Single-Channel Speech Enhancement using Deep Learning

Master Thesis

Gabriel Hülser

Supervisors: Prof. Alois Sontacchi (IEM), Dr. Gerald Bauer (Harman)

Straubing, March 25, 2018

Abstract

The aim of this work is to implement a single channel speech enhancement algorithm utilizing machine learning techniques, in particular deep neural networks (DNNs).
A large set of speech and noise data is collected to train a neural network model, which predicts time-frequency masks from noisy speech signals. The algorithm is tested using various additive noise sources and its performance is evaluated in terms of speech quality and intelligibility. Furthermore, the results are compared to those of a state of the art noise reduction system provided by *HARMAN*.
By using bidirectional long short-term memory (BLSTM) and a frequency weighted loss function, an average improvement of up to 0.3 PESQ and 0.06 STOI compared to the baseline algorithm is achieved. Moreover, a speech recognition benchmark showed an improvement of 8% in terms of speech accuracy.

Zusammenfassung

Das Ziel dieser Arbeit ist die Entwicklung eines Algorithmus zur Reduktion von Störgeräuschen einkanaliger Sprachsignale basierend auf Methoden des maschinellen Lernens, insbesondere neuronale Netze und Deep Learning.
Eine große Anzahl an Sprach- und Geräuschdaten wird gesammelt um ein neuronales Netzwerk zur Schätzung von spektralen Masken zu trainieren. Der Algorithmus wird für verschiedenste Arten additiver Störgeräuschen getestet und im Hinblick auf Sprachqualität und Sprachverständlichkeit evaluiert. Zusätzlich werden die Ergebnisse mit einem bestehenden, von *HARMAN* zur Verfügung gestellten Algorithmus verglichen.
Mithilfe eines bidirectional long short-term memory (BLSTM) Netzwerkes und einer frequenzabhängigen Fehlerfunktion konnte der PESQ um durchschnittlich 0.3 und der STOI um 0.06, im Vergleich zum ursprünglichen Verfahren, verbessert werden. Mithilfe eines vorgegebenen Testverfahrens wird ausserdem gezeigt, dass sich die Erkennungsrate eines Spracherkenners um bis zu 8% verbessern lässt.

Statutory declaration

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 marked all material which has been quoted either literally or by content from the used sources.

Gabriel Hülser
Graz, March 25, 2018

Contents

1 Introduction

Speech enhancement has been an important research topic in the field of signal processing for the last few decades. It comprises a group of methods that aims to improve the quality and intelligibility of speech signals. While in the early days of digital speech communication research, the field of applications was dominated by military purposes, the emergence of mobile communication at the end of the last century introduced it into the every day life of many people. With the rise of mobile phones, telecommunication systems had to cope with a variety of noise scenarios since phones were now extensively used in situations such as automobile environments, public transportation, restaurants or on busy streets, as opposed to a landline were the interfering noise is typically less severe. While the suppression of unwanted background noise may be the most evident goal of speech enhancement, it also includes tasks such as dereverberation, bandwidth extension or packet loss concealment. However, this thesis is focused on the reduction of additive noise and more precisely the application in smart speakers, wireless loudspeakers with an integrated virtual assistant offering handsfree interaction via voice commands. Speech enhancement for the use in smart speakers poses a special challenge since the dialogue system is expected to function properly independent from the placement of the device in relation to the location of the user. Since conventional algorithms are reaching their limits when it comes to severe noise conditions and unstationary noise types such as babble noise, the goal of this thesis is to make use of the recent advantages in the field of machine learning and in particular deep learning and apply them to the problem of speech enhancement and noise reduction.
The thesis is structured as follows. Chapter 2 gives a brief introduction to the key concepts of artificial neural networks and deep learning. In chapter 3 basic signal processing techniques for the spectral representation of speech signals are explained followed by an overview of objective evaluation criteria used to assess the performance of speech enhancement algorithms in chapter 4. Chapter 5 describes different time-frequency masks as used in the field of computational auditory scene analysis (CASA) and their performance in terms of noise reduction is evaluated and compared. The proposed speech enhancement framework and the experimental setup is described in chapter 6 followed by a detailed description of the experimental results in chapter 7. Finally chapter 8 concludes the thesis by summarizing the results as well as prospects for future potential and limits of the proposed speech enhancement system.

1.1 Problem Formulation

The principal goal of speech enhancement or simply noise reduction is to improve perceived quality and intelligibility of noisy speech signals. If solely monaural signals are considered, the problem can be more precisely defined as **Single-Channel Speech Enhancement**. In this context, noisy means that the desired speech signal is degraded by additive noise and/or other effects such as nonlinear distortions or reverberation.
The focus of this work lies on additive noise sources which can be environmental sounds such as wind, traffic or additional speech, or noise induced by the transmission system e.g. quantization noise.
Therefore, we define the noisy signal $x(t)$ as a sum of the clean speech signal $s(t)$ and a the noise signal $n(t)$.

$$x(t) = s(t) + n(t) \tag{1.1}$$

The goal in speech enhancement is then to predict an estimate $\hat{s}(t)$ of the clean signal, given an observation of the noisy signal (see figure 1.1).

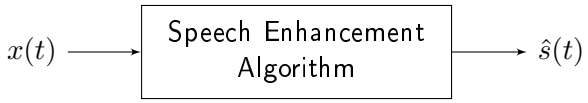

Figure 1.1 – A speech enhancement algorithm outputs an estimate $\hat{s}(t)$ of the clean speech signal, given the noisy signal $x(t)$ as an input

.

1.2 Related Work

At the present day most noise reduction and speech enhancement systems as used in mobile communications, hearing aids or speech recognition technologies, are based on signal processing techniques exploiting the statistical properties of the anticipated signals. Generally this means that simplified assumptions are made about the properties of the speech and even more of the noise sources.
Basic spectral subtraction algorithms obtain a noise estimate from segments of the signal where no speech is present. One critical assumption made here is that the properties of the interfering signal will not change as much from one frame to another, so that the knowledge obtained from the past few frames can be used to separate the speech and the noise signal [1], [2], [3], [4], [5]. However, if the signal does change its properties this leads to an perceptually unpleasant effect known as musical noise and of course if the changes are particularly drastic the suppression mechanism will lose its effect completely. While other adaptive methods such as wiener filtering [6], minimum mean-square error short-time spectral amplitude (MMSE-STSA) [7] or maximum-a-posteriori (MAP) estimation [8] are able to reduce the effect of musical noise by incorporating assumptions about the statistics of the speech signal, they still require complicated methods to estimate the power spectral density (PDS) of the noise signal [9], [10].

So called model-based approaches incorporate a priori information about the spectro-temporal properties of speech and noise signals. While hidden Markov models (HMMs) [11], [12] can be used to include phonetic information to estimate the speech source, typical supervised speech enhancement systems such as methods based on nonnegative matrix factorization (NMF) [13], [14], [15], are trained on different types of noise and speech models.

In recent years there has been an increased interest in deep learning and several studies have been investigating the use of purely data-driven approaches to speech enhancement and source separation. In particular after the publications on Deep Belief Nets (DBN) and greedy layer wise pre-training by Geoffrey Hinton [16] and Yoshua Bengio [17] several authors have applied these concepts on speech enhancement and source separation.

Already in 1989 a neural network speech enhancement technique was proposed by Shin'ichi Tamura [18]. In particular the author used a neural network with 60 hidden units for each of the four layers, the input was a 60-point long noisy speech waveform sampled at 12 kHz and the output was the corresponding clean waveform of the same length. While the study focussed more on the analysis of the trained network and no actual numbers on the performance of the system were published, the authors reported that 'the noise reduction method using a neural network was comparable or better than the conventional spectral subtraction method'.

A few years later Fei Xie and Dirk van Compernolle [19] proposed a family of nonlinear spectral estimators implemented by a multilayer perceptron neural network. They used parameters describing the speech and noise distributions as extra inputs to the network and investigated the performance of a recognition system when using the nonlinear spectral estimator as its front end. In terms of the recognition rate an average improvement of around 10%, compared to the generalized spectral subtraction method, was reported.

However, during that time neural networks were considered unpractical due to their computational complexity and advances were made in other fields of machine learning which led to a decline in popularity which lasted until 2007. At that time came a new wave of neural network research on grounds of the above mentioned publication by Geoffrey Hinton as well as the increase of computational power and dataset sizes [20, p.17-19]. After deep learning was successfully used for noise robust speech recognition [21], [22], [23] as well as binary speech coding [24], several studies on neural network speech enhancement have followed.

Based on [25], Lu et al. [26] built a deep denoising autoencoder (DAE) using greedy layer-wise pretraining plus fine tuning for noise reduction and speech enhancement. Patches of mel frequency power spectra using 40 bands were used as features and the effects of the trainings data size, the number of hidden layers and the depth of the network were investigated. An inverse transform was performed to synthesize the restored speech and to compare the results with the clean reference signals and the results of a baseline algorithm, they were resynthesized from mel sprectra in the same way. While the depth of the DAE did not drastically effect the results, increasing the training data set size

and hidden layer size showed general improvement in terms of noise reduction, distortion as well as perceptual evaluation of speech quality (PESQ) measure. Furthermore, an optimal patch size of 11 frames in terms of speech enhancement performance was reported. In comparison to an MMSE based algorithm (IMCRA) the PESQ for the enhanced speech was up to 1.1 higher, but the effect on noise reduction in dB improved only for some of the test scenarios. Moreover, the evaluation only employed noise that was included in the training data and therefore the network's performance in terms of generalization was not determined.

In [27] deep neural networks (DNN) were used as part of an algorithm to improve speech recognition in noise for hearing-impared listeners. From a 65-channel gammatone filterbank output features were extracted to train multiple subband DNNs. The features included amplitude modulation spectrogram, perceptual linear prediction and mel frequency cepstral coefficients (MFCC) as well as additional delta features. Ideal binary masks were used as training targets and the subband DNNs were used to classify the corresponding time-frequency points as either speech or non-speech. The authors reported increased intelligibility for speech-shaped noise and babble noise scenarios. The results were more distinct for hearing impared listeners, reaching up to 70% of improvement in intelligibility.

A different approach was used in [28]. Here a DAE is used to obtain the power spectrum estimate of clean speech and the a priori signal-to-noise (SNR) ratio is estimated using a posteriori SNR controlled recursive averaging (PCRA). Finally the enhanced speech is obtained by frequency domain wiener filtering. The autoencoder has one layer consisting of 300 hidden units and uses a frequency dependend linear weighting function to improve the perceptual quality. The training process was similar to [26], using unsupervised pre-training and fine-tuning. The method was compared to frequency domain wiener filter with Decision-Directed approach for SNR estimation and achieved similar results in terms of SNR and distortion and slightly better results in terms of PESQ.

Huang et al. [29] studied deep learning for monaural speech separation and proposed joint optimization of the deep learning models with an extra masking layer to enforce a reconstruction constraint. They used DNN and recurrent neural network (RNN) models with 2 layers of 150 hidden units and evaluated conventional as well as log-mel spectra. The proposed models achieved about 3.8 to 4.9 dB Source to Interference Ratio (SIR) gain compared to NMF models and maintained better source-to-distortion (SDR) and source-to-artifact ratio (SAR).

Ding Liu, Paris Smaragdis and Minje Kim [30] presented various experiments using a deep learning model for speech denoising. In contrast to earlier studies, they did not apply any pretraining step and used backpropagation to estimate the model parameters. Compared to an NMF model trained on the same dataset as the neural network (NN) models, the proposed method achieved significant improvements in terms of SDR,

SIR, SAR and short-time objective intelligibility measure (STOI). Furthermore, the experiments showed that this method is adequately robust to unseen mixing situations. Comparison of network topologies showed that the number of hidden layers and units was not crucial although the best results were achieved with a single layer of 2000 units. Additionally, it was shown that rectified linear units (ReLu) were superior to other common activation functions.

In [31] the authors evaluated different training targets for supervised speech enhancement. A DNN with three hidden layers of 1024 rectified linear units was trained using backpropagation without unsupervised pre-training. They compared different spectral masks, including ideal binary mask (IBM), target binary mask (TBM) and ideal ratio mask (IRM) as well as the short-time Fourier transform spectral magnitude, its corresponding mask (FFT-mask) and direct estimation of the gammatone frequency power spectrum. It was reported that when testing with different noise scenarios, softmasks generally produce the best results in terms of STOI and PESQ.

Xu et. al [32], [33] have been further investingating the use of restricted boltzmann machines (RBM) and DBNs in combination with large scale training to learn the mapping between noisy and clean log power spectral features. To build the model a stack of multiple RBMs was pre-trained layer-by-layer with noisy speech and then fine-tuned with noisy and clean speech pairs. Sigmoid activation was used for the hidden units and linear activation for the output layer. In addition global variance equalization was incorporated as a post-processing step. For the experiments in [33] a large training set was built, which consisted of about 2500 hours of training data including 4620 clean speech utterances and 104 different noise types. Different context sizes up to 13 frames were compared in terms of SNR and the training set size was evaluated in terms of PESQ. Furthermore, the generalization capabilities of the trained network were investigated using different unseen noise types. Similar to the results in [26] no significant improvement for context sizes over 11 frames was reported and the best results were achieved using a training set size of 625h although the difference to training with 100h was only marginal. It was stated that the combination of pre-training, variance normalization and dropout improved the performance for unseen noise types about 0.25 dB in average. Furthermore, in a subjective listening test 78% of the test candidates preferred the DNN-based enhanced speech to that obtained using a conventional approach (LogMMSE).
In a later study [34] the use of multi-objective learning and IBM-based post-processing was investigated. As a constraint in the objective function, MFCCs and IBM were used as additional training targets and in combination with the post-processing step about 0.2 PESQ and 0.03 STOI improvements were obtained on average.

Several studies have investigated the use of deep learning in the context of speech separation, which we will mention here because of the similarities to mask-based methods for noise reduction.
In [35] the authors studied discriminatively trained recurrent neural networks for single-channel speech separation. They used ideal ratio masks as training targets and compared

DNNs and RNNs in particular long short-term memory (LSTM), using different network topologies. The best results were achieved using 2 layers of 256 LSTMs, which produced an average SDR of 12.2 dB in comparison to 10.46 dB when using a 3-layer DNN with 1024 hidden units. Furthermore, DFT spectra and mel spectra with both 100 and 40 bands were compared as input features. It was reported that in the DFT power spectrum domain, an 11.4 dB average SDR was obtained while in the mel magnitude domain with 100 bands 12.8 was achieved.

In an additional paper [36], the use of bidirectional LSTMs was investigated and the framework was extended by an automatic speech recognition (ASR) system. It was shown that phase sensitive filters (PSF) generally produce better results than IBM and IRM. Bidirectional LSTMs achieved an average SDR that was up to 0.5 dB higher compared to the results obtained with LSTMs and furthermore a phase sensitive cost function was used, which further improved the results by more than 1.5 dB.

Finally convolutional neural networks (CNNs), which are playing a big role in state of the art image classification results [37], have been used for speech enhancement as well. In [38] the authors investigated the utilization of CNNs for the estimation of soft masks and log power spectra, furthermore the generalization performance for different languages was examined. They used a network consisting of two convolutional layers followed by to fully connected layers with 1024 hidden units. It was shown that when using log power spectra as targets, CNNs produced results that outperformed conventional DNNs by up to 0.14 PESQ and 0.6 least significant difference (LSD). Additionally, it was reported that the network trained with a multilingual dataset was in average 3% better in terms of PESQ than the ones trained with monolingual data.

In summary there has been a growing interest in deep learning and its application to speech enhancement in particular during the past few years. Different neural network architectures as well as signal analysis and synthesis methods have been studied which led to promising results. In the context of this work we will compare some of the mentioned ideas and built a framework which will combine the most promising approaches. The following will give a short introduction to deep learning and its basic concepts.

2 | Deep Learning

The term deep learning describes a category of machine learning algorithms based on **artificial neural networks** (ANNs). In recent years, deep learning has been successfully applied to many tasks in the field of artificial intelligence, such as automatic speech recognition (ASR) [39] and dialogue systems [40], image classification [37], handwriting recognition [41] or machine translation [42].
ANNs are computational models which apply an arbitrary function on a given input x to obtain a desired output y

$$y = f(x; \Theta) \tag{2.1}$$

The model has a set of parameters Θ, which are gradually adjusted during the training phase so that the outputs y are as close as possible to the desired output values t of a given training set $\{\mathbf{x}, \mathbf{t}\}$.

In the following, the basic concepts of deep learning will be coarsely described by first introducing the artificial neuron model as the basic building block of artificial neural networks in chapter 2.1. Then in chapter 2.2 the structure and the mathematical description of feed forward networks will be explained followed by an explanation of the basic training process in chapter 2.3. Finally, in 2.4 three basic network topologies for deep learning, namely autoencoders, convolutional networks and recurrent networks are described.

2.1 Artificial Neurons

ANNs are inspired by the biological nervous system which processes information through networks of neurons connected by synapses. Typically a neural cell consists of so called dendrites, which are able to receive electrochemical stimuli from other neurons, a cell body (soma) and an axon, which transfers a processed signal to the dendrites of other neurons in the network [43, p.35].
An **artificial neuron**, is a significantly simplified mathematical model of the way neurons receive, process and transmit information. In its general form, as depicted in figure 2.1, it is defined by a linear combination of N input values x_i and a subsequent transformation

of the resulting activation a

$$a = \sum_{i=1}^{N} w_i x_i + b \tag{2.2}$$

$$y = g(a) \tag{2.3}$$

where g is an arbitrary transfer function also referred to as the **activation function** and the parameters w_i and b are known as **weights** and **biases** [43, p.72] [44, p.227]. Using vector notation equations 2.2 and 2.3 can be compactly expressed as

$$y = g(\mathbf{w}\mathbf{x} + b) \tag{2.4}$$

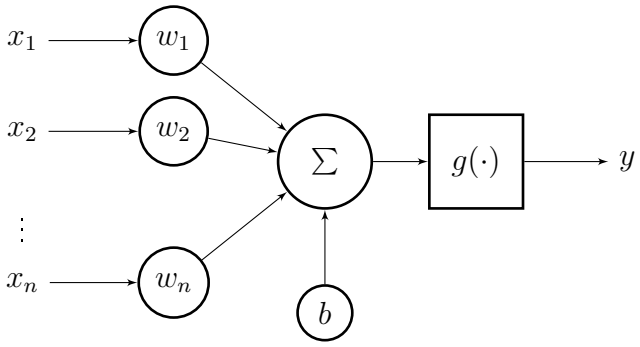

Figure 2.1 – Artificial neuron: The inputs x_i are multiplied by the weights w_i and the linear combination is then transformed by an activation function g.

Commonly used activation functions are shown in figure 2.2 and include **logistic sigmoid**, **hyperbolic tangent** (tanh), **rectified linear unit** (ReLu) or the **softmax** function, which is typically used for classification.

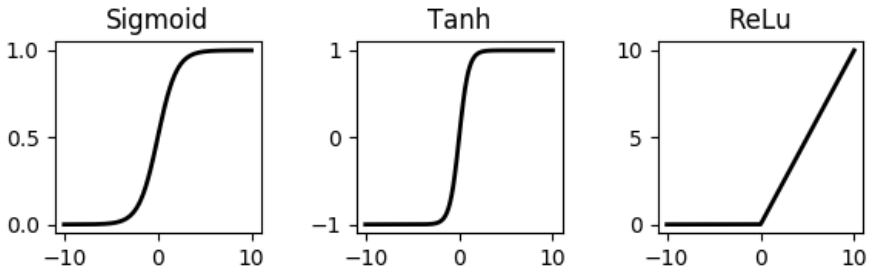

Figure 2.2 – Common activation functions. Sigmoid (left), hyperbolic tangent (middle) and rectified linear unit (right).

In the following it will be described how artificial neurons are used as building blocks for neural network models.

2.2 Feed-Forward Network

The most basic ANN structure is a **feed forward network**, sometimes also called multilayer perceptron (MLP), and has been a common choice for classification tasks. Multiple artificial neurons are combined to form a network organized in so called **layers**. Figure 2.3 shows a simple feed forward neural network consisting of one hidden layer and an output layer. Each layer i is defined by a weight matrix $\mathbf{W}^{(i)}$ and a bias vector $\mathbf{b}^{(i)}$ combining the parameters for each of the neurons inside the layer [20, p.193/194].

$$\mathbf{h}^{(1)} = g^{(1)}\left(\mathbf{W}^{(1)T}\mathbf{x} + \mathbf{b}^{(1)}\right) \tag{2.5}$$

$$\mathbf{y} = g^{(2)}\left(\mathbf{W}^{(2)T}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}\right) = g^{(2)}\left(\mathbf{W}^{(2)T}\left(g^{(1)}\left(\mathbf{W}^{(1)T} + \mathbf{b}^{(1)}\right)\right) + \mathbf{b}^{(2)}\right) \tag{2.6}$$

where the weight matrix $\mathbf{W}$ is formed by the weight vectors corresponding to the neurons inside the associated layer and the bias vector $\mathbf{b}$ by the corresponding bias values accordingly.

$$\mathbf{W}^{(i)} = [\mathbf{w}_1^{(i)}, \mathbf{w}_2^{(i)}, ..., \mathbf{w}_{N_i}^{(i)}] \tag{2.7}$$

$$\mathbf{b}^{(i)} = [b_1^{(i)}, b_2^{(i)}, ..., b_{N_i}^{(i)}]^T \tag{2.8}$$

In general, the outputs from the current layer are treated as the input for the subsequent layer

$$\mathbf{h}^{(k)} = g^{(k)}\left(\mathbf{W}^{(k)T}\mathbf{h}^{(k-1)} + \mathbf{b}^{(k)}\right) \tag{2.9}$$

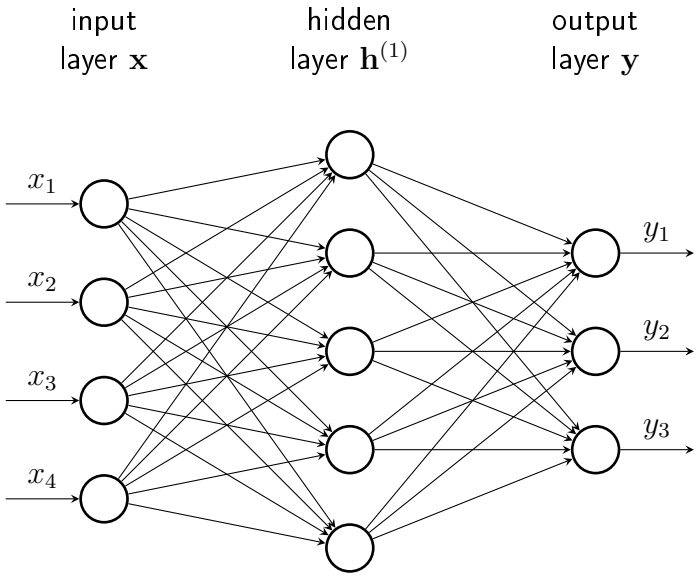

Figure 2.3 – Simple feed forward neural network with one hidden layer. Each of the connections represents an artificial neuron consisting of a weight vector $\mathbf{w}$ a bias value b and an activation function g.

The number of hidden layers connotes the depth of the corresponding network, whereby the term **Deep Neural Network** (DNN) implies an artificial neural network spanning multiple hidden layers. Figure 2.4 shows a deep neural network with three hidden layers, each consisting of $n = 5$ neurons or **hidden units**.

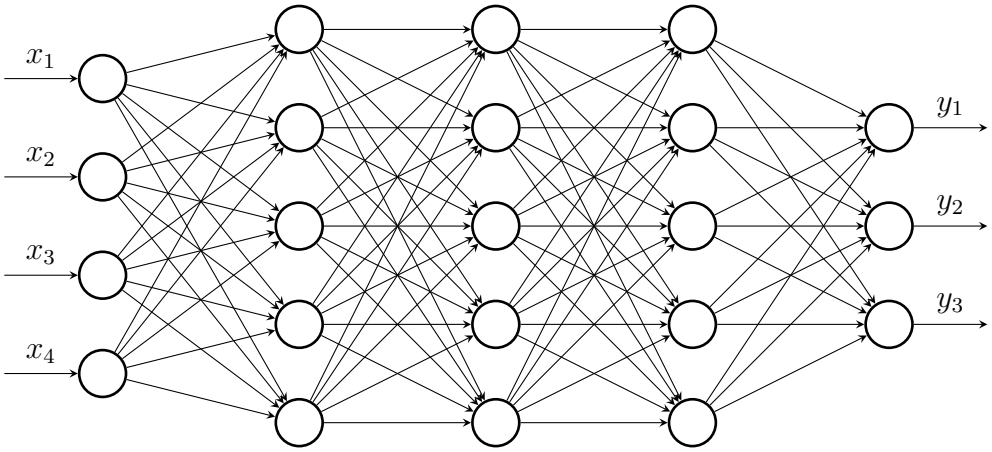

Figure 2.4 – Deep feed forward neural network with three hidden layers.

2.3 Training neural network models

In order to find the parameters of the model function f_Θ, the network is presented with data consisting of input and output pairs. This process is called learning or **training**. A cost function which measures the divergence between the true output and the network's prediction is used to evaluate the model. This will be briefly described in 2.3.1. Then the parameters are adjusted by a gradient based optimization algorithm as described in 2.3.2. In chapter 2.3.3 the backpropagation algorithm is introduced, which is needed to compute the gradient of the loss function with respect to the weights. Finally, regularization for deep neural networks will be briefly discussed in chapter 2.3.4.

2.3.1 Cost Function

During training, we try to minimize a loss function $\mathcal{L}(f(\mathbf{x};\Theta),\mathbf{t})$ with respect to the parameters Θ which is evaluated on a set of N training pairs $\{\mathbf{x}_i,\mathbf{t}_i\}$ $i = 1,..,N$, consisting of feature and target vectors for a given observation i [44, 5.2].

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmin}}\, \mathcal{L}(f(\mathbf{x};\Theta),\mathbf{t}) \tag{2.10}$$

Note that the targets $\mathbf{t}$ can be vectors of class labels or probabilities for classification as well as any kind of vectors, matrices or tensors for regression problems. A common loss function is the **mean-squared error** (MSE) [44, 1.5.5], but depending on the context other objectives measures such as the **Kullback-Leibler divergence** or **cross-entropy** losses are popular as well. All loss functions measure the divergence of the predicted outputs $\mathbf{y}$ from the true outputs $\mathbf{t}$. In case of the MSE we measure the squared distances over all N observations, and take the mean value:

$$\mathcal{L}_{MSE}(f(\mathbf{x};\Theta),\mathbf{t}) = \frac{1}{N}\sum_{i=1}^{N}\left(f(\mathbf{x}_i;\Theta)-\mathbf{t}_i\right)^2 = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{y}_i-\mathbf{t}_i\right)^2 \tag{2.11}$$

2.3.2 Parameter Optimization

To find a solution to equation 2.10 i.e. to find the optimum parameters Θ for a given network an optimization algorithm is used. Typically optimization algorithms use information from the gradient of the cost function to guide the weights into the direction of a minimum. In the following two variants of the gradient descent algorithm are introduced the stochastic gradient descend (SGD) and the *Adam* optimizer.

Stochastic Gradient Descent A common optimization method is SGD [20, p.290], which is technically an approximation of the gradient descent optimizer. Gradient descent finds minima by taking steps in the direction of the negative gradient. The update rule is given by:

$$\Theta^{(t+1)} = \Theta^{(t)} - \eta\nabla_\Theta\mathcal{L}(f(\mathbf{x};\Theta^{(t)}),\mathbf{t}) \tag{2.12}$$

where η is a step size also known as **learning rate** and $\mathcal{L}$ is the loss function evaluated over all training samples

$$\mathcal{L}(f(\mathbf{x};\Theta),\mathbf{t}) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(f(\mathbf{x}_i;\Theta),\mathbf{t}_i) \tag{2.13}$$

By contrast stochastic gradient descent performs the update rule only on one of the training samples at a time

$$\Theta^{(t+1)} = \Theta^{(t)} - \eta\nabla_\Theta\mathcal{L}(f(\mathbf{x}_i;\Theta^{(t)}),\mathbf{t}_i) \tag{2.14}$$

however a lot of times a modified version is used, which employs small subsets of the training data, known as mini-batches. With N_b denoting the **batch size**, the update rule of SGD becomes

$$\Theta^{(t+1)} = \Theta^{(t)} - \eta\nabla_\Theta\sum_{i=1}^{N_b}\mathcal{L}_i(f(\mathbf{x}_i;\Theta^{(t)}),\mathbf{t}_i) \tag{2.15}$$

Adam Another variant of the gradient descent method is *Adam* [45], which we briefly describe here since it proved to be an efficient optimizer and was used for most of the experiments within the scope of this project. The name derives from adaptive moment estimation. In the so called momentum method the previous step is remembered and contributes to the current update. Usually it is multiplied by a forgetting factor β and added to the current parameter. The ADAM method computes adaptive learning rates for the first and second moments. It is described in the following algorithm 1:

Algorithm 1 The *Adam* algorithm

Require: Step size η
Require: Exponential decay rates for the moment estimates $\beta_1, \beta_2 \in [0,1)$ with $\beta_1 \neq \beta_2$
Require: Initial parameters Θ_0
Require: Small constant for numerical stabilization ϵ
  Initialize 1st and 2nd moment vectors $\mathbf{m}_0 = 0$, $\mathbf{v}_0 = 0$
  Initialize time step $t = 0$
  while stopping criterion not met **do**
    $t \leftarrow t + 1$
    $\mathbf{g} \leftarrow \nabla_\Theta \sum_{i=1}^{N_b}\mathcal{L}_i(f(\mathbf{x}_i;\Theta),\mathbf{t}_i)$, Compute gradient on minibatch
    $\mathbf{m} \leftarrow \beta_1\mathbf{m} + (1-\beta_1)\mathbf{g}$, Update biased first moment estimate
    $\mathbf{v} \leftarrow \beta_2\mathbf{v} + (1-\beta_2)\mathbf{g}^2$, Update biased second moment estimate
    $\hat{\mathbf{m}} \leftarrow \frac{\mathbf{m}}{1-\beta_1^t}$, Compute bias-corrected first moment estimate
    $\hat{\mathbf{v}} \leftarrow \frac{\mathbf{v}}{1-\beta_2^t}$, Compute bias-corrected second moment estimate
    $\Theta \leftarrow \Theta - \eta\frac{\hat{\mathbf{m}}}{\sqrt{\hat{\mathbf{v}}}+\epsilon}$, Update parameters
  end while
  return Θ, Resulting parameters

2.3.3 Backpropagation

To calculate the gradient $\nabla_\Theta \mathcal{L}$ the so called **backpropagation** algorithm is used. In the first step, the forward pass, the error is computed by propagating the input through the network to calculate the output for the current parameters and a given training pair or **minibatch**. Then, by using the chain-rule, the error is propagated back through the network to obtain the contribution to the output value for each of the individual weights, this step is known as the backward pass [46].
In the following we will use the symbol $w_{ij}^{(k)}$ to denote a single weight of the parameter set, where i describes the index for the weight vector, j the hidden unit, and k the number of the corresponding layer. For simplification, the bias $b_j^{(k)}$ will be omitted, which could also be interpreted as adding a value $w_{0j}^{(k)}$ to the weight vector and having an additional output value $h_0^{(k-1)}$ of value 1 at the previous layer.

$$\mathbf{w}_j^{(k)} = [w_{0j}^{(k)}, w_{1j}^{(k)}, .., w_{N_{k-1}j}^{(k)}]^T$$
$$\mathbf{h}^{(k)} = [1, h_1^{(k)}, .., h_{N_k}^{(k)}]^T$$

As already mentioned above, the activation $a_j^{(k)}$ of the j-th hidden unit inside the k-th layer is defined as

$$a_j^{(k)} = \sum_{i=0}^{N_{k-1}} w_{ij}^{(k)} h_i^{(k-1)} \tag{2.16}$$

Where h_i is the i-th output value after the nonlinear activation function g

$$h_i = g(a_i) \tag{2.17}$$

In general, the derivation of the loss function $\mathcal{L}$ w.r.t the weight w_{ij} depends on the activation a_j of the corresponding layer before it is passed to the activation function.

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{(k)}} = \frac{\partial \mathcal{L}}{\partial a_j^{(k)}} \frac{\partial a_j^{(k)}}{\partial w_{ij}^{(k)}} \tag{2.18}$$

We denote the left term, also referred to as **errors**, by

$$\delta_j^{(k)} = \frac{\partial \mathcal{L}}{\partial a_j^{(k)}} \tag{2.19}$$

Using equation 2.16, the right term can be written as

$$\frac{\partial a_j^{(k)}}{\partial w_{ij}^{(k)}} = \frac{\partial}{\partial w_{ij}^{(k)}} \sum_{i=0}^{N_{k-1}} w_{ij}^{(k)} h_i^{(k-1)} = h_i^{(k-1)} \tag{2.20}$$

Substituting equations 2.19 and 2.20 into equation 2.18, results in

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{(k)}} = \delta_j^{(k)} h_i^{(k-1)} \tag{2.21}$$

This means, that the partial derivative of a weight $w_{ij}^{(k)}$ is simply the product of the error $\delta_j^{(k)}$ in the current layer and the output $h_i^{(k-1)}$ at node i in the previous layer. With this we can now derive the the partial derivatives for the output layer and any given hidden layer. Assuming a single training sample the error function is

$$\mathcal{L} = (y_j - t_j)^2 = (g^{(m)}(a_j^m) - t_j)^2 \tag{2.22}$$

with m denoting the output layer. Using equation 2.19 we can write

$$\delta_j^{(m)} = \frac{\partial \mathcal{L}}{\partial a_j^{(m)}} = 2(g^{(m)}(a_j^{(m)}) - t_j)g^{(m)\prime}(a_j^{(m)}) = 2(y_j - t_j)g^{(m)\prime}(a_j^{(m)}) \tag{2.23}$$

Which gives us the gradient of the error function for a weight in the output layer:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{(m)}} = \delta_j^{(m)} h_i^{(m-1)} = 2(y - t)g^{(m)\prime}(a_j^{(m)})h_i^{(m-1)} \tag{2.24}$$

As for a arbitrary hidden layer, we use again the chain rule to evaluate the error term

$$\delta_j^{(k)} = \frac{\partial \mathcal{L}}{\partial a_j^{(k)}} = \sum_l \frac{\partial \mathcal{L}}{\partial a_l^{(k+1)}} \frac{\partial a_l^{(k+1)}}{\partial a_j^{(k)}} \tag{2.25}$$

where the sum runs over all nodes l in the subsequent layer $k+1$. Using again equation 2.19, we can write

$$\delta_j^{(k)} = \sum_l \delta_l^{(k+1)} \frac{\partial a_l^{(k+1)}}{\partial a_j^{(k)}} \tag{2.26}$$

With equation 2.16 the right term becomes

$$\frac{\partial a_l^{(k+1)}}{\partial a_j^{(k)}} = \frac{\partial}{\partial a_j^{(k)}} \sum_j w_{jl}^{(k+1)} g^{(k)}(a_j^{(k)}) = w_{jl}^{(k+1)} g^{(k)\prime}(a_j^{(k)}) \tag{2.27}$$

Combining equations 2.26 and 2.27 leads to the backpropagation formula:

$$\delta_j^{(k)} = g^{(k)\prime}(a_j^{(k)}) \sum_l w_{jl}^{(k+1)} \delta_l^{(k+1)} \tag{2.28}$$

Which now can be used to compute the partial derivative of the loss function $\mathcal{L}$ for a given weight $w_{ij}^{(k)}$

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{(k)}} = \delta_j^k h_i^{(k-1)} = g^{(k)\prime}(a_j^{(k)}) h_i^{(k-1)} \sum_l w_{jl}^{(k+1)} \delta_l^{(k+1)} \tag{2.29}$$

The stated derivation of the backpropagation formulas are based on [44, 5.3] but as in [46] the superscript notation for indicating the layer index was added.

2.3.4 Regularization

In machine learning an essential goal is to develop a model that performs well on unknown test data and not only on the training data. This property is described by the term **generalization**. If the training error is small but the model does not perform well on the test data this means that the underlying function is not approximated well. This case is called **overfitting** and is illustrated in the plot *(c)* of figure 2.5. On the contrary, when the true function is not approximated well enough so that both training and test performance are low, the term **underfitting** is used. This case is illustrated in plot *(a)*.

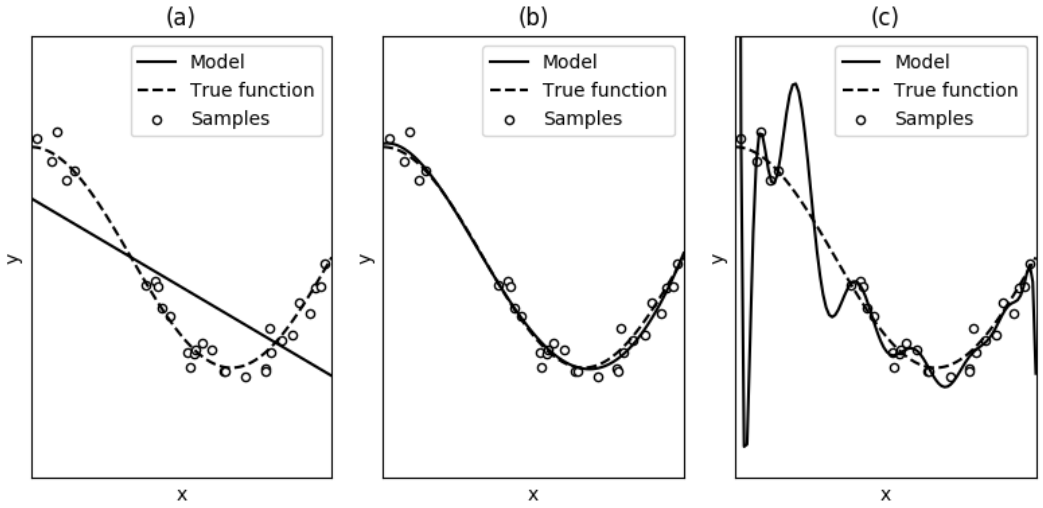

Figure 2.5 – Demonstration of underfitting and overfitting. In plot *(a)* the model is too simple and the true function is not approximated well enough, only a few samples fit to the estimated function. In plot *(c)* the model is too complicated, most of the samples fit the function but the estimated curve is not a good approximation of the true function. Plot *(b)* shows an accurate model of the true function.

To improve generalization and avoid overfitting there exist many strategies collectively known as regularization.
In the following two different regularization methods, namely early stopping and dropout will be described. For further information refer to [20, ch.7].

Early Stopping During training, the loss represents the performance of the model on the taining data. To observe the performance in terms of generalization, the model is evaluated on a small validation set, usually after each epoch [1]. Ideally training and validation loss are decreasing steadily over time. However, in the case of overfitting, an increase of the validation loss may be observed while the training loss is further improving. One regularization strategy is to stop the training procedure if such a drift

1. The term epoch describes one complete pass over the training data. After one epoch the model has been presented with every observation from the training set once.

is observed and the validation loss starts to increase. This strategy is known as early stopping and is illustrated in figure 2.6. The model parameters are saved after each epoch and if the validation loss is not further improving, the training stops and the parameters corresponding to the best validation performance are returned. Usually the validation set is a subset of the training data which is not fed to the model.

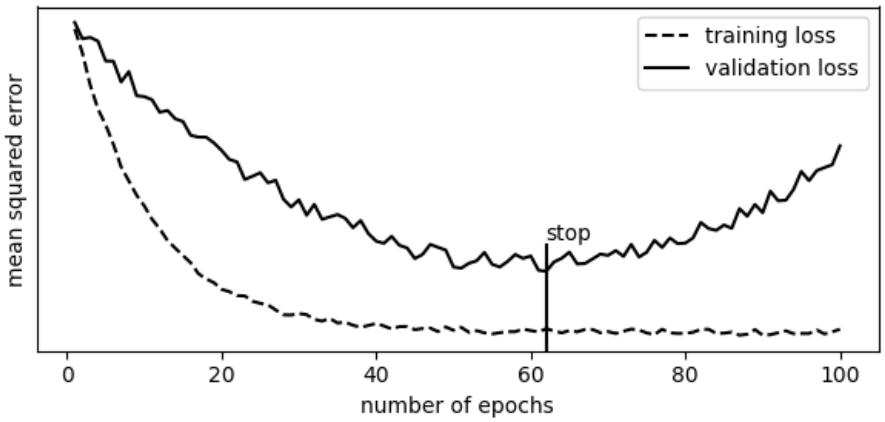

Figure 2.6 – Illustration of early stopping. The validation loss is monitored after each epoch and if no improvements can be observed the training procedure is stopped.

Dropout A different regularization approach is **dropout** training which was first published in 2014 by Srivastava et al. [47].
The idea is to randomly set a certain percentage of the input and hidden units to zero after each weight update. This can also be seen as randomly sampling a binary mask which is applied to all the input and hidden units in the network. The probability of a mask value to be zero is a parameter which has to be selected before training.
There are different ways to implement dropout regularization. The *keras* framework [48] applies regularization separately to each layer, making it possible to use dropout only for selected layers with different parameters.

2.4 Network Topologies

In addition to feed forward networks, deep learning involves numerous network structures, as well as different regularization and optimization proceedures. In the following three common network topologies will be introduced, namely autoencoders, convolutional neural networks and recurrent neural networks. For a more detailed dicussion and further information on practical deep networks, the reader is reffered to part II of the *Deep Learning Book* [20].

2.4.1 Autoencoders

Autoencoders [20, ch.14] are a class of neural networks that attempt to output a copy of the input data. They consist of two parts: an encoder $\mathbf{h} = f(\mathbf{x})$ that produces a hidden representation of the input data and a decoder $\mathbf{r} = g(\mathbf{h})$ that outputs a reconstruction of the input from the hidden representation. The learning process can be described as minimizing a loss function over the input data and the estimated output

$$\mathcal{L}(g(f(\mathbf{x})), \mathbf{x}) \tag{2.30}$$

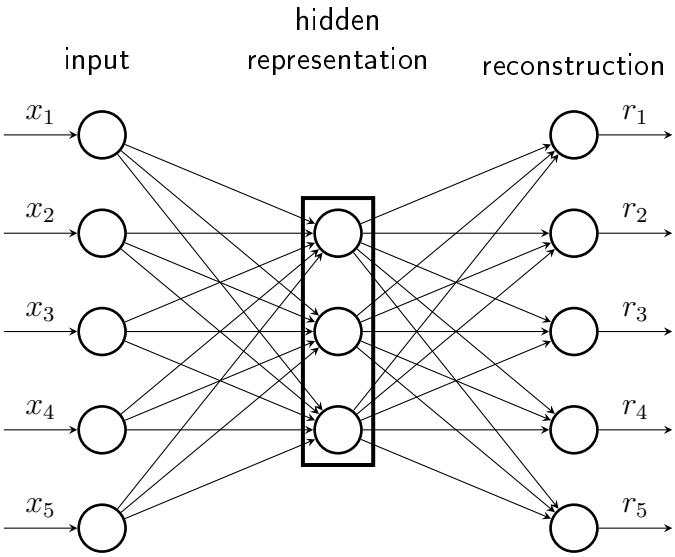

Figure 2.7 – An autoencoder consists of an encoder $\mathbf{h} = f(\mathbf{x})$ that produces a hidden representation of the input data and a decoder $\mathbf{r} = g(\mathbf{h})$ that outputs a reconstruction of the input from the hidden representation.

The model often learns useful properties of the input data which can be used for dimensionality reduction or feature learning.
A variant is the so called **Denoising Autoencoder** (DAE), which tries to reconstruct the data $\mathbf{x}$ from the input $\tilde{\mathbf{x}}$, which is corrupted by some form of noise

$$\mathcal{L}(g(f(\tilde{\mathbf{x}})), \mathbf{x}) \tag{2.31}$$

Autoencoders are often regularized by adding some kind of penalty term to the loss function, for example a sparsity constraint on the code layer or any kind of activity regularization [20, ch.14.2].
Deep autoencoders can be formed the same way as simple deep feedforward networks by increasing the number of layers in the encoder and decoder.

As described in [16], unsupervised pre-training in combination with supervised fine-tuning can be used to build a deep belief net (DBN). Originally restricted boltzmann machines

(RBM) and contrastive divergence were used to build the DBNs, but for practical reasons the RBMs are sometimes substituted with autoencoder networks [25] [26]. Multiple single autoencoders are trained seperately to reconstruct the previous autoencoder's hidden layer. These mutiple autoencoders are then stacked on top of each other and can be trained the same way as a conventional DNN. The benefit of the pre-training stage is that the layer weights are already guided into a useful direction rather than starting with a random weight initialization for the fine-tuning. This can avoid the weights to get stuck at local minima and speeds up the training process.

2.4.2 Convolutional Neural Networks

A commonly used network topology, especially for problems involving image data are **Convolutional Neural Networks** (CNNs) [20, ch.6]. The principal difference to conventional neural networks is that they apply a convolution operation to the input rather than a matrix multiplication.

$$y(t) = (w * x)(t) = \sum_{\tau} w(\tau)x(t-\tau) \tag{2.32}$$

Where w is referred to as the **kernel** and x as the **input**. The output is sometimes referred to as the **feature map**. Often convolutional layers operate on multidimensional input data so the convolution is carried out over multiple axis. In the case of 2-dimensional image data I and filter kernels K the convolution becomes

$$S(i,j) = (K * I)(i,j) = \sum_{u}\sum_{v} K(u,v)I(i-u, j-v) \tag{2.33}$$

Note that there are different ways to handle the convolution near the edges of the input image. Sometimes indices for which the kernel would be convolved with values outside the range of the input image are excluded from the operation, which results in a smaller output image. A different option is to pad the input image outside the edges so that the output is of the same size as the input. Figure 2.8 depicts an example of a convolutional operation. The image on the left is convolved with the Kernel K in the middle and results in the matrix shown on the right. In this example the input image is zero-padded to ensure an output image of the same dimensions.

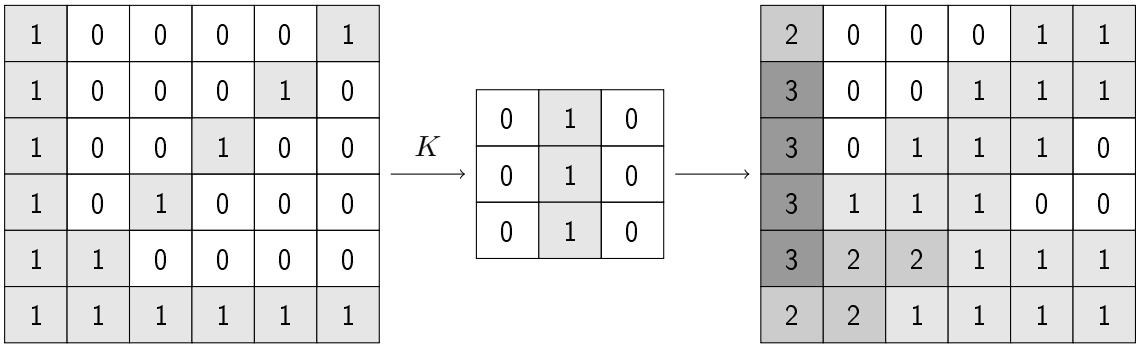

Figure 2.8 – Demonstration of the convolution operation. The image on the left side is convolved with the kernel K (middle), which results in the modified image on the right. Since the kernel represents a vertical line, the resulting image shows increasing values in areas where the input image has vertical lines. In this example, values outside the edges of the input image are treated as zero.

In practice, images often have multiple channels, e.g. three channels $j = 1, 2, 3$ for the colors red, blue and green (RGB). In this case the output of the input layer is summed over all channels

$$\mathbf{Y}_i = \sigma\left(b_i + \sum_{j=1}^{m} \mathbf{K}_i * \mathbf{I}_j\right) \tag{2.34}$$

Here i is the index for the filter kernel in the current layer and b is a bias value. In general a layer k has a fixed number of filter kernels $m^{(k)}$, which means that the output of the layer will add up to $m^{(k)}$ feature maps.

$$\mathbf{Y}_i^{(k)} = \sigma\left(b_i^{(k)} + \sum_{j=1}^{m^{(k-1)}} \mathbf{K}_i^{(k)} * \mathbf{Y}_j^{(k-1)}\right) \tag{2.35}$$

It should be noted that there are different ways to implement a convolutional layer. The above stated definitions are in accordance to the implentations used in the *Keras* API.

Typically a **pooling** operation takes place after the convolution and the subsequent nonlinear transformation of the input. Pooling, sometimes also referred to as subsampling, replaces the output of the net at a certain position with a statistical value computed over the nearby values. **Max pooling** for example, returns the maximum value over a rectangular window of a specific size. Common is also **average pooling** where the mean value within the window is reported. Pooling can reduce the size of the previously computed feature maps. The so called **stride** defines the step size for the shifting of the analysis window. If all strides are set to 1, the output of the pooling operation will have the same size as its input data.

A typical CNN for image classification is depicted in figure 2.10. It usually consists of multiple convolutional layers, each followed by a pooling layer. While this part can be interpreted as a feature extraction step, the following feed-forward network, consisting of multiple fully-connected layers can be seen as the actual classifier. The values inside the feature maps previous to the first fully connected layer are stacked to one large vector.

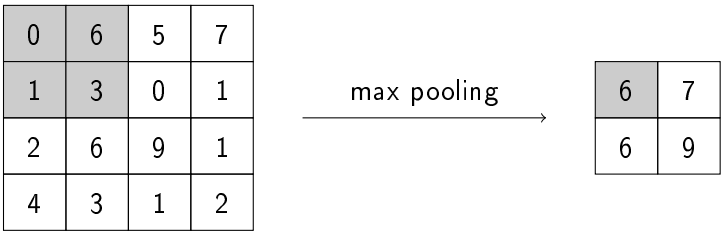

Figure 2.9 – Demonstration of the max pooling operation. The window size and stride in this example is $(2, 2)$, which results in reduced size of the image.

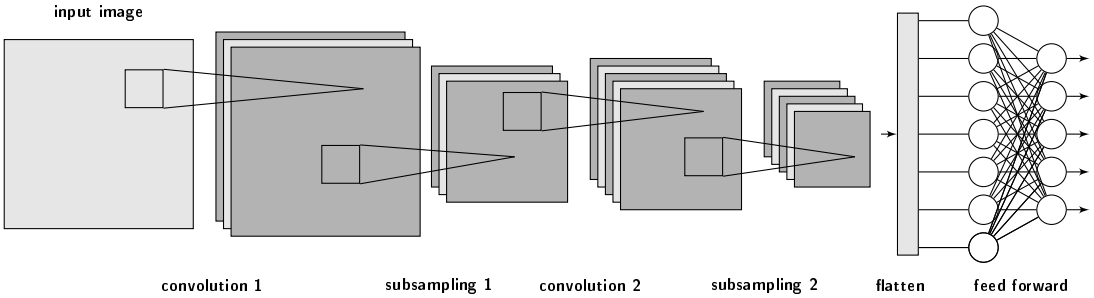

Figure 2.10 – Typical CNN consisting of 2 convolutional layers, 2 pooling layers and 2 fully connected layers. After the second pooling operation all resulting pixels are stacked to one vector, which is the input to the first fully connected layer.

2.4.3 Recurrent Neural Networks and LSTMs

Recurrent Neural Networks (RNN) are a class of neural networks which operate on sequential data and have been successfully applied to numerous tasks such as language processing [49], automatic speech recognition [50] or handwriting recognition [51]. The main difference to feed forward networks is that RNNs incorporate connections to values from previous time-steps. There exist numerous patterns for RNNs, which differ in the way the recurrent connections are placed inside the network. For detailed information about recurrent neural networks, it's design patterns and learning strategies refer to the Deep Learning Book [20, ch.10]. Figure 2.11 shows three simple possibilities for recurrent connections inside a RNN, in the left example the first hidden unit receives its own output from the preceding timestep $\mathbf{h}_1^{(t-1)}$ in addition to the current input values $\mathbf{x}^{(t)}$. In the middle the recurrent connection is between the output of the second hidden layer $\mathbf{h}_2^{(t-1)}$ and the first hidden layer and on the right hand side the first hidden unit receives the net output from the preceding time-step $\mathbf{y}^{(t-1)}$.

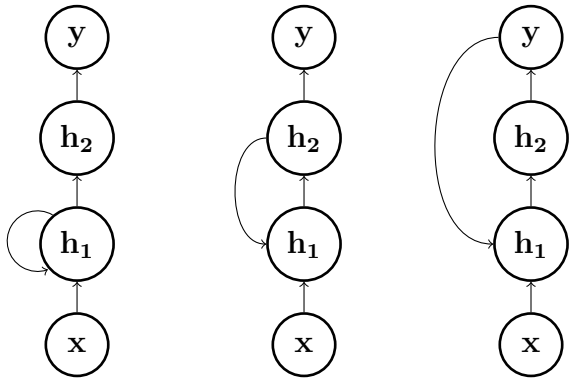

Figure 2.11

During training RNNs are **unfolded** to forward networks which can result in very deep and complicated structures. The computation of the gradient for unfolded networks may include many multiplications of a weight by itself which can cause the gradient to take very small values, aggravating the training process of RNNs. This so called **vanishing gradient** problem led to the design of **long short-term memory cells** or **LSTMs**.
LSTM recurrent networks are organized in cells that have an internal self-loop controlled by a system of gating units. Each gate behaves similar to a single conventional hidden unit, having a bias vector $\mathbf{b}$, a weight matrix $\mathbf{W}$ as well as an additional weight matrix $\mathbf{U}$ controlling the recurrent connection.
Such a LSTM structure is depicted in figure 2.12 and is described in the following.
The current state $\mathbf{s}^{(t)}$ is controlled by the input gate $\mathbf{i}^{(t)}$ and the forget gate $\mathbf{f}^{(t)}$, which adjust the contribution of the current and the previous state.

$$\mathbf{i}^{(t)} = \sigma\left(\mathbf{W_i}\mathbf{x}^{(t)} + \mathbf{U_i}\mathbf{h}^{(t-1)} + \mathbf{b_i}\right) \tag{2.36}$$

$$\mathbf{f}^{(t)} = \sigma\left(\mathbf{W_f}\mathbf{x}^{(t)} + \mathbf{U_f}\mathbf{h}^{(t-1)} + \mathbf{b_f}\right) \tag{2.37}$$

With t denoting the timestep and $\mathbf{x}^{(t)}$ and $\mathbf{h}^{(t-1)}$ denoting the current input and the previous output of the cell. The sigmoid activation denoted by σ sets the weights to values between 0 and 1.
Introducing an intermediate state $\tilde{\mathbf{s}}^{(t)}$

$$\tilde{\mathbf{s}}^{(t)} = \sigma\left(\mathbf{W_c}\mathbf{x}^{(t)} + \mathbf{U_c}\mathbf{h}^{(t-1)} + \mathbf{b_c}\right) \tag{2.38}$$

the current state is obtained as follows

$$\mathbf{s}^{(t)} = \mathbf{i}^{(t)} \odot \tilde{\mathbf{s}}^{(t)} + \mathbf{f}^{(t)} \odot \mathbf{s}^{(t-1)} \tag{2.39}$$

Where the operator $\odot$ denotes the Hadamard product.
Finally, the output gate $\mathbf{o}^{(t)}$ controls the contribution of the current state to the output $\mathbf{h}^{(t)}$.

$$\mathbf{o}^{(t)} = \sigma\left(\mathbf{W_o}\mathbf{x}^{(t)} + \mathbf{U_o}\mathbf{h}^{(t-1)} + \mathbf{b_o}\right) \tag{2.40}$$

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh\left(\mathbf{s}^{(t)}\right) \tag{2.41}$$

There exist many variants of the internal structure of LSTMs as well as alternatives such as **gated recurrent units** (GRUs). In GRUs the number of parameters is reduced by simultaneously controlling the forgetting factor and the decision to update the weight. For further information see chapter 10 of the *Deep Learning Book* [20]. The above stated definitions are in accordance to the implementations used in the *Keras* API.

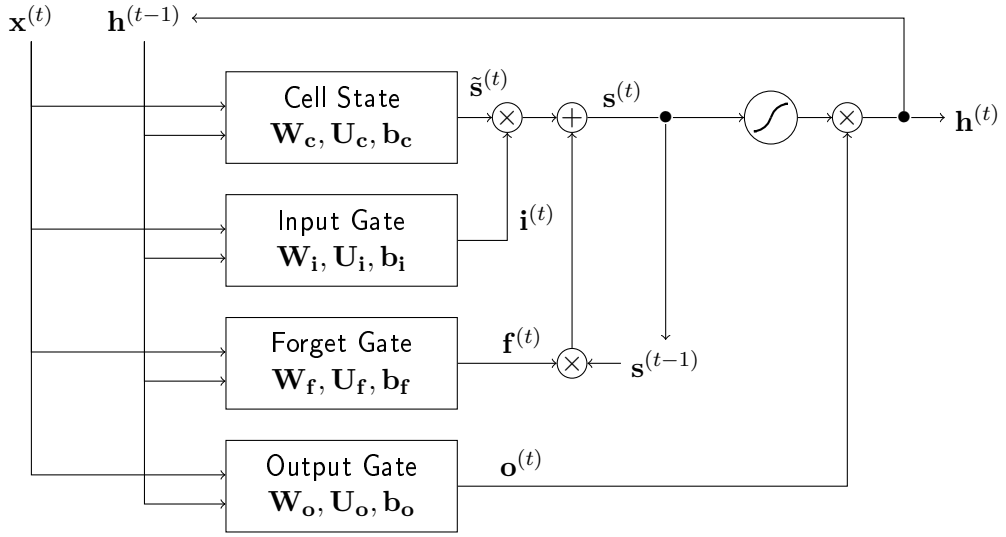

Figure 2.12 – Diagram of a single LSTM cell

To increase the amount of input information, recurrent neural networks are sometimes extended to **bidirectional** recurrent neural networks (BRNNs) [52]. For this the network is trained simultaneously in positive and negative time direction using separate forward and backwards layers, whose outputs are merged and fed to the next layer.

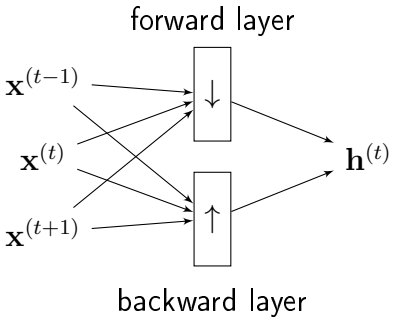

Figure 2.13 – Bidirectional RNNs employ seperate forward and backwards layers to increase the amount of input context.

Bidirectional RNNs have been successfully used for optical character recognition [51], speech recognition [53] as well as speech separation [36].

After this short introduction to deep learning, the next chapter will discuss how to use audio signals in the context of machine learning as well as signal analysis and synthesis for speech processing algorithms.

3 Spectro-Temporal Signal Representation

To exploit the time-frequency characteristics of the underlying signal components, many speech enhancement algorithms operate in the spectral or even the cepstral or modulation domain. Usually this is done by fourier transformation or filterbank analysis. Most algorithms can be divided into three coarse steps as depicted in figure 3.1. In the first stage the signal is analyzed by transformation into the frequency domain and in the second step the signal is modified by filtering out unwanted components. Finally the enhanced waveform is obtained by transforming the modified signal back to the time-domain [54, p.5].

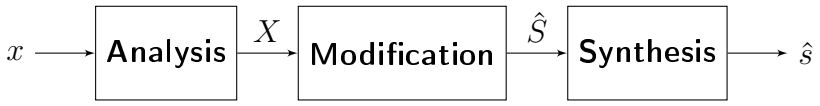

Figure 3.1 – Analysis-modification-synthesis framework as used in many speech enhancement algorithms.

In the case of machine learning based systems the transformed signal X can also be seen as the feature-space for the learning algorithm, this means that the signal representation at this point should also meet specific requirements for efficient learning, such as independency, size and a certain value range.
Chapter 3.1 will give a short introduction to fourier analysis and the spectrogram representation followed by a brief outline of its inversion. In chapter ?? the mel-weighted spectrogram will be introduced.

3.1 Spectrogram

The **Fourier Transform** [TODO cite speechcom] decomposes a time-domain signal into its spectral components. It is defined by the following formula:

$$X(\omega) = \mathcal{F}(x(t)) = \int_{-\infty}^{\infty} x(t) e^{-j\omega t} dt \tag{3.1}$$

Where the t denotes the time index and $\omega = 2\pi f$ the frequency.
The corresponding inverse transform is most commonly defined as

$$x(t) = \mathcal{F}^{-1}(X(\omega)) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\omega)e^{j\omega t}d\omega \tag{3.2}$$

The above mentioned Fourier Transform pair is only defined for continuous signals both in time and frequency.
When working with time-discrete signals, as it is the case with most digital signals and systems the **Discrete Fourier Transform** (DFT) is used. A finite signal of length N, sampled at a rate of f_s will result in a spectrum of frequency coefficients equally spaced by $\Delta f = \frac{f_s}{N}$.

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi kn}{N}} \tag{3.3}$$

$$x(n) = \frac{1}{N}\sum_{k=0}^{N-1} X(k)e^{j\frac{2\pi kn}{N}} \tag{3.4}$$

Here n denotes the time index and k the frequency index. When the DFT is carried out for consecutive frames the **Short-Time Fourier Transform** (STFT) can be computed. The frames are weighted by a window function $w(n)$ and shifted in time by a factor R, this yields the complex spectrum depending on the frame index l

$$X(l,k) = \mathbf{STFT}(x(n); w, R) = \sum_{n=lR}^{lR+N-1} x(n)w(n-lR)e^{-j\frac{2\pi k(n-lR)}{N}} \tag{3.5}$$

A widely used signal representation is the **spectrogram**. It is obtained by considering only the squared magnitude of the complex spectrum. Usually it is expressed in dB, in which case representation is also referred referred to as **log-power spectrum** (LPS).

$$\mathbf{spectrogram}(x(n); w, R) = 10 \cdot \log_{10}(|X(l,k)|^2) \tag{3.6}$$

As shown in figure 3.2 the spectrogram is commonly presented as a two dimensional plot with time on the x-axis and frequency on the y-axis. The magnitude is shown in dB:

Spectrogram Inversion To obtain the signal waveform from the complex STFT, each spectral frame is transformed back to the time domain via inverse DFT .The overlapping time signal frames are then weighted by a synthesis window s and summed together according to the frame shift.

$$x(n) = \mathbf{STFT}^{-1}(X(l,k); w, s, R) = \frac{1}{N}\frac{\sum_l s(n-lR)\sum_{k=0}^{N} X(l,k)e^{j\frac{2\pi k(n-lR)}{N}}}{\sum_l w(n-lR)} \tag{3.7}$$

To reconstruct the time signal from a spectrogram representation it has to be combined with the corresponding phase Φ.

$$X(l,k) = |X(l,k)| \cdot e^{j\Phi(l,k)} \tag{3.8}$$

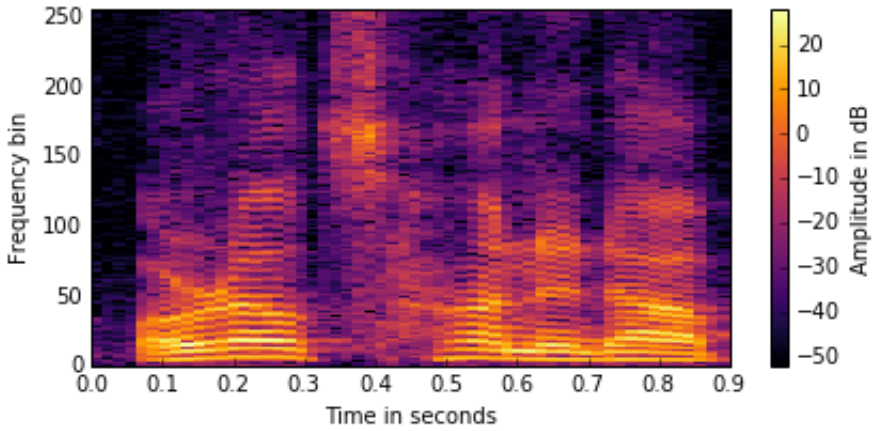

Figure 3.2 – Power spectrogram of a speech signal. The magnitude at a single time-frequency point is given as a color value.

3.2 Mel-Spectrum

The **mel scale** describes the perceptual pitch of a tone. It is based on listening experiments, which revealed that the intervals of equal pitch increments are increasing with frequency. For lower frequency bands, roughly below 500 Hz, the pitch intervalls are equally spaced while they increase gradually for higher frequencies. Since the mel scale is based on listening experiments, there exist mutiple versions of the conversion formula [55], [56].
Usually multiple frequency bins are combined by a wighting curve to form a single mel band [57]. Figure 3.3 shows the weighting curves for $M = 24$ mel bands.

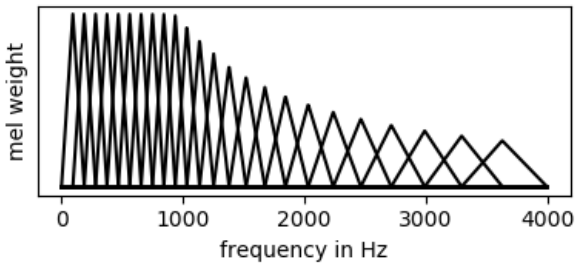

Figure 3.3 – Mel filter as a function of frequency.

To transform a N-point spectrum into the mel domain it is multiplied by a weighting matrix T of size $M \times N$, where M denotes the number of mel bands.

$$X_{mel} = 10 \log_{10} \left(T \cdot |X|^2 \right) \tag{3.9}$$

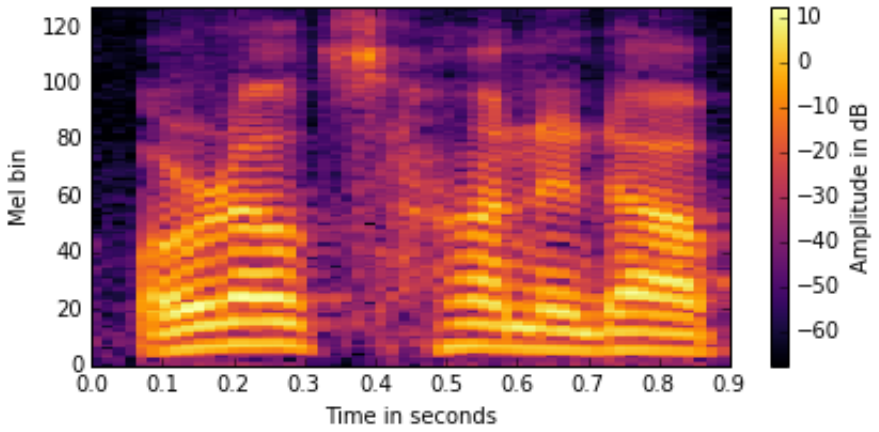

Figure 3.4 – Mel-spectrogram of a speech signal. The magnitude at a single time-frequency point is given as a color value.

4 Evaluation Criteria

To objectively evaluate the performance of speech enhancement systems, the following methods are used in the context of this work. The first two criteria, namely PESQ and STOI are full reference (FR) algorithms, meaning that they need to be provided with a reference signal for comparison. FR algorithms are considered to be more accurate than no reference (NR) systems, which only use the enhanced output signal for evaluation.

4.1 PESQ

PESQ stands for **Perceptual Evaluation of Speech Quality** and was originally developed by the International Telecommunication Union (ITU) as an objective method for speech quality assessment of telephone networks and speech codecs. Since it includes measurements of environmental noise at the sending side when used as a full reference algorithm, it is also a widely used tool for the evaluation of speech enhancement algorithms. The PESQ compares a reference signal with a degraded signal, which in our case is the noisy signal processed by the speech enhancement algorithm. First a series of delays between the two signals are computed for time alignment. Then they are compared by using a perceptual model which takes account of the perceptual frequency and loudness in the human auditory system.
The basic structure is depicted in figure 4.1. For our experiments, the implementation

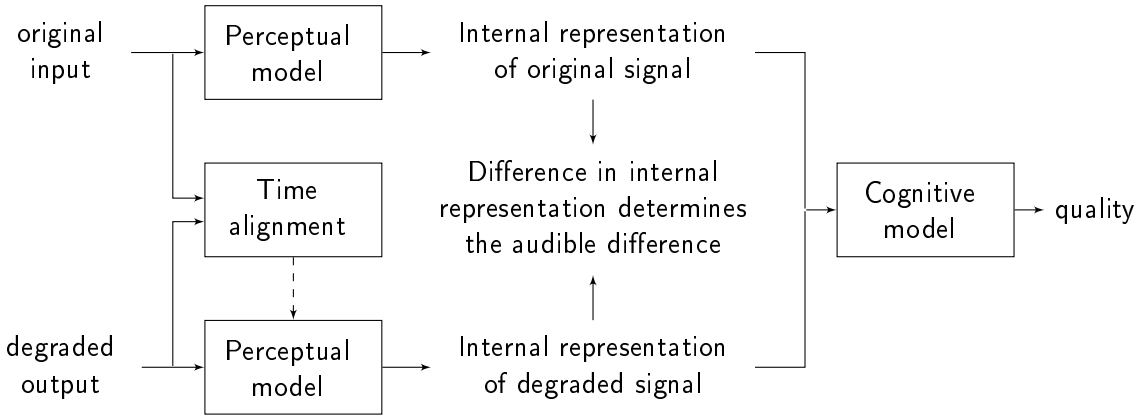

Figure 4.1 – Overview of the basic philiosophy used in PESQ [58]

as described in the ITU-T recommendation P.862 (02/2012) [58] is used. The algorithm reports the raw PESQ value ranging from -0.5 (worst) to 4.5 (best) as well as a mapping onto MOS-LQO (Mean Opinion Score - Listening Quality Objective) scale ranging from 1 to 5. For the results reported in the following chapters the latter will be used, unless otherwise stated.

4.2 STOI

Short-time objective intelligibility measure as described in [59] is an algorithm to predict the intelligibility of noisy speech.
It ranges from 0 (no intelligibilty) to 1 (perfect intelligibilty). In general it analyses short (approximately 400 ms) overlapping time segments and compares the clean and degraded speech signals based on a correlation coefficient between their temporal envelopes.
In a first step the signals are resampled to 10 kHz and framed into 50% overlapping frames with a length of 256 samples. The frames are weighted by a Hann-window and are zero-padded up to 512 samples. To exclude non-speech frames, silent regions are removed by analyzing the energy content within the clean speech frames. After transformation into the frequency domain, the DFT-bins are grouped to 15 one-third octave bands ranging from 150 Hz to 4.3 kHz.

$$X_j(m) = \sqrt{\sum_{k=k_1(j)}^{k_2(j)-1} |\hat{x}(k,m)|^2} \tag{4.1}$$

Where $\hat{x}(k,m)$ denotes the k-th frequency bin of the m-th clean speech frame and j denotes the band index. The band edges are denoted by k_1 and k_2. In the next step the single frames are grouped to short-time regions spanning $N = 30$ frames.

$$\mathbf{x}_{j,m} = [X_j(m-N+1), X_j(m-N+2), ..., X_j(m)]^T \tag{4.2}$$

To componsate for global level differences, the degraded speech regions $\mathbf{y}(n)$ are normalized and clipped.

$$\bar{\mathbf{y}}_{j,m}(n) = \min\left(\frac{\|\mathbf{x}_{j,m}\|}{\|\mathbf{y}_{j,m}\|}\mathbf{y}_{j,m}(n), (1+10^{-\beta/20})\mathbf{x}_{j,m}(n)\right) \tag{4.3}$$

where $\beta = -15$ dB and $\|\cdot\|$ represents the ℓ_2 norm.
The intermediate intelligibility measure for a time-frequency unit is then calculated as the sample correlation coefficient between the clean and the noisy vector.

$$d_{j,m} = \frac{\left(\mathbf{x}_{j,m} - \mu_{\mathbf{x}_{j,m}}\right)^T \left(\bar{\mathbf{y}}_{j,m} - \mu_{\bar{\mathbf{y}}_{j,m}}\right)}{\|\mathbf{x}_{j,m} - \mu_{\mathbf{x}_{j,m}}\| \|\bar{\mathbf{y}}_{j,m} - \mu_{\bar{\mathbf{y}}_{j,m}}\|} \tag{4.4}$$

where μ refers to the mean value of the corresponding vector.
Finally the result is given as the average over all bands and time-segments.

$$d = \frac{1}{JM}\sum_{j,m} d_{j,m} \tag{4.5}$$

4.3 Speech Accuracy Score

Finally the score utility from *Microsoft Speech Platform* is used to assess the performance of the presented algorithms in terms of speech intelligibility. The resulting score reports the accuracy of *Microsoft*'s speech recognition system and returns a percentage titled 'Speech Accuracy Score' and a text file containing pairs of recognitions and transcriptions. For this a noisy audio recording with a duration of 10 minutes, consisting of voice commands by different female, male and children speakers is provided. The raw audio file has a speech accuracy score of 41.9%.
The tool is used at *Harman* for benchmarking in the context of projects in cooperation with *Microsoft* and further details on the technical background are kept confidential.

4.4 Signal-to-Noise Ratio

The signal-to-noise ratio (SNR) compares to power of the wanted signal, in our case the speech signal, to the power of the noise signal and is commonly expressed in decibels.

$$\mathbf{SNR} = 10 \log_{10} \left(\frac{P_{signal}}{P_{noise}} \right) \tag{4.6}$$

It can be calculated using the summed squared magnitudes

$$\mathbf{SNR} = 10 \log_{10} \left(\frac{\sum_t s^2(t)}{\sum_t n^2(t)} \right) \tag{4.7}$$

Note that for this metric, the noise signal has to be known. Considering an additive noise model (see eq. 1.1) the noise signal can be expressed by the difference between the the noisy signal $x(t)$ and the clean speech signal.

$$\mathbf{SNR} = 10 \log_{10} \left(\frac{\sum_t s^2(t)}{\sum_t (x(t) - s(t))^2} \right) \tag{4.8}$$

5 | Time-Frequency Masks

To use deep learning for speech enhancement there are two commonly employed strategies. The first approach is to train the network directly on pairs of noisy and clean amplitude spectra and to combine the estimated amplitude and the noisy phase in the enhancement stage [32] [33]. The other approach is to use time-frequency masks as learning targets, which are then used to segregate the speech and noise components in the noisy spectrum [31] [60].
Time-frequency masks are widely used in computational auditory scene analysis (CASA) [61]. CASA systems analyze the acoustic input such as a cochleagram or correlogram by trying to segregate the different sources. This is often done by computing a mask weight for each time-frequency point, which emphasizes regions dominated by the target source and supresses those dominated by other sources. In this case the estimated signal is obtained by element-wise multiplication of the source signal and the time-frequency mask.

$$\hat{S}(t,f) = M(t,f) \odot X(t,f) \tag{5.1}$$

Where $\odot$ defines the hadamard-product or element-wise multiplication.

In the following three different types of spectral masks, commonly used for source separation and speech enhancement will be presented. Namely Ideal Binary Mask (IBM), Ideal Ratio Mask (IRM) and Phase Sensitive Filter (PSF). Finally the different time-frequency masks will be evaluated and compared in terms of their speech enhancement capabilities.

5.1 Ideal Binary Mask

The **ideal binary mask** or IBM introduced by Wang et al. [62], [63], classifies a single time-frequency bin of the mixture spectrum X as being associated with either the speech or non-speech component. If the local SNR at a single point (i,j) is larger than a local criterion ξ the corresponding mask-value is set to one otherwise it is considered as noise and will be zero to suppress the spectral magnitude at this specific point.

$$M^{\text{IBM}}(t,f) = \begin{cases} 1, & \text{for } SNR(t,f) > \xi \\ 0, & \text{for } SNR(t,f) < \xi \end{cases} \tag{5.2}$$

The local criterion ξ depends on the overall SNR of the mixture and a parameter α. Where 5 dB is a commonly used value for α [31].

$$\xi = SNR_{dB} - \alpha \tag{5.3}$$

The IBM is considered globally optimal in terms of SNR if the window function is rectangular and the time-frequency decomposition is orthogonal [64], [65]. Plot *(c)* in figure 5.2 shows the ideal binary mask for a speech signal *(a)* masked by babble noise *(b)*.

5.2 Ideal Ratio Mask

A commonly used spectral mask especially in the context of DNN based speech enhancement is the **ideal ratio mask** (IRM) [66], [31], [37]. In contrast to the IBM, the IRM belongs to the family of softmasks, meaning that its range is not constrained to binary values. When the phase of the underlying signals s and n are equal, the IRM is considered to produce optimal results [36]. It is closely related to the wiener filter, which is the optimal filter in the minimum mean-square error sense and is defined as follows

$$M^{\text{IRM}} = \frac{|S|^2}{|S|^2 + |N|^2} \tag{5.4}$$

If power spectral densities are used instead of spectral values and the speech and noise signals are uncorrelated, the above mentioned relation would match the wiener filter.
Several studies have found that the IRM generally performs better than the IBM in the context of source separation [31] [65] [36]. Plot *(d)* in figure 5.2 shows the ideal ratio mask for a speech signal *(a)* masked by babble noise *(b)*.

5.3 Phase Sensitive Filter

A different type of soft mask is the **phase sensitive filter**, which was introduced by Hakan Erdogan et al. and obtained promising results in the context of deep learning based speech separation [36]. The PSF is derived from the ideal complex mask $M^{\text{ICM}} = \frac{S}{X}$ by considering only the real part of the filter.

$$M^{\text{psf}} = \Re\left(\frac{S}{X}\right) = \Re\left(\frac{|S|}{|X|} e^{j(\phi_S - \phi_X)}\right) = \frac{|S|}{|X|}\cos\left(\phi_S - \phi_X\right) \tag{5.5}$$

For the practical use as a learning target for neural networks the value range will be constrained by clipping the values to $[0, 1]$. The resulting mask is also referred to as truncated PSF [1] and is illustrated in plot *(e)* of figure 5.2.

1. In the following, the acronym PSF stands exclusively for the truncated PSF.

5.4 Comparison

The described time-frequency masks are compared in an oracle scenario using a test set of 360 utterances and the above mentioned evaluation criteria. The test set includes four female and four male speakers and each of the utterances is superimposed with a noise signal resulting in 360 noisy utterances divided into five different signal to noise ratios, namely -5, 0, 5, 10 and 15 dB[2]. Using the noisy signal and the corresponding clean speech signal the individual time-frequency masks are calculated. The resulting enhanced signals are constructed by multiplying the noisy spectra and the corresponding spectral masks and subsequently transforming the spectra back to the time domain by inverse STFT.

Figure 5.1 shows the PESQ *(a)*, SNR *(b)* and STOI *(c)* for IBM, IRM and PSF in comparison to the noisy signals of the test data. The respective results are averaged for each of the four SNR levels in the test set.

Table 5.1 shows the delta values for the resulting PESQ, SNR and STOI for the indi-

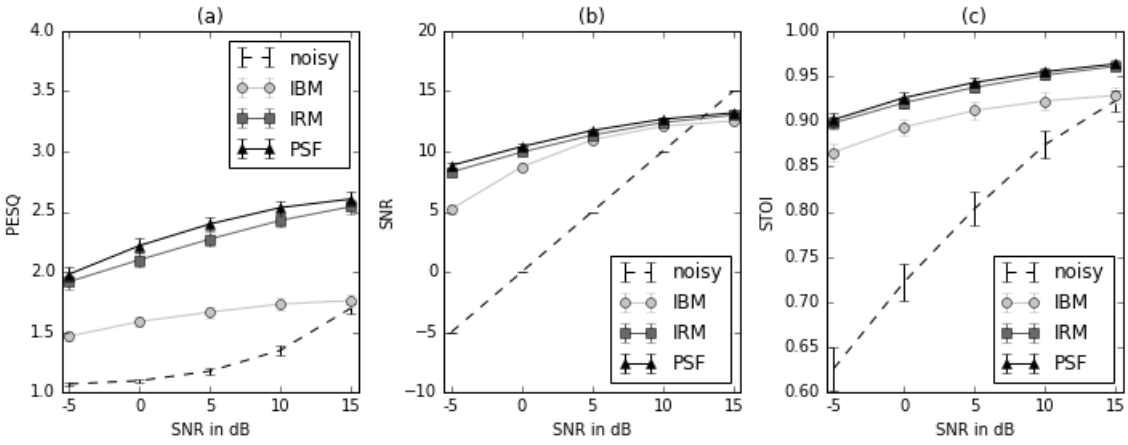

Figure 5.1 – Comparison of different spectral masks. For the metrics PESQ, SNR and STOI, the mean values as well as the 95% confidence intervals are indicated as measured for a test sest of 360 sentences with different SNRs

vidual mask approaches. For each of the resulting enhanced utterances the difference to the original noisy signal is calculated and the delta values are given as the mean value over all measurements.

It can be observed that in terms of speech quality and intelligibility, as measured by the PESQ and STOI metrics, the softmasks clearly outperform the binary mask. As for the SNR of the resulting signals the difference is only significant for severe noise conditions below 0 dB SNR. While the differences are clearly visible for the PESQ results, the performance of the IBM in terms of the resulting SNR improves with better overall SNR of the original noisy utterances. Additionally, the IRM and PSF results are more independent from the SNR of the input signal than the IBM. When looking at the average results for the STOI measurements in figure 5.1 there is almost no visible difference between the

2. For a detailed description of test data, see chapter 6.2

Mask	ΔPESQ	ΔSNR	ΔSTOI
IBM	0.4	5.2	0.14
IRM	1.0	6.5	0.17
PSF	**1.1**	**7.0**	**0.17**

Table 5.1 – PESQ results for different oracle masks at various SNR levels.

performance of the two softmasks and a difference of about 0.4 to that of the binary mask.
The delta values in table 5.1 show that the PSF performs best for all of the three evaluation criteria closely followed by the IRM.
The mentioned results are consistent with the results presented in [31] and [36], which also shows that the phase sensitive filter may be superior to the ideal ratio mask not only in terms of speech separation, but also for speech enhancement tasks.

Figure 5.2 additionally shows a clean speech signal *(a)* and the same signal superimposed with babble noise at 5 dB SNR *(b)* as well as the corresponding IBM *(c)*, IRM *(d)* and PSF *(e)*.

In the next chapter it will be presented how deep neural networks can be used for speech enhancement and the general framework for the final experiments will be introduced.

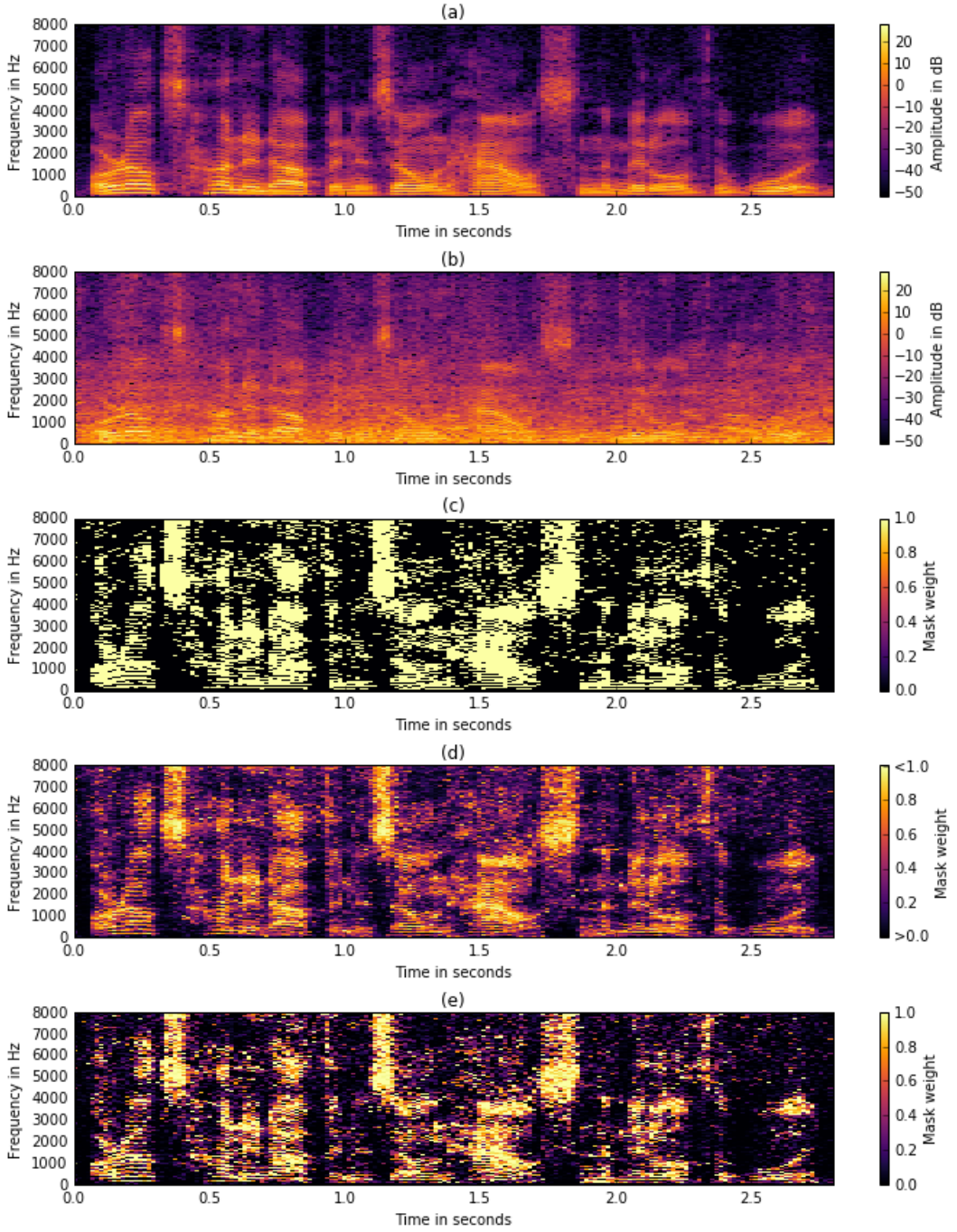

Figure 5.2 – Illustration of different oracle masks. A clean speech signal *(a)* is superimposed with babble noise at 5 dB SNR, which results in the noisy signal in *(b)*. Plots *(c)*, *(d)* and *(e)* show the corresponding IBM, IRM and PSF respectively.

6 Experimental Setup

This chapter will give an overview about the general framework for the conducted experiments. First the basic speech enhancement approach is introduced followed by a description of the dataset and the generation of the training and test data. Furthermore, an explanation of the feature extraction and training procedures will be given along with a short description of the baseline algorithm used for comparison.

6.1 Proposed Framework

For the experiments conducted in the context of this work, the following framework is used. The neural network models are trained to predict spectral masks from noisy speech input. For that the model is fed with features extracted from noisy speech i.e. lps or mel spectra. After training, the neural network model is used to predict the time-frequency mask for the noisy input features frame by frame. Figure 6.1 shows the proposed speech enhancement method. The upper half of the diagram depicts the training process of the neural network model and the lower half shows the enhancement and the testing framework respectively.
From the noisy time signal the spectrum is obtained by Fourier analysis. Then the input features for the neural network are extracted and standardized using the global mean and variance of the training data.

$$X = \frac{X - \mu_{train}}{\sigma^2_{train}} \tag{6.1}$$

A detailed explanation of the feature extraction process as well as the estimation of the global mean and variance of the training data is given in chapter 6.3.
The enhanced complex spectrum is obtained by element-wise multiplication of the estimated spectral mask with the complex spectrum of the noisy signal. Finally, the enhanced speech spectrum is obtained by inverse fourier transform.

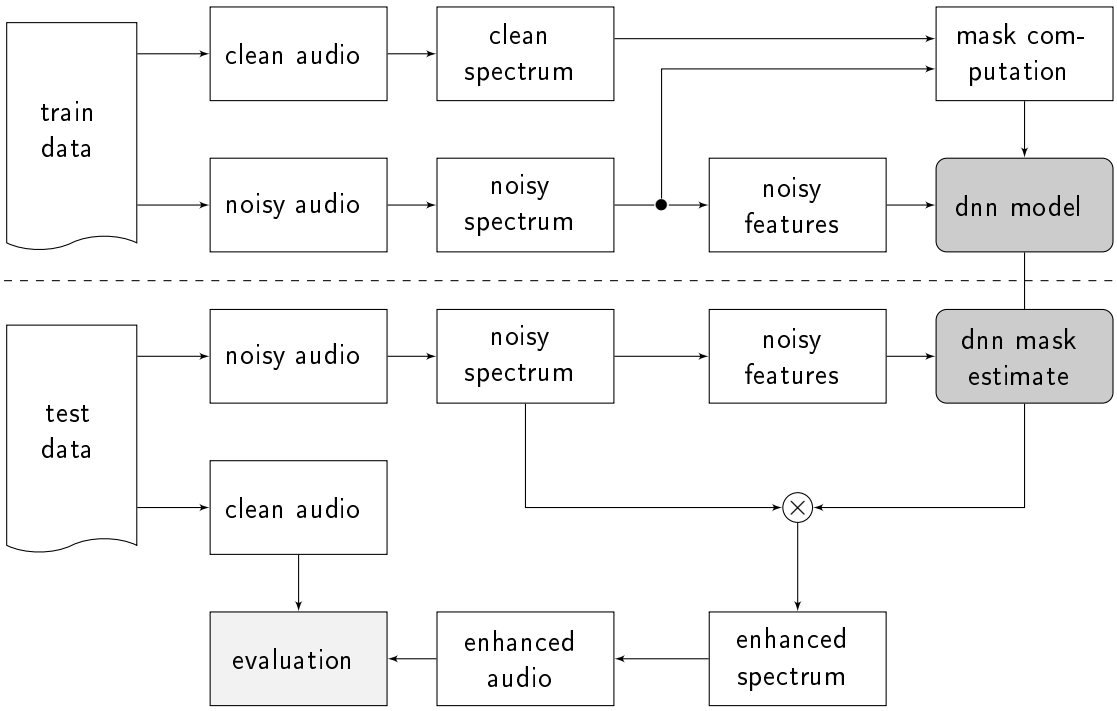

Figure 6.1 – Proposed speech enhancement framework. Features are extracted from the noisy spectrogram and then fed into the neural network model. Subsequently the predicted spectral mask is multiplied by the noisy spectrogram to attain an estimate of the clean speech spectrogram. Finally the enhanced speech signal is obtained by inverse FFT and overlap and add.

To provide the model with information about the temporal context, the input features will span multiple time frames corresponding to exactly one frame of the respective spectral mask. More specifically, a context size of 11 frames was chosen for the experiments and, depending on the particular ANN model, the output will either correspond to the 6th (center) or 11th (end) input frame. It should be noted, that the target mask is estimated frame by frame, which means that there is no temporal dependency between successive target predictions.

6.2 Dataset

For training and testing of the neural network models a large quantity of speech and noise data is needed. Since the goal is to estimate the clean speech signal from a corresponding noisy speech signal, the collected speech data is combined with noise signals to form a large number of noisy speech signals, which can be seen as the input data to the model. In the following the composition of the speech and noise corpora as well as the generation of the noisy speech data will be described.

Speech data The dataset is based on 360 hours of speech data from the *LibriSpeech* corpus [67], which consists of read English speech derived from audiobooks from the *LibriVox* [68] project. In particular 154 female and 154 male speakers are randomly chosen from the clean training set (`train-clean-360.tar.gz` [68]). From each of the speakers, five different utterances each with a duration of 10s were chosen, resulting in a total of 1500 clean utterances for the training data and 40 utterances for the test data. The IDs of the used speakers and utterances are listed in the appendix.

Noise data A combination of the TUT database for acoustice scene classification [69] and background noises from the QUT-NOISE-TIMIT database [70] is used to form a noise corpus with an overall duration of more than 29 hours. The corpus is divided into 7 categories namely: *car*, *city*, *cocktail party*, *home*, *nature*, *office* and *traffic*. Additionally, an eighth category *transient* is formed by using multiple types of non-stationary noises such as ring-tones, dial-tones, sirens etc. manually collected from `http://freesound.org`.
The choice of noise categories is based on what we considered typical environments for the use of a lifestyle audio product. Of course the structure of the available data also played a role, since we wanted each noise category to consist of roughly the same amount of data.
For testing additional noise data was collected from the NOISEX-92 database [71]. The choice of noise data for the test set is based on the fact that the NOISEX-92 database is a popular database for the evaluation of noise reduction algorithms and furthermore offers noise scenarios that differ greatly from those used for the training data, which makes it possible to evaluate the respective deep learning models in terms of their generalization capabilities. Table 6.1 lists the different noise categories contained in the training and test data.

Train	Test
car	babble$*$
city	buccaneer$*$
cocktail party	destroyerengine$*$
home	destroyerops$*$
nature	f16$*$
office	factory$*$
traffic	leopard $*$
transient	machinegun $*$
	workshop

Table 6.1 – Noise categories contained in the data set. The categories marked by $*$are taken from the NOISEX-92 database.

Noisy speech data The clean utterances are overlayed by segments of the noise sounds at a fixed set of SNR-levels. In particular the training set compromises SNR-levels of 10, 5, 0 and -2 dB. It should be noted that since the noise database contains not only

stationary but also non-stationary and even short transient sounds, the momentary or short-term SNR can differ drastically from the above stated SNR-levels.
Clean and noisy speech were summed together using a scaling factor α

$$x[n] = s[n] + \alpha n[n] \tag{6.2}$$

which is determined by the desired SNR-level in dB SNR_{dB}

$$\alpha = \sqrt{\frac{\sum_n s^2[n]}{\sum_n n^2[n] 10^{\frac{SNR_{dB}}{10}}}} \tag{6.3}$$

Five utterances, each with a duration of ten seconds, were chosen randomly from each of the speakers and superimposed by the noise sounds. The procedure, in which the training data was created is depicted in figure 6.2. For every single noise category four different segments were selected to cover the different SNR-levels. This results in 32 different noisy utterances for each clean utterance in the training set. Therefore $2 \times 150 \times 5 = 1500$ clean utterances result in $1500 \times 8 \times 4 = 48000$ noisy utterances, which correspond to 133 hours of audio material.

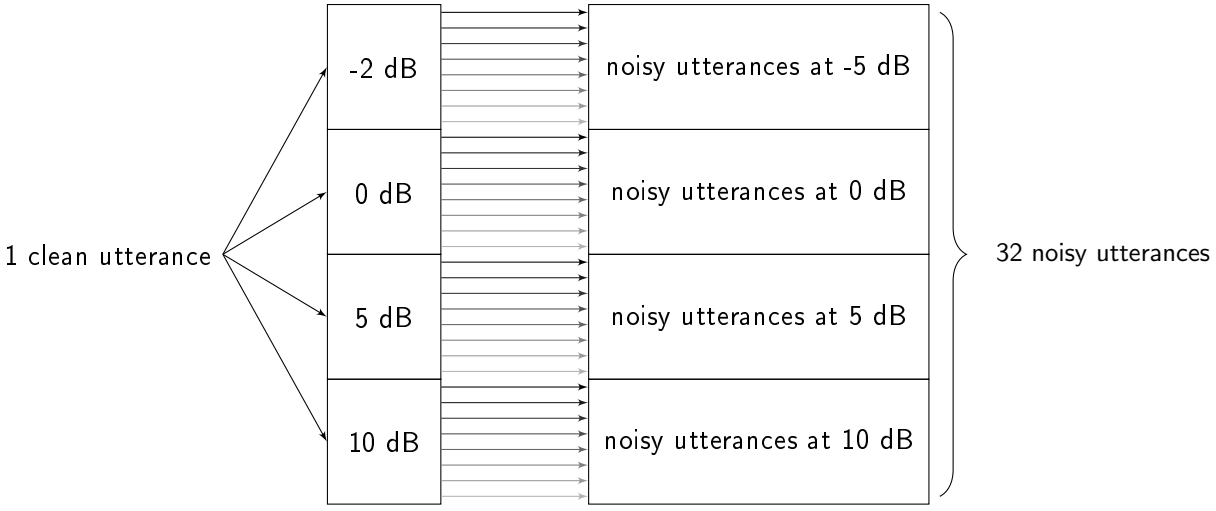

Figure 6.2 – Creation of the noisy training data. A single utterance is superimposed with noise data at four different SNR levels. From each category, four different noise segments are chosen randomly to be superimposed with the clean utterance.

For the test data a slightly different procedure was used to generate the noisy utterances. To minimize the time required for evaluation of the models, only 40 different clean utterances are included in the test data and each one was used only 9 times for the noisy test data resulting in $40 \times 9 = 360$ noisy utterances. Instead of using one random noise segment from each category for each SNR level, the single clean utterances are combined with exactly one random segment from each category and the SNR level for each noisy utterance was selected alternatingly between 15, 10, 5, 0 and -5 dB.

6.3 Feature Extraction

The features and targets are extracted in multiple stages. At first the complex spectrograms are computed for each pair of noisy and clean audio, using a hann window of 32ms length and an overlap of 50%. Subsequently, the noisy spectral features as well as the targets i.e. spectral masks are calculated. Both log power spectra and mel spectra were implemented for later comparison. The feature frames are now combined to feature matrices spanning $M = 11$ frames, each corresponding to exactly one target frame. The

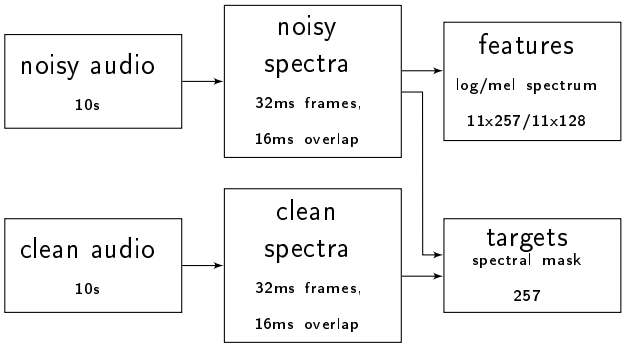

Figure 6.3

choice of the input length is based on studies by Lu et al. [25] and Xu et al. [33], where different context sizes were evaluated. It was shown that the performance improved with the number of input frames and that 'more acoustic context information could smooth the enhanced speech to obtain better hearing sense'. Furthermore, it was observed that the significance of the context size was especially high for low SNR levels. On the downside, more input context also leads to higher complexity regarding the network structures and a longer initial delay in the case of real time implementation and in terms of PESQ and STOI, Xu et. al reported no significant improvement for context sizes larger than 11 frames. However, the above mentioned studies evaluated the context size only for the use with DAEs and feed forward networks and while a context of 11 frames was used in combination with CNNs as well [38], there is no definitive coherence of the above mentioned results with the optimum context size for RNN models. Hence, it would be useful to further investigate the optimal sequence length for RNNs in the context of speech enhancement.
For feature standardization during training and testing the mean and variance over the complete training set is needed. To save computation time they are computed during the feature extraction process using the algorithm described in [72] by Chan et al., which makes it possible to compute the mean and variance in parallel by combining the statistics of multiple subsets X_A, X_B of the complete set of data X. It is described below in algorithm 2.

Algorithm 2 Pairwise mean and variance

Require: Mean values μ_A, μ_B
Require: Variance values σ_A^2, σ_B^2
Require: Sample sizes n_A, n_B
$\delta = \mu_A - \mu_B$
$m_A = \sigma_A^2(n_A - 1)$
$m_B = \sigma_B^2(n_B - 1)$
$M = m_A + m_B + \delta^2 \frac{n_A n_B}{n_A + n_B}$
$\mu = \frac{\mu_A n_A + \mu_B n_B}{n_A + n_B}$
$\sigma^2 = \frac{M}{n_A + n_B - 1}$
return μ, σ^2, combined statistics

6.4 Training Stage

The training procedure was the same for all of the performed experiments as described in chapter 7. A batch size of 128 was chosen empirically to efficiently use the hardware resources [1]. As long as there is enough memory available, larger batch sizes lead to more efficiency and speed up the training process. Furthermore a larger batch represents a more accurate estimate of the gradient. Smaller batch sizes on the other hand, can have a regularization effect and lead to better generalization [20, ch.8].
For optimization, *Adam* was chosen since it converged faster than other optimization algorithms in most of the initial experiments. Since the complete training set was too large to fit into the memory, the data was divided into 100 smaller chunks, each holding around 300.000 observations which corresponds to circa 4 GB. From these chunks the minibatches were provided during training until each sample was seen once and the next chunk was loaded into memory. Before dividing the whole dataset into chunks, the observations were shuffled, additionally the order of the chunks as well as the individual content was permuted as well during each training epoch.
Unless otherwise stated, the mean squared error between the true and the predicted mask output was used as the loss function.
All of the experiments were implemented in *Python 3* and *Keras* with *Tensorflow* [73] as backend.

6.5 Baseline Algorithm

The baseline speech enhancement algorithm is used for comparison with the proposed framework. It is based on a modified wiener filter approach and uses noise shaping to achieve a time-frequency varying attenuation at lower frequencies [74]. Furthermore a noise power spectral density estimation algorithm is used which is based on a multiplicative estimator utilizing multiple increment and decrement time-constants. The

1. For the experiments a pc with 128 mb memory and two *TITAN Xp* graphic cards was available.

time-constants are chosen by observing the long-term trend of the noisy input spectrum [75]. The mentioned approaches, which are originally designed for the use in automobile environments, are modified to maximize the performance for the reference file needed for scoring the word accuracy. The algorithm is currently under development by *Harman*, and further details on the design and implementation are kept confidential.

This chapter gave an overview of the experimental setup and the proposed speech enhancement framework. The next chapter will give a description of the carried out experiments as well as an evaluation of the proposed algorithm in terms of the mentioned evaluation criteria.

7 Experiments and Results

In this chapter the speech enhancement method is tested with a number of different neural network models and the results are compared to those of the baseline method. First a basic LSTM network is investigated followed by an evaluation of different feature target combinations as well as a comparison of different layer sizes. The LSTM model is then extended to employ bidirectional layers and the outcome is discussed and compared to the previous results. After that a novel loss function is introduced and its impact on the performance of the speech enhancement system is investigated. Furthermore, an extended data set, tailored to match the *Cortana* reference file, was used for training and the resulting model was evaluated. Finally, the mentioned methods are compared to a CNN based approach and the overall results are summarized.

7.1 LSTM Networks

For the experiments regarding LSTM units the general network structure is based on [35] and [36], where the authors investigated recurrent neural networks for single-channel speech separation. For all the experiments, the audio data is sampled at 16 kHz and for DFT computations a hann window of 32 ms and an overlap of 50% is used. The basic network is depicted in figure 7.1 and consists of two LSTM layers with 512 hidden units followed by an additional dense layer with 257 hidden units.

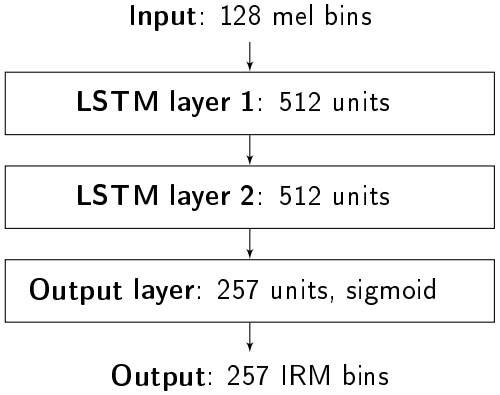

Figure 7.1 – Structure of the investigated LSTM network.

Sigmoid activation is used in the output layer since it ensures the output values to be between 0 and 1, which matches the value range of the IRM. The input features are 128-point mel-spectra and as targets ideal ratio masks with 257 bins are used.
A time sequence of 11 time frames consisting of the current frame and the 10 previous timesteps is used as the input to the model and the spectral mask of the current frame is used as the corresponding output. Although it is theoretically possible to train RNNs with time sequences of variable length, *Keras* expects the input shape to be constant. Additionally a fixed size of the acoustic context allows the dataset to be used for training of other network topologies as well.
To improve the generalization capabilities of the model, dropout of 0.2 is applied to the first two recurrent layers. In an initial experiment it was observed that when using *Adam* the validation loss did not improved after the first epoch (see figure 7.2). This led to decision to train the models exclusively for one epoch in order to minimize the time required for the following experiments.

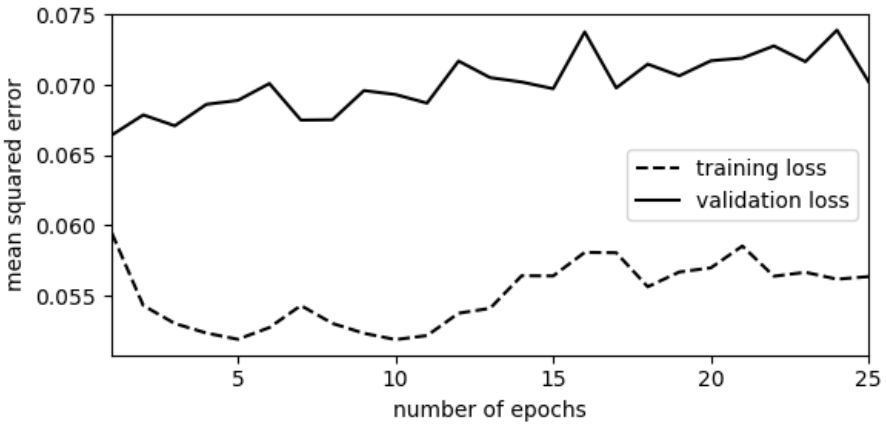

Figure 7.2 – Validation loss and training loss for 25 epochs of training of an LSTM network using *Adam*.

Results The performance of the network is evaluated in terms of the speech accuracy score, and the above mentioned evaluation criteria PESQ and STOI. Figure 7.3 shows the results over the mismatched test set consisting of 360 utterances as the mean value at five different SNR levels. As comparison the metrics for the unprocessed noisy signals, the signals enhanced using the oracle IRM and the results of the baseline algorithm are shown as well. It can be observed that the proposed system outperforms the baseline algorithm in terms of PESQ and STOI significantly. The perceived quality improvement measured by PESQ is relatively small at drastic noise scenarios with SNR levels lower than 0 dB and gets more salient for higher SNR levels but with a distinct increase of the confidence interval. As for the perceived intelligibility the improvements are more distinct at lower SNR levels with the results for the RNN model being 5% higher than the unprocessed speech at -5 dB but with a larger variability. Interestingly, the baseline algorithm seems to decrease intelligibility with an overall STOI lower than the original noisy speech. These measurements indicate that the proposed algorithm performs better

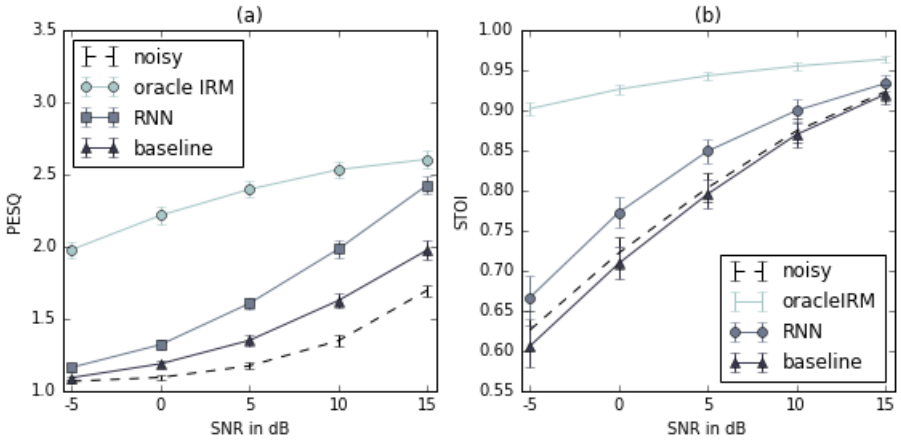

Figure 7.3 – Objective evaluation results for the proposed RNN model, baseline algorithm, oracle IRM and unprocessed speech. PESQ (a) and STOI (b) were measured for a test set of 360 utterances. The plot shows the mean values and confidence intervals for five different SNR levels.

than the baseline algorithm both in terms of speech quality and speech intelligibility. However, the baseline system achieved a higher speech accuracy score with 76% compared to the proposed algorithm with 72%. When considering the fact that the baseline algorithm was tuned using the reference file for the speech accuracy tool, this can still be seen as a satisfactory result. Furthermore, it has to be noted that the speech accuracy score is evaluated using the reference file, which is composed of only a specific kind of babble noise and is therefore not necessarily suitable to measure the general speech enhancement performance. Empirical observations of the resulting audio signals showed

method	acc [%]	ΔPESQ	ΔSTOI
baseline	**76**	0.2	-0.01
LSTM 2x512	72	**0.4**	**0.04**

Table 7.1 – Objective evaluation results for the proposed RNN model and the baseline algorithm. PESQ and STOI are given as the delta values, calculated by averaging the difference between the noisy and enhanced signals from the test set.

that the perceived speech quality is indeed higher for the RNN results compared to the baseline results. Speech processed with the baseline algorithm exhibits high amounts of musical noise especially for low SNR levels. The RNN system on the other hands shows no musical noise even at severe noise conditions. Similar observations can be made when looking at the spectrograms of the resulting signals. Figure 7.4 shows the resulting spectra for the baseline algorithm, the mentioned LSTM model and the corresponding oracle IBM for 3 seconds of speech degraded by babble noise at an overall SNR of 0dB. It can be seen that, in contrast to the baseline algorithm, the result of the neural network is fairly similar to that produced by the oracle mask and to the clean reference signal, respectively.

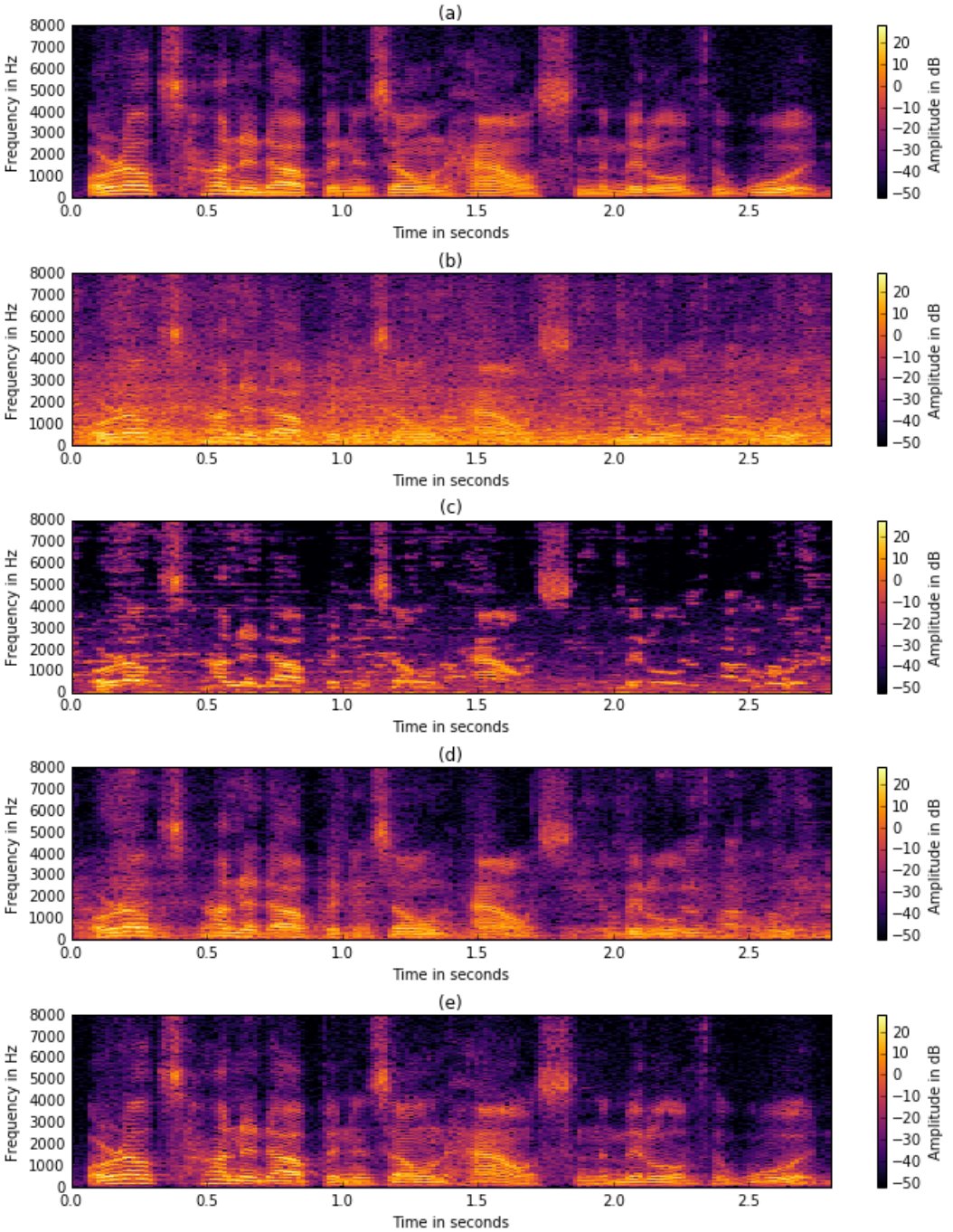

Figure 7.4 – Result of the proposed LSTM network for a speech signal mixed with babble noise at 0dB. Plots *(a)* and *(b)* show the clean and noisy signal respectively. Plot *(c)* the shows the noisy signal processed by the baseline algorithm and plot *(d)* the result of the proposed LSTM system. In *(e)* the noisy signal is multiplied by the oracle IRM.

In table 7.2 the PESQ and STOI improvements are indicated for each of the noise categories included in the test set. It can be seen that the biggest improvements compared to the baseline system are achieved for noise categories, which involve transient signals such as 'machine gun' and 'workshop'. Categories for which the differences between the proposed model and the baseline algorithm are not significant, e.g. the categories 'f16' and 'leopard', typically consist of more stationary and narrow-band noise. The drop in performance for these noise types could be explained by the fact that the predictions of the proposed system are based on an acoustic context spanning 11 overlapping frames, which in some cases may not be enough to distinguish between the harmonic structure of a vowel and that of a sinusoidal masker. This issue could be addressed by applying a recursive evaluation scheme for the mask weights similar to [3] [76], by which information from the preceding mask predictions are taken into account.

noise type	method	ΔPESQ	ΔSTOI
babble	LSTM	0.3 (±0.06)	0.02 (±0.005)
	baseline	0.1 (±0.03)	-0.03 (±0.005)
buccaneer	LSTM	0.5 (±0.07)	0.04 (±0.005)
	baseline	0.2 (±0.04)	-0.01 (±0.003)
destroyer engine	LSTM	0.4 (±0.06)	0.04 (±0.006)
	baseline	0.3 (±0.05)	0.01 (±0.002)
destroyer ops	LSTM	0.4 (±0.06)	0.03 (±0.004)
	baseline	0.2 (±0.03)	-0.02 (±0.002)
f16	LSTM	0.5 (±0.06)	0.05 (±0.006)
	baseline	0.3 (±0.05)	0.01 (±0.003)
factory	LSTM	0.5 (±0.07)	0.04 (±0.006)
	baseline	0.2 (±0.04)	-0.02 (±0.003)
leopard	LSTM	0.4 (±0.06)	0.03 (±0.004)
	baseline	0.4 (±0.05)	-0.01 (±0.001)
machine gun	LSTM	0.3 (±0.04)	0.03 (±0.006)
	baseline	-0.1 (±0.02)	-0.01 (±0.001)
workshop	LSTM	0.5 (±0.07)	0.05 (±0.007)
	baseline	-0.0 (±0.04)	-0.01 (±0.003)

Table 7.2 – Comparison of the proposed LSTM network with the baseline algorithm. The results of the measured evaluation criteria are shown individually for each of the categories from the test. The 95% confidence interval is given in brackets next to the mean value.

7.1.1 Comparison of Features

In this experiments different feature target combinations for the described network were implemented and evaluated. The phase sensitive filter as described in chapter 5 was used in [36] in the context of speaker seperation. The authors reported the PSF to perform better than the IRM in terms of signal to distortion ratio. Similar to the experiments in chapter 5 the spectral masks were compared in an oracle scenario, meaning that the optimum masks calculated from the reference signal were used. However, it is not proven that these results correlate with the suitability of the respective time-frequency masks as learning targets for deep learning models.
Table 7.3 shows the result for four different combinations. The column 'features' represents the input features to the neural network model, the column 'targets' represents the type of spectral mask used as the output feature.

model	features	targets	acc [%]	ΔPESQ	ΔSTOI
LSTM 2x512	lps	psf	64	0.3	0.04
LSTM 2x512	lps	irm	70	0.3	0.04
LSTM 2x512	mel	psf	67	0.3	**0.05**
LSTM 2x512	mel	irm	**72**	**0.4**	0.04

Table 7.3 – Objective evaluation of different feature-target combinations.

The results are fairly similar for all combinations in terms of PESQ and STOI and the best word accuracy scores are obtained using IRM as output target. The observed superiority of the IRM over the PSF mask contradicts the oracle experiment in chapter 5, where the PSF performed best in terms of PESQ and SNR. This leads to the assumption that although the PSF is theoretically better in terms of the resulting speech quality, the IRM is more suitable as a learning target since it only requires information about the magnitude, which is more or less explicitly provided by the input features. In contrast, the phase information included in the PSF is not explicitly given by the model input and can only be estimated from the harmonic structure of the input spectrum. In table 7.3 it can also be seen that in general, mel spectra achieve better results than log power spectra except for the STOI values, where PSF masks yield the best results. However, from the best to the worst result for the STOI metric the difference is only 0.8%.
Figure 7.5 shows the PESQ measurements in detail. The results are averaged over each of the four different SNR levels.

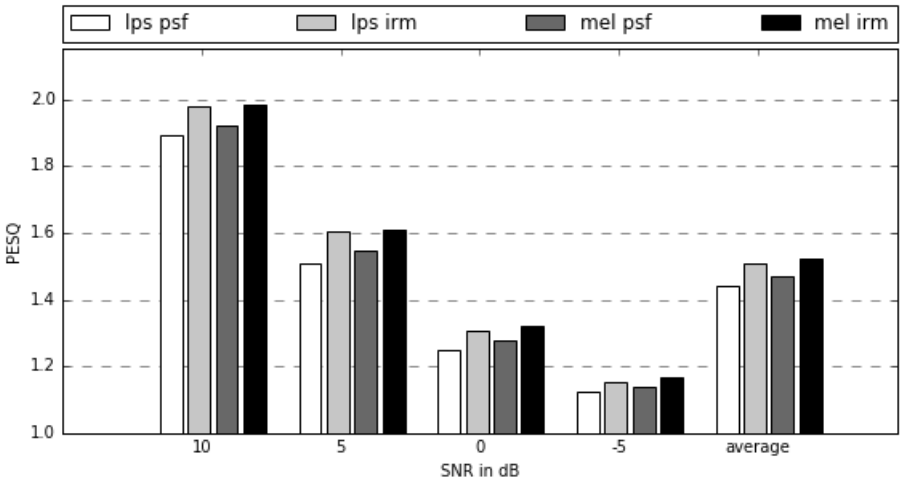

Figure 7.5 – PESQ measured for different feature combinations averaged over four different SNR levels.

It can be observed that the relations between the different feature combinations stay fairly equal over all noise levels, since the tendencies are independent from the underlying SNR level. However, since the results are more articulate for higher SNR levels, it can be concluded that, the lower the SNR of the test signal, the lower the differences between the feature combinations.
The results for the STOI metric is shown in figure 7.6. As already observed in table 7.3, the results are very similar for all four feature combinations, with the combination mel spectra and phase sensitive filter performing best for each of the four SNR levels.

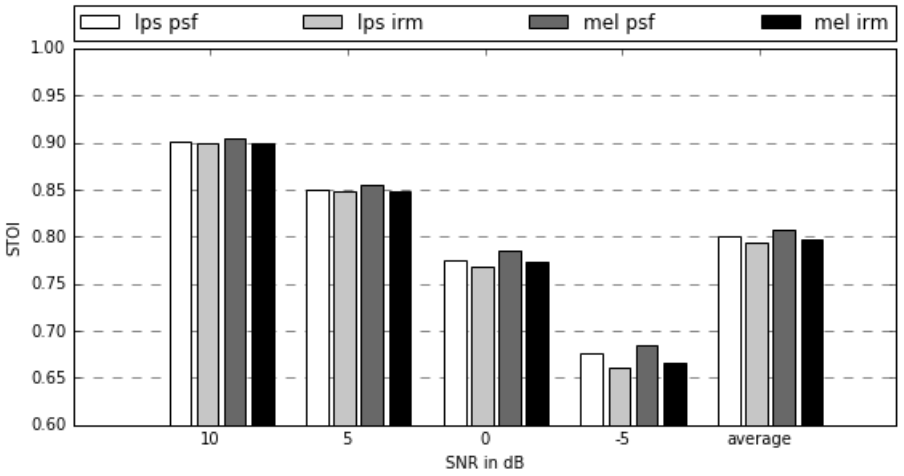

Figure 7.6 – STOI measured for different feature combinations averaged over four different SNR levels.

7.1.2 Comparison of Layer Sizes

For this experiments the above mentioned network consisting of two LSTM layers and a sigmoid output layer was used to evaluate the performance for different layer sizes. Mel spectra were used as input features and ideal ratio masks as output features, since this configuration showed the most promising results in the previous experiment. Table 7.5 shows the achieved speech accuracy score as well as the average results for PESQ and STOI for different numbers of hidden units in the first two layers.
When looking at the speech accuracy score it is clearly visible that the performance improves with the increase in the number of hidden units per layer.

model	features	targets	acc [%]	ΔPESQ	ΔSTOI
LSTM 2x64	mel	irm	69	0.2	0.03
LSTM 2x128	mel	irm	69	0.3	0.03
LSTM 2x256	mel	irm	71	0.3	0.04
LSTM 2x384	mel	irm	70	0.4	0.05
LSTM 2x512	mel	irm	72	0.4	0.04
LSTM 2x1024	mel	irm	**74**	**0.4**	**0.05**

Table 7.4 – Objective evaluation of different layer sizes

The other metrics, also shown in figures 7.7 and 7.8, show a clear rise in performance from 64 to 256 hidden units. Interestingely PESQ and STOI show a dip in performance at 2x512 hidden units while the development stays linear for the speech accuracy score at this configuration.
However, above 384 hidden units a saturation can be observed, with only minor differences in terms of PESQ and STOI. This is an important observation when considering that total number of parameters increases exponentially in relation to the number of hidden units (see figure 7.9). As an example, an increase from 256 to 512 hidden units per layer results in an increase from 985,601 to 3,543,809 total parameters for the corresponding NN model.

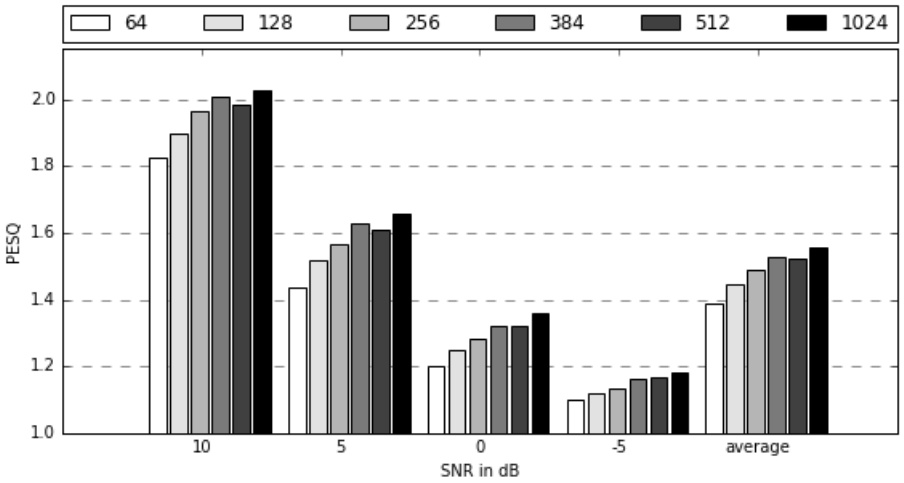

Figure 7.7 – PESQ measured for different LSTM layer sizes averaged over four different SNR levels.

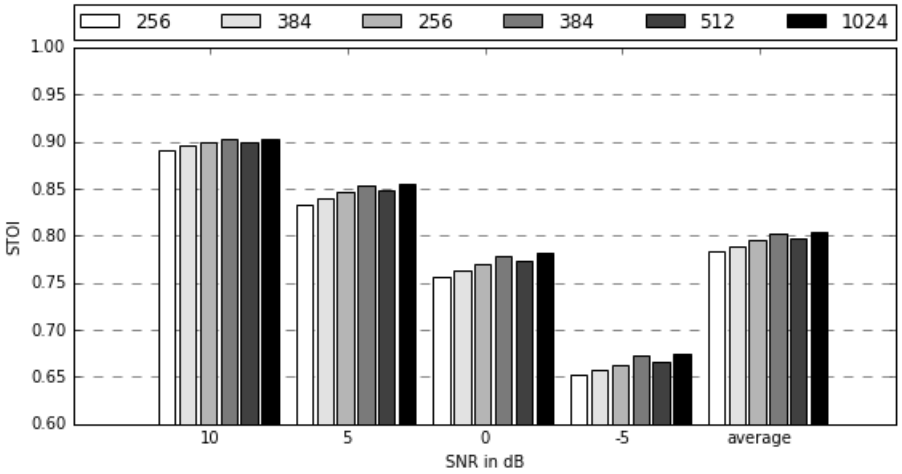

Figure 7.8 – STOI measured for different LSTM layer sizes averaged over four different SNR levels.

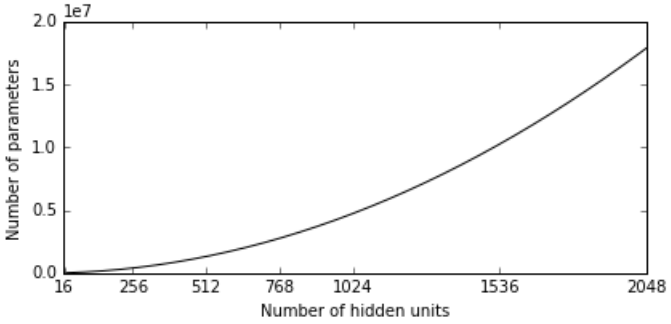

Figure 7.9 – Number of parameters of an LSTM layer as a function of the layer size.

7.2 Bidirectional LSTMs

The above stated network structure is now modified to employ bidirectional LSTMs, since they are considered to produce better results especially in the context of speech processing [52], [36], [53]. Figure 7.10 shows the measurements for a bidirectional LSTM network consisting of two layers with 512 hidden units each. Since each layer incorporates a forward layer as well as a backward layer the number of hidden units doubles, resulting in a total of 1024 trainable units. As a reference the measurements are compared to that of a LSTM network with 1024 hidden units per layer. Again the mismatched test set consisting of 360 utterances is used for evaluation and the results are given as the mean value over each of the four different SNR levels. It can be observed

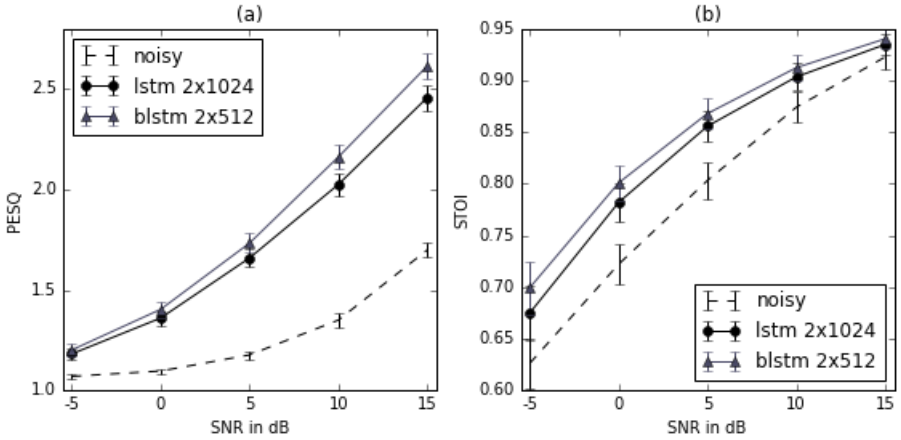

Figure 7.10 – Objective evaluation results for an LSTM network with 1024 hidden units per layer, BLSTM network with 512 hidden units per layer and unprocessed speech. PESQ (a) and STOI (b) were measured for a test set of 360 utterances.

that the bidirectional network performs slightly better than the standard LSTM network. In comparison to the noisy test signals, the BLSTM model achieved an improvement of 0.54 in terms of PESQ and 0.05 in terms of STOI. In comparison the LSTM improved the PESQ by 0.46 and the STOI by 0.04. The measurements correlate with the speech accuracy score where the BLSTM network obtains 74.1% compared to 73.7% for the LSTM network. Furthermore it should be noted that the BLSTM network has a smaller parameter space than the LSTM network with 9.4 million compared to around 13.4 million. This means that the use of bidirectional layers improves the performance while reducing the total number of parameters of the NN model, which is especially important in case the hardware resources are limited e.g. for FPGA implementations.

7.2.1 Comparison of Layer Sizes

Different feature combinations namely mel/psf and mel/irm as well as different layer sizes were investigated similar to the previous experiments. The results are shown in table 7.5. Note that in the case of bidirectional LSTMs the number of hidden units doubles since each layer consists of one forward and one backward layer. In table 7.5 the total number of hidden units per layer is given in brackets.

model	features	targets	acc [%]	ΔPESQ	ΔSTOI
BLSTM 2x192 (384)	mel	psf	63	0.3	0.06
BLSTM 2x256 (512)	mel	psf	67	0.4	0.06
BLSTM 2x512 (1024)	mel	psf	**71**	0.4	0.07
BLSTM 2x768 (1536)	mel	psf	68	0.4	0.06
BLSTM 2x32 (64)	mel	irm	64	0.2	0.03
BLSTM 2x64 (128)	mel	irm	69	0.3	0.04
BLSTM 2x128 (256)	mel	irm	71	0.4	0.06
BLSTM 2x192 (384)	mel	irm	72	0.4	0.06
BLSTM 2x256 (512)	mel	irm	**74**	0.4	0.06
BLSTM 2x512 (1024)	mel	irm	72	**0.5**	0.06
BLSTM 2x768 (1536)	mel	irm	71	0.5	**0.06**

Table 7.5 – Objective evaluation of different layer sizes and feature combinations. Each BLSTM layer consists of one forward and one backward layer which doubles the number of hidden units. The total number of hidden units is given in brackets.

When considering only the speech accuracy score both configurations seem to have an optimum layer sizes after which the performance starts to decrease. For the combination mel spectra/psf masks this is 2x512 hidden units and for mel spectra/irm masks the optimal size is smaller with 2x256. This indicates that the phase sensitive filter requires more complicated networks to be estimated precisely than the ideal ratio mask, which can be explained by the fact that psf masks incorporate additional phase information which is not explicitely contained by the mel spectrum.
When looking at the PESQ and STOI measurements, the performance linearly increases with the number of hidden units per layer, unlike the speech accuracy there is no observable border indicating a decrease in performance.
As mentioned before, one should also consider the number of parameters of the resulting network when choosing the layer size, since it increases exponentially with the number of hidden units per layer.

7.3 Weighted Loss Function

In this experiment a modified loss function similar to the weighted reconstruction loss in [28] is used. Instead of multiplying the reconstruction loss with a linear decreasing weighting function, a slightly modified weight $F_w(k)$ is used which has a constant value up until the frequency bin corresponding to 1000 Hz (see figure 7.11).

$$F_w(k) = \begin{cases} \frac{K-\gamma}{K} & \text{for } k \leq \gamma \\ \frac{K-k}{K} & \text{for } k > \gamma \end{cases} \tag{7.1}$$

where K denotes the number of frequency bins and γ the frequency bin corresponding to 1000 Hz.
The modified loss function is the mean squared error between target and prediction multiplied by the weighting function.

$$\mathcal{L}_w(\mathbf{y}, \mathbf{t}) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{F_w} \cdot (\mathbf{y}_i - \mathbf{t}_i)^2 \tag{7.2}$$

The break at 1000 Hz is justified by the mel scale, which states that the perceived

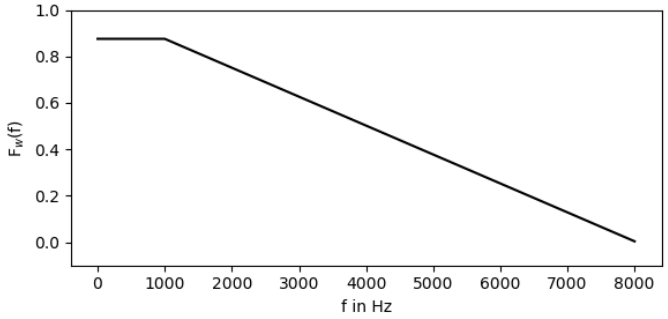

Figure 7.11 – Loss weight as a function of frequency.

pitch increments correspond to equally spaced pitch intervals on the frequency scale up to a point between 500 and 1000 Hz [55], [56]. The frequency weighting has the consequence that the error of a single frequency bin corresponding to a low frequency is more significant than that of a single high frequency bin. This not only matches the frequency resolution of the human auditory system, but also helps to increase the quality of voiced speech which is mainly located between 100 and 4000 Hz.

Figure 7.12 shows the PESQ and STOI measurements in comparison to the same BLSTM model trained using a conventional MSE loss.

Additionally, in table 7.6 the *Cortana* speech accuracy as well as the average improve-

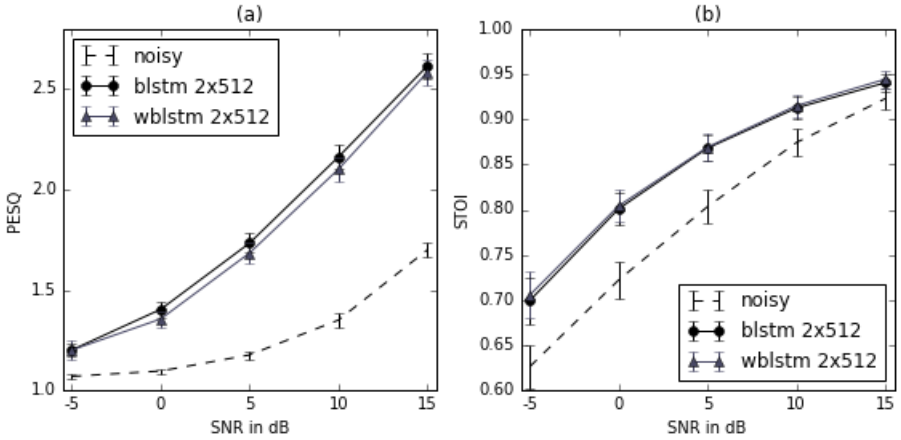

Figure 7.12 – PESQ and STOI measurements for a 2x512 BLSTM network trained with the weighted cost function (wblstm) and the conventional MSE loss (blstm).

ments in terms of PESQ and STOI are compared for two different BLSTM strtuctures with and without the modified MSE loss.

It can be observed that while the frequency weighting of the cost function results in an increase of up to 3% in speech accuracy, the differences in terms of PESQ and STOI are minimal and even indicate a slight decrease in performance in the case of the weighted cost function.

model	features	targets	acc [%]	ΔPESQ	ΔSTOI
BLSTM 2x256 (512)	mel	irm	74	0.4	0.06
wBLSTM 2x256 (512)	mel	irm	**76**	0.4	0.06
BLSTM 2x512 (1024)	mel	irm	72	**0.5**	0.06
wBLSTM 2x512 (1024)	mel	irm	75	0.4	**0.07**

Table 7.6 – Objective evaluation of the weighted loss function. While BLSTM refers to a bidirectional network trained using a conventional MSE loss, wBLSTM denotes the same network structure trained using the above mentioned weighted MSE.

7.4 Matched Training

To further improve the speech accuracy score, noise data extracted from the speech platform test file is used to form an extended training set. For this, hand labeled sections from the test file, in which no speech is present are mixed with clean speech from the dataset. This resulted in additional training data consisting of 6000 utterances (17 hours), increasing the total number of utterances included in the training data from 48,000 to 54,000.
Using the matched training data, the system improves by up to 8% in additional speech accuraccy. Table 7.7 shows the detailed results with and without the extended training set. It can be seen that the matched training set has no effect on the PESQ and STOI measurements for the 360 utterances contained in the test set, this means that the generalization capabilities are not effected by an extension of the training data while the improvements are large in case of the particular test case.

model	features	targets	train data	acc [%]	ΔPESQ	ΔSTOI
BLSTM 2x256	mel	irm	unmatched	74	0.4	0.06
BLSTM 2x256	mel	irm	matched	80	0.4	0.06
wBLSTM 2x256	mel	irm	unmatched	76	0.4	0.06
wBLSTM 2x256	mel	irm	matched	82	0.4	0.06
wBLSTM 2x512	mel	irm	unmatched	75	0.4	0.07
wBLSTM 2x512	mel	irm	matched	83	0.4	0.07

Table 7.7 – Objective evaluation for different RNN models with and without the matched training set.

Table 7.8 summarizes the results from the previous experiments, LSTM, BLSTM, BLSTM with weighted MSE and the same model trained with the extended training data are compared in terms of speech accuracy, PESQ and STOI improvement. It can be observed that the adaption of the training data to match the *Cortana* reference file brings the largest improvement in terms of speech accuracy, which clearly shows that the configuration of the training data is more effective and more important than fine-tuning of the network topology, hyper-parameters or training procedure.

model	train data	acc [%]	ΔPESQ	ΔSTOI
LSTM 2x512	unmatched	72	0.4	0.04
BLSTM 2x256	unmatched	74	0.4	0.06
wBLSTM 2x256	unmatched	76	0.4	0.06
wBLSTM 2x256	matched	82	0.4	0.06

Table 7.8 – Comparison of different optimization approaches. The matched training data brings the biggest improvements in terms of speech accuracy.

7.5 Convolutional Neural Networks

For this experiment a convolutional neural network based on the structure used in [38] is implemented because promising results have been reported. Figure 7.13 shows the basic structure of the model, which is made up of two 2D convolutional layers with a max pooling operation in between. The stride of the convolutional layers is set to [1,1] and that of the pooling layer to [2,2]. After the second convolutional layer the output is flattened an fed into the first fully-connected layer of the feed-forward part. After two layers with 1024 hidden units using ReLu activation follows the sigmoid output layer. The training procedure is the same as in the previous experiments, except that the input is given as matrices of size 128 x 11 instead of single time-step vectors, as it is the case for the mentioned LSTM models.

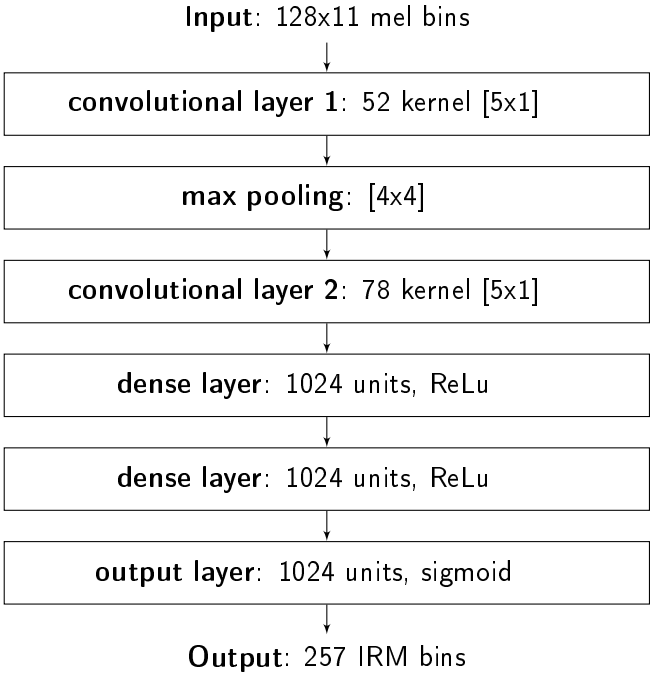

Figure 7.13 – Structure of the investigated CNN.

Table ?? shows the results achieved with the CNN compared to the baseline method and the 2x256 BLSTM network. It can be observed that, although the measurements in terms of PESQ and STOI show a better performance of the CNN model compared to the baseline system, the RNN performs significantly better than the CNN. More significantly, the resulting speech accuracy of 43% is only +1% higher than that of the unprocessed cortana reference file which scores at 42% accuracy. The poor performance of the CNN is logical when considering the fact that they are inspired by the principles of visual perception and are mostly used for image processing tasks such as object recognition. Furthermore, the employed filter kernels are only able to represent patterns along the frequency axis and not along the time axis, which means that the temporal

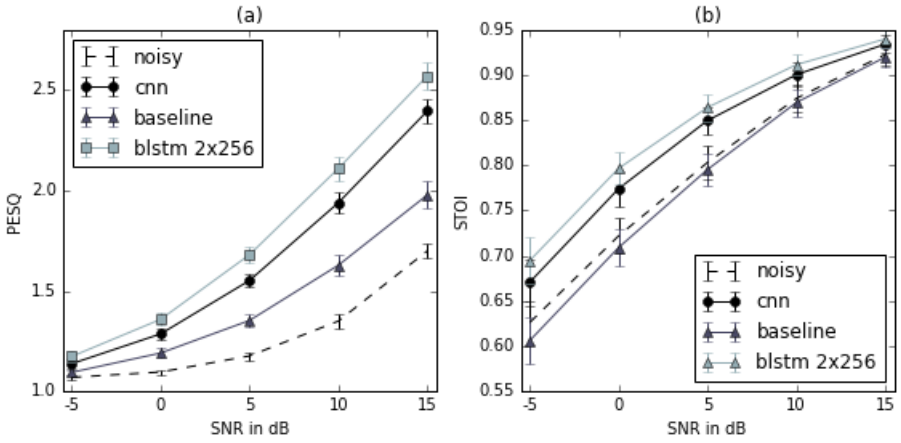

Figure 7.14 – PESQ (a) and STOI (b) measurements for the proposed CNN model compared to a BLSTM with 256 hidden units per layer, the baseline algorithm and the corresponding unprocessed speech signals.

information can only be modeled by the two dense layers on top of the convolutional layers. It should also be noted that the described structure employs a large number of

method	acc [%]	ΔPESQ	ΔSTOI
baseline	**77**	0.2	-0.01
BLSTM 2x256	74	**0.5**	**0.05**
CNN 2x[5,1]	43	0.4	0.04

Table 7.9 – Objective evaluation of the investigate CNN structure compared to a BLSTM with 256 hidden units per layer and the baseline algorithm.

parameters compared to the above mentioned RNNs. In sum the CNN has a total of 19,545,535 parameters compared to 3,543,809 parameters used for an LSTM network with two layers of 512 hidden units. Although the number of parameters could be reduced by optimizing the network structure, e.g. the kernel sizes, stride size or number of filter kernels, it is unclear if the model will achieve results comparable to those by the proposed RNN models. Nevertheless, the advantage of CNNs in comparison to RNNs is that the computational effort needed for the prediction of a single mask frame is much lower, since only a single forward pass through the network is necessary as opposed to 11 sequential passes needed by the proposed LSTM network.

In this chapter the experimental results for different network topologies and training procedures are presented and discussed. The next chapter concludes this thesis by summarizing the results and giving an outlook on future potential of the findings.

8 Conclusion

In this thesis the use of deep learning in the context of speech enhancement is investigated. This is motivated by the recent success of neural networks in other research areas and the limitations of conventional speech enhancement algorithms in the case of severe noise conditions. The focus is on RNNs and LSTM networks in particular, due to their suitability to machine learning problems involving timeseries prediction. Different network topologies are compared and evaluated using conventional evaluation criteria as well as a speech recognition tool provided by *Microsoft*.

The investigated RNN based algorithm outperforms the baseline method in terms of objective evaluation criteria as well as subjectively observed speech quality. Using the proposed algorithm, the amount of musical noise is reduced significantly compared to the results produced by the baseline system. In terms of speech accuracy, which is measured using a specific *Cortana* ASR test case, the results are comparable to that of the baseline algorithm. It is observed that the use of bidirectional LSTMs consistently improved the model accuracy in comparison to conventional LSTM networks.
Additionally it is shown that RNNs are superior to CNN structures for the use within the proposed speech enhancement scheme.

Although a perceptual frequency weighting of the cost function helps to improve the performance of the speech enhancement system for certain test cases, in general no significant improvements in comparison to a conventional MSE loss are reported. As a recommendation for future research, it would make sense to investigate an approach where, instead of the estimated spectral mask, the reconstruction of the complex spectrum is used to compute the cost as in [36], to include phase information to the training process.

The biggest boost in performance in terms of *Cortana* speech accuracy is achieved using an extended training set, configured to match the background noise from the corresponding reference file. This means that for the use with a lifestyle audio product such as a voice-controlled smart speaker, the possibility to adapt the model to the particular acoustic environment has to be considered, in order to maximize the performance of the speech enhancement algorithm. Therefore, future research may be carried out in order to investigate the possibility of transfer learning [77] in combination with an initial

calibration phase, and/or expert systems using multiple pre-trained models for different usecases and noise scenarios respectively.
Furthermore, taking into account the temporal trend of consecutive mask estimates, for example by using a recursive evaluation scheme as introduced in [76], could help to further improve the performance especially for stationary narrow-band noise.

As a final remark, the feasibility of a real-time implementation of the proposed algorithm for the use within mobile devices such as smartphones or cloud speakers, has yet to be explored. Most of the presented algorithms employ neural network models defined by a large number of parameters, which would require too much memory to be implemented on conventional DSP chips. But even if enough memory would be available, a real-time implementation would still be challenging since the temporal context of the input data has to be processed sequentially. However, with the growing interest in artificial intelligence and deep learning, it can be expected that more and more research and development will be done in terms of custom hardware for neural network inference and training.

Bibliography

[1] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 27, no. 2, pp. 113–120, 1979.

[2] M. Berouti, R. Schwartz, and J. Makhoul, "Enhancement of speech corrupted by acoustic noise," *ICASSP '79. IEEE Int. Conf. Acoust. Speech, Signal Process.*, vol. 4, no. 1, pp. 208–211, 1978.

[3] R. Hoeldrich and M. Lorber, "Non-linear spectral subtraction with combined smoothing strategies for broadband noise reduction," in *Audio Engineering Society Convention 103*, Sep 1997.

[4] A. R. Fukane and S. L. Sahare, "Different approaches of spectral subtraction method for enhancing the speech signal in noisy environments," *International Journal of Scientific and Engineering Research*, vol. 2, May 2011.

[5] S. Kamath and P. C. Loizou, "A multi-band spectral subtraction method for enhancing speech corrupted by colored noise.," in *ICASSP*, p. 4164, IEEE, 2002.

[6] B. Widrow, J. Glover, J. McCool, J. Kaunitz, C. Williams, R. Hearn, J. Zeidler, J. Eugene Dong, and R. Goodlin, "1975 Adaptive noise cancelling: Principles and applications," *Proc. IEEE*, vol. 63, no. 12, pp. 1692–1716, 1975.

[7] Y. Ephraim and D. Malah, "Speech enhancement using a minimum mean-square error log-spectral amplitude estimator," *IEEE Trans. Acoust.*, vol. 33, no. 2, pp. 443–445, 1985.

[8] T. Lotter and P. Vary, "Speech enhancement by MAP spectral amplitude estimation using a super-Gaussian speech model," *EURASIP J. Appl. Signal Processing*, vol. 2005, no. 7, pp. 1110–1126, 2005.

[9] I. Cohen, "Noise spectrum estimation in adverse environments: Improved minima controlled recursive averaging," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 5, pp. 466–475, 2003.

[10] R. Martin, "Noise power spectral density estimation based on optimal smoothing and minimum statistics," *IEEE Trans. Speech Audio Process.*, vol. 9, no. 5, pp. 504–512, 2001.

[11] Y. Ephraim, D. Malah, and B. H. Juang, "On the application of hidden Markov-models for enhancing noisy speech," *1988 Int. Conf. Acoust. Speech, Signal Process.*, no. 4, pp. 1846–1856, 1988.

[12] Y. Ephraim, D. Malah, and B. Juang, "Speech enhancement based upon hidden markov modeling," *Icassp*, pp. 353–356, 1989.

[13] K. W. Wilson, B. Raj, P. Smaragdis, and A. Divakaran, "Speech denoising using nonnegative matrix factorization with priors," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4029–4032, March 2008.

[14] K. W. Wilson, B. Raj, and P. Smaragdis, "Regularized non-negative matrix factorization with temporal dependencies for speech denoising," *Proc. Annu. Conf. Int. Speech Commun. Assoc. Interspeech*, pp. 411–414, 2008.

[15] Y.-X. Wang and Y.-J. Zhang, "Nonnegative Matrix Factorization: A Comprehensive Review," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1336–1353, 2013.

[16] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets.," *Neural Comput.*, vol. 18, no. 7, pp. 1527–54, 2006.

[17] Y. Bengio, "Learning deep architectures for ai," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[18] S. Tamura, "An analysis of a noise reduction neural network," *Int. Conf. Acoust. Speech, Signal Process.*, pp. 2001–2004, 1989.

[19] F. X. F. Xie and D. V. Compernolle, "A family of MLP based nonlinear spectral estimators for noise reduction," *Proc. ICASSP '94. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. ii, pp. 53–56, 1994.

[20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[21] A. L. Maas, Q. V. Le, T. M. O'Neil, O. Vinyals, P. Nguyen, and A. Y. Ng, "Recurrent Neural Networks for Noise Reduction in Robust ASR.," *Interspeech*, pp. 3–6, 2012.

[22] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," *2013 IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 7398–7402, 2013.

[23] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep Neural Networks for Acoustic Modeling in Speech Recognition," *IEEE Signal Process. Mag.*, no. November, pp. 82–97, 2012.

[24] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A. rahman Mohamed, and G. E. Hinton, "Binary coding of speech spectrograms using a deep autoencoder.," in *Interspeech* (T. Kobayashi, K. Hirose, and S. Nakamura, eds.), pp. 1692–1695, ISCA, 2010.

[25] X. Lu, S. Matsuda, C. Hori, and H. Kashioka, "Speech restoration based on deep learning autoencoder with layer-wised learning," *Interspeech, Portland, Oregon*, pp. 1504–1507, 2012.

[26] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech Enhancement Based on Deep Denoising Autoencoder," *Interspeech*, no. August, pp. 436–440, 2013.

[27] E. W. Healy, S. E. Yoho, Y. Wang, and D. Wang, "An algorithm to improve speech recognition in noise for hearing-impaired listeners," *The Journal of the Acoustical Society of America*, vol. 134, no. 4, pp. 3029–3038, 2013.

[28] B. Y. Xia and C. C. Bao, "Speech enhancement with weighted denoising auto-encoder," *Proc. Annu. Conf. Int. Speech Commun. Assoc. Interspeech*, no. August, pp. 3444–3448, 2013.

[29] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep Learning for Monoaural Speech Separation," *Proc. 39th IEEE Int. Conf. Acoust. Speech, Signal Process. ICASSP 2014*, pp. 1562–1566, 2014.

[30] D. Liu, P. Smaragdis, and M. Kim, "Experiments on deep learning for speech denoising.," in *Interspeech* (H. Li, H. M. Meng, B. Ma, E. Chng, and L. Xie, eds.), pp. 2685–2689, ISCA, 2014.

[31] Y. Wang, A. Narayanan, and D. Wang, "On Training Targets for Supervised Speech Separation," *Ieee/Acm Trans. Audio*, vol. 22, no. 12, pp. 1849–1858, 2014.

[32] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "An Experimental Study on Speech Enhancement Based on Deep Neural Networks," *IEEE Signal Process. Lett.*, vol. 21, no. 1, pp. 65–68, 2014.

[33] Y. Xu, J. Du, L.-r. Dai, C.-h. Lee, Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee, "A Regression Approach to Speech Enhancement Based on Deep Neural Networks," *IEEE Trans. Audio, Speech Lang. Process.*, vol. 23, no. 1, pp. 7–19, 2015.

[34] Y. Xu, J. Du, Z. Huang, L. Dai, and C. Lee, "Multi-objective learning and mask-based post-processing for deep neural network based speech enhancement," *CoRR*, vol. abs/1703.07172, 2017.

[35] F. Weninger, J. R. Hershey, J. Le Roux, and B. Schuller, "Discriminatively trained recurrent neural networks for single-channel speech separation," *2014 IEEE Glob. Conf. Signal Inf. Process. Glob. 2014*, pp. 577–581, 2014.

[36] H. Erdogan, J. R. Hershey, S. Watanabe, and J. Le Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," *Proc. 40th IEEE Int. Conf. Acoust. Speech, Signal Process. ICASSP 2015*, pp. 708–712, 2015.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2012.

[38] T. Kounovsky and J. Malek, "Single channel speech enhancement using convolutional neural network," in *2017 IEEE International Workshop of Electronics, Control, Measurement, Signals and their Application to Mechatronics (ECMSM)*, pp. 1–5, May 2017.

[39] A. Graves, A. r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, May 2013.

[40] L. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building End-to-End Dialogue Systems Using Generative Hierarchical Neural Network Models," *arXiv*, pp. 3776–3783, 2016.

[41] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems 21* (D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, eds.), pp. 545–552, Curran Associates, Inc., 2009.

[42] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[43] A. Zell, *Simulation neuronaler Netze*. Oldenbourg, 2000.

[44] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[46] R. Hecht-Nielsen, "Theory of the Backpropagation Neural Network," *Proc. Int. Jt. Conf. Neural Networks*, vol. 1, pp. 593–605, 1989.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, Jan. 2014.

[48] F. Chollet *et al.*, "Keras." `https://github.com/keras-team/keras`, 2015.

[49] G. Mesnil, Y. Dauphin, K. Yao, Y. Bengio, L. Deng, D. Hakkani-Tur, X. He, L. Heck, G. Tur, D. Yu, and G. Zweig, "Using recurrent neural networks for slot filling in spoken language understanding," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, pp. 530–539, March 2015.

[50] Y. Miao, M. Gowayyed, and F. Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 167–174, Dec 2015.

[51] T. M. Breuel, A. Ul-Hasan, M. A. Al-Azawi, and F. Shafait, "High-performance ocr for printed english and fraktur using lstm networks," in *2013 12th International Conference on Document Analysis and Recognition*, pp. 683–687, Aug 2013.

[52] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, pp. 2673–2681, Nov 1997.

[53] A. Graves, N. Jaitly, and A. r. Mohamed, "Hybrid speech recognition with deep bidirectional lstm," in *2013 IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 273–278, Dec 2013.

[54] P. Mowlaee, *Phase-Aware Signal Processing in Speech Communication: Theory and Practice*. Signal Processing, John Wiley & Sons, November 2016.

[55] S. S. Stevens, J. Volkmann, and E. B. Newman, "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[56] S. Umesh, L. Cohen, and D. Nelson, "Fitting the mel scale," in *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, vol. 1, pp. 217–220 vol.1, Mar 1999.

[57] S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, pp. 357–366, Aug 1980.

[58] "Itu-t recommendation p.862 : Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," tech. rep., Telecommunication Standardization Sector of ITU, 02/2001.

[59] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An Algorithm for Intelligibility Prediction of Time Frequency Weighted Noisy Speech," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 19, no. 7, pp. 2125–2136, 2011.

[60] Y. Zhao, D. Wang, I. Merks, and T. Zhang, "Dnn-based enhancement of noisy and reverberant speech," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6525–6529, March 2016.

[61] D. Wang and G. J. Brown, *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. Wiley-IEEE Press, 2006.

[62] N. Roman, D. Wang, and G. J. Brown, "Speech segregation based on sound localization," in *Neural Networks, 2001. Proceedings. IJCNN '01. International Joint Conference on*, vol. 4, pp. 2861–2866 vol.4, 2001.

[63] G. Hu and D. Wang, "Monaural speech segregation based on pitch tracking and amplitude modulation," in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 1, pp. I–553–I–556, May 2002.

[64] A. Jourjine, S. Rickard, and Ö. Yilmaz, "Blind separation of disjoint orthogonal signals: Demixing N sources from 2 mixtures," *ICASSP, IEEE Int. Conf. Acoust. Speech Signal Process. - Proc.*, vol. 5, pp. 2985–2988, 2000.

[65] Y. Li and D. Wang, "On the optimality of ideal binary time-frequency masks," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3501–3504, March 2008.

[66] N. Roman and D. Wang, "On Binary and Ratio Time-Frequency Masks for Robust Speech Recognition," in *Proc. International Conference on Spoken Language Processing (ICSLP)*, pp. 2541–2544, 2004.

[67] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An asr corpus based on public domain audio books," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210, April 2015.

[68] "Librivox." `https://librivox.org/`. Accessed: 2018-01-19.

[69] A. Mesaros, T. Heittola, and T. Virtanen, "Tut database for acoustic scene classification and sound event detection," in *2016 24th European Signal Processing Conference (EUSIPCO)*, pp. 1128–1132, Aug 2016.

[70] R. V. M. M. D. Dean, S. Sridharan, "The qut-noise databases and protocols.." `https://doi.org/10.4225/09/58819f7a21a21`, 2010. Accessed: 2018-01-19.

[71] A. Varga and H. J. M. Steeneken, "Assessment for automatic speech recognition ii: Noisex-92: A database and an experiment to study the effect of additive noise on speech recognition systems," *Speech Commun.*, vol. 12, pp. 247–251, July 1993.

[72] T. Chan, G. Golub, R. LeVeque, and S. U. C. D. o. C. Science., "Updating formulae and a pairwise algorithm for computing sample variances," *Ann. Phys. (N. Y).*, vol. 54, p. 258, 1979.

[73] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[74] V. K. Rajan, C. Baasch, M. Krini, and G. Schmidt, "Improvement in listener comfort through noise shaping using a modified wiener filter approach," in *Speech Communication; 11. ITG Symposium*, pp. 1–4, Sept 2014.

[75] C. Baasch, V. K. Rajan, M. Krini, and G. Schmidt, "Low-complexity noise power spectral density estimation for harsh automobile environments," *2014 14th Int. Work. Acoust. Signal Enhanc. IWAENC 2014*, pp. 218–222, 2014.

[76] Y. Ephraim and D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, pp. 1109–1121, Dec 1984.

[77] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, Oct 2010.