the integration of sophisticated algorithms in automatic speech recognition (ASR) using deep neural networks (DNN), which have a strong reliance on their training data. In order to further improve the performance of ASR systems, this thesis investigates the augmentation of the training data set for ASR by means of a DNN-based text-to-speech (TTS) synthesis system.
For this purpose, the following approach is used: Speaker-specific information in the form of speaker embedding vectors is extracted from an existing corpus and the interpolation between these vectors allows for the generation of new speaker characteristics by passing them to a speech synthesis system. Speech synthesised with the resulting new voices is then used to train a GMM/HMM-based ASR system. For the development of the ASR system, the GRASS corpus has already been used, whose texts are to be synthesised with the newly generated voices in order to acoustically augment the GRASS corpus. Subsequent experiments with the augmented corpus investigate how adding utterances from the synthesised voices to the training set affects the performance of the speech recognition system.
The experiments show, that adding a limited amount of synthesised voices leads to an improvement of the speech recognition system, but with a predominance of synthesised voices in the training set the performance deteriorates significantly.
Since this work deals exclusively with read speech, where recognition performances reached are quite high already, it would be interesting to further conduct experiments with conversational speech, where data sparsity is even more of an issue.

Abstract (German)

Die stetige Weiterentwicklung von Spracherkennungssystemen in den letzten Jahren hat für eine breite Anwendung dieser in den verschiedensten Bereichen gesorgt. Dieser Erfolg ist vor allem auf die Integration hochentwickelter Algorithmen in die automatische Spracherkennung (ASR) mittels neuronaler Netzwerke (DNN) zurückzuführen, die stark auf ihre Trainingsdaten angewiesen sind. Um die Performance der ASR-Systeme weiterhin zu verbessern, untersucht diese Arbeit die Erweiterung der Trainingsdaten für ein ASR-System mit Hilfe eines DNN-basierten Text-to-Speech (TTS) Synthesesystems.
Dafür wird folgender Ansatz gewählt: Aus einem bereits existierenden Corpus werden sprecherspezifische Informationen in Form von Vektoren extrahiert, deren Interpolation die Erzeugung neuer Sprechercharakteristiken erlaubt, indem sie dem Sprachsynthesesystem zugeführt werden. Die Sprache, welche mit den resultierenden Stimmen synthetisiert wurde, wird anschließend für das Training eines GMM/HMM-basierten ASR-Systems verwendet. Für die Entwicklung des ASR-Systems wird bereits der GRASS Corpus eingesetzt, dessen Texte mit den neu erzeugten Stimmen synthetisiert werden sollen, um so den GRASS Corpus akustisch zu erweitern. Nachfolgende Experimente mit dem erweiterten Corpus untersuchen, inwiefern sich die Erweiterung des Trainingsets mit synthetisierten Stimmen auf die Performance des Spracherkenners auswirkt.
Es kann gezeigt werden, dass das Hinzufügen einer begrenzten Anzahl an synthetisierten Stimmen zu einer Verbesserung des Spracherkenners führt, bei Überwiegen der synthetisierten Stimmen im Trainingsset verschlechtert sich die Performance jedoch erheblich.
Da sich diese Arbeit ausschließlich mit gelesener Sprache beschäftigt, bei welcher die automatische Spracherkennung bereits recht zufriedenstellende Ergebnisse erzielt, wäre es durchaus interessant, in weiterführenden Arbeiten Experimente mit spontaner Sprache durchzuführen, bei welcher die mangelnde Verfügbarkeit an Daten ein noch größeres Problem darstellt.

Contents

Statutory Declaration **3**

Acknowledgement **5**

Danksagung **5**

Abstract (English) **7**

Abstract (German) **9**

Contents **11**

1 Introduction **13**

2 Neural Network-based Speech Synthesis **17**
2.1 Text-to-Speech (TTS) Synthesis 17
2.1.1 History 17
2.1.2 Conventional TTS Synthesis 17
2.1.3 Development of TTS Synthesis 17
2.1.4 Neural Network-based Speech Synthesis 18
2.2 Merlin - a Neural Network Speech Synthesis System 19
2.2.1 Model Components 19
2.2.2 Dataset 23
2.2.3 Speaker Adaptation 24

3 Speaker Encoding and Interpolation **27**
3.1 Speaker Encoding 27
3.1.1 Conventional Approaches 28
3.1.2 Neural Encoding 29
3.2 Neural Speaker Embedding Extraction based on PyTorch and Kaldi 30
3.2.1 Pipeline Speaker Recognition System 30
3.2.2 Evaluation Metrics 34
3.2.3 Experiments 36
3.2.4 Speaker Embedding 46
3.3 Interpolation of Speaker Embeddings 46

4 Speaker Embedding-based Speaker Adaptation and Data Augmentation **49**
4.1 Speaker Embedding-based Speaker Adaptation 49
4.1.1 Results 50
4.2 Data Augmentation 51
4.2.1 GRASS Corpus 51
4.2.2 GRASS Corpus Augmentation 51
4.2.3 Evaluation 53

5 Automatic Speech Recognition **59**
5.1 Introduction to Automatic Speech Recognition (ASR) 59
5.1.1 Field of Application 59
5.1.2 Challenges 59
5.1.3 Development of ASR Systems 60

5.2 Building a Kaldi-based ASR System . . . 61
5.2.1 Data Preparation . . . 61
5.2.2 Feature Extraction . . . 63
5.2.3 Dictionary . . . 63
5.2.4 Language Model . . . 63
5.2.5 Acoustic Model . . . 65
5.2.6 Hidden Markov Models (HMM) . . . 65
5.2.7 Evaluation . . . 66

6 ASR Experiments **67**
6.1 Experiment Settings . . . 67
6.2 Baseline Experiment . . . 68
6.2.1 Result . . . 68
6.3 Corpus Augmentation using Synthesised Speech . . . 69
6.3.1 Amount of Synthesised Speech . . . 69
6.3.2 Selection of Utterances . . . 69
6.3.3 Results . . . 69
6.3.4 Optimisation of Data Augmentation . . . 72
6.3.5 Results . . . 72
6.4 Replacement using Synthesised Speech . . . 75
6.4.1 Results . . . 75
6.5 Using the Synthesised Corpus for Training . . . 78
6.5.1 Results . . . 78
6.6 Using the WASS Corpus for Training . . . 81
6.6.1 Corpus Mismatch . . . 82
6.6.2 Result . . . 82

7 Discussion **87**
7.1 Lessons learnt from Speech Synthesis . . . 87
7.2 Lessons learnt from ASR Experiments . . . 89
7.3 Outlook . . . 91

8 Conclusion **93**

A Appendix **95**
A.1 Chapter 6 . . . 96
A.1.1 Section 6.3 . . . 96
A.1.2 Section 6.4 . . . 98
A.1.3 Section 6.5 . . . 104
A.1.4 Section 6.6 . . . 105

References **107**

List of Figures **111**

List of Tables **115**

1 Introduction

In recent years, the performance of automatic speech recognition (ASR) has increased significantly. This improvement led to a broad use of ASR in everyday life, making ASR more in demand than ever. However, ASR is a highly complex process, that still faces many challenges and solving them requires knowledge from a wide range of fields. Therefore, continuous development of ASR systems is essential in order to further improve their performance.
The state-of-the-art ASR systems are based on neural networks. An essential aspect of the success of ASR systems is the training process for which thousands of hours of transcribed speech data is required [1]. This transcription mainly relies on humans, which makes the collection of training data for DNN-based ASR systems very time consuming and expensive. Additionally, for robust speech recognition, lexical and acoustic diversity is necessary, requiring not only hours of voice recordings with a variety of linguistic features but also a large number of individual speakers to guarantee the acoustic diversity [2]. Therefore, the collection of training data with a sufficient acoustic variety is costly and an essential issue of developing neural network-based ASR systems.
Parallel to the development of ASR systems, speech synthesis systems have also improved their performance in recent years. Especially the successful integration of neural networks in the speech synthesis process allowed the generation of more natural-sounding, human-like speech. State-of-the-art speech synthesis systems at an end-to-end level, e.g., Tacotron 2, generate speech with high naturalness based on a sequence-to-sequence prediction network with attention mechanism [3]. However, more conventional speech synthesis systems using FFDNN for acoustic modeling still produce intelligible speech at high quality, e.g, Merlin [4].

To combine the progresses made in both fields and to overcome the effort of collecting training data from numerous real speakers in terms of acoustic diversity for the ASR training, the training corpus could be augmented with natural-sounding synthesised speech. The aim of this thesis is to investigate whether the augmentation of an existing ASR training corpus with synthesised voices, i.e., newly generated speakers, leads to an improvement in ASR performance. This hypothesis is based on previous work from Rosenberg [2] and Fazel [1], which have already experimented with the use of synthesised speech in training sets, showing promising results.
Rosenberg et al. (2019) [2] tested the hypothesis of improving the performance of speech recognition systems by adding more transcribed training data generated by speech synthesis systems. For speech synthesis, Tacotron 2 with a multi-speaker speech synthesis model with WaveRNN vocoder was used, which produced speech from multiple speakers based on a fixed sized speaker embedding. Besides the acoustic diversity in terms of speaker variability, they also investigated the inclusion of unseen utterances produced by speech synthesis in the training data, i.e., to increase the lexical diversity. In Rosenberg [2], speaker information is generated via three different d-vector approaches, based on which speech was synthesised for data augmentation in terms of acoustic diversity. They found that sampling from observed speaker representations provided the best results in terms of speaker diversity (gain of 4% WER). By using the sampling approach to speaker conditioning, the performance of data augmentation was investigated. 960 hours of synthesised material were used in combination with different portions of real speech to investigate the influence of the reduction of real speech on ASR performance. They showed that

with less real speech data, data augmentation becomes more effective, although the performance degrades when the amount of real speech is reduced in the training process.
Fazel et al. (2021) [1] proposed to use synthetic speech for ASR training in order to reduce the reliance on human transcribed data by training ASR models on production like data synthesised from text-to-speech (TTS) engines. In two experiments the acoustic as well as the lexical diversity was increased to improve ASR performance. They increased the acoustic diversity by synthesising input text with randomly selected voice profiles, which provides synthetic speech with diverse speaker attributes. The system used by Fazel et al. [1] contained a multi-context TTS engine to synthesise speech and a RNN-T model for speech recognition, which were trained in a multi-style training using real speech recordings on TTS-based synthetic speech. In order to generate synthetic speech with diverse speaker attributes, in [1] the speaker identities were controlled with voice profile embeddings obtained from a speaker verification system. Additionally they followed a variational auto-encoder (VAE) approach to increase the inter- and intra-speaker variability and furthermore the speaker diversity. They showed that by using synthetic speech with randomly selected voice profiles in addition to real speech data, the performance of speech recognition is improved compared to speech recognition using real speech only. Similar to [2], they showed that using synthetically generated speech in addition to real data, ASR performance is improved. However there is still a decrease in performance when replacing parts of real speech by synthetic speech.

For building an ASR system for read Austrian German, a Kaldi-based system was developed using the GRASS corpus, which is also the corpus to be augmented in this work. The aim of this thesis is to confirm the hypothesis that an augmentation of the GRASS corpus by means of synthesised voices leads to an improvement in ASR performance. Therefore, only the read speech part of the GRASS corpus will be used in this first step and conversational speech including cross talk or interaction between speakers is not considered within the scope of this thesis.
Previous research [1, 2] has already shown that by including speaker-specific information in the synthesis process, speech with diverse speaker attributes can be generated. The speech synthesis in this work is performed by using Merlin, a neural network-based speech synthesis system, which is described in more detail in Chapter 2 after a brief introduction to conventional text-to-speech synthesis (TTS). Merlin uses an average voice model (AVM) approach, which contains information from multiple speakers that allows a speaker adaptation by adding speaker-specific information to the AVM.
Figure 1.1 illustrates the approach applied in this thesis: Based on a given dataset, speaker-specific information is obtained by using a speaker recognition system (cf. Chapter 3). This speaker-specific information is extracted from audio files in form of vectors, called speaker embedding. In order to generate new voices, the obtained speaker embedding vectors are interpolated, resulting in vectors containing new speaker characteristics. The interpolated speaker embedding vectors are then used to augment the GRASS corpus (cf. Chapter 4). Therefore, the embedding is given to the speech synthesis system in addition to the linguistic features extracted from the GRASS corpus. The synthesised utterances based on the newly generated voices augment the training set of the ASR system (cf. Chapters 5 and 6). A subsequent discussion analyses the process of speaker interpolation-based data augmentation and interprets the results of the ASR system. In addition, the outcome is compared to previous work and an outlook on further research is given (cf. Chapter 7). The final conclusion summarises the relevant parts of this thesis and points out the most essential findings (cf. Chapter 8).

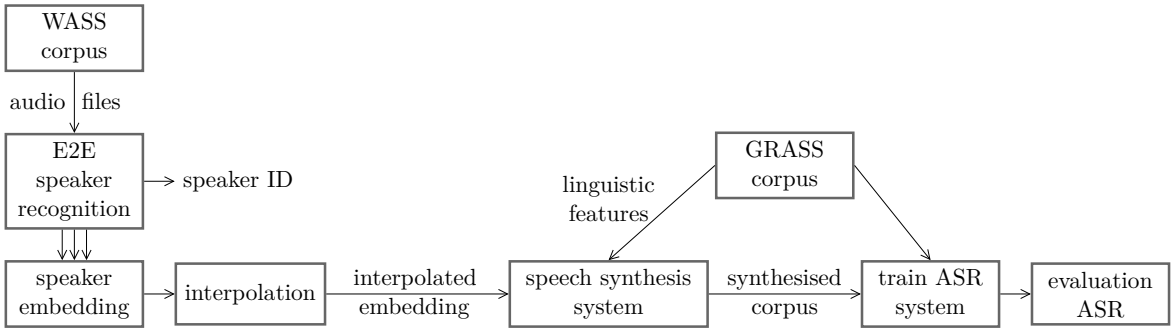


Figure 1.1: Process of speaker interpolation-based data augmentation: Extraction of speaker embedding vectors containing speaker-specific information from an end-to-end speaker recognition system based on the audio files of the WASS corpus. The speaker embedding vectors are interpolated and used as additional input to the linguistic features for the speech synthesis system, in order to generate new voices for augmenting the GRASS corpus. The augmented corpus is used as training set for the ASR system, whose performance is evaluated afterwards.

2 Neural Network-based Speech Synthesis

2.1 Text-to-Speech (TTS) Synthesis

2.1.1 History

The primary task of speech synthesis systems is to generate a corresponding speech signal from a given text. As early as the second half of the 18th century, science was concerned with the field of speech synthesis. One of the first speech synthesisers was Kempelen's voice synthesiser. The invention of the computer gave rise to the first computer-based speech synthesis systems leading to the statistical parametric speech synthesis (SPSS) system. This system learnt acoustic parameters from linguistic features by using Hidden Markov Models (HMM) [5]. With the increasing success of neural networks around 2010, the HMM-based acoustic model of the SPSS was replaced by a neural network which still predicts acoustic features based on linguistic input. Further improvements in terms of neural networks led to the state-of-the-art end-to-end (E2E) speech synthesis systems, which predict acoustic features from phoneme sequences rather than linguistic features [5].

2.1.2 Conventional TTS Synthesis

A conventional TTS system, e.g., a statistical parametric speech synthesis (SPSS) system, consists of the components shown in Figure 2.1.

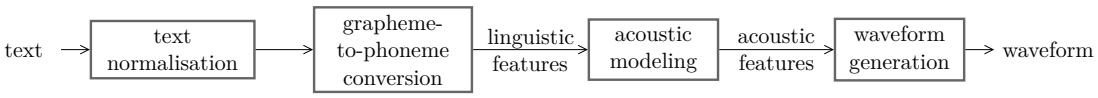


Figure 2.1: Components of a conventional TTS synthesis system to convert a given text into a speech signal including text normalisation, grapheme-to-phoneme conversion, acoustic modeling and waveform generation [5].

The first step of conventional TTS systems is the text analysis, the so-called frontend, to produce linguistic features out of a given text input. This step involves text normalisation, which converts raw written text into spoken-form words and is mainly used for numbers or abbreviations. Afterwards the obtained characters (graphemes) are converted into pronunciations (phonemes) by using for example a look-up dictionary. An acoustic model follows which plays an essential role in speech synthesis as it generates acoustic features from linguistic features or even from phonemes or characters. The resulting acoustic features are then passed to a waveform generator (i.e., a vocoder) to generate the desired speech waveform [5].

2.1.3 Development of TTS Synthesis

Due to the progress made in the field of speech synthesis, the individual components of the TTS synthesis system have changed over time and as a result different realisations of TTS

synthesis systems exist. Especially the acoustic model has been strongly influenced by improved approaches. Figure 2.2 shows the development of the contribution of different concepts to the speech synthesis process. At the end of the 20th century there was an almost balanced distribution of these concepts. In recent years, HMM have become increasingly popular until neural network approaches came to dominate speech synthesis around 2019 [6].

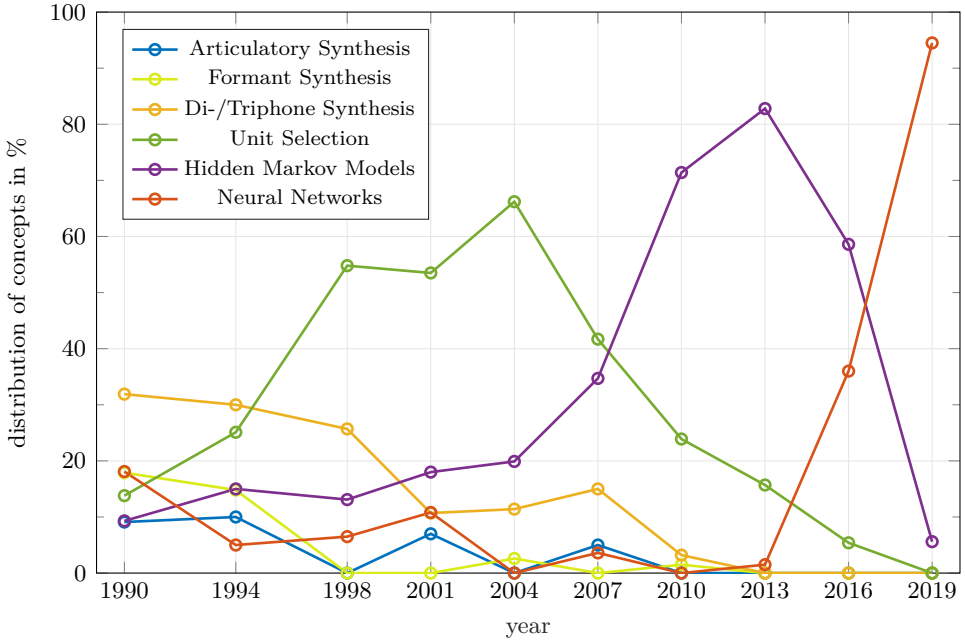


Figure 2.2: Distribution of speech synthesis concepts since 1990 [6].

In this work, the process of speech synthesis is done by means of Merlin [4], a neural network speech synthesis system (cf. Section 2.2). Therefore, the process of speech synthesis based on neural networks is described in more detail in Section 2.1.4.

2.1.4 Neural Network-based Speech Synthesis

As early as in the 1990s, neural networks were used to predict acoustic features from linguistic features. The networks used nowadays differ mainly in their increased complexity in terms of hidden layers and training algorithms due to more advanced computational resources [4]. This progress in the field of neural networks led to the use of neural networks in SPSS for acoustic modeling [5], in which they replaced HMMs to convert linguistic features obtained by text analysis into acoustic features (e.g., fundamental frequency (F0), mel-generalised coefficients (MGC), band aperiodicity (BAP)). Numerous models still follow this TTS approach (cf. Figure 2.2), where not only the acoustic model is replaced by neural networks but also other components, e.g., neural network-based vocoders.
As a next step, end-to-end models were designed (e.g., Tacotron 1/2 [7]) which directly take characters or phonemes as input, instead of labels. Further development led to fully end-to-end TTS systems in order to generate waveforms directly from text. The usage of these systems reduces the reliance on human contributions and thus the number of possible error sources. Apart from that the integration of neural networks in speech synthesis yields a higher voice quality in terms of the naturalness and intelligibility of the synthesised speech, which explains their wide use in state-of-the-art TTS systems [5].

2.2 Merlin - a Neural Network Speech Synthesis System

Merlin, an open source neural network toolkit for speech synthesis, provides DNN model building for statistical parametric speech synthesis (SPSS) [4]. For speech synthesis, a feed-forward deep neural network (FFDNN) is trained based on linguistic input and acoustic output features. In contrast to already mentioned end-to-end systems (cf. Section 2.1.4), which allow speech synthesis from raw text input, Merlin requires an external frontend, e.g., Festival, to convert text into HTS-style labels prior to the speech synthesis process (cf. Section 2.2.2). Merlin transforms these HTS-style formatted labels into a linguistic feature vector, which is used as input for the model during training. The acoustic features, which are extracted from the provided audio files, are used as output to the model to be trained (cf. Figure 2.3, top).
The trained model that has learnt the relation between linguistic and acoustic features is employed during speech synthesis to predict the corresponding acoustic features based on the given input labels, which are then passed to a vocoder to produce the desired speech waveform [4] (cf. Figure 2.3, bottom).

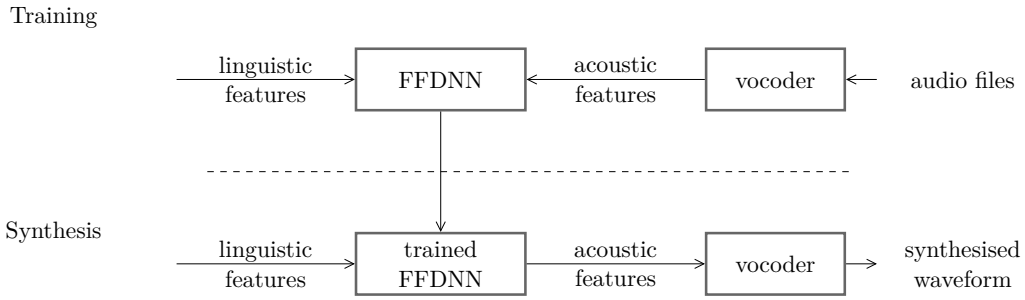


Figure 2.3: Model training and speech synthesis based on Merlin: During training the model learns the relation between linguistic and acoustic features (top). The linguistic feature vectors are used as input for the trained model (duration model and acoustic model) to predict acoustic output features which are passed to a vocoder to generate the speech waveform (bottom).

2.2.1 Model Components

Linguistic Input Features

As the linguistic information must be in the form of labelled text in order to be processed, Merlin applies an external frontend which produces labels in HTS-style format [8] from a given text. Labels can be either phone aligned or state aligned, where each phone in state alignment consists of multiple states. The provided labels are phone aligned and already HTS-style formatted. These formatted labels were first transformed into feature vectors by using a question file which outputs a feature vector containing linguistic contextual and positional information. The HTS-style question file is language dependent and contains questions about the current phoneme to finally represent this information in compact form. Before using these feature vectors as input for the neural network, a min-max normalisation was carried out to normalise features to the interval [0.01 0.99] (cf. Figure 2.4) [4].

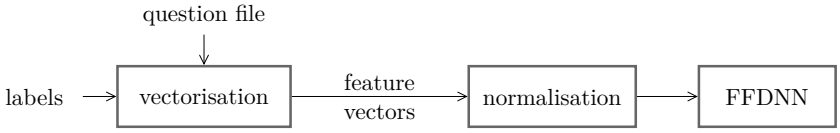


Figure 2.4: Pre-processing of linguistic features: vectorised and normalised HTS-style formatted labels used as input feature vectors for the FFDNN (duration model and acoustic model).

Acoustic Output Features

While linguistic features are used as input for the neural network, acoustic features represent the output of the neural network during the training process. The required acoustic features were extracted from the provided audio files by means of a vocoder. By applying mean-variance normalisation, the acoustic output features were normalised to zero mean and unit variance (cf. Figure 2.5). The acoustic output features consist of Mel-Cepstral Coefficients (MCCs), band aperiodicities (BAPs) and fundamental frequency on log scale (log F_0) with their deltas and delta-deltas. Additionally a voiced/unvoiced (V/UV) binary feature is included in the acoustic output features.

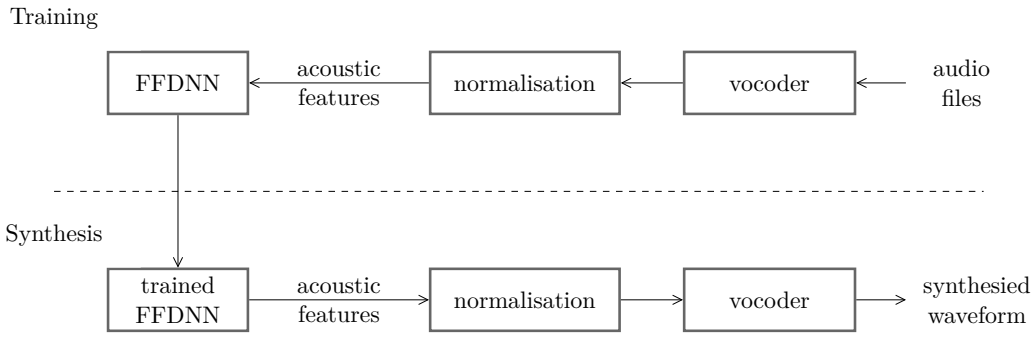


Figure 2.5: Normalisation of acoustic features: extraction of acoustic features by using a vocoder, which are normalised (mean-variance normalisation) and given as output features to the FFDNN during training (top). At the synthesis step the trained FFDNN predicts acoustic features which are normalised and given to a vocoder in order to generate the corresponding waveform (bottom).

Vocoder

Merlin mainly supports two vocoders: STRAIGHT and WORLD [4]. In this work, the WORLD vocoder was used. First, the WORLD vocoder acted as speech analyser, which extracted acoustic features from the provided audio files given as output features to the neural network during training (cf. Figure 2.5, top). The following acoustic features were extracted from the provided audio files by using the WORLD vocoder [4]:

- bap: 5-dimensional band aperiodicities; power ratio between speech signal and the aperiodic component of the signal
- lf0: fundamental frequency on log scale at 5 ms frame intervals
- mgc: 60-dimensional MCCs (Mel-Cepstral Coefficients)

Second, the vocoder reconstructed the desired speech waveform from the predicted acoustic output features at the synthesis step (cf. Figure 2.5, bottom).

Training Process

In this work, FFDNN were used for modeling the parameters of the speech synthesis system [4]. Training the speech synthesis system means learning the relationship between input and output features, i.e., learning how to map linguistic feature vectors onto acoustic vocoder parameters. The employed FFDNN learns to predict the acoustic output frame by frame using several hidden layers to perform nonlinear activation functions [4]:

$$\mathbf{h}_t = \mathcal{H}(\mathbf{W^{xh}}\mathbf{x}_t + \mathbf{b}^h),$$
$$\mathbf{y}_t = \mathbf{W^{hy}}\mathbf{h}_t + \mathbf{b}^y,$$

where $\mathcal{H}(\cdot)$ describes the nonlinear activation function in a hidden layer, $\mathbf{W^{xh}}$ and $\mathbf{W^{hy}}$ are the weight matrices, $\mathbf{b}^h$ and $\mathbf{b}^y$ are bias vectors and $\mathbf{W^{hy}}\mathbf{h}_t$ is a linear regression to predict target features from the activations in the preceding hidden layer. An example of a feed forward neural network is shown in Figure 2.6, in which a neural network with 4 hidden layers predicts vocoder parameters based on linguistic input vectors.

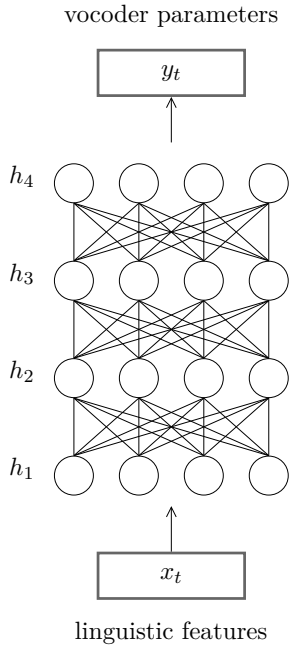


Figure 2.6: A feed forward deep neural network with four hidden layers to predict vocoder parameters from given linguistic input features [4].

For the training process, a pair <label, audio file> was given to the toolkit in order to provide the required linguistic input features and acoustic output features. Merlin consists of two main parts, namely the duration model and the acoustic model, which are both FFDNN-based and trained separately [4]. Therefore, training the speech synthesis systems means to train the duration model as well as the acoustic model [9].
The duration model was trained on the aligned data, i.e., the labels, which contain alignments for the phones from the dataset, in order to predict phone-level duration. Additionally, the acoustic model learnt the relation between acoustic features and the given labels (cf. Figure 2.7).

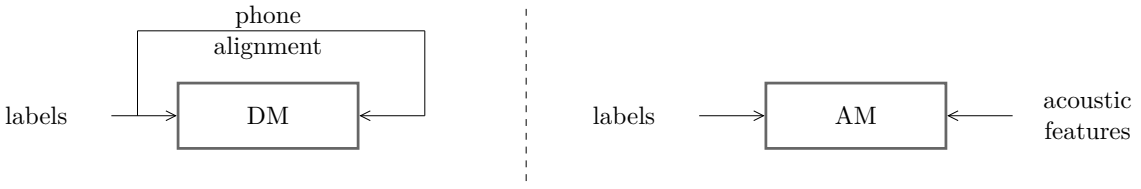


Figure 2.7: Separate training of duration model (DM) and acoustic model (AM): the DM takes phone alignments as input in order to learn how to predict phone-level duration (left), likewise the AM receives the label files as input as well as the acoustic output features to learn their relationship (right).

To train the duration model and the acoustic model the settings were stored in the corresponding configuration files. First the ratio between training, validation and test set was determined. The training set consisted of 80% of the label files, and validation and test set each contained 10% of the label files. The network to be trained was a 6-layer FFDNN using hyperbolic tangent units (TANH), where each layer had a size of 1024. Table 2.1 shows the settings for the duration and acoustic model and the output dimensions of both models are shown in Table 2.2. For the vocoder, the waveform specifications were defined as shown in Table 2.3.

Table 2.1: Settings regarding the network architecture as stored in the configuration file for the duration model and the acoustic model.

parameter	duration model	acoustic model
dropout rate	0.0	
batch size	64	256
decay	exponential	
learning rate	0.002	
optimiser	stochastic gradient descent	
warm-up epochs	10	
training epochs	25	

Table 2.2: Settings regarding the dimension of the output features as stored in the configuration file for the duration model and the acoustic model.

parameter	duration model	acoustic model
dur	1	-
mgc/dmgc	-	60/180
bap/dbap	-	5/15
lf0/dlf0	-	1/3

Table 2.3: Settings regarding the waveform as stored in the configuration file for the duration model and the acoustic model.

parameter	duration model	acoustic model
sampling rate	-	$48000kHz$
frame length	-	2048
frequency warping (α)	-	0.77
minimum phase order	-	1023

Synthesis

For speech synthesis, duration was predicted first by giving HTS-style formatted labels as input to the previously trained duration model. The predicted duration was used as an input to

the acoustic model [9] in addition to the linguistic input features, i.e., the labels. The trained acoustic model then generated acoustic features based on the linguistic features and the duration, which were then passed to the vocoder to synthesise the desired speech waveform (cf. Figure 2.3, bottom and Figure 2.8). In summary, Merlin only needs HTS-style formatted labels to first predict the duration and furthermore the acoustic features for generating the corresponding speech waveform using the previously trained duration and acoustic model.

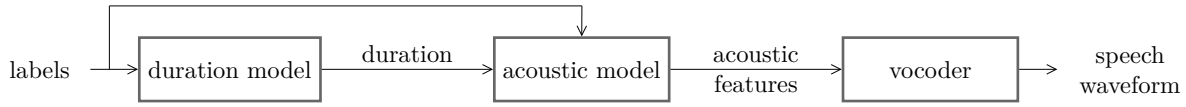


Figure 2.8: The trained duration model takes labels as input to predict duration, which is further used as input to the trained acoustic model in addition to the labels in order to produce the corresponding acoustic features required for the speech waveform generation by means of a vocoder.

2.2.2 Dataset

WASS Corpus

The given dataset, in this work called WASS corpus (Wiener Corpus of Austrian Varieties for Speech Synthesis), has already been collected in the projects VSDS (Viennese Sociolect and Dialect Synthesis), AVDS (Adaptive Audio-Visual Dialect Synthesis) and SALB (Speech synthesis of auditory lecture books for blind children) [10, 11, 12] and consists of 19 different speakers of which 13 are male and 6 are female. It contains, for example, sentences from the Berlin-Marburg corpus and the Kiel corpus, resulting in a total of 8293 utterances, of which about 4540 contain a different word sequence (cf. Table 2.4). A professional speaker was recorded, who constitutes a large part of the dataset (about half of the utterances). The standard for all speakers was "standard Austrian German" and speakers were recorded individually reading the utterances, so there is no interaction between speakers.

Labels and Audio Files

For each utterance there was an audio file (.wav) available with matching label files containing linguistic information (.lab) in HTS-style format using phone alignment [8] (cf. Infobox 2.1). Audio files were provided to get the acoustic features of the utterance whereas label files give the temporal boundaries of quinphones (in $10^{-7}s$), i.e., a context with a centerphone and 2 phones to the left and 2 phones to the right was taken into account. Additionally the linguistic context and the positional information were considered when using HTS-formatted labels as well as prosodic information in terms of stressed syllables.

```
  ...       ...                                 ...
5620000   6060000    GS^I-s+t=S@3_2/A:0_-1_2/B:1-0-4@1-1&3-7#1-3$1-3!2-1;2-3|0/C:1+0+2/D:content_2/
                     E:content+1@2+3&2+2#1+1/F:content_2/G:0_0/H:9=4@1=1|L-L%/I:0=0/J:9+4-1
6060000   6400000    I^s-t+S=P2h@4_1/A:0_-1_2/B:1-0-4@1-1&3-7#1-3$1-3!2-1;2-3|0/C:1+0+2/D:content_2/
                     E:content+1@2+3&2+2#1+1/F:content_2/G:0_0/H:9=4@1=1|L-L%/I:0=0/J:9+4-1
  ...       ...                                 ...
```

Infobox 2.1: Provided labels in HTS-style format for two quinphones which are part of an utterance in the provided dataset. The first two columns describe the temporal boundaries to the corresponding quinphone and the third column contains linguistic information.

Table 2.4: Proportion of the 8293 utterances per speaker in the WASS corpus and additional information.

speaker ID	number of different utterances	gender	description
bje-at	89	male	little data available
bsc-at	223	female	
csc-at	223	male	
dsc-at	223	male	
esc-at	320	male	
fwa-at	89	male	little data available
gun-at	223	female	
hoi-at	223	male	
hpo-at	221	male	
ioka-at	223	male	strong accent
jage-at	223	female	
kep-at	297	male	
lsc-at	223	male	
mpu-at	223	male	
nke-at	223	female	
psc-at	223	female	
spo-at	4378	male	professional speaker
tfe-at	223	female	
wke-at	223	male	

2.2.3 Speaker Adaptation

Average Voice Model (AVM)

Training speaker-dependent neural networks requires a large amount of data from individual speakers. To overcome this problem, first a multi-speaker model was generated (average voice model, AVM), trained by the data from all available speakers in the database (cf. Section 2.2.2). Based on the generated AVM, speaker-dependent models were adapted by sharing all layers from the average voice model. The use of an AVM as intermediate step made it possible to train speaker-specific neural networks even with a small amount of speaker data and to generate speaker-specific models from the average voice model [13].
The Centre for Speech Technology Research (CSTR) in Edinburgh provides a script for speaker adaptation based on Merlin [14]. In order to build an AVM, the steps 01 to 07 from [14] were carried out. They include the preparation of the label files, the extraction of acoustic features, the training of the duration and acoustic model as well as the speech synthesis. The settings shown in Tables 2.1 - 2.3 have been defined in the configuration file for the duration model and the acoustic model.
For training the AVM, different combinations of speakers from the given dataset (cf. Section 2.2.2) were tested. Firstly an AVM was trained by using all available speakers in the dataset.

Additionally, male and female voices were considered separately by generating a male AVM and a female AVM from which male and female speakers were adapted.

Stand-alone Model

For comparison reasons, stand-alone models were created for all speakers in the dataset. Therefore steps 01 to 07 from [14] were executed similar to the AVM, but compared to the AVM, which uses information from all speakers, the stand-alone models were generated by using only the corresponding speaker.

Speaker Adaptation

Based on the trained AVM, individual speakers were adapted according to steps 08 to 13 from [14]. These steps are similar to steps 01 to 07 from the AVM process but instead of training the duration and acoustic models from the beginning, they use the parameters of the previously trained AVM. As the neural networks for the individual speakers do not have to be trained from scratch, this method is very efficient.
To train the neural network for the individual speaker, the speaker-specific linguistic input and acoustic output features were given to the neural network. Additionally, the parameters from the previously trained AVM were passed to the neural network during training. This allowed generating a speaker-dependent neural network even with a small number of speaker-specific data. By passing the acoustic output features of the speaker-dependent model to the vocoder, a "new speaker" was generated (cf. Figure 2.9).

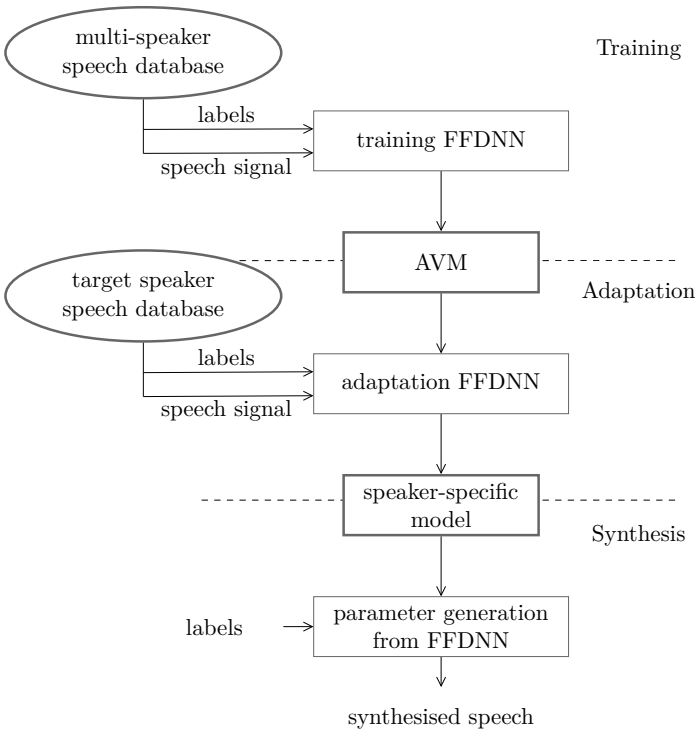


Figure 2.9: Speaker adaptation based on Merlin including the training of the AVM by using multi-speaker information, the adaptation of the speaker-specific model and the subsequent speech synthesis.

Results

According to the speaker adaptation description in [14], the first steps include building an average voice model (AVM) over multiple speakers. Therefore, all speakers from the database (cf. Section 2.2.2) were used to generate an AVM.
Before individual speakers have been adapted from this AVM, stand-alone models were built in order to compare them to the adapted models. Building these stand-alone models showed that the amount of training data for training the duration and acoustic model is essential for the performance of the speech synthesis system. Especially for speakers with little training data (only about 90 training utterances), the synthesised speech was really noisy and not intelligible, whereas performance increased for speakers with more training data. This suggested the use of an AVM instead of stand-alone models.

Afterwards speakers were adapted from the AVM. The adaptation led to more individuality in speech in contrast to the monotonous average voice.
When comparing the synthesised speech based on speaker adaptation to the speech generated from stand-alone models, synthesis performance increased mainly for speakers with less training data. Speakers with an adequate number of training data did not improve by using speaker adaptation.

In addition, the process of building an AVM was repeated to create separate AVM for male and female speakers. The comparison of the different AVMs demonstrated that the AVM based on female speakers has a higher pitch than the mixed AVM and, conversely, the AVM based on male speakers has a significantly lower pitch than the mixed AVM. As the database contains only a low number of female speakers, the quality of the AVM based on female speakers was lower than that based on male speakers. Since the subsequent adaptation indicated that splitting male and female speakers does not lead to an improvement in overall quality, the mixed AVM was chosen as basis for further work.

The analysis of the adapted voices from the mixed AVM showed that the quality of the synthesised speech was not sufficient for two speakers, resulting in the exclusion of these speakers (ioka-at and jage-at). Therefore, the AVM used in this work was trained by using only 17 out of the provided 19 speakers.

3 Speaker Encoding and Interpolation

3.1 Speaker Encoding

In order to obtain a compact representation of speaker-specific information from a given dataset, an appropriate extraction and encoding of speaker-specific features is required. Different approaches on the representation of speaker information have been developed in history, which have mainly been employed in speaker recognition models. In speaker recognition systems, speaker-specific information is extracted from a given waveform based on which the speaker recognition process is performed. The term speaker recognition includes the process of speaker identification, i.e., the classification of a speaker to a specific identity, as well as the speaker verification, which determines whether a speech sample belongs to a specific speaker identity [15].
Conventional speaker recognition systems consist of four main parts [15]:

- local feature description to describe a variable-length input sequence as features, e.g., mel-frequency cepstral coefficients (MFCCs),
- a dictionary, which contains several temporal orderless center components to learn and describe statistics,
- vector encoding to aggregate the variable-length input feature sequence into an utterance-level vector representation based on the dictionary,
- decision generator for speaker recognition.

This conventional pipeline in speaker recognition systems works well also for a limited amount of data. As the number of labelled data and the computational capability has increased in recent years, speaker recognition systems are moving towards an end-to-end learning approach, which employs a general encoding layer in order to encode variable-length input sequence to an utterance-level representation [15] (cf. Section 3.1.2). This end-to-end approach allows the direct modeling from utterances and often reduces the model complexity and the number of required concepts [16].
For both approaches, the representation of speaker-specific information in a compact form is essential in the process of speaker recognition. The encoding of speaker information reduces high-dimensional data into a low dimensional feature vector, which should be ideally independent of the utterance length. Despite the reduction of the dimension, most of the information is preserved by using an appropriate encoding method [15].
The timeline in Figure 3.1 gives an overview of the development from conventional speaker encoding approaches based on Gaussian Mixture Models (GMM) leading to the state-of-the-art neural encoding (x-vectors) [17].

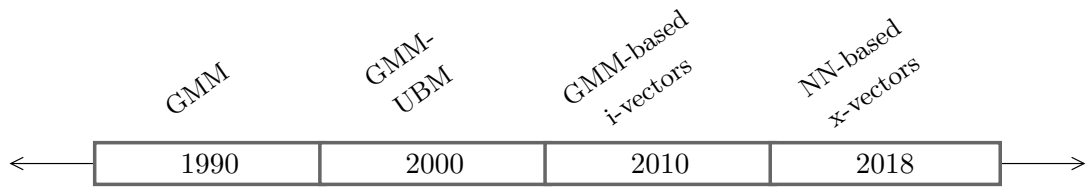


Figure 3.1: Timeline of automatic speaker recognition: Conventional speaker recognition is based on GMMs and their adaptations (e.g., GMM-UBM, i-vector approach). More state-of-the-art encoding techniques employ neural networks for robust speaker recognition using x-vectors [17].

3.1.1 Conventional Approaches

Gaussian Mixture Models (GMM)

The conventional approach of speaker encoding in speaker recognition systems is based on GMMs. A GMM $p(x|\lambda)$ consists of a finite sum of Gaussians $\mathcal{N}(x|\mu_i, \Sigma_i)$ with mean μ_i and covariance matrix Σ_i, weighted by the model weights α_i [18]:

$$p(x|\lambda) = \sum_{i=1}^{N} \alpha_i \mathcal{N}(x|\mu_i, \Sigma_i). \tag{3.1}$$

In Equation 3.1 the summed and weighted components reflect vocal tract configurations and therefore, GMMs are useful to model speaker-specific speech by choosing appropriate model parameters $\lambda = \{\alpha_i, \mu_i, \Sigma_i\}$. During model training, the model parameters λ have to be estimated using the so-called maximum likelihood estimation. Given a distribution of feature vectors, i.e., mel-frequency cepstral coefficients (MFCCs), the model parameters have to be learned in order to maximise the likelihood of the GMM [18].

An adapted version of the GMM is used in speaker recognition systems, called Universal Background Model (UBM). An UBM is a large GMM which is trained on a huge amount of speech data from several speakers to represent a speaker-independent model, consisting of mixture weights and a normal distribution with mean and covariance of the Gaussians [18].

GMM Supervector

By using the maximum a posteriori (MAP) adaptation, the means of the UBM are updated according to a given speaker utterance. The updated mean values yield the so-called GMM supervector, which represents speaker characteristics in a compact form [19]. The speaker-independent UBM and the resulting speaker-specific GMM with adapted means are then compared by calculating the log-likelihood ratio between the two models, which is furthermore used for the speaker verification process [18].

GMM i-vector

Representing speaker information in form of an i-vector is the most recent approach based on an universal background model (UBM) and is derived from the GMM supervector approach. A speaker-independent supervector $\mathbf{m}$ derived from an UBM is used to form a speaker-specific supervector $\mathbf{M}$ including a total variability matrix $\mathbf{T}$ (combining speaker and channel variabilities) and a random vector $\mathbf{w}$ containing normal distributed total-variability factors, called i-vector (identity vector) [18]:

$$\mathbf{M} = \mathbf{m} + \mathbf{T}\mathbf{w}. \tag{3.2}$$

After normalisation and whitening of the resulting i-vector, the score between model and test i-vectors is calculated and a classifier, e.g., probabilistic linear discriminant analysis (PLDA), is employed for speaker verification [18].

3.1.2 Neural Encoding

A further development in speaker encoding is the extraction of speaker information from an end-to-end neural network-based speaker recognition system (cf. Figure 3.2) [15].

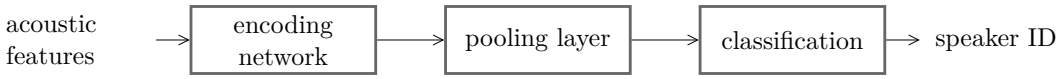


Figure 3.2: Almost end-to-end neural network speaker recognition system consisting of an encoding network, a statistical pooling layer and a classifier to identify a speaker based on a given speech segment [20].

This end-to-end system takes a variable length input sequence, e.g., MFCCs, as input and an encoding network acts as frame-level feature extractor (TDNN - time delay neural network). A subsequent layer, the encoding layer or pooling layer, aggregates this frame-level features over the entire sequence to get an utterance-level result which is passed to the output layer for decision making (classification). Between the pooling layer and the classification step, the encoded fixed-dimensional utterance-level representation is extracted, also called speaker embedding, which describes speaker characteristics in a compact form (cf. Figure 3.3). Therefore, the choice of an appropriate encoding layer for the representation of speaker information is essential [20].

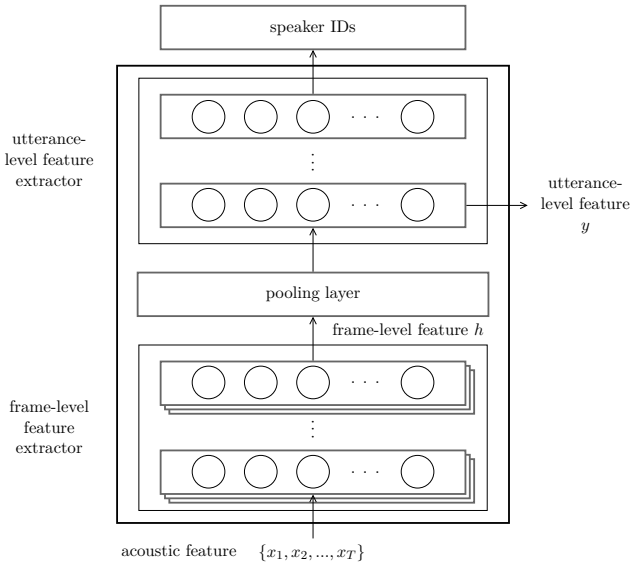


Figure 3.3: DNN for extracting an utterance-level speaker representation (speaker embedding) between the statistical pooling layer and the classifier [21].

One possible encoding layer is the temporal average pooling layer (TAP), which averages the sequence of extracted frame-level features over time. As not all frames contribute equally to the utterance-level representation in terms of speaker information, the self-attentive pooling layer (SAP) is considered. This layer pays attention to more important frames and neglects frames which do not contribute much (e.g., non-speech frames) by applying a self-attention mechanism to learn weights for mean and standard deviation vectors over each frame [15].
In order to take a more detailed look at the statistics, the learnable dictionary encoding (LDE) layer was introduced. Compared to the TAP and SAP layer, which compute mean and standard deviation of the frame-level representations in form of a single vector, the LDE layer clusters

frame-level representations by using soft clustering and determines mean and standard deviation for each cluster [20]. As the LDE layer was used as encoding layer in this work, the following section discusses it in more detail.

Learnable Dictionary Encoding (LDE)

The learnable dictionary encoder combines the two steps, dictionary learning and vector encoding, from the conventional approach in one single layer [15]. The encoding layer obtains a variable length feature sequence $x_1, x_2, ..., x_L$ extracted from a temporal input data sequence. In the LDE layer the dictionary component centers $\mu_1, \mu_2, ..., \mu_c$ are trained by stochastic gradient descent and each frame of features is then assigned to each dictionary component center by using learnable weights w_{tc}. The difference $x_t - \mu_c$ (residual) and the corresponding assigned weights w_{tc} are aggregated for each component center μ_c to result in a fixed-dimensional encoded vector $e_1, e_2, ..., e_c$, which can be further used for speaker recognition [15] (cf. Figure 3.4).

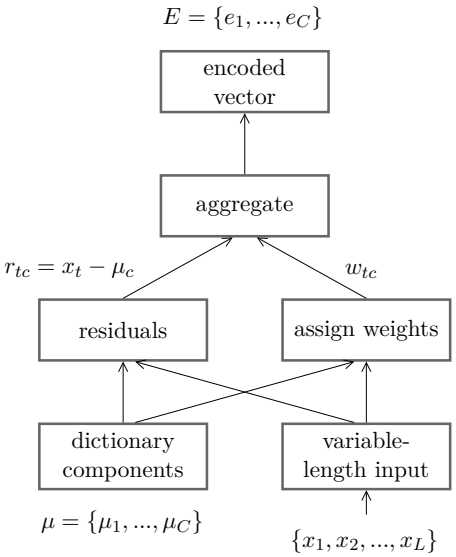


Figure 3.4: LDE pooling layer: The variable-length input $x = \{x_1, x_2, ..., x_L\}$ is related to the learnable dictionary components $\mu = \{\mu_1, ..., \mu_C\}$ by the residual $r_{tc} = x_t - u_c$ and the assigned weight w_{tc}. The encoding layer applies an aggregation operation in order to obtain the desired fixed-dimensional encoder output $E = \{e_1, ..., e_C\}$ [15].

3.2 Neural Speaker Embedding Extraction based on PyTorch and Kaldi

Based on the LDE (cf. Section 3.1.2) used in state-of-the-art neural speaker recognition systems, the utterance-level speaker embedding vectors were extracted from the audio files in the provided dataset. Therefore, a 12-stage speaker recognition pipeline was used, including feature extraction, neural network training, network decoding and a post processing step ([22], [20]). Figure 3.5 shows a simplified representation of the pipeline which is used for speaker identification as well as for speaker verification.

3.2.1 Pipeline Speaker Recognition System

Data Preparation

The audio files from the 19 speakers provided in the dataset (cf. Section 2.2.2) were first divided into portions of training and test sets (90% training data and 10% test data), with the

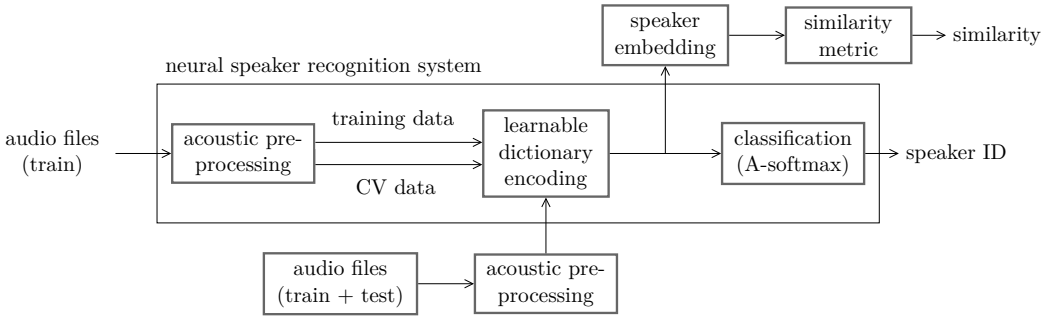


Figure 3.5: Utterance-level speaker embedding extraction from an end-to-end neural network speaker recognition system. The resulting speaker embedding is used for speaker verification (i.e., the similarity between a speech sample and a speaker ID) and the calculated speaker ID for speaker identification.

training data further subdivided into training set and cross-validation set (CV). For the speaker recognition task, not only the audio files were needed but also additional files, that contain the path to the corresponding audio files (cf. Infobox 3.1) or the speaker ID of a particular utterance (cf. Infobox 3.2). The files had to be created in advance, whereby the format of the required files is described on the Kaldi website [23].

utterance ID	**audio file**
bsc-berlin-035	data_base/test/bsc_berlin_035.wav
bsc-berlin-038	data_base/test/bsc_berlin_038.wav
...	...

Infobox 3.1: File `wav.scp` *contains the utterance ID and the path to the corresponding audio file.*

utterance ID	**speaker ID**
bsc-berlin-035	bsc
bsc-berlin-038	bsc
...	...

Infobox 3.2: File `utt2spk` *contains the utterance ID and the corresponding speaker ID.*

The provided script also offers the possibility of data augmentation for training. For this purpose, the MUSAN corpus (consisting of music, speech and noise) is used to augment the original corpus. However, this method of data augmentation was not used in this work, because the provided dataset already achieved sufficient results.

Acoustic Pre-Processing

In advance of the training process, an acoustic pre-processing step was carried out (cf. Figure 3.6).

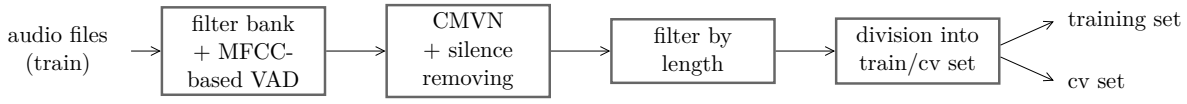


Figure 3.6: Acoustic pre-processing prior to training the speaker recognition system.

First, a filter bank analysis was performed, in which the speech signal was divided into filter bands and the mel frequency cepstral coefficients (MFCCs) were calculated for each filter band. The MFCCs obtained were used for energy-based speech activity detection.
For the training process, so-called x-vector features were created. The raw feature vectors were normalised by using cepstral-mean-variance-normalisation (CMVN) and additionally non-speech frames were removed.
Afterwards utterances were filtered by frame length, i.e., utterances with a frame length below a certain limit value were excluded from the training process. In order to get an appropriate limitation of the frame length, different values were tested (cf. Table 3.1 and Section 3.2.3).

Table 3.1: Settings for the acoustic pre-processing regarding the minimum frame length of the utterances used for the training process.

parameters	values	description
minimum frame length	200/300/500/800	utterances with less frames are excluded from training process
number of speakers	16/17/18/19	speakers used for training (depending on minimal frame length)

The remaining utterances were then divided into training (90%) and cross-validation (CV) (10%) sets. In order to avoid dominance of individual speakers during training, it was evaluated how many utterances belong to each speaker. For speakers with a small number of utterances, the utterances were repeated more often in the training process compared to speakers with a large number of utterances, resulting in a balanced training process.

Neural Network Training

During the pooling process in the LDE layer, a variable-length input sequence was transformed into a fixed-dimensional representation according to the predefined number of dictionary components. Afterwards a linear transformation was carried out that takes the fixed-dimensional output of the LDE layer as input and outputs a vector with the dimension of the hidden layer. In this stage, the encoding network is responsible for frame-level feature extraction. The subsequent learnable dictionary encoder acts as pooling layer which represents the frame-level features in form of an utterance-level vector, the so-called utterance-level speaker embedding vector. Based on the calculated utterance embedding vectors, a softmax function is used as classifier which outputs the estimated speaker ID (cf. Figure 3.5).
For the training process, several settings had to be adjusted which have a significant influence on the speaker recognition results. The minimum frame length required for the training utterances was specified to exclude utterances with a frame length that is too short from the training process. Depending on the minimum required frame length, the length of utterance chunks used for training was set by the chunk size, which is defined by choosing a minimum and maximum chunk size. During training, for each batch a chunk size between a given minimum and maximum chunk size was randomly selected. A random start time was then generated for each utterance, starting from which samples of the length of the previously randomly determined chunk size were used for the training process.
The dimension of the input layer (input-dim) was defined as well as the dimension of the hidden layers (hidden-dim) and the number of LDE dictionary components (C). For the dimension of the input layer and the number of LDE dictionary components, the values were chosen according to [20]. The dimension of the hidden layer was determined experimentally because the choice of the hidden layer dimension is important as it defines the length of the extracted speaker embedding

vector. Again, the values in [20] were taken as reference. The most significant parameters for training the speaker verification system are shown in Table 3.2. For some parameters, the table contains multiple possibilities, because the values were determined experimentally (cf. Section 3.2.3).

Table 3.2: Settings for the training process of the speaker recognition system.

parameters	values	description
chunk size interval (in frames)	[150,200]/[150,300]/ [200,200]/[200,300]/ [200,500]/[300,300]/ [500,500]/[800,800]	size of chunks used during training (depending on minimum frame length)
model	ResNet34	convol. NN used for classification
input-dim	30	input layer size (dimension of x-vector features)
hidden-dim	32/128/512	hidden layer size (speaker embedding dimension)
C	32	number of dictionary clusters for pooling layer (LDE)
pooling	mean	pooling method (mean or mean and standard deviation) for pooling layer (LDE)
network-type	LDE	learnable dictionary layer (pooling layer method)
distance-type	sqrt	sqrt or norm
A-softmax	true	loss function
m	2	integer to control size of angular A-softmax

Network Decoding

After training the network a decoding step followed to investigate the performance of the trained speaker recognition system using the previously defined test set. For testing the speaker verification task, a trial file is required, which contains pairs of utterances from the test set to be compared. If utterances of the same speaker are compared, the target is 1, otherwise 0. A total of 528 utterance pairs were compared in the trial file, with target and non-target occurring alternately (cf. Infobox 3.3).
For the decoding of the neural network, the same pre-processing as for training the neural network was applied (cf. Figure 3.6). The pre-processed frames were afterwards used for the input of the previously trained neural network (cf. Figure 3.5). For the neural network, the same settings as before were used (cf. Table 3.2). After the pooling layer, the fixed-dimensional speaker embedding vectors for train and test set were extracted.

utterance ID 1	utterance ID 2	target/non-target
bsc-berlin-031	bsc-berlin-097	1
bsc-berlin-035	csc-marburg-015	0
...	...	...

Infobox 3.3: File `trial` *contains the utterance IDs of the utterances that are compared in the speaker verification task and the target/non-target information.*

Post Processing

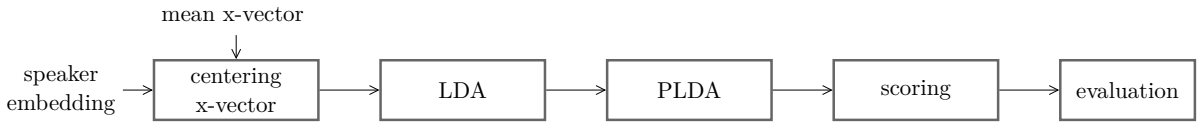


Figure 3.7: Post processing of the calculated utterance-level speaker embedding vectors in order to obtain pairwise scores based on which the speaker verification is carried out. A subsequent evaluation shows the performance of the utterance embedding vectors.

The extracted speaker embedding vectors were first centered by computing the mean over all embeddings and afterwards a linear discriminant analysis (LDA) was carried out to reduce the dimensionality of the embeddings. A subsequent trained probabilistic linear discriminant analysis (PLDA) model was applied to calculate the pairwise score between utterances according to the trial file (cf. Infobox 3.4). The calculated scores gave the log likelihood which was used for decision same or different speaker depending on a defined threshold and further evaluated to show the extracted embedding quality.

utterance ID 1	utterance ID 2	score
bsc-berlin-031	bsc-berlin-097	2.5431
bsc-berlin-035	csc-marburg-015	-3.1475
...	...	...

Infobox 3.4: File `score` *contains the utterance IDs of the utterances that are compared in the speaker verification task according to the trial file and the corresponding pairwise scores.*

3.2.2 Evaluation Metrics

The speaker verification process (cf. Section 3.2.1) was carried out using different combinations of parameters (cf. Table 3.2) in order to show their influence on the performance of the speaker verification system. For performance evaluation, two important performance metrics were considered: the equivalent error rate (EER) and the minimum detection cost function value (minDCF). Additionally, performance was evaluated graphically by reducing the dimensionality of the speaker embeddings using the t-distributed stochastic neighbour embedding (t-SNE) and plotting the obtained low-dimensional embedding vectors in a 2-dimensional map.

Equal Error Rate (EER)

In the traditional approach of the evaluation of speaker recognition systems, a trial file was predefined in which a value of 0 or 1 was assigned to each pair to be evaluated (non-target

or target trial). According to the trial file, scores between the test set utterance combinations (cf. Infobox 3.4) were calculated based on the previously trained speaker recognition system. A threshold was defined for the scores to distinguish between non-target and target or in other words between same and different speakers. For defining a threshold, it is essential to minimise the number of possible errors, whereby a distinction can be made here between false positives and false negatives. False positives, also called false alarms, classify a non-target trial as target trial and false negatives, also known as misses, classify a target trial as a non-target trial. To keep both errors as low as possible, a threshold value is chosen for which the probabilities for false positives P_{FA} and false negatives P_{miss} are equal, as both change in opposite direction when changing the threshold. The equal error rate (EER) in % is the value of P_{FA} and P_{miss} at this threshold. The lower the EER the higher the accuracy of the speaker recognition system [24].

Minimum Detection Cost Function (minDCF)

In the evaluation part of the pipeline, the minimum detection cost function (minDCF) is calculated in addition to the EER, which is a common error metric used in speaker recognition. Compared to equal error rate, which assigns equal weight to false negatives (P_{miss}) and false positives (P_{FA}), the minDCF weights P_{miss} and P_{FA} differently, depending on the application. As in some applications, e.g., achieving a low false positive rate is more important than achieving a low false negative rate, the minDCF has become an essential metric in the evaluation process [24]. Equation 3.3 describes the calculation of the detection cost function C_{det} [24]:

$$C_{det}(P_{miss}, P_{FA}) = C_{miss}P_{miss}P_{tar} + C_{FA}P_{FA}(1 - P_{tar}). \tag{3.3}$$

In this evaluation method, a weighting of the two normalised error rates P_{miss} and P_{FA} with the prior probability of targets is proposed as well as weighting the costs of the two error types. This results in application dependent parameters C_{miss}, C_{FA} and P_{tar}, which influence the value of the detection cost function C_{det} according to Equation 3.3. C_{miss} and C_{FA} describe the estimated costs of the two error types (false negatives and false positives), whereas P_{tar} is the prior probability that a target event occurs in the application. A low value for the minDCF provides that both EER is low and the threshold has been set well [24].

t-Distributed Stochastic Neighbour Embedding (t-SNE)

The t-SNE is a nonlinear dimensionality reduction method which is widely used in visualising high dimensional data in a low dimensional map (2- or 3-dimensional). Therefore, similar objects are modeled by nearby points.
In this algorithm, high-dimensional distances (i.e., Euclidean distances) are transformed into conditional probabilities to indicate similarities between two objects. This similarity between data points is expressed by conditional probabilities, where neighbours are picked in proportion to their probability density function [25]. In order to minimise a data representation that reduces the mismatch between the similarities (probabilities) for two data points in the high- and low-dimensional space, the Kullback-Leibler divergence has to be minimised by means of a gradient descent method. Since the t-SNE is based on probabilities, it is not a deterministic method, hence the result will vary to some extent even with the same dataset [25].

3.2.3 Experiments

Based on the presented evaluation methods, the optimal combination of parameters were to be determined in order to extract representative speaker embeddings. Especially three parameters were tuned in this investigation:

- the minimum required frame length of training utterances,
- the chunk size interval of the training utterances,
- and the dimension of the extracted speaker embedding vectors.

For the minDCF evaluation, the parameters shown in Table 3.3 were used. As the cost of missed detection C_{miss} and false alarm C_{FA} was equal, both were set to 1. The prior probability of the target speaker in a trial P_{tar} was calculated according to the number of target speakers (same speakers) divided by the number of all possible speaker pairs.

Table 3.3: Settings for the minDCF evaluation in terms of the estimated costs of misses and false alarms (C_{miss} and C_{FA}) and the prior target probability P_{tar}.

parameter	value	description
C_{miss}	1	cost of a missed detection
C_{FA}	1	cost of a false alarm
P_{tar}	0.1	prior probability of the target speaker in a trial

Table 3.4 gives an overview of tested parameter combinations and their influence on the EER and the minDCF. In the following sections, the experiments and results are discussed in more detail.

Table 3.4: Influence of the minimum required frame length (and the resulting number of training utterances), the chunk size interval of the training utterances and the speaker embedding dimension on the speaker recognition process.

number of utterances (min. frame length)	chunk size interval	embedding dimension	LDA dimension	EER in %	minDCF
181 (800)	[300, 300]	128	100	28.81	0.86
181 (800)	[800, 800]	128	100	27.60	0.85
709 (500)	[200, 200]	128	100	19.13	0.66
709 (500)	[200, 200]	32	32	22.76	0.78
709 (500)	[200, 200]	512	200	19.37	0.72
709 (500)	[500, 500]	128	100	22.28	0.87
709 (500)	[200, 500]	128	100	19.61	0.75
709 (500)	[200, 500]	512	200	22.03	0.81
1776 (300)	[200, 300]	128	100	12.59	0.49
1776 (300)	[200, 300]	512	200	12.59	0.51
1776 (300)	[150, 200]	128	100	12.83	0.47
1776 (300)	[150, 300]	128	100	10.90	0.51
4768 (200)	[150, 200]	128	100	6.05	0.27

Minimum required Frame Length

The choice of an appropriate minimum required frame length for the training utterances is essential. On the one hand, previous work has shown that for successful speaker verification, longer training chunks improve performance which supports the use of longer utterances for the training process [26]. On the other hand, a high minimum required frame length would exclude quite a few utterances and even speakers from the training set, since the provided dataset consists of many short utterances.
In the original pipeline [20] only utterances with a frame length of more than 800 were used for training which corresponds to a length of 8s. Utterances with less than 8s were therefore excluded from the training process. When considering the distribution of frame lengths of the utterances in the dataset (cf. Figure 3.8), this limitation of 800 frames per utterance led to a drastic reduction of training utterances, as the majority of utterances have a length of less than 400 frames.

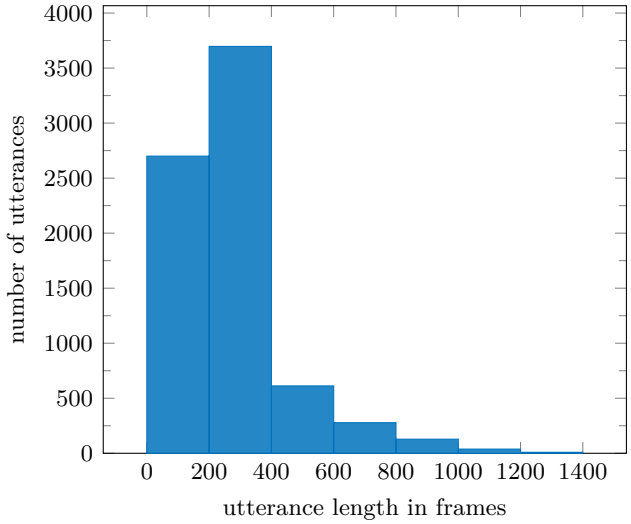


Figure 3.8: Distribution of frame lengths of the train utterances from the WASS corpus.

By choosing a limit value of 800 frames, only 181 utterances out of 7468 total training utterances were kept. When decreasing the minimum frame length per utterance to 500, about 700 utterances maintained for the training process. Furthermore a required frame length of only 300 frames increased the number of training utterances to 1776 and a frame length of 200 frames resulted in a total of 4768 utterances for training (cf. Figure 3.9).

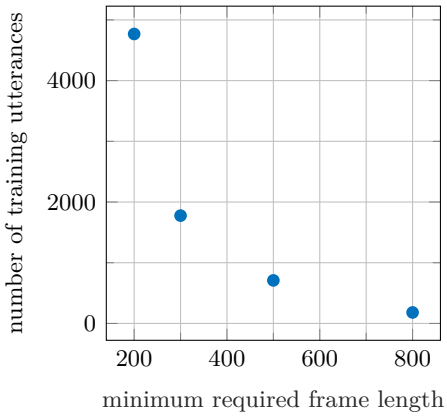


Figure 3.9: Number of training utterances depending on the minimum required frame length.

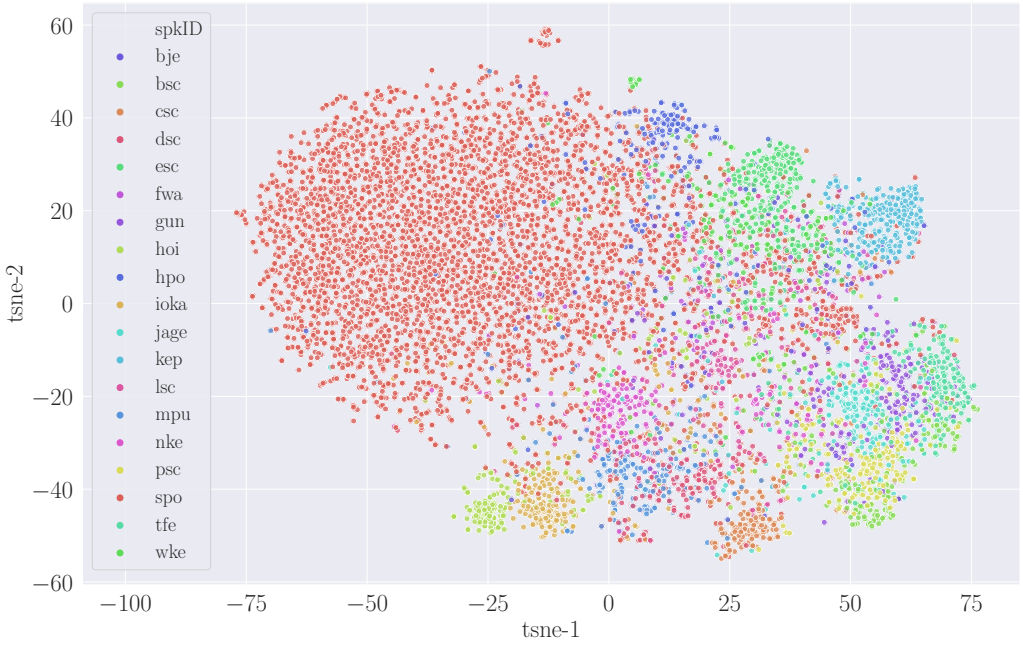
Table 3.4 shows a clear trend: the smaller the minimum required frame length and thus the higher the number of training utterances, the smaller the EER and the minDCF, indicating that the speaker recognition performance increases. With similar intervals of chunk sizes and embedding dimension, the result seemed to strongly depend on the number of utterances.
To investigate this assumption, a training dataset containing utterances with a frame size above 300 frames was randomly reduced from 1776 to 776 utterances. The result shown in Table 3.5 confirms the dependence of the speaker recognition performance on the number of training utterances. The minDCF as well as the EER were reduced drastically by increasing the number of training utterances.

Table 3.5: Influence of the reduction of the training utterances on the EER and the minDCF while keeping the other parameters constant.

number of utterances (min. frame length)	chunk size interval	embedding dimension	LDA dimension	EER in %	minDCF
1776 (300)	[150, 300]	128	100	10.90	0.51
776 (300)	[150, 300]	128	100	19.13	0.75

Additionally, the graphical comparison shows the benefit of using more training data (cf. Figure 3.10), which results in clearer distinction between individual speakers.

(a) t-SNE for 19 speakers from the WASS corpus with the following parameters: minimum frame length: 300; chunk size interval: [150, 300]; embedding dimension: 128; LDA dimension: 100.



(b) t-SNE for 19 speakers from the WASS corpus with the following parameters: minimum frame length: 300; chunk size interval: [150, 300]; embedding dimension: 128; LDA dimension: 100.

Figure 3.10: Influence of the number of training utterances on the speaker recognition performance: By reducing the number of utterances with at least a length of 300 frames from 1776 training utterances (top) to 776 training utterances (bottom), performance of the embedding-based speaker separation degrades.

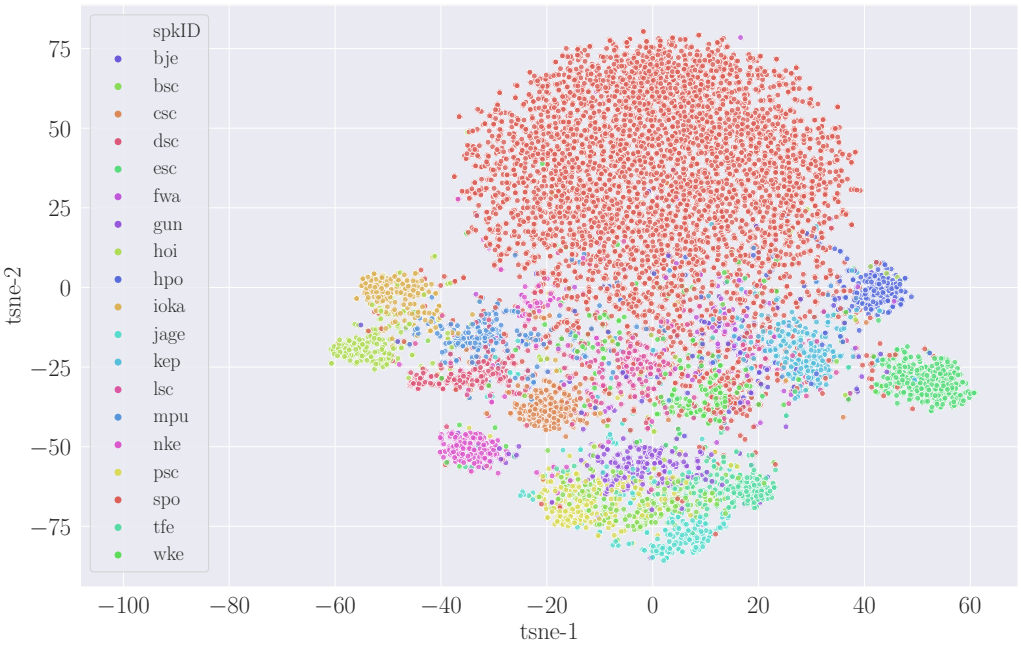
Chunk Size

According to [26], the chunk size of the training samples has a strong influence on the speaker recognition performance. Chen et al. [26] investigated the difference when using chunks of length $2s$ to $4s$ and chunks of length $5s$ to $8s$ for the training of the speaker verification system. They showed that by choosing longer chunks for training, performance of speaker recognition is improved. Therefore, a similar value for the chunk size was initially chosen in this work. However, the maximum chunk size depends on the minimum required frame length of the training utterances, because a maximum chunk size of 800 frames requires a minimum frame length of 800 frames per utterance. As mentioned previously, this limit of 800 frames per utterance excluded most of the data for the training process (cf. Figure 3.8) which may result in performance degradation. This leads to a trade-off between the number of utterances and the chunk size interval. Therefore, it was investigated experimentally, which combinations of different parameters (minimum frame length, chunk size) lead to "best" results in terms of the evaluation (cf. Section 3.2.2).
The experiments (cf. Table 3.4) do not confirm the observation from [26]. For larger chunk sizes, a tendency towards higher EERs and minDCFs was observed (cf. Table 3.6), while keeping the number of training utterances and the embedding dimension constant. This tendency is also shown in Figure 3.11, which shows improved separation when using shorter utterance chunks for the training process. Apart from that, a better distinction between male and female speakers was achieved when shorter chunks of the training utterances were used.

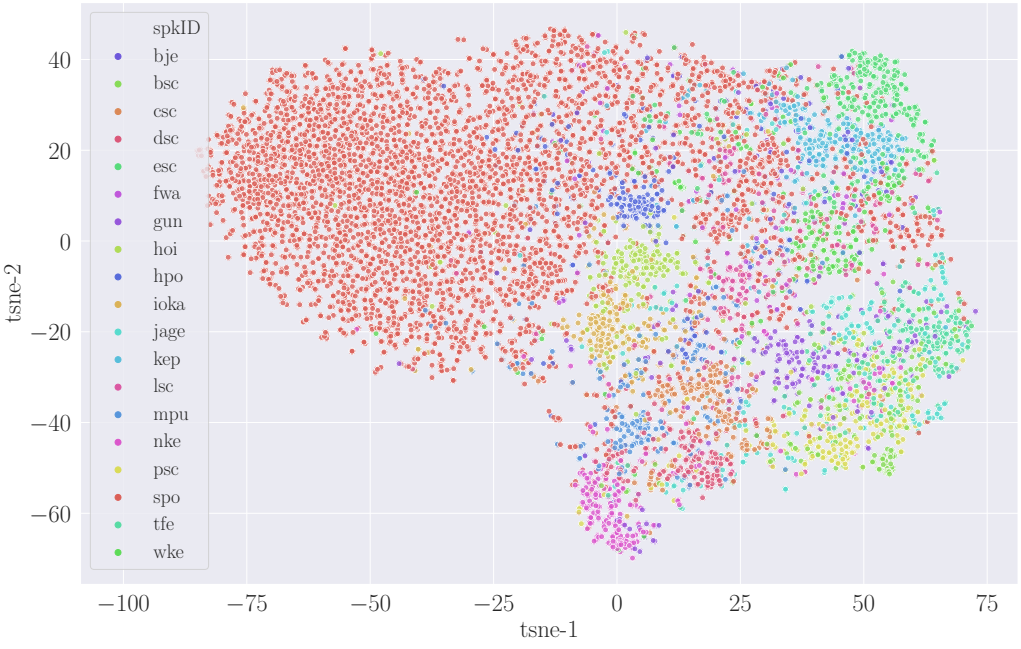
Table 3.6: Comparison of different chunk sizes and their influence on the EER and the minDCF when keeping the minimum required frame length and speaker embedding dimension constant.

number of utterances (min. frame length)	chunk size interval	embedding dimension	LDA dimension	EER in %	minDCF
709 (500)	[200, 200]	128	100	19.13	0.66
709 (500)	[500, 500]	128	100	22.28	0.87
709 (500)	[200, 500]	128	100	19.61	0.74
709 (500)	[100, 200]	128	100	16.95	0.69

This outcome would support the choice of low values for the minimum required frame length for the training utterances, resulting in an increasing number of training utterances and further in an improvement in terms of speaker recognition.



(a) t-SNE for 19 speakers from the WASS corpus with the following parameters: minimum frame length: 500; chunk size interval: [200, 200]; embedding dimension: 128; LDA dimension: 100.



(b) t-SNE for 19 speakers from the WASS corpus with the following parameters: minimum frame length: 500; chunk size interval: [500, 500]; embedding dimension: 128; LDA dimension: 100.

Figure 3.11: Influence of the chunk size of the training utterances on the speaker recognition performance: A shorter chunk size (200 frames, top) achieves better speaker separation and distinction between male and female speakers compared to a longer chunk size (500 frames, bottom). The minimum required frame length is set to 500 (709 training utterances) and speaker embedding dimension is 128 in both cases.

Feature Embedding Dimension

The hidden layer dimension of the neural network in the speaker recognition process is essential for the dimension of the feature embeddings as the speaker embeddings are the features extracted from the penultimate layer. Therefore, the dimension of the hidden layer determines that of the speaker embeddings (cf. Section 3.2.1, Neural Network Training). In the past, different speaker embedding dimensions have already been tested. The following table gives an overview of previously used speaker embedding dimensions (cf. Table 3.7), which supports the selection of an appropriate speaker embedding dimension for this work.

Table 3.7: Speaker embedding dimensions already used in the literature.

literature	embedding dimension
Cai et al. [15]	128
Cooper et al. [20]	512/256/200
Chen et al. [26]	400

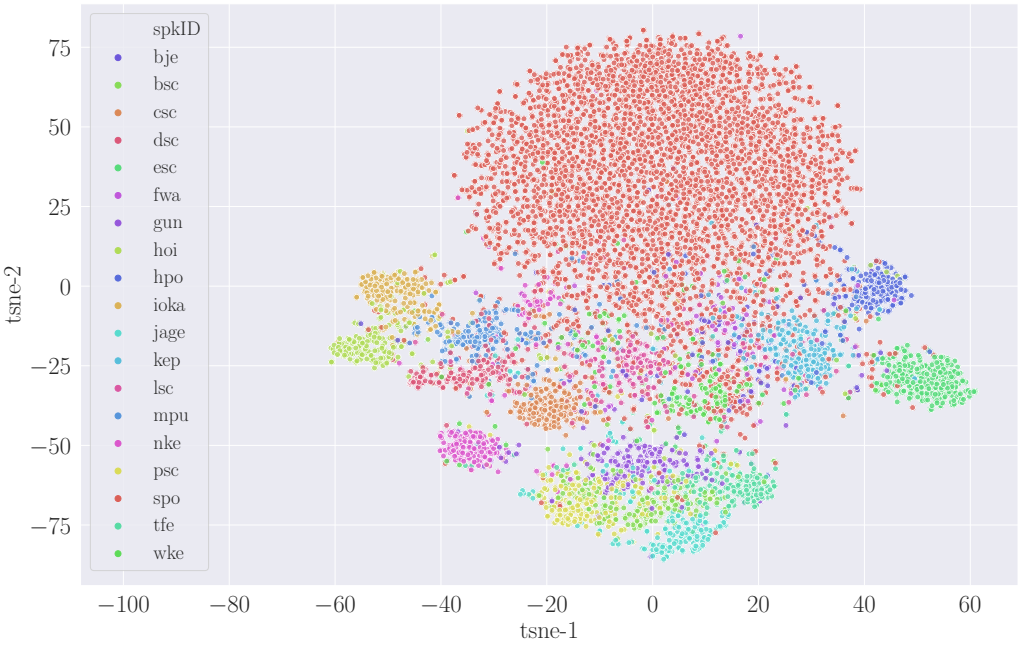
Cooper et al. [20] have already shown that better results in terms of EER and minDCF are achieved for smaller speaker embedding dimensions. In order to obtain an adequate dimension for the speaker embeddings, different dimensions were chosen in this thesis and the influence on speaker recognition performance was investigated by comparing EER, minDCF (cf. Table 3.8) and the t-SNE-based visualisation (cf. Figure 3.12).

Table 3.8: Influence of the speaker embedding dimension on the speaker recognition performance (EER and minDCF) by keeping the minimum frame length and chunk size interval constant and varying only the speaker embedding dimension {32, 128, 512}.

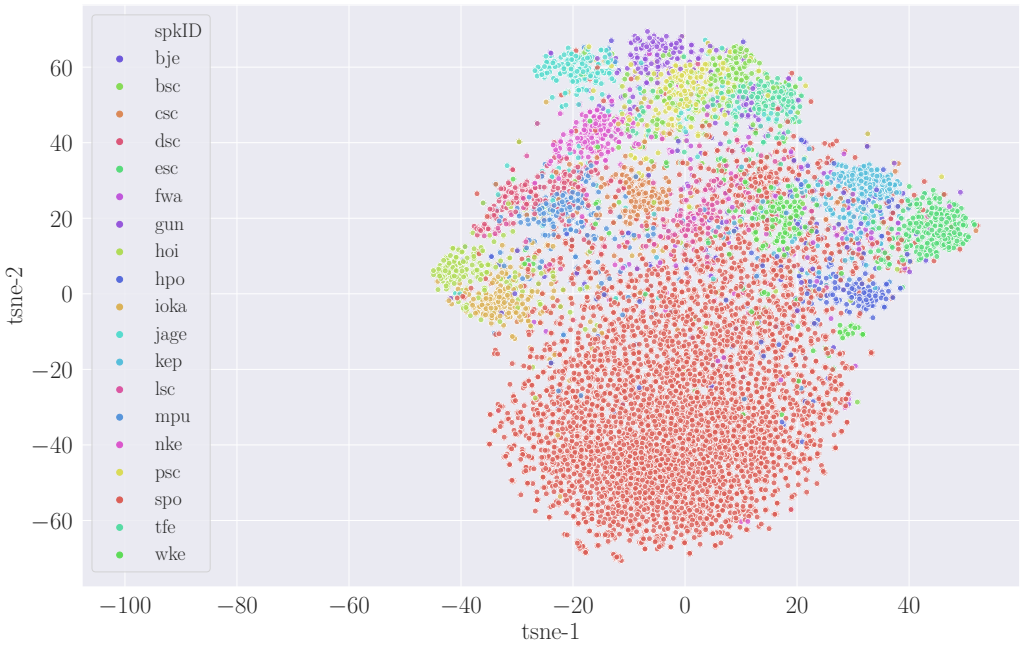
number of utterances (min. frame length)	chunk size interval	embedding dimension	LDA dimension	EER in %	minDCF
709 (500)	[200, 200]	32	32	22.76	0.78
709 (500)	[200, 200]	128	100	19.13	0.65
709 (500)	[200, 200]	512	200	19.37	0.72

The observation in [20] was partially confirmed in this thesis by comparing the results of the speaker recognition system with the same training and test utterances based on three different speaker embedding sizes {32, 128, 512} (cf. Table 3.8 and Figure 3.12). For a speaker embedding dimension of 128, performance increased in terms of EER and minDCF as well as for t-SNE compared to a dimension of 512. However, decreasing the speaker embedding size to only 32, the performance was even worse than with an embedding dimension of 512. Furthermore, a speaker embedding dimension of 128 achieved the best results in separating male and female speakers in terms of the t-SNE.

(a) t-SNE for 19 speakers from the WASS corpus with the following parameters: minimum frame length: 500; chunk size interval: [200, 200]; embedding dimension: 32; LDA dimension: 32.



(b) t-SNE for 19 speakers from the WASS corpus with the following parameters: minimum frame length: 500; chunk size interval: [200, 200]; embedding dimension: 128; LDA dimension: 100.



(c) t-SNE for 19 speakers from the WASS corpus with the following parameters: minimum frame length: 500; chunk size interval: [200, 200]; embedding dimension: 512; LDA dimension: 200.

Figure 3.12: Influence of the speaker embedding dimension on the speaker recognition performance: Increasing the speaker embedding dimension from 32 to 128 improves performance whereas an increase from 128 to 512 results in degradation of performance for same minimum frame length (500) and chunk size interval ([200, 200]).

Discussion

Taking into account the above observations, the quality of speaker embedding strongly depends on a sufficiently low minimum frame length of utterances to ensure a high number of training utterances. This parameter had to be chosen in relation to the chunk size interval of the training utterances, because the maximum chunk size is predefined by the minimum frame length. According to the distribution of the frame lengths of the utterances used (cf. Figure 3.8), a minimum frame length of 200 would be adequate to prevent excluding a high number of utterances from the training process. Furthermore this choice allowed a chunk size interval ([150, 200]) that does not differ too much from previously tested intervals in the literature [26], [20]. A speaker embedding dimension of 128 showed better results in terms of EER, minDCF and t-SNE compared to the other speaker embedding dimensions tested in this work.
Table 3.4 confirms these observations, because the chosen parameter combination (cf. Table 3.9) showed best results in terms of the evaluation parameters (EER, minDCF). Apart from that the visual evaluation (t-SNE) supports these parameter combination (cf. Figure 3.13). Figure 3.13 shows a clear separation of the individual speakers. However, the grouping in male and female speakers is worse compared to other parameter combinations (cf. Figure 3.12). This is probably due to the small chunk size, which can no longer represent this property sufficiently well.

Table 3.9: Experimentally derived choice of parameters for the speaker embedding extraction.

number of utterances (min. frame length)	chunk size interval	embedding dimension	LDA dimension
4768 (200)	[150, 200]	128	100

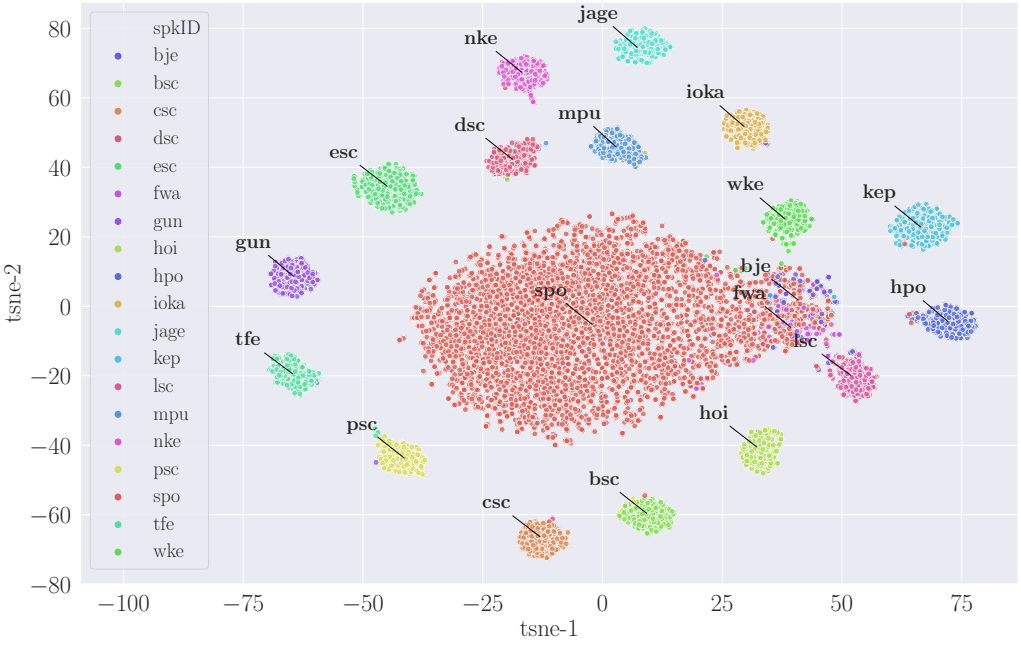


Figure 3.13: t-SNE for 19 speakers from the WASS corpus with the following parameters: minimum frame length: 200; chunk size interval: [150, 200]; embedding dimension: 128; LDA dimension: 100.

3.2.4 Speaker Embedding

By running the pipeline with the parameters shown in Table 3.9, the corresponding utterance-level speaker embeddings were extracted and stored in a file as vectors. So far, utterance-level representations were created. Therefore, each utterance had its own embedding vector. In order to represent the speaker characteristics rather than the utterance information, the calculated utterance embeddings were averaged per speaker [1]. Each resulting fixed-dimensional speaker embedding vector was then transformed into a vector of type `float64` and stored in a separate binary file. According to the conventional speaker adaptation (cf. Section 2.2.3), two speakers were not considered in the speaker embedding-based speaker adaptation, resulting in 17 speaker embedding vectors.

3.3 Interpolation of Speaker Embeddings

The extracted speaker embedding vectors describing 17 speakers from the WASS corpus served as basis to generate "new" speakers. By applying linear interpolation between two base speaker embedding vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, an intermediate speaker $\mathbf{v}_{12}$ was obtained:

$$\mathbf{v}_{12} = \alpha \cdot \mathbf{v}_1 + (1 - \alpha) \cdot \mathbf{v}_2. \tag{3.4}$$

An interpolation factor $\alpha \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$ was considered to scale the amount of the two individual speakers. For $\alpha = 0.0$ and $\alpha = 1.0$, respectively, the original speakers were synthesised. All possible combinations between the 17 base speakers were calculated, leading to a total number of 816 speaker embeddings. Considering for example speakers `bjedsc00` and `bscdsc00`, which have 0 parts ($\alpha = 0.0$) from speaker `bje` and `bsc` respectively, but 1 part ($1 - \alpha = 1.0$) from speaker `dsc`, they both describe the same original speaker `dsc`:

$$\begin{aligned} \text{bjedsc00} &= 0.0 \cdot \text{bje} + (1.0 - 0.0) \cdot \text{dsc} = \text{dsc} \\ \text{bscdsc00} &= 0.0 \cdot \text{bsc} + (1.0 - 0.0) \cdot \text{dsc} = \text{dsc} \\ &\dots = \dots \end{aligned}$$

To avoid redundancy, only one representation of the original speakers was retained (cf. Table 3.10), reducing the number of 816 speakers to 561 different speakers. The exclusion of the synthesised base speakers ($\alpha = 0.0$ and $\alpha = 1.0$) led to a total number of 544 "new" speakers.

Table 3.10: Assignment of the original speaker ID to the respective synthesised speaker ID.

original speaker	respective speaker	original speaker	respective speaker
bje	bjebsc10	kep	bjekep00
bsc	bjebsc00	lsc	bjelsc00
csc	bjecsc00	mpu	bjempu00
dsc	bjedsc00	nke	bjenke00
esc	bjeesc00	psc	bjepsc00
fwa	bjefwa00	spo	bjespo00
gun	bjegun00	tfe	bjetfe00
hoi	bjehoi00	wke	bjewke00
hpo	bjehpo00		

The obtained speaker embedding vectors were stored in separate files, for which the file name consisted of the embedding ID including the two combined speaker IDs and the interpolation factor α. In order to generate speech from the newly created speakers, the interpolated speaker embedding vectors were integrated in the speech synthesis process as described in the following chapter.

4

Speaker Embedding-based Speaker Adaptation and Data Augmentation

4.1 Speaker Embedding-based Speaker Adaptation

In Section 2.2.3 the process of speaker adaptation based on an average voice model (AVM) was described. This approach uses an AVM trained on labels and audio files from all available speakers in the dataset. In order to adapt speaker-specific models from the AVM, speaker-specific data was given to the trained AVM, which allowed the training of a speaker-specific model used for the generation of speaker-specific speech (cf. Figure 4.2, left). This method has the advantage that speaker-specific models can also be adapted for speakers with a small amount of data. However, the generation of speaker-specific speech requires speaker-specific models, which not only demand a large amount of storage space, but also take some time to train.
To overcome this problem, a modified AVM approach was used for speech synthesis in this thesis. According to the calculation in Section 2.2.3 an AVM was trained based on the 17 speakers from the dataset. In addition to the linguistic input features, the previously calculated speaker embedding vectors were used as input for training the AVM. Therefore, the vector describing the linguistic features was concatenated with the speaker embedding vector and the resulting vector was given as input to the neural network during training (cf. Figure 4.1).

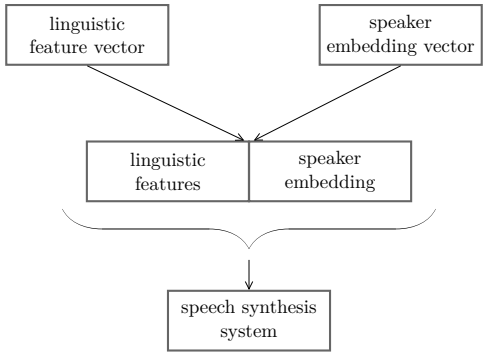


Figure 4.1: Concatenation of the linguistic feature vector and the speaker embedding vector, resulting in the input vector for the speech synthesis system.

Based on the concatenated input vector, the AVM learnt not only the relation between the linguistic input features and the acoustic output, but also the relation between speaker information and the acoustic output. The trained AVM was able to synthesise speaker-specific speech by having access to the labels and the speaker embedding of the desired speaker (cf. Figure 4.2, right).

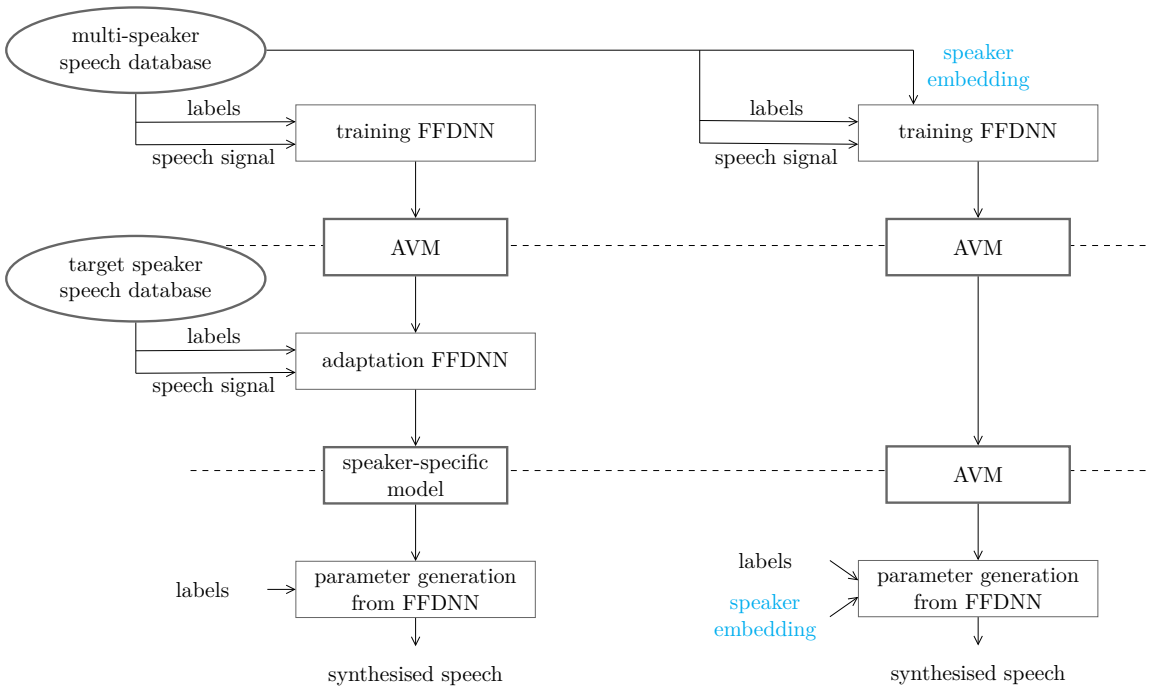


Figure 4.2: Comparison between the conventional speaker adaptation method including the training of the AVM, the adaptation of the speaker-specific model and the subsequent speech synthesis (left) and the speaker embedding-based speaker adaptation that allows speaker-specific speech synthesis by using only a single AVM trained with speaker embedding vectors (right).

By integrating the speaker embedding vectors in the training process, a subsequent speaker adaptation was performed without the need of individual trained models. In contrast to the speaker adaptation in Section 2.2.3, where each speaker required its own trained speaker-specific model, the integration of speaker embedding vectors in the AVM training process allowed a subsequent speaker adaptation based on a single AVM.
For the speaker-specific speech synthesis, the speaker embedding of the desired speaker had to be given to the trained AVM in addition to the corresponding label file. As in the training process, the linguistic information and the embeddings were combined to a single vector and passed on to the previously trained AVM to generate the corresponding acoustic features used for speaker-specific speech synthesis (cf. Figure 4.2, right).

4.1.1 Results

The resulting speaker-specific speech waveforms were compared to the generated speech waveform obtained from the conventional speaker adaptation method (cf. Section 2.2.3) in order to validate the use of speaker embeddings for speaker adaptation. The speakers synthesised with the speaker embedding-based AVM and the speakers synthesised with the conventional speaker adaptation method are very close, although the conventional approach used individual AVMs for the adaptation of each speaker. This result validates the speaker adaptation based on a single AVM using speaker embedding vectors as additional input during training.

4.2 Data Augmentation

4.2.1 GRASS Corpus

The ASR system at the SPSC was developed using the Graz corpus of Read And Spontaneous Speech (GRASS) from the SPSC [27]. The GRASS corpus is a large scale speech database for Austrian German consisting of speech of 38 speakers (male and female) from similar social and different regional backgrounds. Not only read speech and elicited commands were recorded but also a large number of spontaneous conversations, which makes the GRASS corpus appealing for linguistic and phonetic studies. In addition to the audio files, the GRASS corpus contains corresponding orthographic transcriptions, including hesitations, repetitions and disfluencies as well as laughter, breathing, dialect words and overlapping talk.
This work mainly deals with the question, if increasing the acoustic variability of a training corpus leads to an improvement of the ASR performance. In this first step, only read speech was investigated and conversational speech with for example cross talk was not considered. Therefore, only the part of the GRASS corpus that contains recordings of read speech was covered in this thesis.

4.2.2 GRASS Corpus Augmentation

In this work, the GRASS corpus was to be augmented acoustically. The question arose, as to which utterances should be augmented from the corpus. For the recent development of the ASR system, the GRASS corpus has already been divided into train, development and test set, as is shown in Table 4.1. Therefore, it seemed quite reasonable to augment only the training set acoustically and to keep development and test set as they are.

Table 4.1: Separation of the utterances from the read speech part of the GRASS corpus into training, evaluation and test set. The values in brackets show the number of different utterances in each set.

overall	train	evaluation	test
4323	3820	249	254
(802)	(548)	(127)	(127)

The speaker embedding interpolation generated a total of 561 voices. Synthesising the training set of the GRASS corpus, which contains 548 individual utterances, with all generated speakers would result in about 300.000 newly generated utterances. The high amount of data may lead to some challenges in terms of memory limitations and training duration. Therefore, it had to be clarified in advance whether sufficient storage space is available. In addition, an amount of data should be selected with which the training of the ASR system can be carried out in a realistic time. Apart from that it was interesting to investigate, if the performance of the ASR system reaches saturation at some point where ingesting more amount of training data would be unnecessary.

Text to Label Conversion

To perform speech synthesis by means of Merlin, the system needs corresponding label files to generate speech out of a given text (cf. Section 2.2.2). Therefore, the frontend Festival [28] was employed to generate label files from the text of the GRASS corpus. Festival is trained by having access to text and the corresponding audio files in order to align the audio with the text. But it is also possible to use a pre-trained model to get labels by only providing text utterances. A pre-trained Austrian German (AT) voice model can be found at ([11], [29]), which is a voice model based on Hidden Markov Models (HTS). This voice model was used in this work

to obtain label files from the training utterances of the GRASS corpus (cf. Section 4.2.1), which were either mono labels or full-context labels (cf. Infobox 4.1). The full-context labels represent contextual information of phonemes, which are formatted in HTS style [30] (cf. Section 2.2.2).

```
      0     1000000     sil
1000000     2000000     h
2000000     3000000     a
3000000     4000000     l
4000000     5000000     oh
5000000     6000000     v
6000000     7000000     E
7000000     8000000     l
8000000     9000000     t
9000000    10000000     sil
```

```
      0     1000000     x^x-sil+h=a@x_x/A:0_0_0/B:x-x-x@x-x&x-x#x-x$x-x!x-x;x-x|x/C:0+-1+2/D:0_0/
                        E:x+x@x+x&x+x#x+x/F:content_2/G:0_0/H:x=x@1=1|0/I:3=2/J:3+2-1
1000000     2000000     x^sil-h+a=l@1_2/A:0_0_0/B:0-1-2@1-2&1-3#1-3$1-1!0-1;0-0|0/C:1+0+2/D:0_0/
                        E:content+2@1+2&1+1#0+1/F:content_1/G:0_0/H:3=2@1=1|NONE/I:0=0/J:3+2-1
2000000     3000000     sil^h-a+l=oh@2_1/A:0_0_0/B:0-1-2@1-2&1-3#1-3$1-1!0-1;0-0|0/C:1+0+2/D:0_0/
                        E:content+2@1+2&1+1#0+1/F:content_1/G:0_0/H:3=2@1=1|NONE/I:0=0/J:3+2-1
3000000     4000000     h^a-l+oh=v@1_2/A:0_-1_2/B:1-0-2@2-1&2-2#1-2$1-1!0-1;0-0|0/C:1+0+4/D:0_0/
                        E:content+2@1+2&1+1#0+1/F:content_1/G:0_0/H:3=2@1=1|NONE/I:0=0/J:3+2-1
4000000     5000000     a^l-oh+v=E@2_1/A:0_-1_2/B:1-0-2@2-1&2-2#1-2$1-1!0-1;0-0|0/C:1+0+4/D:0_0/
                        E:content+2@1+2&1+1#0+1/F:content_1/G:0_0/H:3=2@1=1|NONE/I:0=0/J:3+2-1
5000000     6000000     l^oh-v+E=l@1_4/A:1_0_2/B:1-0-4@1-1&3-1#2-1$1-1!1-0;0-0|0/C:0+0+0/D:content_2/
                        E:content+1@2+1&2+0#1+0/F:0_0/G:0_0/H:3=2@1=1|NONE/I:0=0/J:3+2-1
6000000     7000000     oh^v-E+l=t@2_3/A:1_0_2/B:1-0-4@1-1&3-1#2-1$1-1!1-0;0-0|0/C:0+0+0/D:content_2/
                        E:content+1@2+1&2+0#1+0/F:0_0/G:0_0/H:3=2@1=1|NONE/I:0=0/J:3+2-1
7000000     8000000     v^E-l+t=sil@3_2/A:1_0_2/B:1-0-4@1-1&3-1#2-1$1-1!1-0;0-0|0/C:0+0+0/D:content_2/
                        E:content+1@2+1&2+0#1+0/F:0_0/G:0_0/H:3=2@1=1|NONE/I:0=0/J:3+2-1
8000000     9000000     E^l-t+sil=x@4_1/A:1_0_2/B:1-0-4@1-1&3-1#2-1$1-1!1-0;0-0|0/C:0+0+0/D:content_2/
                        E:content+1@2+1&2+0#1+0/F:0_0/G:0_0/H:3=2@1=1|NONE/I:0=0/J:3+2-1
9000000    10000000     l^t-sil+x=x@x_x/A:1_0_4/B:x-x-x@x-x&x-x#x-x$x-x!x-x;x-x|x/C:0+0+0/D:content_1/
                        E:x+x@x+x&x+x#x+x/F:0_0/G:3_2/H:x=x@1=1|0/I:0=0/J:3+2-1
```

Infobox 4.1: Conversion of the input utterance "Hallo Welt" to the corresponding mono labels (top) and full-context labels (bottom) by using the frontend Festival and the pre-trained Austrian German (AT) voice model.

Speech Synthesis

The previously trained AVM obtained by integrating the 17 original speaker embedding vectors as additional input during training (cf. Section 4.1 and Figure 4.2, right) was used as trained model in order to generate speech from the interpolated speaker embeddings. By giving the generated labels and the interpolated speaker embedding vectors (cf. Section 3.3) as input to the trained AVM, the acoustic features of the "new" speakers were calculated and further passed to a vocoder to synthesise the desired speaker-specific speech waveform (cf. Figure 4.3).

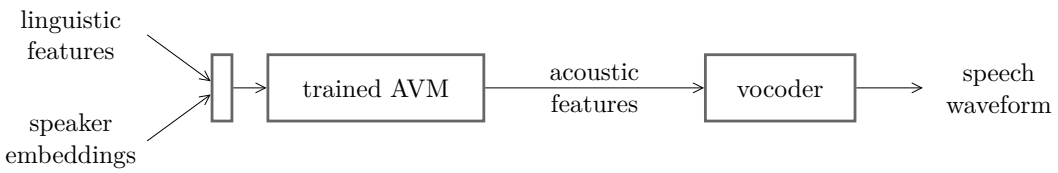


Figure 4.3: Generation of speaker-specific speech by passing the interpolated speaker embedding vectors in addition to the linguistic features to the previously trained AVM. A vocoder takes the obtained acoustic features as input and generates the new voice.

4.2.3 Evaluation

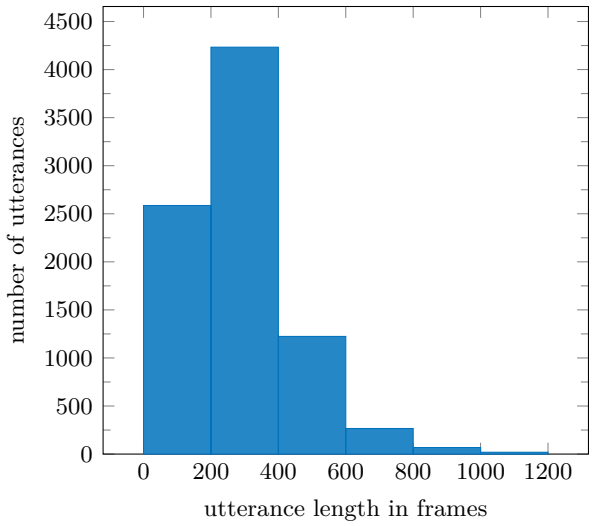
An efficient method to investigate the quality of the interpolated speaker embedding vectors and the resulting synthesised utterances is the evaluation based on the speaker recognition system discussed in Section 3.2. Therefore, the steps of the pipeline in Section 3.2.1 were performed. These steps include the extraction of the speaker embedding vectors from the synthesised speech waveforms and the subsequent evaluation by means of EER, minDCF and t-SNE (cf. Section 3.2.2).
In order to validate the quality of the speaker embedding vectors, four experiments were performed using the synthesised utterances based on the interpolated speaker embeddings. The first experiment analysed the speaker embeddings which represent the 17 speakers from the WASS corpus (base speakers), i.e., $\alpha = 0.0$ and $\alpha = 1.0$ respectively. In three subsequent experiments the other interpolation factors were investigated, for which three base speaker combinations and their interpolation were examined to show the transition between two base speakers and their separability (cf. Table 4.2).

Table 4.2: Base speaker combinations (bscwke, keplsc and fwaspo) and their interpolations used to evaluate the interpolated speaker embeddings.

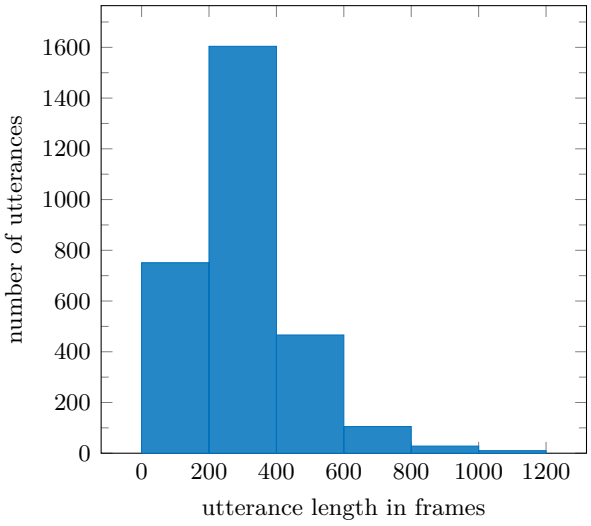
base speakers	interpolated speakers					
bsc, wke	bscwke00	bscwke02	bscwke04	bscwke06	bscwke08	bscwke10
kep, lsc	keplsc00	keplsc02	keplsc04	keplsc06	keplsc08	keplsc10
fwa, spo	fwaspo00	fwaspo02	fwaspo04	fwaspo06	fwaspo08	fwaspo10

Pre-Processing

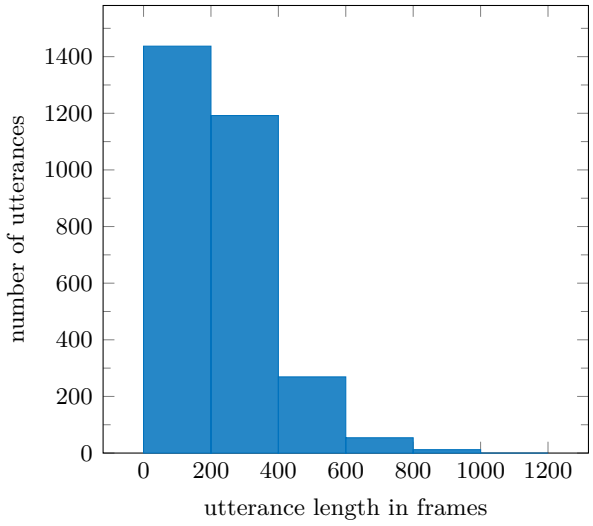
After the generation of the files containing information from the audio files (e.g., `wav.scp`, `utt2spk`), the acoustic pre-processing was carried out. In this step, the MFCCs and the VAD were calculated, the features were normalised and silence was removed. The filtering by length was performed to exclude utterances with less frames than a predefined limit value. For this purpose, the frame lengths of the audio files for the 17 base speakers and for three speaker combinations (cf. Table 4.2) were examined and graphically represented in terms of histograms (cf. Figures 4.4a - 4.4d). The analysis showed that the distribution of frame lengths per utterance strongly depends on the speaker combination, because the duration for each phone varies for each speaker. As a result the minimum frame length had to be defined for each speaker individually, which is shown in Table 4.3. An appropriate choice ensured that enough utterances remained for the training process and yet that utterances that were too short were excluded.



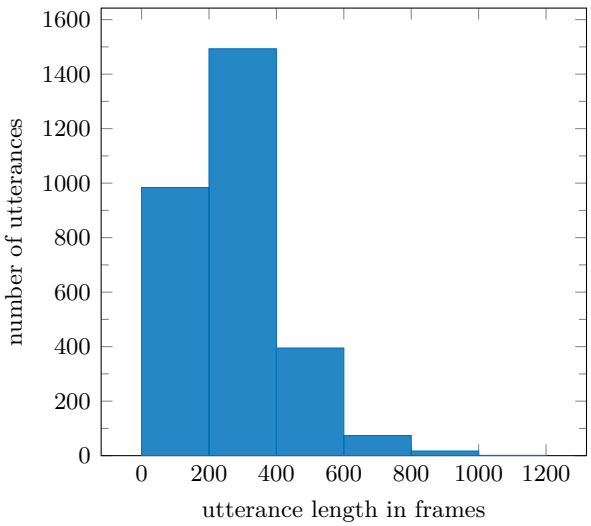
(a) base speakers



(b) speaker combinations for bscwke



(c) speaker combinations for keplsc



(d) speaker combinations for fwaspo

Figure 4.4: Distribution of frame lengths of the train utterances for different speaker combinations.

Neural Network Training and Decoding

Based on the parameter settings for the speaker recognition system discussed in Section 3.2.3 (cf. Table 3.2), the settings for training and decoding the neural network were chosen, with most of the parameter combinations taken over. However, the frame length distribution for the speaker combination keplsc (cf. Figure 4.4c) supported the reduction of the minimum required frame length and hence a modified chunk size interval (cf. Table 4.3).

Table 4.3: Parameter setting for the speaker embedding extraction depending on the speaker combination.

speaker combinations	minimum frame length	utterances left/ original utterances	chunk size interval	embedding dimension
base speakers	200	5812/8398	[150, 200]	128
bscwke	200	2213/2964	[150, 200]	128
keplsc	150	2485/2964	[130, 150]	128
fwaspo	200	1980/2964	[150, 200]	128

Post Processing

The post processing step included the centering of the x-vectors and the dimensionality reduction by using LDA with dimension 100, as well as the scoring based on PLDA and a predefined trial file (cf. Section 3.2.1 and Figure 3.7).

Results

A final evaluation was performed based on the evaluation metrics shown in Section 3.2.2: the equivalent error rate (EER), the minimum detection cost function (minDCF) and the t-distributed stochastic neighbour embedding (t-SNE). The cost of the missed detections C_{missed} and false alarms C_{FA} were set to 1 and the prior target probability P_{tar} was calculated for each experiment separately (cf. Section 3.2.3).
Table 4.4 shows the EER and the minDCF of the four different experiments, whereas the outcome of the t-SNE is illustrated in Figures 4.5 - 4.8. When comparing the outcome of the base speaker experiment to the result from the experiment with the original WASS corpus (cf. Section 3.2.3 and Table 3.4), similar values were achieved in terms of the EER and the minDCF. Additionally the graphical evaluation confirmed a good separability between the individual speakers. The performance of the speaker recognition system degraded when considering the additional interpolation factors of three base speaker combinations. However, the transition of the interpolation factor from $\alpha = 0.0$ to $\alpha = 1.0$ is clearly visible in Figures 4.6 - 4.8.

Table 4.4: Performance of the speaker recognition system for different speaker combinations.

speaker combinations	EER (in %)	minDCF(P_{tar})
base speakers	5.23	minDCF(0.110) = 0.37
bscwke	19.14	minDCF(0.287) = 0.53
keplsc	16.05	minDCF(0.287) = 0.53
fwaspo	26.54	minDCF(0.287) = 0.79

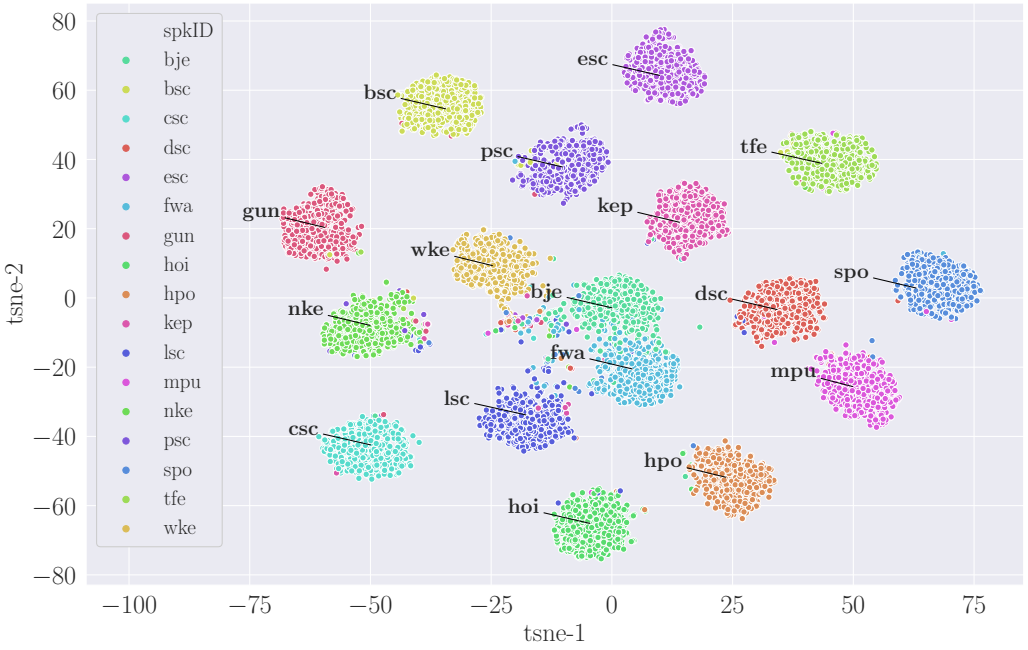


Figure 4.5: t-SNE for 17 base speakers representing the original speakers from the WASS corpus with the following parameters: minimum frame length: 200; chunk size interval: [150, 200]; embedding dimension: 128; LDA dimension: 100.

Figure 4.6: t-SNE for speaker combination bscwke with the following parameters: minimum frame length: 200; chunk size interval: [150, 200]; embedding dimension: 128; LDA dimension: 100.

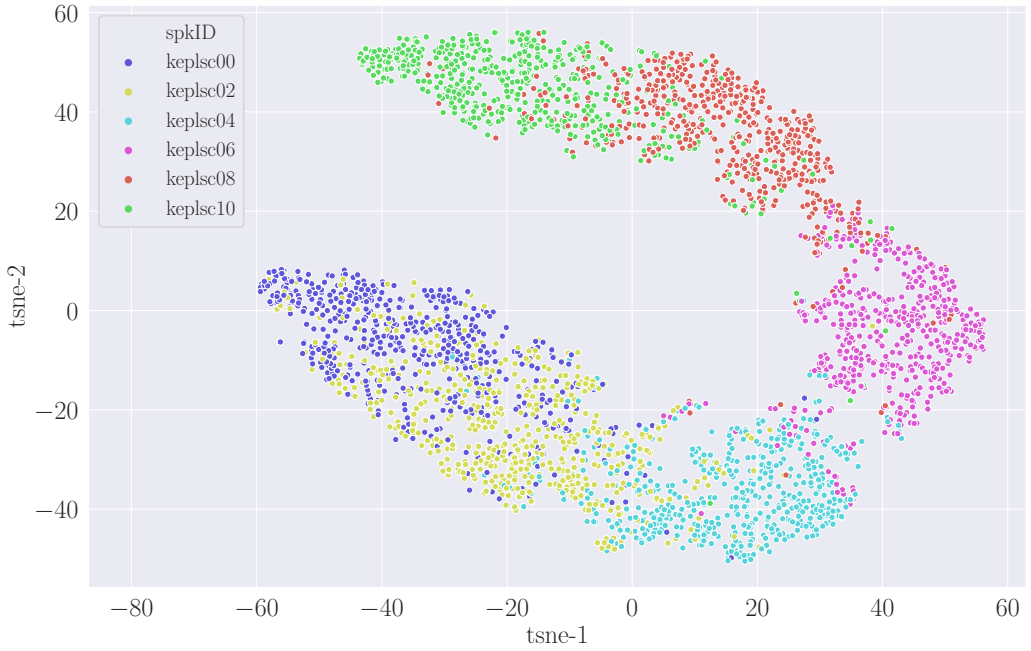


Figure 4.7: t-SNE for speaker combination keplsc with the following parameters: minimum frame length: 150; chunk size interval: [130, 150]; embedding dimension: 128; LDA dimension: 100.

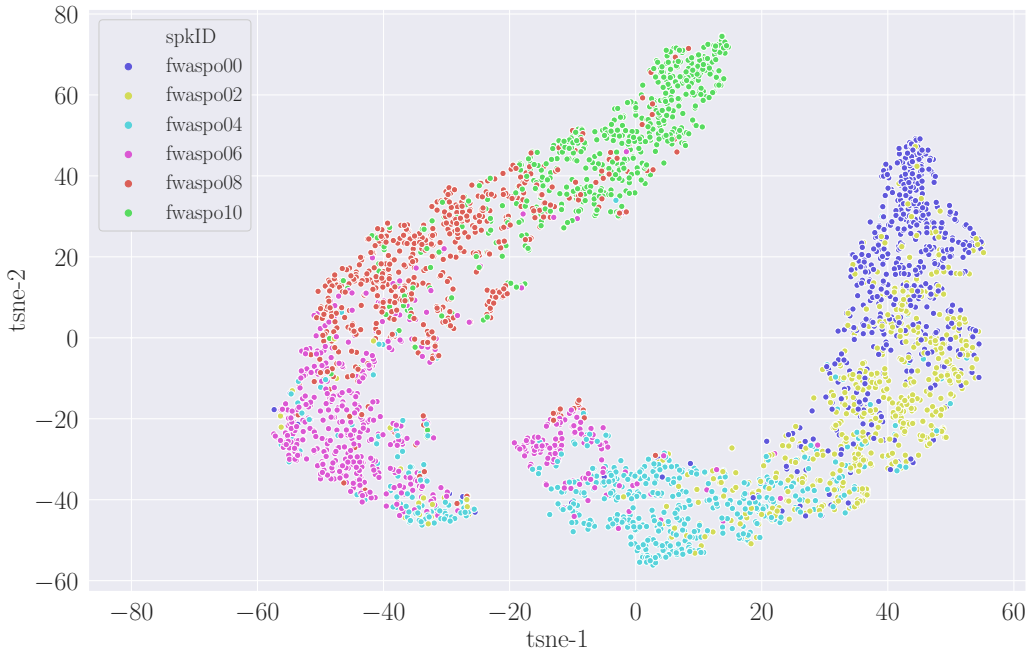


Figure 4.8: t-SNE for speaker combination fwaspo with the following parameters: minimum frame length: 200; chunk size interval: [150, 200]; embedding dimension: 128; LDA dimension: 100.

5 Automatic Speech Recognition

5.1 Introduction to Automatic Speech Recognition (ASR)

Karat, Yankelovich and Lai [31] describe automatic speech recognition (ASR) as process of decoding and transcribing spoken language. In more detail, a common ASR system receives acoustic input from a speaker, analyses it using some pattern, model, or algorithm, and produces an output, usually in the form of a text [31] (cf. Figure 5.1).

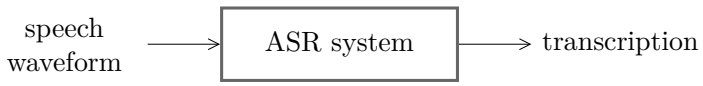


Figure 5.1: Conceptual ASR system: The incoming speech waveform is converted into a sequence of text hypotheses by the ASR system [32].

5.1.1 Field of Application

Due to performance improvements of ASR systems, they are finding ever broader areas of application and are increasingly being integrated into everyday life. Many people associate voice recognition with state-of-the-art consumer products such as voice control in mobile phones, car navigation systems and household appliances to make everyday life easier. Apart from these applications, automatic speech recognition is also used in healthcare to assist, for example, blind and visually impaired people. In education, ASR is used in the field of translation as well as in various language learning softwares. In addition, approved ASR systems are increasingly taking the place of people when it comes to information services for banking services or flight booking services [33], for which the ASR system is integrated into a dialogue system.
It is likely that with performance increases of ASR systems also for more naturalistic speech the fields of application for ASR systems will continue to grow in the future.

5.1.2 Challenges

The success of ASR is mainly due to the integration of machine learning in the ASR training process. The largest advancements in ASR in the last 10 years have come due to DNNs [34]. However, training highly complex models requires a large amount of transcribed speech recordings, which are often challenging to obtain.
Furthermore, in state-of-the-art ASR systems, various components are combined for which knowledge from disciplines such as linguistics or computer science is necessary, but also expertise in the fields of acoustics, communication theory or physiology is beneficial [33].
This multidisciplinary viewpoint is required to deal with the challenges that ASR faces. These challenges mainly concern linguistics, as there arises the question which units (words, syllables, phonemes, etc.) should be used for recognition. Furthermore, syntactic and semantic ambiguities as well as the complexity of language influence ASR. But speech variability also plays an important role, including different voices, accents, styles or speech rates. In addition to the linguistic challenges already mentioned, environmental conditions such as background noise or simultaneous speech are an important issue in ASR [33].

5.1.3 Development of ASR Systems

As mentioned previously, an ASR system converts an incoming speech waveform into a corresponding text representation. This process is based on a statistical model that infers a text sequence $W = w_1, w_2, ..., w_m$ from a sequence of acoustic features vectors $X = x_1, x_2, ..., x_n$ (cf. Equation 5.1) [35, 36]:

$$W^* = \underset{W}{\operatorname{argmax}} P(W|X) = \underset{W}{\operatorname{argmax}} P(X|W)P(W), \tag{5.1}$$

where $P(X|W)$ describes the likelihood of the observed feature vector X given the word sequence W and $P(W)$ represents the probability for a string of words W in a specific language [35]. In order to recognise the input speech, three main components have to be considered. For speech recognition it is not only important to analyse the acoustic input waveform but to take a language model into account as well, which contains linguistic information, e.g., n-gram language modeling. Additionally a lexicon is used to build the relation between words and their corresponding pronunciation. This interaction of several components is evident in Equation 5.1, which consists of a first part $P(X|W)$ representing the lexicon and the acoustic model and a second part $P(W)$ describing the language model.
Figure 5.2 shows the flowchart of a conventional ASR system, in which the main components are considered separately. It includes a pre-processing step as well as a frontend for feature extraction and the following three main parts [37]:

- a **lexicon**, also called pronunciation model is a mapping between words and their corresponding phones,
- an **acoustic model** models a sequence of feature vectors given a sequence of phones,
- a **language model** gives the likelihood of a word sequence based on an underlying grammar.

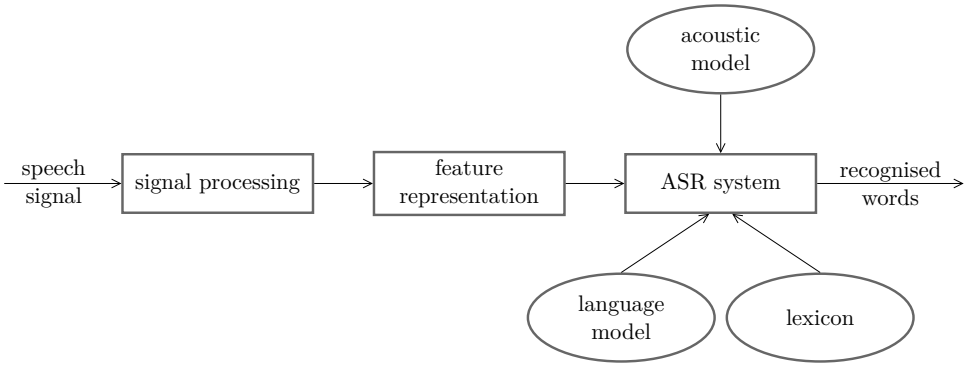


Figure 5.2: Conventional ASR system including a preprocesssing step and three main components (lexicon, acoustic model and language model) [38].

The conventional approach according to [39] uses Gaussian Mixture Models (GMM) to model the feature distribution for a phone and a Hidden Markov Model (HMM) models the transition between phones and the observed features. Based on the statistics an optimal state sequence is to be determined for a sequence of input features, which requires information from three separately trained models, namely the language model, the acoustic model and the lexicon.
Around 2006, Li Deng and Hinton ([39], [40]) suggested to integrate deep learning in the speech recognition process. Hybrid HMM/DNN systems show great performance regarding the speech recognition performance. However, the separate training of the three individual models make them quite time-consuming, which leads towards an end-to-end (E2E) approach. This reduces

the needed knowledge of the user regarding the single components because E2E ASR systems only require extracted features as input and output the desired transcription [39]. Since E2E systems are not used for the experiments presented in this thesis, the reader is referred to [41] for an overview on current developments in the field of E2E ASR systems.

5.2 Building a Kaldi-based ASR System

This work addresses the challenge of speaker variability, especially with regard to different voices (cf. Section 5.1.2). By augmenting the training corpus with interpolated voices, the ASR system was to be trained for individual voice characteristics in order to improve its performance.
Therefore, the already developed ASR system at the SPSC Graz was applied in this work, which employs a recipe developed by Julian Linke ([42]). The ASR system is based on Kaldi, a widely used state-of-the-art speech recognition toolkit [43] that follows the conventional GMM/HMM approach (cf. Section 5.1.3) and consists of three main components, namely the lexicon, the language model and the acoustic model (cf. Figure 5.2). The training of the individual components was performed according to the conventional method, i.e. each component was trained separately (cf. Section 5.1.3). The Kaldi-based ASR system also has the option of GMM/DNN acoustic modeling, but for read speech the GMM/HMM approach already yielded good results, therefore this approach was used in this work.
Figure 5.3 gives an overview of the individual steps which were performed in order to train and evaluate the ASR system. A detailed description of these steps is given in the following sections, which include the file preparation and the feature extraction, the creation of the lexicon and the language model as well as the training and evaluation of the acoustic model.

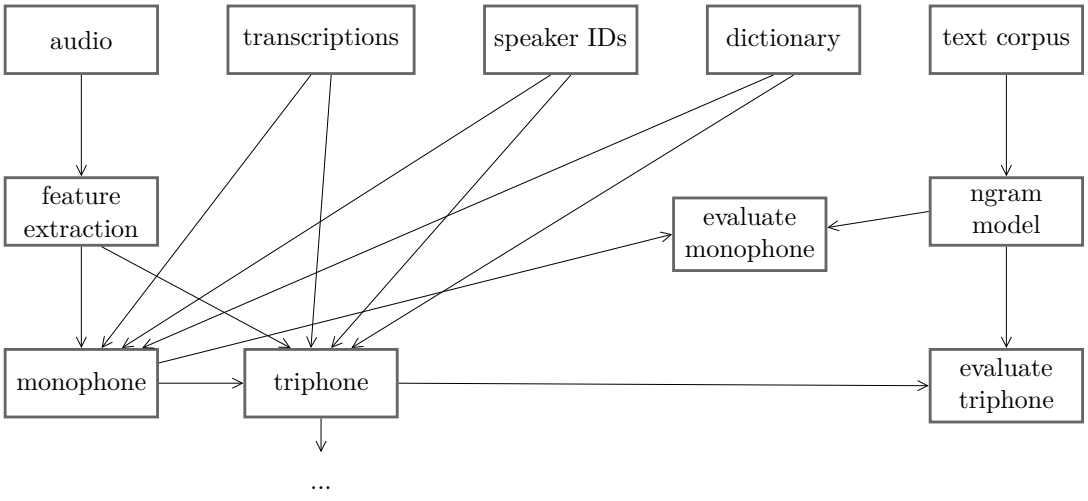


Figure 5.3: An illustration of the proceeding showing the individual steps during the ASR training and evaluation. It includes the basic elements and pre-processing step that are necessary for the training and evaluation process as well as the main components, namely the language model, the acoustic model and the lexicon [44].

5.2.1 Data Preparation

Prior to training the individual components of an ASR system, the required data had to be prepared. According to the description of the Kaldi toolkit, three files needed to be generated in advance to the training process, namely `wav.scp`, `text` and `utt2spk`. The pre-processing in this work included the generation of the information for the newly created speech waveforms as well as the subsequent combination with the already existing information. Since in this work the

training set of the GRASS corpus was augmented while development and test sets remained unchanged, only the training set was considered in the pre-processing.

Audio files

The first step concerned indexing all newly created audio files with the corresponding utterance IDs. As shown in Infobox 5.1 the first column represents the utterance ID, containing not only the utterance but also the speaker ID separated by a dash, and the second column gives the absolute path to the audio file. Therefore, the utterance ID in the first row in Infobox 5.1 `tfewke02-BE_001` includes the utterance `BE_001` and the speaker ID `tfewke02`, where the latter indicates that the utterance was spoken by a speaker obtained from the interpolation between speaker `tfe` and speaker `wke` with an interpolation factor of $\alpha = 0.2$.

utterance ID	**path to audio file**
tfewke02-BE_001	/clusterFS/[...]/data/train/corpus-synth/tfewke02-BE_001.wav
gunwke10-SC_014	/clusterFS/[...]/data/train/corpus-synth/gunwke10-SC_014.wav
...	...

Infobox 5.1: File `wav.scp` *contains the utterance ID and the absolute path to the corresponding audio file, whereby the utterance ID contains the speaker information in form of a speaker ID.*

Transcriptions

Each audio file was then related to its corresponding transcription for the train, evaluation and test set. This information is described in the file `text` (cf. Infobox 5.2), where the first column shows the utterance ID and the second column contains the corresponding transcription.

utterance ID	**transcription**
tfewke02-BE_001	HEUTE IST SCHÖNES FRÜHLINGSWETTER
gunwke10-SC_014	DREH DAS LICHT ÜBER DER ABWASCH AB
...	...

Infobox 5.2: File `text` *contains the utterance ID and the corresponding transcription, whereby the utterance ID contains the speaker information in form of a speaker ID.*

Speaker Information

To train speaker-specific models it was necessary to assign each utterance to its corresponding speaker. This information was stored in the file `utt2spk`. But the inverse information was required as well, which summarised the utterances that belong to the same speaker and was stored in the file `spk2utt`. Infobox 5.3 and 5.4 show the content of file `utt2spk` and `spk2utt`.

utterance ID	**speaker ID**
tfewke02-BE_001	tfewke02
gunwke10-SC_014	gunwke10
...	...

Infobox 5.3: File `utt2spk` *assigns the utterance ID with the corresponding speaker ID.*

speaker ID	utterance ID
tfewke02	tfewke02-BE_001 tfewke02-BE_002 tfewke02-BE_003 tfewke02-BE_004 tfewke02-BE_005 ...
gunwke10	gunwke10-SC_001 gunwke10-SC_002 gunwke10-SC_003 gunwke10-SC_004 gunwke10-SC_005 ...
...	...

Infobox 5.4: File spk2utt *summarises the utterances that belong to the same speaker and assigns them to the corresponding speaker ID.*

5.2.2 Feature Extraction

The feature extraction in form of MFCCs from the audio file is an essential step in ASR, as they describe the sequence of observations, based on which the ASR process was performed. For details on the settings, the reader is referred to Section 6.1. The MFCCs were stored in a file which described the utterance ID followed by the MFCCs in form of a $m \times n$ matrix where m is the number of features and n is number of frames. Then a normalisation step followed (mean and variance normalisation) for each speaker, because of high variability of the speech signal.

5.2.3 Dictionary

One of the main components in an ASR system is the dictionary or lexicon, which contains all words appearing in the corpus with their corresponding pronunciation, whereby each word should be unique. The canonical representation of the words was obtained by using a grapheme-to-phoneme (G2P) conversion.
A very popular G2P conversion tool is the Munich Automatic Segmentation, called MAUS, which was developed by the Institute of Phonetics and Speech Processing in Munich [45]. In order to generate the lexicon, a word list that contained all words from the corpus was uploaded to the MAUS web service [46] and the corresponding list of phonemic representations was returned. This process gave the pronunciation for a word according to statistical rules leading to a pronunciation dictionary.
The G2P conversion is based on rules appearing in standard *German* German, which are slightly different compared to the rules in standard *Austrian* German. Therefore the generated dictionary was corrected according to phonological rules for *Austrian* German, including especially the conversion from voiced /z/ to unvoiced /s/ and the typical pronunciation of the final <g> in *German* German /C/ was changed into the *Austrian* German version /k/:

$$\begin{aligned} f'E6 + tI\mathbf{C} &\longrightarrow f'E6 + tI\mathbf{k} \\ fo:6\mathbf{z}ICt &\longrightarrow fo:6\mathbf{s}ICt \\ In\mathbf{Z}@nj2:6 &\longrightarrow In\mathbf{S}@nj2:6 \end{aligned}$$

Additionally noise and silence phones were defined in order to describe sounds that do not correspond to speech. These noise and silence phones prevent the ASR system from assigning noise or silence with phones from the phone list, resulting in a dictionary containing silence and nonsilence phones as well as noise (cf. Infobox 5.5).

5.2.4 Language Model

The language model provides the ASR system with information from the underlying language. It models the transition between words and defines probabilities of certain word sequences on a symbolic level. In this work a so-called n-gram model was used, where n-gram describes a sequence of n words [47]. The n-gram model is a probabilistic language model for predicting the next word based on a $(n-1)$-th order Markov model. The Markov model assumes that the

unique words	canonical pronunciation
!SIL	SIL
<NOISE>	NSN
<SPOKEN_NOISE>	SPN
<UNK>	SPN
AB	a p
ABDREHEN	a p d r e: @ n
ABDUNKELN	a p d U N k @ l n
...	...
ÜBERRAGTEN	y: b 6 r a: k t @ n
ÜPPIGEN	Y p I g @ n

Infobox 5.5: File `lexicon.txt` *is a list of all spoken words with their corresponding canonical pronunciation and additionally silence phones, noise and unknown words (SIL, NSN, SPN) are considered in the dictionary.*

probability of a word depends only on the previous $(n-1)$ words [47]:

$$P(w_n|w_{1:(n-1)}) \approx P(w_n|w_{(n-N+1):(n-1)}), \tag{5.2}$$

i.e., the probability of the n-th word conditioned on the entire context $P(w_n|w_{1:(n-1)})$ is approximated with the probability of the n-th word considering only the last $N-1$ words $P(w_n|w_{(n-N+1):(n-1)})$, where N describes the n-gram size. Based on the bigram assumption the sequence of words $w_{1:n}$ can be further simplified as [47]:

$$P(w_{1:n}) \approx \prod_{k=1}^{n} P(w_k|w_{k-1}), \tag{5.3}$$

where $P(w_k|w_{k-1})$ is the probability of a bigram. Bigrams are estimated using the maximum likelihood estimation (MLE). This approach counts the bigram probability $C(w_{n-1}w_n)$ of a word w_n given a previous word w_{n-1} and normalises the count by the unigram count $C(w_{n-1})$ of the word w_{n-1} [47]:

$$P(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)}{C(w_{n-1})}. \tag{5.4}$$

The SRI International's Speech Technology and Research (STAR) Laboratory provides a Language Modeling Toolkit (SRILM) that includes a method to count n-grams and estimate language models from them based on a text input [48]. Training the SRI language model required the utterance transcriptions from the training set, which were stored in a separate file, one utterance per line (cf. Infobox 5.6). Based on the given utterance transcriptions (`corpus.txt`)

corpus utterances
HEUTE IST SCHÖNES FRÜHLINGSWETTER
DIE SONNE LACHT
AM BLAUEN HIMMEL ZIEHEN DIE WOLKEN
...

Infobox 5.6: File `corpus.txt`*: Collection of all spoken utterances obtained from the transcription file.*

and the predefined language model order `lm_order = 3`, the SRILM generated a n-gram count and estimated a n-gram language model. The language model was used to prepare a grammar which was converted into a FST (finite state transducer) format.

5.2.5 Acoustic Model

Gaussian Mixture Models (GMM)

The acoustic model builds the relationship between the observed acoustic feature vectors (MFCCs) extracted from a speech waveform and the underlying phones. In the conventional approach the acoustic model is based on Gaussian Mixture Models (GMMs), in which the distribution of feature vectors for a given phone are modeled by a weighted sum of Gaussians (cf. Equation 5.5):

$$P(x|h) = \sum_i p_i \mathcal{N}(\mu_i, \Sigma_i), \tag{5.5}$$

where $P(x|h)$ describes the likelihood of a feature vector x given the phone h. Based on the training data, the parameters of the Gaussians are to be estimated, i.e., the means μ_i and standard deviations Σ_i. The trained GMMs are used to model the acoustic feature vectors, based on which the underlying phone is to be predicted [36].

5.2.6 Hidden Markov Models (HMM)

By employing GMM, only short frames are analysed without including the temporal dependencies and transitions between phones. The temporal dependencies can be modeled using a Hidden Markov Model (HMM). Similar to a Markov Model, that models a sequence of observable random variables (states), in a Hidden Markov Model these states are unknown, i.e., hidden. The HMM has only access to an observation sequence based on which a HMM should infer the hidden state sequence [36],[49].
An HMM is mainly characterised by emission probabilities and transition probabilities. The previously mentioned likelihood of the acoustic feature vectors given a phone modeled by the GMM (cf. Equation 5.5) is considered in the HMM by means of emission probabilities. In order to consider the temporal structure of phones, the HMM uses transition probabilities, which include the information obtained from the lexicon and the language model to describe the transition between phones [36],[49]. The combination of emission and transition probabilities allows to calculate the probability of the observed feature vector $P(x)$ by considering all phones h that lead to the observation x:

$$P(x) = \sum_h P(h) \cdot P(x|h), \tag{5.6}$$

where $P(h)$ describes the probability of a phone and $P(x|h)$ is the likelihood of features given a phone modeled using a GMM [36, 49].

Training HMM

Training an HMM means to learn the emission and transition probabilities from the given data. During the training process, the HMM has access to the observation sequence, i.e. the feature vectors, and to the underlying hidden state sequence, i.e., the phones. As can be seen in Equation 5.6, information from the acoustic model, but also from the lexicon and the language model is required for determining the emission and transition probabilities of the HMM.
The training of the acoustic model was divided into two parts. Firstly a monophone model was trained by considering only one single phone, in which any contextual information, i.e., neighbouring phones were neglected. In each training iteration, the parameters of the HMM were estimated. For refining the model parameters each training step was followed by an alignment step, that aligned the audio and the transcript according to the current acoustic model. The trained monophone model was used as basis for the triphone model, which considered neighboring phones, i.e., integrated the effect of co-articulation. With the triphone model, different phoneme variants were represented, depending on the left and right phoneme [50].

Decoding HMM

During decoding, the performance of the ASR system was evaluated. In this step the previously trained models (monophone models and triphone models) received a sequence of feature vectors extracted from an unknown speech waveform. Based on the learnt emission and transition probabilities, the model calculated the most likely state sequence, i.e., the most likely phone sequence that had produced the given feature vectors [36, 49].

5.2.7 Evaluation

The most commonly used method for evaluating ASR performance is to compare the recognised string of words to a reference transcription, resulting in an evaluation metric called Word Error Rate (WER). This approach counts the differences between the two strings, distinguishing between insertions, deletions and substitutions. The number of insertions describes how often a word is recognised although not being present in the reference string. In contrast, a deletion occurs when a word is not recognised but in the reference transcription and for a substitution the wrong word was recognised compared to the reference string. Insertions, deletions and substitutions count equally, resulting in the equation for the WER in %:

$$\frac{I + D + S}{N}, \tag{5.7}$$

where I, D and S describe the number of insertions, deletions and substitutions and N describes the total number of words in the reference transcript [51].
Based on the comparison between the recognised string and the reference string a second evaluation metric was considered, the Sentence Error Rate (SER), which is given in % and counts the number of incorrectly recognised utterances compared to the reference utterances.

6 ASR Experiments

This chapter describes the ASR experiments carried out with the augmented training corpus by means of speaker interpolation (cf. Chapter 4). The augmented training corpus offered the possibility to train the ASR system with different data combinations and to compare the results afterwards. Based on these experiments, a training dataset was determined in order to improve the performance of the ASR system.
The subsequent sections document the experiments performed in this work, which followed the same approach described in Section 5.2 and differed only in the composition of the training, development and test sets. The experiments included the data and file preparation as well as the generation of a language model and an acoustic model, the latter being trained based on monophones and triphones. For evaluation the results of the experiments are presented by means of the WER and the SER in tabular and in graphical form.

6.1 Experiment Settings

An overview of the most relevant settings for the ASR system, which remained unchanged for all experiments, is given in this section. For the extraction of the MFCCs from the audio files (sample frequency of $48kHz$) a frame length of $20ms$ and a frame shift of $12.5ms$ (while setting use-energy to false) was used. These settings showed the best performance in earlier experiments by J. Linke. The configurations of the language model and the training of monophone and triphone model are given in Table 6.1. For the decoding, a first-beam of 10.0 and a regular beam of 13.0 was used, while the lattice beam was set to 6.0.

Table 6.1: Experimental setup of the general parameters, including the settings for the language model and the acoustic model (monophone and triphone model).

	parameters	values	description
language model	lm_order	3	order of the language model
acoustic model (monophone)	num_iters	40	number of training iterations
	initial_beam	6	beam used in the first iteration
	regular_beam	10	beam used after first iteration
	retry_beam	40	beam used after retry
	totgauss	1000	target number of Gaussians
	boost_silence	1.0	boost silence likelihoods in alignment
acoustic model (triphone)	num_iters	35	number of training iterations
	beam	10	regular beam
	retry_beam	40	beam used after retry
	totgauss	11000	target number of Gaussians
	numleaves	2000	number of leaves
	boost_silence	1.0	boost silence likelihoods in alignment

6.2 Baseline Experiment

In the first experiment, the ASR system was trained, developed and tested with utterances from the original training set of the GRASS corpus. This experiment was intended to reproduce the initial results of the ASR system and served as basis for further experiments. Table 6.2 shows the distribution of the GRASS corpus into train, development and test set.

Table 6.2: Division of the utterances from the GRASS corpus into training, development and test set: The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	GRASS			
		utterances		speakers	
		total	different	m	f
baseline	train	3820	1248	17	16
	dev	249	154	1	1
	test	254	162	1	1

6.2.1 Result

The results for the baseline experiment are shown in Table 6.3. For the monophone model as well as for the triphone model, the ASR system showed promising results if being trained, developed and tested with the original GRASS corpus. For both models a WER around 1% was achieved, which served as a basis for further experiments.

Table 6.3: Performance of the ASR system in terms of the WER and the SER based on the original training, development and test set of the GRASS corpus. The results of the monophone model and the triphone model are shown separately.

id	train	dev/test	monophone model		triphone model	
			WER (%)	SER (%)	WER (%)	SER (%)
baseline	GRASS	GRASS	1.12	4.33	0.88	3.54

6.3 Corpus Augmentation using Synthesised Speech

The following experiments dealt with the augmentation of the original GRASS training set using the synthesised utterances based on the speaker interpolation (cf. Figure 6.1). Different experiments were conducted regarding the amount of synthesised speech and the way the synthesised utterances were selected for augmentation. These experiments investigated how these conditions affect the performance of the ASR system. For development and testing, the original development and test sets from the GRASS corpus were used according to the sets in the baseline experiment (cf. Section 6.2).

Figure 6.1: Composition of the training, development and test set in the experiments with the augmented GRASS training corpus, with development and test set from the original GRASS corpus.

6.3.1 Amount of Synthesised Speech

An essential parameter that was adjusted when augmenting the training set was the amount of synthesised utterances added to the original training set. For this purpose, the synthesised speech was successively added to the original training corpus in order to find the optimal ratio between original and synthesised speech for training the ASR system. Table A.1 shows the different training set compositions for the experiments conducted at a varying amount of synthesised speech added to the (constant sized) GRASS training set.

6.3.2 Selection of Utterances

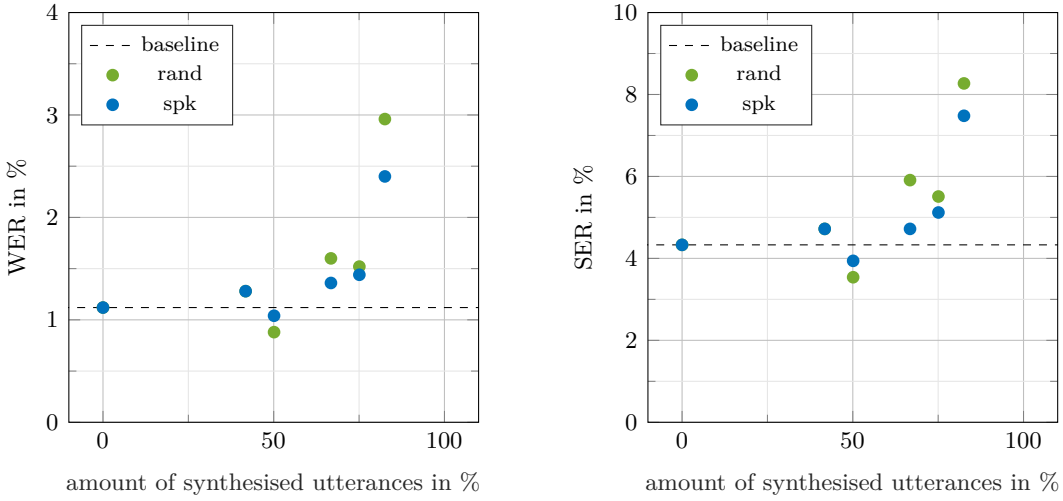
The synthesised utterances used for augmenting the training set were, in separate experiments, either chosen per speaker or randomly. If the training set was chosen per speaker, all utterances from chosen speakers were selected. Therefore a first experiment investigated the performance of the ASR system when taking utterances from a fixed set of randomly chosen speakers whereas in a subsequent experiment the same amount of utterances were randomly chosen from a set including all speakers and utterances. The distribution of the sets for training, development and testing are listed in Table A.1, which compares the results for speaker-specific selection (spk) and the results when randomly choosing the same amount from all speakers and utterances (rand).

6.3.3 Results

Table 6.4 shows the results which indicate that a ratio of 50% between original and synthesised utterances in the training set improved the performance of the ASR system. Apart from that it was observed that for this range of utterances, the random selection mode led to better results compared to the speaker-specific selection mode.

Table 6.4: Performance of the ASR system in terms of the WER and the SER based on the training set of the GRASS corpus augmented by using utterances from the synthesised GRASS corpus (s). The results of the monophone model and the triphone model are shown separately.

id	train	dev/test	monophone model		triphone model	
			WER (%)	SER (%)	WER (%)	SER (%)
baseline	GRASS	GRASS	1.12	4.33	0.88	3.54
1	GRASS $\text{GRASS(s)}_{\text{spk,2740}}$	GRASS	1.28	4.72	0.96	3.54
2	GRASS $\text{GRASS(s)}_{\text{rand,2740}}$	GRASS	1.28	4.72	0.96	3.54
3	GRASS $\text{GRASS(s)}_{\text{spk,3836}}$	GRASS	1.04	3.94	0.80	3.15
4	GRASS $\text{GRASS(s)}_{\text{rand,3800}}$	GRASS	0.88	3.54	0.64	2.36
5	GRASS $\text{GRASS(s)}_{\text{spk,7672}}$	GRASS	1.36	4.72	0.80	3.15
6	GRASS $\text{GRASS(s)}_{\text{rand,7672}}$	GRASS	1.60	5.91	0.80	3.15
7	GRASS $\text{GRASS(s)}_{\text{spk,11508}}$	GRASS	1.44	5.12	0.88	3.15
8	GRASS $\text{GRASS(s)}_{\text{rand,11508}}$	GRASS	1.52	5.51	0.88	3.54
9	GRASS $\text{GRASS(s)}_{\text{spk,18084}}$	GRASS	2.40	7.48	1.36	3.94
10	GRASS $\text{GRASS(s)}_{\text{rand,18084}}$	GRASS	2.96	8.27	1.36	5.51



(a) WER based on the monophone training

(b) SER based on the monophone training

(c) WER based on the triphone training

(d) SER based on the triphone training

Figure 6.2: Performance of the ASR system depending on the amount of utterances from the synthesised GRASS corpus, with the number of utterances from the original GRASS corpus remaining constant. The development and test set are based on the original GRASS corpus. The dashed line represents the result of the baseline experiment.

6.3.4 Optimisation of Data Augmentation

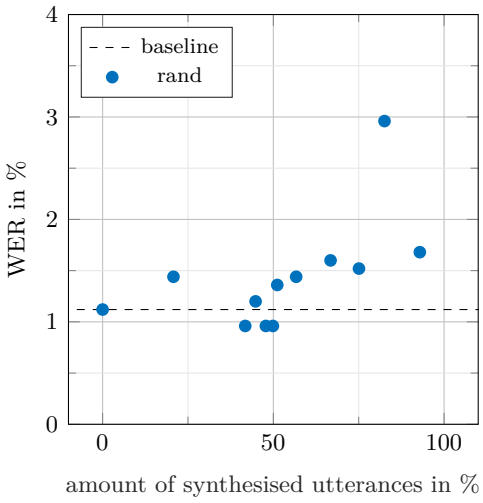
Based on the previous experiments regarding the amount of data augmentation using synthesised utterances and the experiments concerning the selection mode of training data, the optimal ratio between original and synthesised utterances is around 50%. In this data range it is advantageous to randomly select the training data. Therefore the following experiments only considered the random selection mode and the amount of data augmentation is variable (cf. Table A.2).

6.3.5 Results

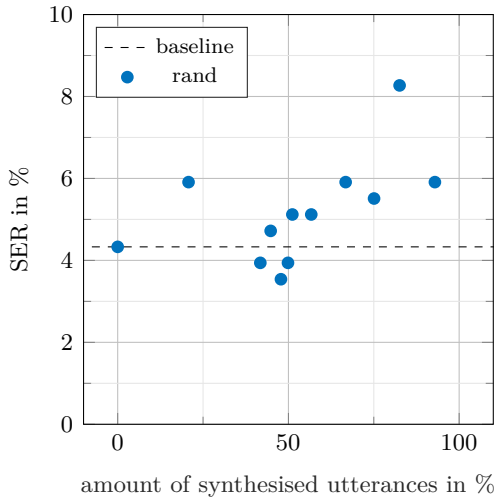
Table 6.5 shows the results indicating that the data augmentation using synthesised speech yields a reduction in WER and SER and thus an improvement in the performance of the ASR system. The optimal amount of data augmentation using synthesised speech is in the range of the amount of utterances from the original GRASS dataset. When interpreting the results, it is important to take into account that due to the random selection of the training data, the results may vary within a small range and cannot always be reproduced exactly.

Table 6.5: Performance of the ASR system in terms of the WER and the SER based on the training set of the GRASS corpus augmented by using utterances from the synthesised GRASS corpus (s). The results of the monophone model and the triphone model are shown separately.

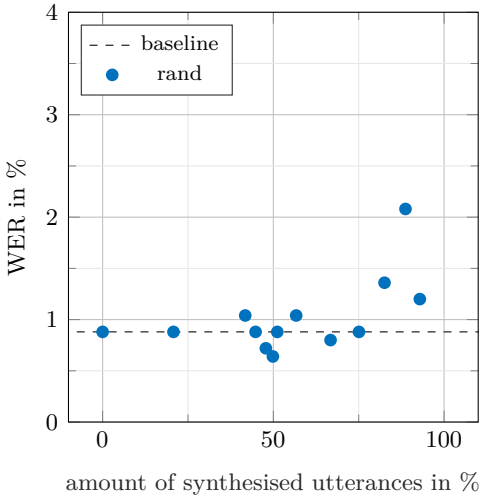
id	train	dev/test	monophone model		triphone model	
			WER (%)	SER (%)	WER (%)	SER (%)
baseline	GRASS	GRASS	1.12	4.33	0.88	3.54
11	GRASS $\text{GRASS(s)}_{\text{rand,1000}}$	GRASS	1.44	5.91	0.88	3.54
2	GRASS $\text{GRASS(s)}_{\text{rand,2740}}$	GRASS	0.96	3.94	1.04	4.33
12	GRASS $\text{GRASS(s)}_{\text{rand,3100}}$	GRASS	1.20	4.72	0.88	3.15
13	GRASS $\text{GRASS(s)}_{\text{rand,3500}}$	GRASS	0.96	3.54	0.72	2.76
4	GRASS $\text{GRASS(s)}_{\text{rand,3800}}$	GRASS	0.96	3.94	0.64	2.36
14	GRASS $\text{GRASS(s)}_{\text{rand,4000}}$	GRASS	1.36	5.12	0.88	3.15
15	GRASS $\text{GRASS(s)}_{\text{rand,5000}}$	GRASS	1.44	5.12	1.04	3.94
6	GRASS $\text{GRASS(s)}_{\text{rand,7672}}$	GRASS	1.60	5.91	0.80	3.15
8	GRASS $\text{GRASS(s)}_{\text{rand,11508}}$	GRASS	1.52	5.51	0.88	3.54
10	GRASS $\text{GRASS(s)}_{\text{rand,18084}}$	GRASS	2.96	8.27	1.36	5.51
16	GRASS $\text{GRASS(s)}_{\text{rand,30000}}$	GRASS	4.17	10.63	2.08	6.30
17	GRASS $\text{GRASS(s)}_{\text{rand,50000}}$	GRASS	1.68	5.91	1.20	4.33



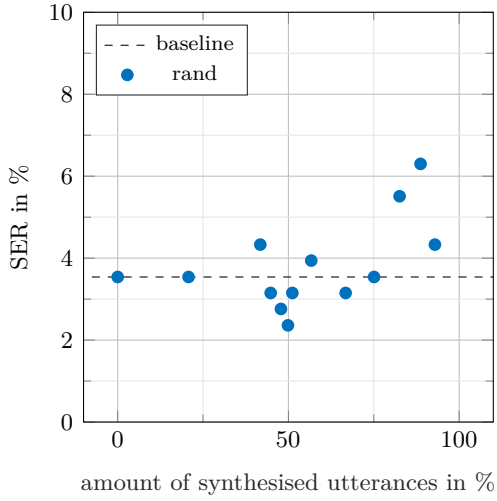
(a) WER based on the monophone training



(b) SER based on the monophone training



(c) WER based on the triphone training



(d) SER based on the triphone training

Figure 6.3: Performance of the ASR system depending on the amount of utterances from the synthesised GRASS corpus, with the number of utterances from the original GRASS corpus remaining constant. The development and test set are based on the original GRASS corpus. The dashed line represents the result of the baseline experiment.

6.4 Replacement using Synthesised Speech

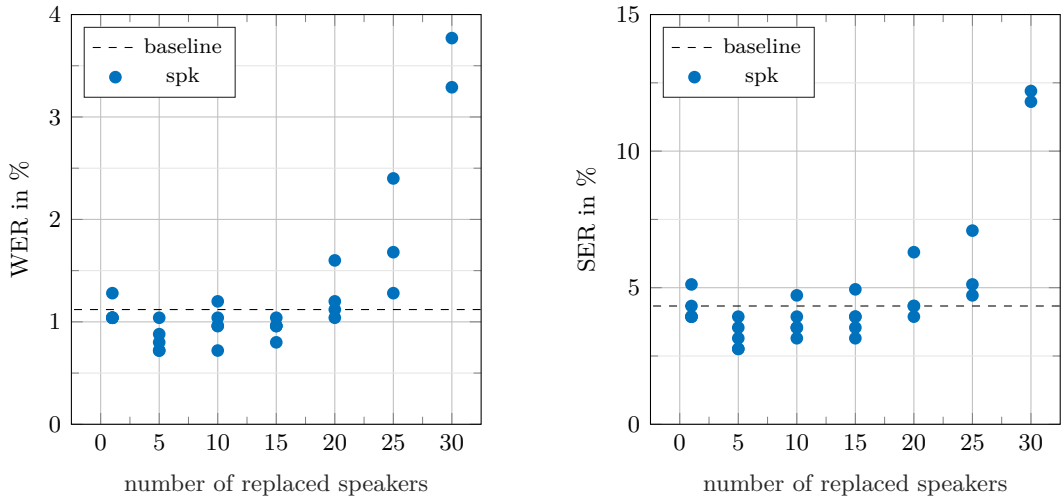
In the previous experiments, all utterances and speakers were taken from the original GRASS training dataset and synthesised utterances were added to the existing training dataset. As a next step, the speakers from the original training set were successively replaced by speakers from the synthesised GRASS corpus. For this purpose a set of speakers was randomly selected and replaced by another random set of synthesised speakers, keeping the utterance IDs the same.

6.4.1 Results

Table 6.6 contains the results for the monophone and triphone model. As the "speakers to be replaced" and the synthesised speakers were selected randomly, the performance of the ASR system varies. Therefore the experiments were conducted multiple times (cf. Tables A.3 - A.9) and the average of these experiments led to the final result in Table 6.6 and Figure 6.4. An analysis of the WER and the SER showed that especially the replacement of speakers up to 15 speakers led to an improvement in the monophone model, however for a replacement of more than 15 speakers the performance degraded. When considering the triphone model it was observed that the replacement of speakers increased the WER and the SER.

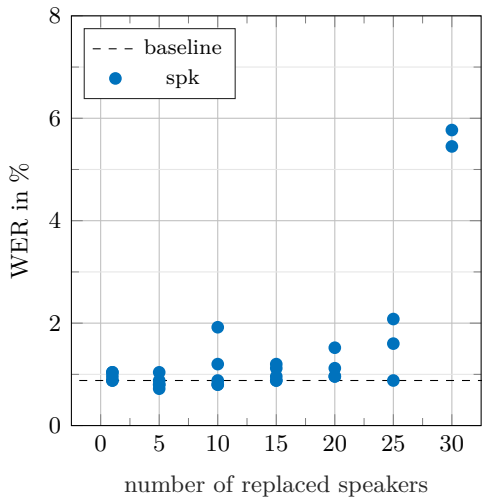
Table 6.6: Performance of the ASR system in terms of the WER and the SER based on the training set of the GRASS corpus where a number of speakers from the original GRASS corpus were replaced by speakers from the synthesised GRASS corpus (s). The results of the monophone model and the triphone model are shown separately.

id	train	dev/test	monophone model		triphone model	
			WER (%)	SER (%)	WER (%)	SER (%)
baseline	GRASS	GRASS	1.12	4.33	0.88	3.54
18	GRASS $\text{GRASS(s)}_{\text{spk(1)}}$	GRASS	1.09	4.25	0.96	3.78
19	GRASS $\text{GRASS(s)}_{\text{spk(5)}}$	GRASS	0.83	3.23	0.85	3.31
20	GRASS $\text{GRASS(s)}_{\text{spk(10)}}$	GRASS	0.98	3.78	1.12	4.25
21	GRASS $\text{GRASS(s)}_{\text{spk(15)}}$	GRASS	0.94	3.70	1.00	3.94
22	GRASS $\text{GRASS(s)}_{\text{spk(20)}}$	GRASS	1.24	4.73	1.14	4.33
23	GRASS $\text{GRASS(s)}_{\text{spk(25)}}$	GRASS	1.79	5.64	1.52	4.98
24	GRASS $\text{GRASS(s)}_{\text{spk(30)}}$	GRASS	3.53	12.00	5.61	15.75

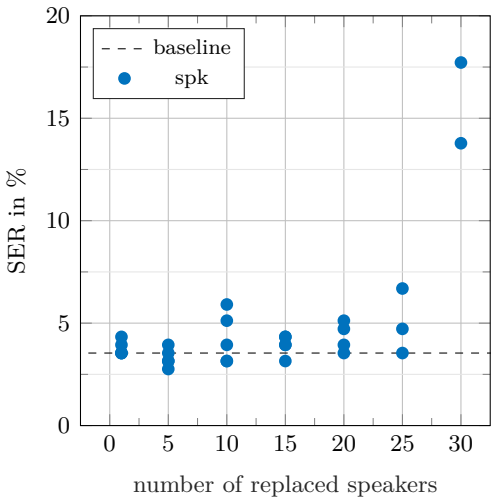


(a) WER based on the monophone training

(b) SER based on the monophone training



(c) WER based on the triphone training



(d) SER based on the triphone training

Figure 6.4: Performance of the ASR system depending on the number of speakers replaced by speakers from the synthesised GRASS corpus, with the number and ID of utterances remaining constant. The development and test set are based on the original GRASS corpus. The dashed line represents the result of the baseline experiment.

6.5 Using the Synthesised Corpus for Training

The previous experiments (cf. Section 6.3) showed that augmenting an existing corpus containing original voices with utterances from synthesised voices decreased the WER and SER of ASR systems. Additionally it was interesting to investigate the performance of the ASR system when being trained only with utterances from the synthesised GRASS corpus. Therefore the synthesised voices obtained by speaker interpolation were used as training set for the ASR system. The utterances for the development and test set were either from the original GRASS corpus (cf. Figure 6.5) or from the synthesised GRASS corpus (cf. Figure 6.6). The exact distribution of utterances into training, development and test set is shown in Table A.10. For all experiments in this section the training data was selected randomly from all synthesised utterances.

Figure 6.5: Composition of the training, development and test set in the experiments with the synthesised GRASS training set, with development and test set from the original GRASS corpus.

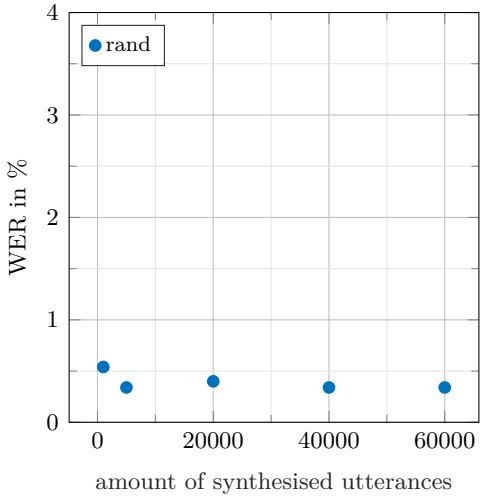
Figure 6.6: Composition of the training, development and test set in the experiments with the synthesised GRASS training set, with development and test set from the synthesised GRASS corpus.

6.5.1 Results

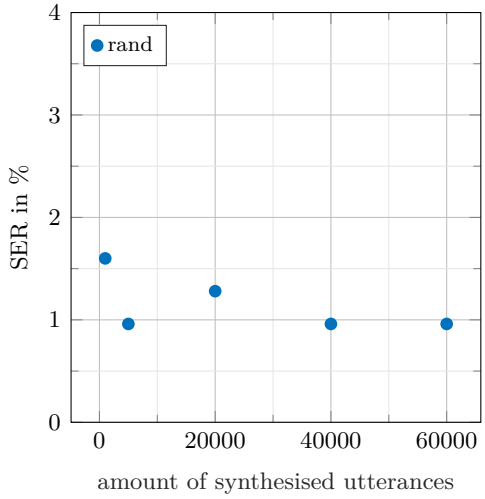
The results in Table 6.7 show a clear mismatch between the synthesised GRASS corpus in the training set and the original GRASS corpus in the development and test set. Regardless of the number of utterances, the WER and SER for both models were around 100%.
If the same training set was used but the development set and test set were made up with utterances from the synthesised voices, the achieved results were excellent, i.e., below 1% WER. An increasing number of synthesised utterances led to an improvement of the ASR performance. However, the improvement saturated at around 40000 utterances, where including more training data did not decrease WER and SER remarkably.

Table 6.7: Performance of the ASR system in terms of the WER and the SER based on the training set with utterances from the synthesised GRASS corpus (s). For development and testing, both corpora were used. The results of the monophone model and the triphone model are shown separately.

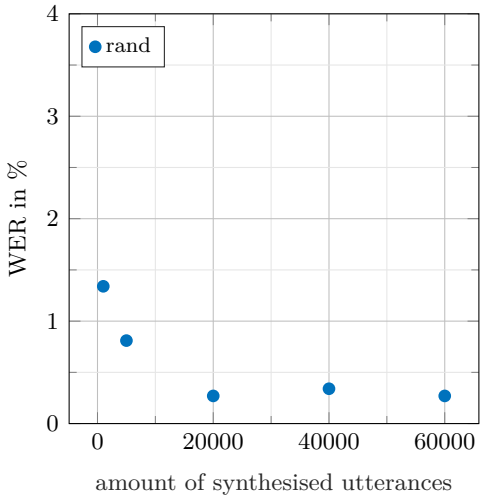
id	train	dev/test	monophone model		triphone model	
			WER (%)	SER (%)	WER (%)	SER (%)
25	$\text{GRASS(s)}_{\text{rand,5000}}$	GRASS	72.92	100.00	95.03	100.00
26	$\text{GRASS(s)}_{\text{rand,20000}}$	GRASS	73.32	100.00	97.36	100.00
27	$\text{GRASS(s)}_{\text{rand,1000}}$	GRASS(s)	0.54	1.60	1.34	2.88
28	$\text{GRASS(s)}_{\text{rand,5000}}$	GRASS(s)	0.34	0.96	0.81	1.92
29	$\text{GRASS(s)}_{\text{rand,20000}}$	GRASS(s)	0.40	1.28	0.27	0.64
30	$\text{GRASS(s)}_{\text{rand,40000}}$	GRASS(s)	0.34	0.96	0.34	0.96
31	$\text{GRASS(s)}_{\text{rand,60000}}$	GRASS(s)	0.34	0.96	0.27	0.64



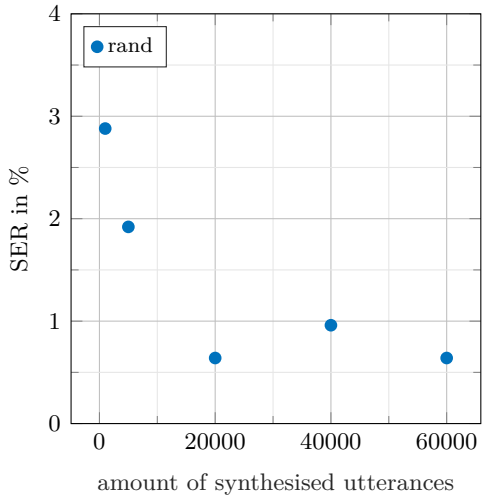
(a) WER based on the monophone training



(b) SER based on the monophone training



(c) WER based on the triphone training



(d) SER based on the triphone training

Figure 6.7: Performance of the ASR system depending on the amount of utterances from the synthesised GRASS corpus. The development and test set are based on the synthesised GRASS corpus.

6.6 Using the WASS Corpus for Training

For these experiments the training corpus consisted of the utterances from the WASS corpus, based on which the AVM and the speaker embeddings were calculated (cf. Section 2.2.2). The WASS corpus contains 17 speakers for which 12 are male and 5 are female, leading to a total number of 7847 utterances. The corpus was divided into train, development and test set according to Table 6.8, which also shows the total number of utterances and the duration in seconds for each set. One female and one male speaker were selected for development and test set.

Table 6.8: Division of the WASS corpus into training, development and test set with the corresponding number of utterances and duration in seconds.

set	speakers		utterances	duration (s)
	m	f		
train	10	3	6955	20483.80
dev	1	1	446	1433.49
test	1	1	446	1395.67

Similarly to the experiments in Section 6.3, the utterances from the original WASS corpus were used for training and augmented by utterances from the synthesised WASS corpus. To generate the synthesised WASS corpus, the interpolated speaker embedding vectors calculated in 3.3 were integrated in the speaker adaptation process in order to create a synthesised WASS corpus based on interpolated speakers.
The utterances for development and test set were either from the original WASS corpus (cf. Figure 6.8) or from the original GRASS corpus (cf. Figure 6.9), the latter used to investigate the mismatch between both corpora. Furthermore, the performance of the ASR system was examined exclusively with utterances from the original WASS corpus or from the synthesised WASS corpus. The influence of the amount of utterances on the ASR performance was not analysed, therefore a fixed number of training utterances was used in all experiments.

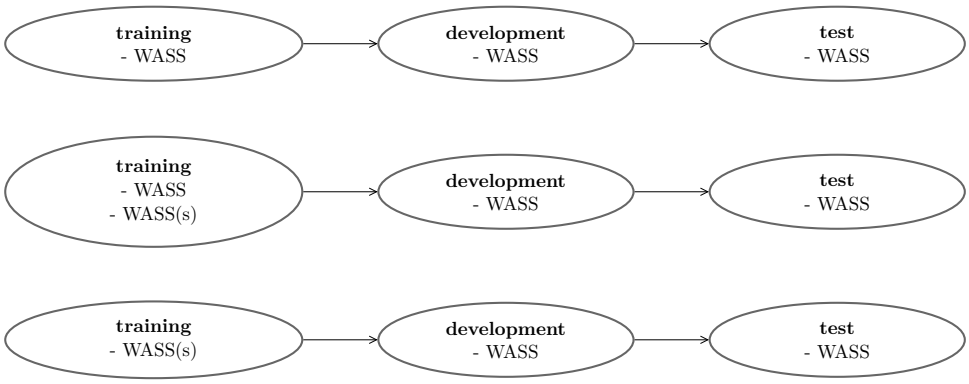


Figure 6.8: Composition of the training, development and test set in the experiments with the WASS corpus. The training set consisted either of the original WASS corpus, the synthesised WASS corpus or a composition of both with development and test set from the original WASS corpus.

6.6.1 Corpus Mismatch

It has been shown in Section 6.5, that a mismatch between the synthesised utterances based on the voices in the WASS corpus and the utterances from the GRASS corpus occurred. Therefore an experiment was performed that uses the original voices from the WASS corpus as training set and the original voices from the GRASS corpus as development and test set. Synthesised voices were successively added to the training set to investigate its influence.

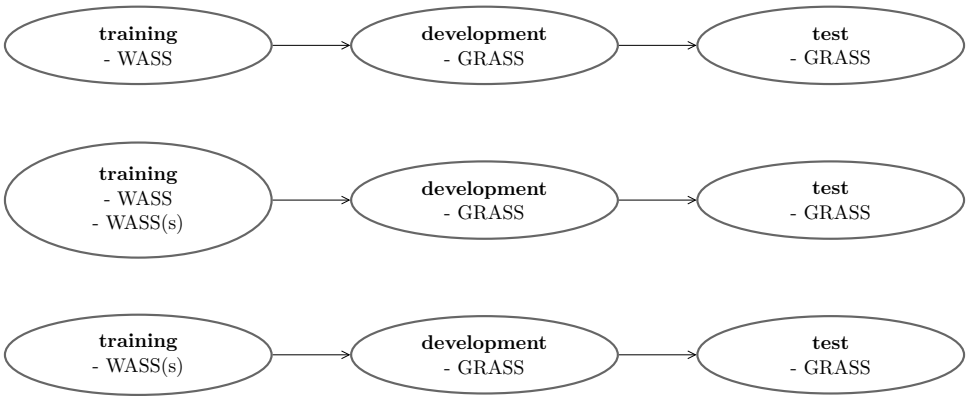


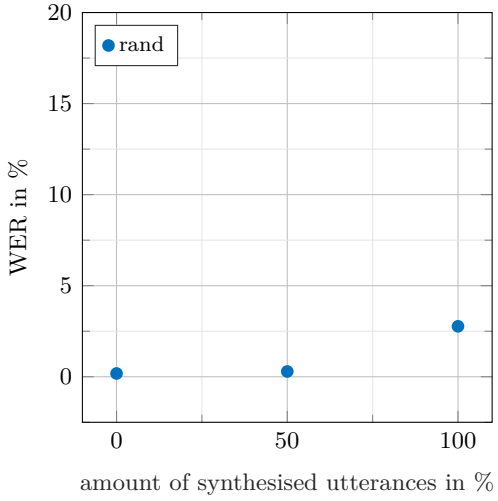
Figure 6.9: Composition of the training, development and test set in the experiments with the WASS corpus. The training set consisted either of the original WASS corpus, the synthesised WASS corpus or a composition of both with development and test set from the original GRASS corpus.

6.6.2 Result

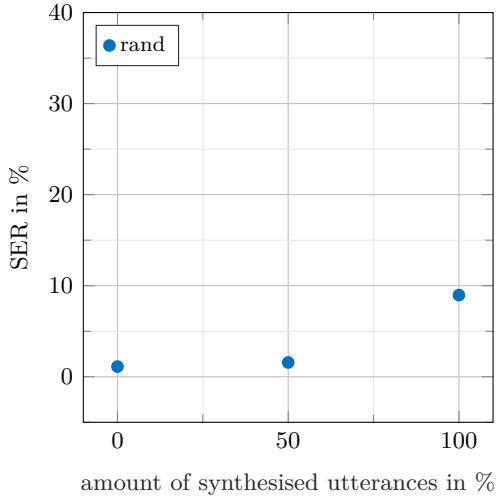
When using the original WASS corpus as training, development and test set, the performance of the ASR was exceptionally good in terms of the WER (0.55%) and the SER (2.69%). However, adding the same amount of interpolated voices to the training set, decreased the WER and SER of the triphone model even further (WER = 0.33%, SER = 1.57%). Training the ASR system only with synthesised voices led to acceptable results (cf. Figure 6.10).
The same observations were made when using the utterances from the original GRASS corpus as development and test set. The experiments show, that the mismatch between different corpora was smallest when using the original corpus for training, development and test set and was highest if the ASR system was trained only with synthesised voices (cf. Figure 6.11). The same trend as in Section 6.3 was visible here: adding the same proportion of synthesised utterances to the original utterances slightly improved the performance for the triphone model while the performance for the monophone model remained almost the same.

Table 6.9: Performance of the ASR system in terms of the WER and the SER based on the training set of the WASS corpus augmented by using utterances from the synthesised WASS corpus (s). For development and testing, the WASS corpus and the GRASS corpus are used. The results of the monophone model and the triphone model are shown separately.

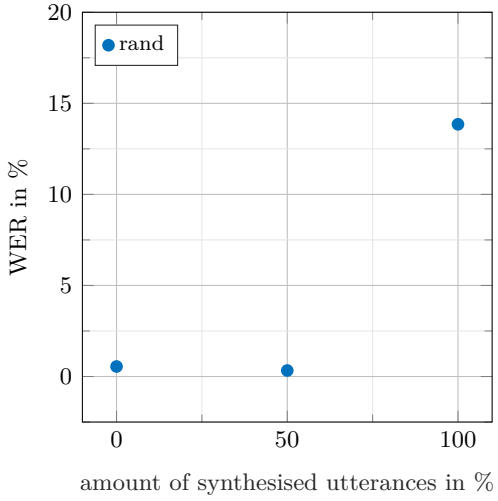
id	train	dev/test	monophone model		triphone model	
			WER (%)	SER (%)	WER (%)	SER (%)
32	WASS	WASS	0.18	1.12	0.55	2.69
33	WASS WASS(s)	WASS	0.29	1.57	0.33	1.57
34	WASS(s)	WASS	2.77	8.97	13.85	36.55
35	WASS	GRASS	5.21	12.60	8.09	25.20
36	WASS WASS(s)	GRASS	6.01	12.99	5.29	15.35
37	WASS(s)	GRASS	19.95	36.22	74.20	85.83



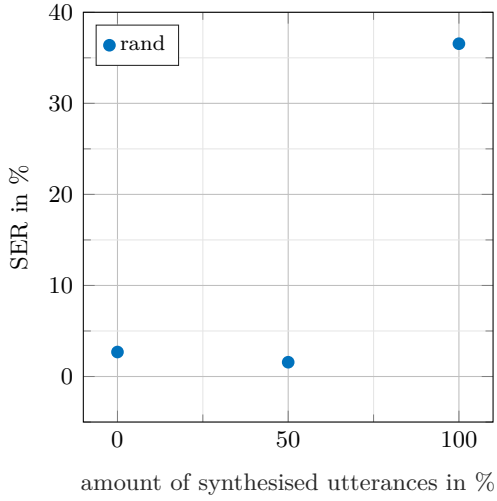
(a) WER based on the monophone training



(b) SER based on the monophone training

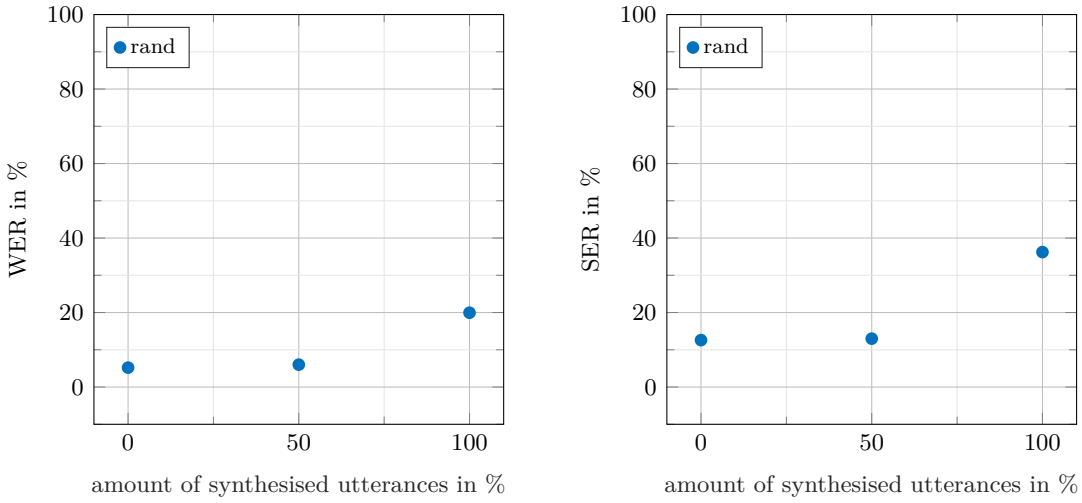


(c) WER based on the triphone training



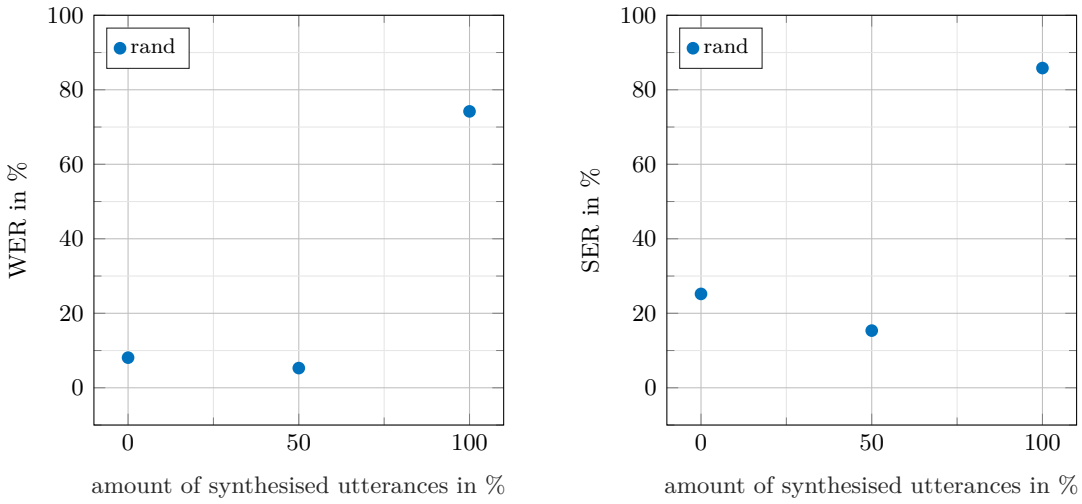
(d) SER based on the triphone training

Figure 6.10: Performance of the ASR system depending on the amount of utterances from the synthesised WASS corpus. The development and test set are based on the original WASS corpus.



(a) WER based on the monophone training

(b) SER based on the monophone training



(c) WER based on the triphone training

(d) SER based on the triphone training

Figure 6.11: Performance of the ASR system depending on the amount of utterances from the synthesised WASS corpus. The development and test set are based on the original GRASS corpus.

7 Discussion

A big challenge in training state-of-the-art ASR systems is the data acquisition, as training highly complex ASR systems requires hours of speech including acoustic and lexical diversity. This issue was the motivation behind this work, that deals with the augmentation of an existing training corpus for an ASR system by using synthesised speech. More precisely, new speaker characteristics were to be generated using a provided corpus by means of interpolation. The created speaker information was integrated in the speech synthesis process in order to create new voices which were furthermore used for data augmentation of the training set of an ASR system. The aim of this work was to test the hypothesis whether increasing the acoustic diversity of the training dataset of the ASR system developed at the SPSC by means of synthesised voices improves its performance.

7.1 Lessons learnt from Speech Synthesis

Merlin, a neural network-based speech synthesis toolkit was employed in this work for speech synthesis. Based on an average voice model including the speakers from the WASS corpus, a speaker adaptation process was carried out to show the quality of the AVM. It was shown that an AVM based on male and female speakers outperforms the AVM trained by exclusively using male and female speakers. Apart form that, 2 speakers were excluded from the AVM, as the speaker adaptation resulted in bad sound quality for 2 speakers.
Since the speaker adaptation showed promising results, the AVM including 17 speakers was used further for the speaker embedding-based speaker adaptation. The speaker embedding-based speaker adaptation included speaker information in form of vectors in the training process of the AVM, why the model learnt the mapping between speaker information and audio file. As a result, speaker-specific speech was generated based on the trained AVM by having access to the speaker information in addition to the linguistic features. This has the advantage that speaker embedding-based speaker adaptation requires only a single trained AVM. However, compared to the speech synthesised using the conventional speaker adaptation method, no noticeable difference was observed in speech quality.
Cooper et al. [20] investigated the role of speaker embeddings for speaker adaptation in zero-shot TTS. In their work, Tacotron was used to generate speech based on speaker embedding vectors, which were extracted from a speaker recognition system similar to this work (cf. Chapter 4). By applying different similarity metrics they showed, that speaker adaptation using speaker embeddings showed good results in terms of speaker similarity and naturalness of synthesised speech.

In order to obtain speaker-specific information in form of speaker embedding vectors from a given dataset, a speaker recognition system was used, that created the speaker embedding vectors from audio files. The speaker embedding extraction was performed by using the LDE approach, where speaker information was extracted after the learnable dictionary encoding layer.
This way of speaker extraction allowed the adjustment of different parameters. It was shown in Section 3.2.3 that especially the number of utterances and the length of the audio chunks used

for training as well as the dimension of the speaker embedding vectors influence the quality of the speaker embedding vectors. By choosing appropriate parameters, the speaker embedding vectors contained the most relevant information to describe the individual speakers.

The number of utterances used for the training process depends on the minimum required frame length for the training utterances. Since a high value for the minimum required frame length excludes utterances with a small frame length, it seemed to be advantageous to choose a small value to include as many utterances as possible. However, another parameter is important, namely the size of the chunks used for the training process. This value depends as well on the minimum required frame length, because the maximum possible chunk size is defined by the minimum required frame length. Previous work has shown that a larger chunk size leads to better results in terms of speaker recognition [26].
To investigate this trade-off, different combinations of minimum required frame length and chunk size interval were tested in various experiments. It was shown that a larger minimum required frame length and hence a decreased number of training utterances leads to performance degradation (cf. Table 3.5). Concerning the chunk size it was expected, that a smaller chunk size degrades performance as well. However this expectation was not confirmed, because smaller chunk sizes yielded acceptable results as shown in Table 3.6. But there was a limit to making the value arbitrarily small: at a chunk size below 200, the grouping into male and female speakers deteriorated, indicating that speaker information has been lost by reducing the chunk size (cf. Figure 3.13).
Additionally different dimensions of the speaker embedding vectors were investigated according to previous analysis in recent work and their influence on the performance of the speaker recognition system was analysed. The outcome of previous work can be confirmed to a certain extent: In [20] better results for smaller speaker embedding dimensions were achieved and in this work, similarly, better values were achieved for a dimension of 128 than for a dimension of 512. However decreasing the dimension to 32, the performance of the speaker recognition system was worse than with a dimension of 512 (cf. Table 3.8). The aforementioned observations led to a specific combination of parameters for speaker embedding extraction, which were used for speaker interpolation and furthermore for the generation of new voices.

For the generation of new speaker characteristics, the extracted speaker embedding vectors were interpolated according to Equation 3.4. The newly generated speaker embedding vectors were integrated in the speech synthesis process, i.e., speaker embedding-based speaker adaptation, to synthesise the utterances from the GRASS corpus with new voices. Therefore, an AVM trained on speaker embedding vectors was used, which took the interpolated vectors as input for the synthesis and generated the utterances from them. The synthesised utterances were analysed using the evaluation metrics (cf. Section 3.2.2), the results of which can be seen in Table 4.4 and Figures 4.5 - 4.8. Compared to the analysis of the original WASS corpus, it was shown that the synthesised corpus also has good separability for the base speakers, which is evident from the low values for EER and minDCF. Figures 4.6 - 4.8 show a transition from one base speaker to another for the interpolated speakers. This evaluation demonstrates the successful generation of new voices by means of interpolated speaker embedding vectors.
The comparison with previous work shows a similar outcome. Korostik et al. [52] applied linear interpolation between the learned speaker embeddings of a CNN network in order to create new voices. A subsequent objective and subjective evaluation led to good results in terms of speaker similarity.

7.2 Lessons learnt from ASR Experiments

With the help of the synthesised GRASS corpus, numerous experiments with the ASR system were carried out and the performance was evaluated.
For the baseline experiment the original training, development and test set of the ASR system were used, which consist of utterances from the original GRASS corpus (cf. Table 6.2). This experiment was used as a reference for further experiments and shows the good performance of the ASR system, as a WER below 1% was achieved for the triphone model.

In the first step, the influence of the training data augmentation on the ASR performance was investigated, in which the original training set was augmented with different amounts and compositions of the synthesised GRASS corpus (cf. Section 6.3). The training set consisted of utterances from the original GRASS corpus and the synthesised GRASS corpus, whereby the utterances from the synthesised GRASS corpus were chosen randomly or per speaker and the amount of added utterances varied. For development and testing the original GRASS corpus was used. The results (cf. Table 6.4) indicate that adding a small amount of synthesised utterances decreases performance (cf. experiments 1 and 2). If the amount of synthesised utterances was increased further (experiments 3 - 5), performance was increased compared to the baseline experiment. However if the amount of synthesised utterances dominated, performance degraded again. Additionally the performance of the ASR system in terms of SER and WER is depicted in Figure 6.2. This representation allows the comparison between the speaker-specific selection and the random selection. The rough evaluation showed that best results were achieved when the number of synthesised utterances was in the same order as the number of original utterances, i.e., the amount of synthesised utterances was about 50%, whereby a random selection mode showed slightly better results.
Generally it was shown in previous work from Fazel et al. and Rosenberg et al. [1, 2] that augmenting an existing corpus using synthesised voices improves ASR performance. This observation can be confirmed based on the results in this work. Section 6.3 shows that the WER and the SER of the triphone model were reduced when the amount of synthesised utterances added is in the range of the amount of the original utterances.

The assumption that keeping the ratio between synthesised and original utterances around 50% was investigated further in subsequent experiments (cf. Table 6.5). Indeed it was shown in experiments 4 and 13, that the WER and SER of the triphone model can be increased when adding utterances from the synthesised GRASS corpus. The WER was decreased from 0.88% to 0.64% and the SER from 3.54% to 2.36% when considering the triphone model. The results for the monophone model were improved as well to some extent.
Concerning the amount of synthesised utterances to be added to an original training corpus in order to improve ASR performance, Rosenberg et al. [2] used a fixed amount of synthesised speech and reduced the amount of utterances from the original corpus. It was shown that best results were achieved when the amount of synthesised and original utterances were equal (at 50%). However, as the proportion of original utterances decreased, the performance of the ASR system degraded, which is also consistent with the results of this work.

In Fazel et al. [1] the replacement from original utterances with synthesised utterances was examined and thus also tested in this work. For this purpose, also in this thesis, different amounts of speakers from the original training corpus were replaced by synthesised speakers, whereby the utterance ID remained the same. Since the results showed to be strongly dependent on the speaker combinations, multiple experiments were carried out (cf. Tables A.3 - A.9). The analysis of the experiments showed that the more original speakers were replaced by synthesised speakers, the higher the WER and SER was. However, for some experiments, performance was improved by the replacement (cf. experiment 19), but the trend shows a clear degradation by

replacing original speakers.
The results in [1] indicate that a replacement from original speech using synthesised speech degrades ASR performance. These results are only partly consistent with the results of this work (cf. Section 6.4). For a certain amount of replaced speakers, the WER and SER were improved, which was not the case in [1]. But it has to be mentioned here that the amount of synthesised utterances used by [1] was significantly higher than that of the original speech. If the amount of synthesised speech was increased considerably, performance also degraded in this work.

In the previous experiments different amounts of the original training set remained. Additional experiments were conducted using only the synthesised utterances from the GRASS corpus as training set. The development and test set consisted either of synthesised GRASS utterances or of the original GRASS utterances (cf. Table 6.7). When training the ASR system with utterances from the synthesised GRASS corpus and developing and testing the performance using the original GRASS corpus, the performance of the ASR system is really poor with a WER and SER around 100% for the triphone model (cf. experiments 25 and 26). Compared to the results in Rosenberg et al. [2], in which ASR performance was investigated when using only synthesised utterances, the WER was around 32% for clean test data.
Additionally the experiments were carried out using the synthesised GRASS corpus as development and test set (cf. experiments 27 - 31). The evaluation of the monophone and triphone model shows quite good results with a trend towards better results the more training data is used (cf. Figure 6.7). This is probably due to the higher acoustic and linguistic diversity resulting from the higher amount of data. However, the performance improvement saturated, meaning that adding more training data no longer reduced the WER. Compared to the experiment with the original GRASS corpus, the ASR performance was significantly better. An analysis of both experiment showed a mismatch between the original and the synthesised GRASS corpus. When the system was trained with the synthesised GRASS corpus but developed and tested using the original GRASS corpus, not only nonsilence phones were detected incorrectly but also silence phones, which indicated problems in the alignment process during training the ASR system, which were not analysed further in this work.

According to the previous experiments carried out using the GRASS corpus, the WASS corpus was used for ASR. One part of the experiments used the WASS corpus as training set and as development and test set. The other part used the WASS corpus for training but was developed and tested with the original GRASS corpus.
For training, three different compositions of training sets were investigated: one system was trained on the original WASS corpus, one on the synthesised WASS corpus and a third experiment used the combination of both for training (cf. Table 6.9). The evaluation based on the WASS corpus showed promising results when adding a part of the synthesised utterances to the original corpus, leading to an improvement of the WER and SER of the triphone model (cf. experiments 32 and 33). Training the ASR system exclusively with the synthesised WASS corpus (cf. experiment 34) degraded performance drastically. Again the results are similar to those obtained from the results in Rosenberg et al. [2]. Adding a fixed amount amount of synthesised speech to the original corpus improves performance. The WER when using only synthesised utterances are in the range of this work.
The evaluation using the original GRASS corpus however showed a clear mismatch between both corpora, which was smallest for the original WASS and GRASS corpus. The same trend as in the previous experiments was observed: adding synthesised utterances to the original WASS set during training, decreased the WER and SER for the triphone model remarkably (cf. experiments 35 and 36). Again, performance was worst when training the ASR system only with synthesised utterances (cf. experiment 37) and for these experiments the WER was significantly higher than the WER obtained in the work from Rosenberg et al. [2].

7.3 Outlook

Since this work deals exclusively with read speech, it is intended to give an overview of whether an acoustic augmentation of the training corpus by means of synthesised voices provides any improvement at all. However, since conversational speech in particular poses great challenges to the field of automatic speech recognition, experiments with conversational speech would also be interesting. A subsequent comparison with the results of this work is appreciated.

Furthermore, it would also be beneficial to use more state-of-the-art speech synthesis systems for the generation of new voices and utterances and to compare the results with those obtained from the FFDNN-based speech synthesis toolkit Merlin.
In addition, when interpolating the speaker embedding vectors, there is the possibility of using different interpolation factors α and investigating their influence on speech recognition. In particular, it would be interesting to see how a finer grid between the interpolated speakers, e.g., $\alpha = 0.1$, affects speech recognition.

As already shown, training the speech recognition system with only synthesised voices did not lead to good results in automatic speech recognition. The discrepancy between the training corpus of synthesised voices and the development and test set of original voices led to problems in speech recognition. For the future, it would be beneficial to address this issue, because training an ASR system with only synthesised voices achieving acceptable results significantly reduces the effort of training data acquisition.

8 Conclusion

The aim of this work was the augmentation of an existing training corpus for an ASR system by using synthesised speech. In the context of this work, the successful use of a FFDNN-based speech synthesis system with spectral feature and duration models was shown. Although numerous systems with more advanced methods already exist, the generation of an average voice model and the subsequent speaker adaptation using speaker embedding vectors works remarkably well with the FFDNN used by Merlin, since this method shows good speaker interpolation properties.
The speaker embedding vector extraction prior to the speech synthesis process was based on a speaker recognition system. It was shown that appropriate parameters for the speaker recognition system allow a good separation between individual speakers.
By interpolating the speaker embedding vectors from the original speakers, new voice characteristics were generated and by passing the resulting interpolated vectors to the speech synthesis system, natural sounding new voices were created. Based on the newly generated voices, the original GRASS corpus was synthesised in order to increase the acoustic diversity of the training corpus for the ASR system.
Due to the high number of new voices, experiments with different compositions of the training dataset were carried out. The results show that augmenting an original corpus with the same amount of synthesised utterances, performance of the ASR system was improved. However, increasing the amount of synthesised voices further, the WER and the SER of the ASR system deteriorated considerably. In further experiments, the influence on ASR performance was investigated when replacing the original speakers from the training set with synthesised voices, and it was found that the replacement worsens the ASR performance. If the ASR system was trained exclusively with synthesised voices, ASR becomes almost impossible.

A comparison with the results from previous work shows a similar outcome: augmenting an existing corpus with real speech with synthesised utterances leads to an improvement in ASR performance if both speech corpora are in the same order of magnitude. However, if the amount of synthesised speech dominates the amount of real speech, ASR performance degrades.

A
Appendix

A.1 Chapter 6

A.1.1 Section 6.3

Table A.1: Division of the utterances from the GRASS corpus into training, development and set and the successive augmentation of the training set using the synthesised GRASS corpus (s). Utterances were selected either per speaker (spk) or randomly (rand). The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	GRASS			
			utterances		speakers	
			total	different	m	f
1	train		3820	1248	17	16
	train(s)	spk (5)	2740	720	5	0
	dev		249	154	1	1
	test		254	162	1	1
2	train		3820	1248	17	16
	train(s)	rand	2740	700	393	164
	dev		249	154	1	1
	test		254	162	1	1
3	train		3820	1248	17	16
	train(s)	spk (7)	3836	755	4	3
	dev		249	154	1	1
	test		254	162	1	1
4	train		3820	1248	17	16
	train(s)	rand	3800	753	396	165
	dev		249	154	1	1
	test		254	162	1	1
5	train		3820	1248	17	16
	train(s)	spk (14)	7672	899	7	7
	dev		249	154	1	1
	test		254	162	1	1
6	train		3820	1248	17	16
	train(s)	rand	7672	890	396	165
	dev		249	154	1	1
	test		254	162	1	1
7	train		3820	1248	17	16
	train(s)	spk (21)	11508	974	18	3
	dev		249	154	1	1
	test		254	162	1	1
8	train		3820	1248	17	16
	train(s)	rand	11508	959	396	165
	dev		249	154	1	1
	test		254	162	1	1
9	train		3820	1248	17	16
	train(s)	spk (33)	18084	1080	22	11
	dev		249	154	1	1
	test		254	162	1	1
10	train		3820	1248	17	16
	train(s)	rand	18084	1064	396	165
	dev		249	154	1	1
	test		254	162	1	1

Table A.2: Division of the utterances from the GRASS corpus into training, development and set and the successive augmentation of the training set with synthesised speech (s). Utterances were selected randomly (rand). The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	GRASS			
			utterances		speakers	
			total	different	m	f
11	train		3820	1248	17	16
	train(s)	rand	1000	518	319	143
	dev		249	154	1	1
	test		254	162	1	1
12	train		3820	1248	17	16
	train(s)	rand	3100	740	392	165
	dev		249	154	1	1
	test		254	162	1	1
13	train		3820	1248	17	16
	train(s)	rand	3500	755	395	165
	dev		249	154	1	1
	test		254	162	1	1
14	train		3820	1248	17	16
	train(s)	rand	4000	756	396	165
	dev		249	154	1	1
	test		254	162	1	1
15	train		3820	1248	17	16
	train(s)	rand	5000	823	396	165
	dev		249	154	1	1
	test		254	162	1	1
16	train		3820	1248	17	16
	train(s)	rand	30000	1149	396	165
	dev		249	154	1	1
	test		254	162	1	1
17	train		3820	1248	17	16
	train(s)	rand	50000	811	395	165
	dev		249	154	1	1
	test		254	162	1	1

A.1.2 Section 6.4

Table A.3: Division of the utterances from the GRASS corpus into training, development and set and the replacement of 1 speaker with 1 speaker from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	GRASS				triphone model	
			utterances		speakers		WER in %	SER in %
			total	different	m	f		
18a	train	spk		1225	16	16	1.04	3.94
	train(s)			120	1	0		
	dev		249	154	1	1		
	test		254	162	1	1		
18b	train	spk		1229	17	15	0.88	3.54
	train(s)			108	1	0		
	dev		249	154	1	1		
	test		254	162	1	1		
18c	train	spk		1232	16	16	0.96	3.54
	train(s)			106	0	1		
	dev		249	154	1	1		
	test		254	162	1	1		
18d	train	spk		1237	16	16	0.88	3.54
	train(s)			120	1	0		
	dev		249	154	1	1		
	test		254	162	1	1		
18e	train	spk		1224	16	16	1.04	4.33
	train(s)			115	1	0		
	dev		249	154	1	1		
	test		254	162	1	1		

Table A.4: Division of the utterances from the GRASS corpus into training, development and set and the replacement of 5 speakers with 5 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	GRASS				triphone model	
			utterances		speakers		WER in %	SER in %
			total	different	m	f		
19a	train	spk		1153	13	15	0.80	3.15
	train(s)			380	4	1		
	dev		249	154	1	1		
	test		254	162	1	1		
19b	train	spk		1148	14	14	1.04	3.94
	train(s)			359	2	3		
	dev		249	154	1	1		
	test		254	162	1	1		
19c	train	spk		1164	13	15	0.72	3.15
	train(s)			475	4	1		
	dev		249	154	1	1		
	test		254	162	1	1		
19d	train	spk		1148	14	14	0.88	3.54
	train(s)			384	5	0		
	dev		249	154	1	1		
	test		254	162	1	1		
19e	train	spk		1180	15	13	0.80	2.76
	train(s)			339	4	1		
	dev		249	154	1	1		
	test		254	162	1	1		

Table A.5: Division of the utterances from the GRASS corpus into training, development and set and the replacement of 10 speakers with 10 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	GRASS				triphone model	
			utterances		speakers		WER in %	SER in %
			total	different	m	f		
20a	train	spk		1050	13	10	0.88	3.94
	train(s)			572	6	4		
	dev		249	154	1	1		
	test		254	162	1	1		
20b	train	spk		1070	12	11	0.80	3.15
	train(s)			664	7	3		
	dev		249	154	1	1		
	test		254	162	1	1		
20c	train	spk		1032	12	11	1.92	5.91
	train(s)			627	10	0		
	dev		249	154	1	1		
	test		254	162	1	1		
20d	train	spk		1042	13	10	1.20	5.12
	train(s)			605	7	3		
	dev		249	154	1	1		
	test		254	162	1	1		
20e	train	spk		1055	13	10	0.80	3.15
	train(s)			615	8	2		
	dev		249	154	1	1		
	test		254	162	1	1		

Table A.6: Division of the utterances from the GRASS corpus into training, development and set and the replacement of 15 speakers with 15 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	GRASS				triphone model	
			utterances		speakers		WER in %	SER in %
			total	different	m	f		
21a	train	spk		924	8	10	0.88	3.15
	train(s)			786	13	2		
	dev		249	154	1	1		
	test		254	162	1	1		
21b	train	spk		887	8	10	0.88	3.94
	train(s)			709	12	3		
	dev		249	154	1	1		
	test		254	162	1	1		
21c	train	spk		933	7	11	1.20	4.33
	train(s)			729	9	6		
	dev		249	154	1	1		
	test		254	162	1	1		
21d	train	spk		960	8	10	0.96	3.94
	train(s)			756	11	4		
	dev		249	154	1	1		
	test		254	162	1	1		
21e	train	spk		907	8	10	1.12	4.33
	train(s)			813	10	5		
	dev		249	154	1	1		
	test		254	162	1	1		

Table A.7: Division of the utterances from the GRASS corpus into training, development and set and the replacement of 20 speakers with 20 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	GRASS				triphone model	
			utterances		speakers		WER in %	SER in %
			total	different	m	f		
22a	train	spk		808	5	8	0.96	3.94
	train(s)			908	11	9		
	dev		249	154	1	1		
	test		254	162	1	1		
22b	train	spk		787	8	5	1.12	4.72
	train(s)			898	17	3		
	dev		249	154	1	1		
	test		254	162	1	1		
22c	train	spk		754	5	8	0.96	3.54
	train(s)			909	11	9		
	dev		249	154	1	1		
	test		254	162	1	1		
22d	train	spk		723	6	7	1.52	5.12
	train(s)			886	17	3		
	dev		249	154	1	1		
	test		254	162	1	1		

Table A.8: Division of the utterances from the GRASS corpus into training, development and set and the replacement of 25 speakers with 25 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	GRASS				triphone model	
			utterances		speakers		WER in %	SER in %
			total	different	m	f		
23a	train	spk		540	3	5	0.88	3.54
	train(s)			975	18	7		
	dev		249	154	1	1		
	test		254	162	1	1		
23b	train	spk		508	5	3	1.60	4.72
	train(s)			955	22	3		
	dev		249	154	1	1		
	test		254	162	1	1		
23c	train	spk		508	5	3	2.08	6.69
	train(s)			953	17	8		
	dev		249	154	1	1		
	test		254	162	1	1		

Table A.9: Division of the utterances from the GRASS corpus into training, development and set and the replacement of 30 speakers with 30 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	GRASS				triphone model	
			utterances		speakers		WER in %	SER in %
			total	different	m	f		
24a	train	spk		293	2	1	5.77	13.78
	train(s)			1001	19	11		
	dev		249	154	1	1		
	test		254	162	1	1		
24b	train	spk		309	1	2	5.45	17.72
	train(s)			1019	20	10		
	dev		249	154	1	1		
	test		254	162	1	1		

A.1.3 Section 6.5

Table A.10: Division of the utterances from the synthesised GRASS corpus and the original GRASS corpus into train, development and test set. Utterances were selected randomly (rand). The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	GRASS			
			utterances		speakers	
			total	different	m	f
25	train(s)	rand	5000	798	396	165
	dev		249	154	1	1
	test		254	162	1	1
26	train(s)	rand	20000	1087	396	165
	dev		249	154	1	1
	test		254	162	1	1
27	train(s)	rand	1000	507	332	136
	dev(s)		312	184	1	1
	test(s)		312	190	1	1
28	train(s)	rand	5000	798	396	165
	dev(s)		312	184	1	1
	test(s)		312	190	1	1
29	train(s)	rand	20000	1087	396	165
	dev(s)		312	184	1	1
	test(s)		312	190	1	1
30	train(s)	rand	40000	1206	396	165
	dev(s)		312	184	1	1
	test(s)		312	190	1	1
31	train(s)	rand	60000	1232	396	165
	dev(s)		312	184	1	1
	test(s)		312	190	1	1

A.1.4 Section 6.6

Table A.11: Division of the utterances from the original WASS corpus into training, development and test set and the successive augmentation of the training set using the synthesised WASS corpus (s). Utterances were selected randomly (rand). The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	WASS			
			utterances		speakers	
			total	different	m	f
32	train	rand	6955	4328	10	3
	dev		446	223	1	1
	test		446	223	1	1
33	train	rand	6955	4328	10	3
	train(s)	rand	6955	4328	396	165
	dev		446	223	1	1
	test		446	223	1	1
34	train(s)	rand	6955	4328	396	165
	dev		446	223	1	1
	test		446	223	1	1

Table A.12: Division of the utterances from the original WASS corpus into training, development and test set and the successive augmentation of the training set using the synthesised WASS corpus (s). Utterances were selected randomly (rand). The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers.

id	set	selection	WASS				GRASS			
			utterances		speakers		utterances		speakers	
			total	different	m	f	total	different	m	f
35	train	rand	6955	4328	10	3	-	-	-	-
	dev		-	-	-	-	249	154	1	1
	test		-	-	-	-	254	127	1	1
36	train	rand	6955	4328	10	3	-	-	-	-
	train(s)	rand	6955	4328	396	165	-	-	-	-
	dev		-	-	-	-	249	154	1	1
	test		-	-	-	-	254	127	1	1
37	train(s)	rand	6955	4328	10	3	-	-	-	-
	dev		-	-	-	-	249	154	1	1
	test		-	-	-	-	254	127	1	1

Bibliography

[1] Amin Fazel, Wei Yang, Yulan Liu, Roberto Barra-Chicote, Yixiong Meng, Roland Maas, and Jasha Droppo. "SynthASR: Unlocking Synthetic Data for Speech Recognition." June 14, 2021. arXiv: 2106.07803v1 [cs.LG].

[2] Andrew Rosenberg, Yu Zhang, Bhuvana Ramabhadran, Ye Jia, Pedro Moreno, Yonghui Wu, and Zelin Wu. "Speech Recognition with Augmented Synthesized Speech." Dec. 2019, pp. 996–1002. DOI: 10.1109/ASRU46091.2019.9003990.

[3] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ Skerry-Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. *Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions*. 2017. DOI: 10.48550/ARXIV.1712.05884. URL: https://arxiv.org/abs/1712.05884.

[4] Zhizheng Wu, Oliver Watts, and Simon King. "Merlin: An Open Source Neural Network Speech Synthesis System." Sept. 2016, pp. 202–207. DOI: 10.21437/SSW.2016-33.

[5] Xu Tan, Tao Qin, Frank Soong, and Tie-Yan Liu. "A Survey on Neural Speech Synthesis." June 2021. arXiv: 2106.15561 [eess.AS].

[6] Carina Lozo, Jan Luttenberger, and Michael Pucher. "The thought collective behind thirty years of progress in speech synthesis." *3rd Workshop of History of Speech Communication Research* (2019).

[7] Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Z. Chen, Samy Bengio, Quoc V. Le, Yannis Agiomyrgiannakis, Robert A. J. Clark, and Rif A. Saurous. "Tacotron: Towards End-to-End Speech Synthesis." *INTERSPEECH*. 2017. DOI: 10.21437/INTERSPEECH.2017-1452.

[8] Heiga Zen. *An example of context-dependent label format for HMM-based speech synthesis in English*. Mar. 2006. URL: https://wiki.inf.ed.ac.uk/twiki/pub/CSTR/F0parametrisation/hts_lab_format.pdf (visited on 01/25/2022).

[9] Srikanth Ronanki, Zhizheng Wu, Oliver Watts, and Simon King. "A Demonstration of the Merlin Open Source Neural Network Speech Synthesis System." *SSW*. 2016.

[10] Michael Pucher, Markus Toman, Dietmar Schabus, Cassia Valentini-Botinhao, Junichi Yamagishi, Bettina Zillinger, and Erich Schmid. "Influence of speaker familiarity on blind and visually impaired children's perception of synthetic voices in audio games." Sept. 2015. DOI: 10.21437/Interspeech.2015-376. URL: https://sociolectix.org/papers/2015_is_voiceperc.pdf.

[11] Markus Toman and Michael Pucher. "An Open Source Speech Synthesis Frontend for HTS." *Proceedings of the 18th International Conference on Text, Speech, and Dialogue - Volume 9302*. 2015, pp. 291–298. DOI: 10.1007/978-3-319-24033-6_33. URL: https://sociolectix.org/papers/2015_tsd_salb.pdf.

[12] Michael Pucher, Michaela Rausch-Supola, Sylvia Moosmüller, Markus Toman, Dietmar Schabus, and Friedrich Neubarth. "Open data for speech synthesis of Austrian German language varieties." *12. Tagung Phonetik und Phonologie im deutschsprachigen Raum*. 2016, pp. 147–150. URL: https://sociolectix.org/papers/opendata_pp2016.pdf.

[13] Shan Yang, Zhizheng Wu, and Lei Xie. "On the training of DNN-based average voice model for speech synthesis." *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*. 2016, pp. 1–6. DOI: 10.1109/APSIPA.2016.7820818.

[14] The Centre for Speech Technology Research (CSTR) Edinburgh. *Speaker adaptation*. 2017. URL: https://github.com/CSTR-Edinburgh/merlin/tree/master/egs/speaker_adaptation/s1 (visited on 12/10/2021).

[15] Weicheng Cai, Jinkun Chen, and Ming Li. "Exploring the Encoding Layer and Loss Function in End-to-End Speaker and Language Recognition System." Apr. 2018. DOI: 10.21437/Odyssey.2018-11.

[16] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer. *End-to-End Text-Dependent Speaker Verification*. 2015. DOI: 10.48550/ARXIV.1509.08062. URL: https://arxiv.org/abs/1509.08062.

[17] Finnian Kelly, Anil Alexander, Oscar Forth, and David van der Vloed. "From i-vectors to x-vectors –a generational change in speaker recognition illustrated on the NFI-FRIDA database." *IAFPA conference, Istanbul*. July 2019.

[18] Đorđe Grozdić, Slobodan Jovičić, Zoran Saric, and Irina Subotić. "Comparison of GMM/UBM and i-vector based speaker recognition systems." *International Conference on Fundamental and Applied Aspects of Speech and Language, Belgrade*. Oct. 2015.

[19] William Campbell, Douglas Sturim, Douglas Reynolds, and A. Solomonoff. "SVM Based Speaker Verification using a GMM Supervector Kernel and NAP Variability Compensation." *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing, 2006*. Vol. 1. June 2006, pp. I–I. DOI: 10.1109/ICASSP.2006.1659966.

[20] Erica Cooper, Cheng-I Lai, Yusuke Yasuda, Fuming Fang, Xin Wang, Nanxin Chen, and Junichi Yamagishi. "Zero-Shot Multi-Speaker Text-To-Speech with State-of-the-art Neural Speaker Embeddings." *2020 IEEE International Conference on Acoustics, Speech and Signal Processing*. Oct. 23, 2019. DOI: 10.1109/ICASSP40776.2020.9054535. arXiv: 1910.10838v2 [eess.AS].

[21] Koji Okabe, Takafumi Koshinaka, and Koichi Shinoda. "Attentive Statistics Pooling for Deep Speaker Embedding." *Interspeech 2018*. Mar. 2018. DOI: 10.21437/Interspeech.2018-993. arXiv: 1803.10963 [eess.AS].

[22] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Leibny Paola García-Perera, Fred Richardson, Réda Dehak, et al. "State-of-the-art speaker recognition with neural network embeddings in nist sre18 and speakers in the wild evaluations." *Computer Speech & Language* (2019), p. 101026. DOI: 10.1016/j.csl.2019.101026.

[23] Daniel Povey. *Kaldi Data Preparation*. Aug. 2015. URL: https://kaldi-asr.org/doc/data_prep.html (visited on 06/03/2022).

[24] David Van Leeuwen and Niko Brummer. "An Introduction to Application-Independent Evaluation of Speaker Recognition Systems." *Speaker Classification I: Fundamentals, Features, and Methods*. Vol. 4343. Jan. 2007, pp. 330–353. ISBN: 978-3-540-74186-2. DOI: 10.1007/978-3-540-74200-5_19.

[25] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE." *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[26] Nanxin Chen, Jesús Villalba, and Najim Dehak. "Tied Mixture of Factor Analyzers Layer to Combine Frame Level Representations in Neural Speaker Embeddings." *Interspeech 2019*. Sept. 2019, pp. 2948–2952. DOI: 10.21437/Interspeech.2019-1782.

[27] Signal Processing and Speech Communication Laboratory, Graz University of Technology; Inffeldgasse 16c/EG; A-8010 Graz. *GRASS: the Graz corpus of Read And Spontaneous Speech*. URL: https://www.spsc.tugraz.at/databases-and-tools/grass-the-graz-corpus-of-read-and-spontaneous-speech.html (visited on 07/14/2022).

[28] The Centre for Speech Technology Research (CSTR). *The Festival Speech Synthesis System*. June 2022. URL: https://www.cstr.ed.ac.uk/projects/festival/ (visited on 07/02/2022).

[29] SourceForge Headquarters. *Austrian German Voices for Festival*. Aug. 2015. URL: https://sourceforge.net/projects/at-festival/ (visited on 03/11/2022).

[30] Computer Science Department. *Creating Label Files for Training Data*. June 2017. URL: http://www.cs.columbia.edu/~ecooper/tts/labels.html (visited on 03/11/2022).

[31] Clare-Marie Karat, Nicole Yankelovich, and Jennifer Lai. "Conversational Speech Interfaces and Technologies." *The Human-Computer Interaction Handbook*. 2007. DOI: 10.1201/9781410615862.CH19.

[32] Cristian Tejedor-García. "Design and Evaluation of Mobile Computer-Assisted Pronunciation Training Tools for Second Language Learning." PhD thesis. Sept. 2020. DOI: 10.35376/10324/43663.

[33] John Levis and Ruslan Suvorov. "Automatic Speech Recognition." *Encyclopedia of Applied Linguistics*. Nov. 2012. DOI: 10.1002/9781405198431.wbeal0066.

[34] William Song and Jim Cai. "End-to-end deep neural network for automatic speech recognition." *Standford CS224D Reports* (2015).

[35] Frederick Jelinek. *Statistical Methods for Speech Recognition*. Cambridge, MA, USA: MIT Press, 1998. ISBN: 0262100665.

[36] Jonathan Hui. *Speech Recognition — GMM, HMM*. Jan. 2021. URL: https://jonathan-hui.medium.com/speech-recognition-gmm-hmm-8bb5eff8b196 (visited on 04/20/2022).

[37] Jonathan Hui. *Speech Recognition — Acoustic, Lexicon & Language Model*. Nov. 2020. URL: https://jonathan-hui.medium.com/speech-recognition-acoustic-lexicon-language-model-aacac0462639 (visited on 04/20/2022).

[38] William Goldenthal and James Glass. "Statistical trajectory models for phonetic recognition." *ICSLP*. Jan. 1994.

[39] Song Wang and Guanyu Li. "Overview of end-to-end speech recognition." *Journal of Physics: Conference Series* 1187.5 (Apr. 2019), p. 052068. DOI: 10.1088/1742-6596/1187/5/052068. URL: https://doi.org/10.1088/1742-6596/1187/5/052068.

[40] Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups." *IEEE Signal Processing Magazine* 29.6 (2012), pp. 82–97. DOI: 10.1109/MSP.2012.2205597.

[41] Jinyu Li. *Recent Advances in End-to-End Automatic Speech Recognition*. 2021. DOI: 10.48550/ARXIV.2111.01690. URL: https://arxiv.org/abs/2111.01690.

[42] Julian Linke. *ASR Script*. unpublished. 2020.

[43] Daniel Povey. *Kaldi*. July 2015. URL: https://kaldi-asr.org/doc (visited on 07/15/2022).

[44] Tobias Schrank and Saskia Wepner. "Speech Communication Laboratory, Automatic Speech Recognition." Signal Processing and Speech Communication Laboratory, Graz University of Technology. 2021.

[45] Florian Schiel. *The Munich Automatic Segmentation System MAUS*. Apr. 2016. URL: https://www.bas.uni-muenchen.de/Bas/BasMAUS.html (visited on 07/25/2022).

[46] Institute of Phonetics and Speech Processing. *BAS Web Services*. Apr. 2020. URL: https://clarin.phonetik.uni-muenchen.de/BASWebServices/interface (visited on 06/20/2022).

[47] Daniel Jurafsky and James H. Martin. *Draft: Speech and Language Processing - N-gram Language Models*. Dec. 2021. URL: https://web.stanford.edu/~jurafsky/slp3/3.pdf (visited on 07/25/2022).

[48] SRI International's Speech Technology and Research Laboratory. *Ngram-Count*. May 2008. URL: http://www.speech.sri.com/projects/srilm/manpages/ngram-count.1.html (visited on 05/25/2022).

[49] Daniel Jurafsky and James H. Martin. *Draft: Speech and Language Processing - Hidden Markov Models*. Dec. 2021. URL: https://web.stanford.edu/~jurafsky/slp3/A.pdf (visited on 07/25/2022).

[50] Eleanor Chodroff. *Kaldi Tutorial*. July 2015. URL: https://www.eleanorchodroff.com/tutorial/kaldi/index.html (visited on 08/10/2022).

[51] Ahmed Ali and Steve Renals. "Word Error Rate Estimation for Speech Recognition: e-WER." *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 20–24. DOI: 10.18653/v1/P18-2004. URL: https://aclanthology.org/P18-2004.

[52] Roman Korostik, Javier Latorre, Sivanand Achanta, and Yannis Stylianou. "Assessing Speaker Interpolation in Neural Text-to-Speech." *Speech and Computer*. Ed. by Alexey Karpov and Rodmonga Potapova. Springer International Publishing, 2021, pp. 360–371. ISBN: 978-3-030-87802-3.

List of Figures

1.1 Process of speaker interpolation-based data augmentation: Extraction of speaker embedding vectors containing speaker-specific information from an end-to-end speaker recognition system based on the audio files of the WASS corpus. The speaker embedding vectors are interpolated and used as additional input to the linguistic features for the speech synthesis system, in order to generate new voices for augmenting the GRASS corpus. The augmented corpus is used as training set for the ASR system, whose performance is evaluated afterwards. 15

2.1 Components of a conventional TTS synthesis system to convert a given text into a speech signal including text normalisation, grapheme-to-phoneme conversion, acoustic modeling and waveform generation [5]. 17
2.2 Distribution of speech synthesis concepts since 1990 [6]. 18
2.3 Model training and speech synthesis based on Merlin: During training the model learns the relation between linguistic and acoustic features (top). The linguistic feature vectors are used as input for the trained model (duration model and acoustic model) to predict acoustic output features which are passed to a vocoder to generate the speech waveform (bottom). 19
2.4 Pre-processing of linguistic features: vectorised and normalised HTS-style formatted labels used as input feature vectors for the FFDNN (duration model and acoustic model). 19
2.5 Normalisation of acoustic features: extraction of acoustic features by using a vocoder, which are normalised (mean-variance normalisation) and given as output features to the FFDNN during training (top). At the synthesis step the trained FFDNN predicts acoustic features which are normalised and given to a vocoder in order to generate the corresponding waveform (bottom). 20
2.6 A feed forward deep neural network with four hidden layers to predict vocoder parameters from given linguistic input features [4]. 21
2.7 Separate training of duration model (DM) and acoustic model (AM): the DM takes phone alignments as input in order to learn how to predict phone-level duration (left), likewise the AM receives the label files as input as well as the acoustic output features to learn their relationship (right). 21
2.8 The trained duration model takes labels as input to predict duration, which is further used as input to the trained acoustic model in addition to the labels in order to produce the corresponding acoustic features required for the speech waveform generation by means of a vocoder. 23
2.9 Speaker adaptation based on Merlin including the training of the AVM by using multi-speaker information, the adaptation of the speaker-specific model and the subsequent speech synthesis. 25

3.1 Timeline of automatic speaker recognition: Conventional speaker recognition is based on GMMs and their adaptations (e.g., GMM-UBM, i-vector approach). More state-of-the-art encoding techniques employ neural networks for robust speaker recognition using x-vectors [17]. 28
3.2 Almost end-to-end neural network speaker recognition system consisting of an encoding network, a statistical pooling layer and a classifier to identify a speaker based on a given speech segment [20]. 29
3.3 DNN for extracting an utterance-level speaker representation (speaker embedding) between the statistical pooling layer and the classifier [21]. 29

3.4 LDE pooling layer: The variable-length input $x = \{x_1, x_2, ..., x_L\}$ is related to the learnable dictionary components $\mu = \{\mu_1, ..., \mu_C\}$ by the residual $r_{tc} = x_t - u_c$ and the assigned weight w_{tc}. The encoding layer applies an aggregation operation in order to obtain the desired fixed-dimensional encoder output $E = \{e_1, ..., e_C\}$ [15]. 30
3.5 Utterance-level speaker embedding extraction from an end-to-end neural network speaker recognition system. The resulting speaker embedding is used for speaker verification (i.e., the similarity between a speech sample and a speaker ID) and the calculated speaker ID for speaker identification. 31
3.6 Acoustic pre-processing prior to training the speaker recognition system. 31
3.7 Post processing of the calculated utterance-level speaker embedding vectors in order to obtain pairwise scores based on which the speaker verification is carried out. A subsequent evaluation shows the performance of the utterance embedding vectors. 34
3.8 Distribution of frame lengths of the train utterances from the WASS corpus. . . . 37
3.9 Number of training utterances depending on the minimum required frame length. 37
3.10 Influence of the number of training utterances on the speaker recognition performance: By reducing the number of utterances with at least a length of 300 frames from 1776 training utterances (top) to 776 training utterances (bottom), performance of the embedding-based speaker separation degrades. 39
3.11 Influence of the chunk size of the training utterances on the speaker recognition performance: A shorter chunk size (200 frames, top) achieves better speaker separation and distinction between male and female speakers compared to a longer chunk size (500 frames, bottom). The minimum required frame length is set to 500 (709 training utterances) and speaker embedding dimension is 128 in both cases. 41
3.12 Influence of the speaker embedding dimension on the speaker recognition performance: Increasing the speaker embedding dimension from 32 to 128 improves performance whereas an increase from 128 to 512 results in degradation of performance for same minimum frame length (500) and chunk size interval ([200, 200]). 44
3.13 t-SNE for 19 speakers from the WASS corpus with the following parameters: minimum frame length: 200; chunk size interval: [150, 200]; embedding dimension: 128; LDA dimension: 100. 45

4.1 Concatenation of the linguistic feature vector and the speaker embedding vector, resulting in the input vector for the speech synthesis system. 49
4.2 Comparison between the conventional speaker adaptation method including the training of the AVM, the adaptation of the speaker-specific model and the subsequent speech synthesis (left) and the speaker embedding-based speaker adaptation that allows speaker-specific speech synthesis by using only a single AVM trained with speaker embedding vectors (right). 50
4.3 Generation of speaker-specific speech by passing the interpolated speaker embedding vectors in addition to the linguistic features to the previously trained AVM. A vocoder takes the obtained acoustic features as input and generates the new voice. 53
4.4 Distribution of frame lengths of the train utterances for different speaker combinations. 54
4.5 t-SNE for 17 base speakers representing the original speakers from the WASS corpus with the following parameters: minimum frame length: 200; chunk size interval: [150, 200]; embedding dimension: 128; LDA dimension: 100. 56

4.6 t-SNE for speaker combination bscwke with the following parameters: minimum frame length: 200; chunk size interval: [150, 200]; embedding dimension: 128; LDA dimension: 100. 56
4.7 t-SNE for speaker combination keplsc with the following parameters: minimum frame length: 150; chunk size interval: [130, 150]; embedding dimension: 128; LDA dimension: 100. 57
4.8 t-SNE for speaker combination fwaspo with the following parameters: minimum frame length: 200; chunk size interval: [150, 200]; embedding dimension: 128; LDA dimension: 100. 57

5.1 Conceptual ASR system: The incoming speech waveform is converted into a sequence of text hypotheses by the ASR system [32]. 59
5.2 Conventional ASR system including a preprocesssing step and three main components (lexicon, acoustic model and language model) [38]. 60
5.3 An illustration of the proceeding showing the individual steps during the ASR training and evaluation. It includes the basic elements and pre-processing step that are necessary for the training and evaluation process as well as the main components, namely the language model, the acoustic model and the lexicon [44]. 61

6.1 Composition of the training, development and test set in the experiments with the augmented GRASS training corpus, with development and test set from the original GRASS corpus. 69
6.2 Performance of the ASR system depending on the amount of utterances from the synthesised GRASS corpus, with the number of utterances from the original GRASS corpus remaining constant. The development and test set are based on the original GRASS corpus. The dashed line represents the result of the baseline experiment. 71
6.3 Performance of the ASR system depending on the amount of utterances from the synthesised GRASS corpus, with the number of utterances from the original GRASS corpus remaining constant. The development and test set are based on the original GRASS corpus. The dashed line represents the result of the baseline experiment. 74
6.4 Performance of the ASR system depending on the number of speakers replaced by speakers from the synthesised GRASS corpus, with the number and ID of utterances remaining constant. The development and test set are based on the original GRASS corpus. The dashed line represents the result of the baseline experiment. 77
6.5 Composition of the training, development and test set in the experiments with the synthesised GRASS training set, with development and test set from the original GRASS corpus. 78
6.6 Composition of the training, development and test set in the experiments with the synthesised GRASS training set, with development and test set from the synthesised GRASS corpus. 78
6.7 Performance of the ASR system depending on the amount of utterances from the synthesised GRASS corpus. The development and test set are based on the synthesised GRASS corpus. 80
6.8 Composition of the training, development and test set in the experiments with the WASS corpus. The training set consisted either of the original WASS corpus, the synthesised WASS corpus or a composition of both with development and test set from the original WASS corpus. 81

6.9 Composition of the training, development and test set in the experiments with the WASS corpus. The training set consisted either of the original WASS corpus, the synthesised WASS corpus or a composition of both with development and test set from the original GRASS corpus. 82
6.10 Performance of the ASR system depending on the amount of utterances from the synthesised WASS corpus. The development and test set are based on the original WASS corpus. 84
6.11 Performance of the ASR system depending on the amount of utterances from the synthesised WASS corpus. The development and test set are based on the original GRASS corpus. 85

List of Tables

2.1 Settings regarding the network architecture as stored in the configuration file for the duration model and the acoustic model. . . . 22
2.2 Settings regarding the dimension of the output features as stored in the configuration file for the duration model and the acoustic model. . . . 22
2.3 Settings regarding the waveform as stored in the configuration file for the duration model and the acoustic model. . . . 22
2.4 Proportion of the 8293 utterances per speaker in the WASS corpus and additional information. . . . 24

3.1 Settings for the acoustic pre-processing regarding the minimum frame length of the utterances used for the training process. . . . 32
3.2 Settings for the training process of the speaker recognition system. . . . 33
3.3 Settings for the minDCF evaluation in terms of the estimated costs of misses and false alarms (C_{miss} and C_{FA}) and the prior target probability P_{tar}. . . . 36
3.4 Influence of the minimum required frame length (and the resulting number of training utterances), the chunk size interval of the training utterances and the speaker embedding dimension on the speaker recognition process. . . . 36
3.5 Influence of the reduction of the training utterances on the EER and the minDCF while keeping the other parameters constant. . . . 38
3.6 Comparison of different chunk sizes and their influence on the EER and the minDCF when keeping the minimum required frame length and speaker embedding dimension constant. . . . 40
3.7 Speaker embedding dimensions already used in the literature. . . . 42
3.8 Influence of the speaker embedding dimension on the speaker recognition performance (EER and minDCF) by keeping the minimum frame length and chunk size interval constant and varying only the speaker embedding dimension {32, 128, 512}. . . . 42
3.9 Experimentally derived choice of parameters for the speaker embedding extraction. 45
3.10 Assignment of the original speaker ID to the respective synthesised speaker ID. . 46

4.1 Separation of the utterances from the read speech part of the GRASS corpus into training, evaluation and test set. The values in brackets show the number of different utterances in each set. . . . 51
4.2 Base speaker combinations (bscwke, keplsc and fwaspo) and their interpolations used to evaluate the interpolated speaker embeddings. . . . 53
4.3 Parameter setting for the speaker embedding extraction depending on the speaker combination. . . . 55
4.4 Performance of the speaker recognition system for different speaker combinations. 55

6.1 Experimental setup of the general parameters, including the settings for the language model and the acoustic model (monophone and triphone model). . . . 67
6.2 Division of the utterances from the GRASS corpus into training, development and test set: The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. . . . 68
6.3 Performance of the ASR system in terms of the WER and the SER based on the original training, development and test set of the GRASS corpus. The results of the monophone model and the triphone model are shown separately. . . . 68

6.4 Performance of the ASR system in terms of the WER and the SER based on the training set of the GRASS corpus augmented by using utterances from the synthesised GRASS corpus (s). The results of the monophone model and the triphone model are shown separately. 70
6.5 Performance of the ASR system in terms of the WER and the SER based on the training set of the GRASS corpus augmented by using utterances from the synthesised GRASS corpus (s). The results of the monophone model and the triphone model are shown separately. 73
6.6 Performance of the ASR system in terms of the WER and the SER based on the training set of the GRASS corpus where a number of speakers from the original GRASS corpus were replaced by speakers from the synthesised GRASS corpus (s). The results of the monophone model and the triphone model are shown separately. 76
6.7 Performance of the ASR system in terms of the WER and the SER based on the training set with utterances from the synthesised GRASS corpus (s). For development and testing, both corpora were used. The results of the monophone model and the triphone model are shown separately. 79
6.8 Division of the WASS corpus into training, development and test set with the corresponding number of utterances and duration in seconds. 81
6.9 Performance of the ASR system in terms of the WER and the SER based on the training set of the WASS corpus augmented by using utterances from the synthesised WASS corpus (s). For development and testing, the WASS corpus and the GRASS corpus are used. The results of the monophone model and the triphone model are shown separately. 83

A.1 Division of the utterances from the GRASS corpus into training, development and set and the successive augmentation of the training set using the synthesised GRASS corpus (s). Utterances were selected either per speaker (spk) or randomly (rand). The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 96
A.2 Division of the utterances from the GRASS corpus into training, development and set and the successive augmentation of the training set with synthesised speech (s). Utterances were selected randomly (rand). The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 97
A.3 Division of the utterances from the GRASS corpus into training, development and set and the replacement of 1 speaker with 1 speaker from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 98
A.4 Division of the utterances from the GRASS corpus into training, development and set and the replacement of 5 speakers with 5 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 99
A.5 Division of the utterances from the GRASS corpus into training, development and set and the replacement of 10 speakers with 10 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 100
A.6 Division of the utterances from the GRASS corpus into training, development and set and the replacement of 15 speakers with 15 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 101

A.7 Division of the utterances from the GRASS corpus into training, development and set and the replacement of 20 speakers with 20 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 102
A.8 Division of the utterances from the GRASS corpus into training, development and set and the replacement of 25 speakers with 25 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 102
A.9 Division of the utterances from the GRASS corpus into training, development and set and the replacement of 30 speakers with 30 speakers from the synthesised GRASS corpus. The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 103
A.10 Division of the utterances from the synthesised GRASS corpus and the original GRASS corpus into train, development and test set. Utterances were selected randomly (rand). The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 104
A.11 Division of the utterances from the original WASS corpus into training, development and test set and the successive augmentation of the training set using the synthesised WASS corpus (s). Utterances were selected randomly (rand). The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 105
A.12 Division of the utterances from the original WASS corpus into training, development and test set and the successive augmentation of the training set using the synthesised WASS corpus (s). Utterances were selected randomly (rand). The total number of utterances as well as the number of different utterances per set is shown, separately for female and male speakers. 105