Master's thesis

Residual Echo Suppression using Deep Neural Networks

Felix Perfler
Matr. No.: 01637739

Electrical Engineering and Audio Engineering, Master's Program - UV 066 413
University of Music and Performing Arts Graz
Graz University of Technology

Supervisor: Univ.Prof. Dipl.-Ing. Dr.techn. Alois Sontacchi

Graz, November 23, 2022

Erklärung

Hiermit bestätige ich, dass mir der Leitfaden für schriftliche Arbeiten an der KUG bekannt ist und ich die darin enthaltenen Bestimmungen eingehalten habe. Ich erkläre ehrenwörtlich, dass ich die vorliegende Arbeit selbständig und ohne fremde Hilfe verfasst habe, andere als die angegebenen Quellen nicht verwendet habe und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Felix Perfler, 01637739
Graz, den 23. November 2022

[illegible]

Unterschrift des Verfassers

Zusammenfassung

Bei Verwendung einer Freisprechstelle kann es zur Übertragung eines akustischen Echos kommen. Die Person, welche am Zwiegespräch teilnimmt, wird über den Lautsprecher, beim kontaktierten Gegenüber, wiedergegeben. Dabei wird sie auch vom Mikrofon dieser Sprechstelle erfasst und zur eigenen Sprechstelle zurück übertragen. Ebendort wird sie dann akustisch über den Lautsprecher wiedergegeben, wodurch sich die Person als Echo hört. Dieses Echosignal sollte deshalb vor der Übertragung gefiltert werden. Obwohl das Lautsprechersignal bekannt ist, muss die darauf einwirkende Impulsantwort des akustischen Übertragungsweges, welche nicht zeitlich konstant und unter anderem die Raumakustik beinhalted, gschätzt werden.

Ein bestehendes System, welche das eingangs beschriebene Problem auf Basis konventioneller digitaler Signalverarbeitung behandelt soll durch die Inkludierung eines Filters auf Basis eines neuronalen Netzwerkes verbessert werden. Dazu wird zunächst die passende Modelarchitektur gefunden. Da es wichtig ist, während des Trainings dem Modell möglichst verschiedene Daten zu zeigen wird ein System zur Generierung synthetischer Trainingsdaten erstellt, wodurch verschiedene Einflussfaktoren in der Praxis, wie etwa die Raumakustik an der Sprechstelle, variiert werden können.

Letztendlich konnte durch Inkludierung des Filters die Echounterdrückung verbessert werden. Dies kann auch quantitativ unterlegt werden, da im Mittel ein AECMOS Echo (Speech Quality Assessment Metric for Echo Impairment) von 4.4, im Vergleich zu 3.1 für ein System ohne Filter, an den Validierungsdaten nachgewießen wird. Darüberhinaus konnte auch die Sprachverständlichkeit aufgebessert werden, was auch in der Steigerung des PESQ (Perceptual Speech Quality Measure) auf 2.3 von 1.8 für das System ohne Filter ersichtlich ist.

Abstract

When using a speakerphone an acoustic echo can be transmitted, if not taken care of beforehand. This happens as the signal of the far-end speaker is played back by the loudspeaker and this signal gets picked up by the microphone of the communication interface. Thus, this unwanted echoed signal should be removed from the transmitted microphone signal, yet cancelling out this echo can pose challenges. Even though the loudspeaker signal is known the impulse response including the room acoustic transfer path is not known and needs to be estimated.

Such an existing system shall be improved by the inclusion of a post filter based on a neural network. Therefore, an appropriate network architecture is found. It is also important to show the network many examples during the training process. This is why the training data is created in a synthetic way while varying parameters like the room where the device is placed in.

Ultimately it can be shown that by including the neuronal post filter the overall echo suppression can be improved, while retaining the near end signal picked up by the microphone. Quantitatively this can be seen in the increase of the AECMOS echo (Speech Quality Assessment Metric for Echo Impairment) metric to 4.4 on average, compared to 3.1 for a system without the filter on a validation dataset. The increase in speech intelligibility is also lines up with the increase of the PESQ (Perceptual Speech Quality Measure) metric to 2.3 compares to 1.8 for a filterless system.

Contents

Chapter 1

Introduction

When using a hands-free telephony set up, an acoustic echo can be transmitted (see figure 1.1). In a typical intercom call situation the two parties participating in the conversation are in different rooms with intercom devices, which include at least one microphone and loudspeaker, placed in front of them. The near-end speaker gets picked up by the microphone and is played back in the far-end room by the loudspeaker of the intercom system. This signal is then again picked up by the microphone of the same intercom system and transmitted to the near-end room, where it is played back by the intercom device situated in the near-end room. Therefore, the near-end speaker is hearing a delayed version of themselves.

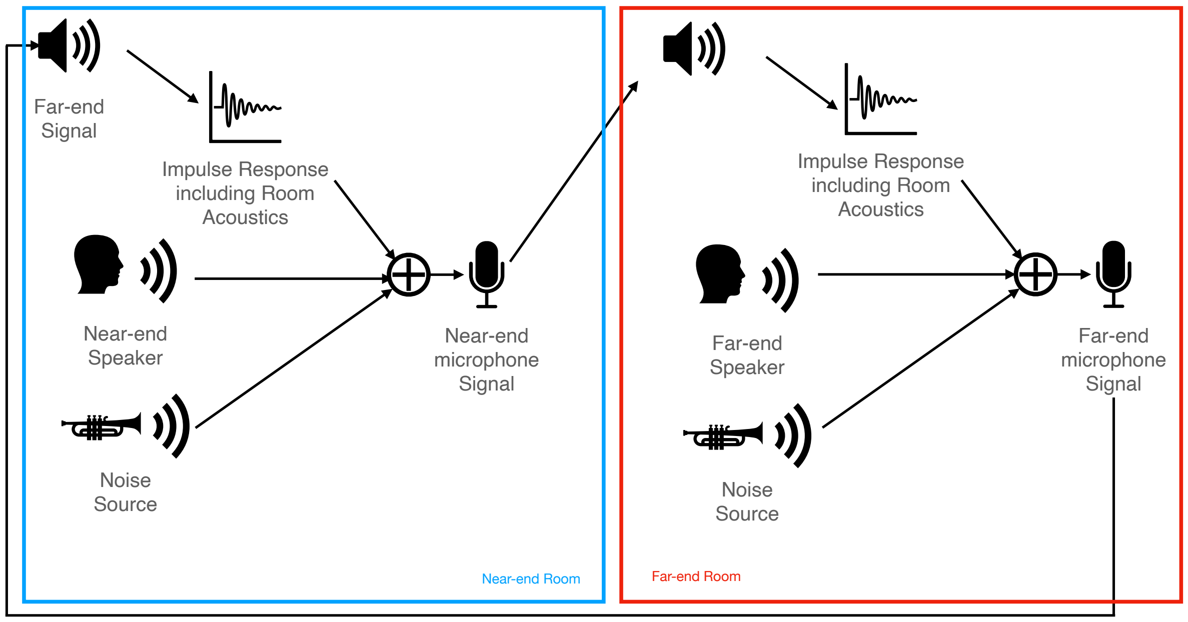
Far-end Signal
Impulse Response including Room Acoustics
Near-end Speaker
Near-end microphone Signal
Noise Source
Near-end Room
Impulse Response including Room Acoustics
Far-end Speaker
Far-end microphone Signal
Noise Source
Far-end Room

Figure 1.1: Signals transmitted by the hands-free telephony set up

For situations where the speakers interrupt each other, as is common in day to day conversations, the acoustic echo interrupts the conversation flow even more. Therefore, it is important that this acoustic echo is not transmitted, but rather filtered out.

The loudspeaker signal is known, what is unknown is the acoustic properties of the surrounding of the device, including but not limited to the nonlinearities of the loudspeaker and room impulse response. Therefore, in order to remove the far-end signal from the microphone signal this impulse response including the room acoustics needs to be estimated. What make this task even harder is that the estimated impulse response is time variant, as the speaker position is not constant, which means that the impulse response needs to be estimated constantly.

Commend International offers such a hands-free telephony system, for example the ID5 as pictured in figure 1.2. Currently, the pictured system deals with the problem described above by applying conventional digital signal processing techniques.

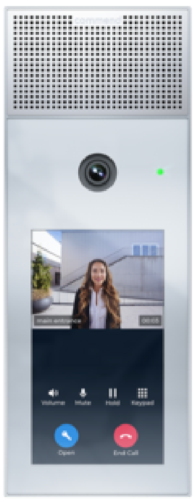

Figure 1.2: ID5 intercom system by Commend International[1]

The performance of the acoustic echo cancellor of this system should be improved. Especially for high volume settings the residual echo cannot be suppressed satisfactory. Therefore, a neural network should be investigated. In comparison to an end-to-end solution, where the whole system would be replaced by a neural net, a post-processing filter is used to augment the existing system.

In order to determine the right architecture for the task at hand, different ones are considered and evaluated. With the temporal changes in the speech signal being of importance special architectures for the network are considered and described in chapter 3. Since the neural network augments an existing system it is important that the computational complexity is kept minimal, so that the CPU is not taxed unnecessary. Therefore, the in-

[1]`https://www.commend.com`; accessed 02/07/2022

clusion of an auditory filter bank is investigated. The aim is to reduce the computational complexity, while retaining the same level of perceived audio performance.

The importance of the data the network is trained with cannot be overstated. It is important to show the network varied training data so once deployed it can handle as many situations as possible. Therefore, the training data should include different room acoustics situations, varied near-end speaker distances, different speaker volume settings, and various noise sources. Since it is not feasible to record all this data using the device, simulating the training data is necessary, which is also done in the course of the thesis. Since examples where the device volume is turned up all the way are of great interest it is of importance to consider the nonlinearities of the loudspeaker. This is achieved through the fitting of a Hammerstein model. The Hammerstein model fitting is described in chapter 7.2.2.

Once the training data is created and suitable model architectures are defined the training process can start. The training process is done in a supervised way. This means the current output is compared to the ideal known output. The network is then updated to better achieve the ideal output. This optimization is done incremental until, until a (local) optimum of the training objective, i.e. the cost function is found. It is important to choose the right cost function for the task. Different ones are defined in chapter 5. In chapter 4 the way how neural networks learn, i.e. backpropagation is described.

In order to compare the models and find the best one the evaluation metrics used are described in chapter 6.

Finally, to bring it all together in chapter 8 the best cost function for the task is determined and different model architectures are evaluated. This then leads to the proposed post processing filter, which is able to significantly improve the residual echo suppression, while not stressing the CPU usage.

Chapter 2

Acoustic Echo Cancellation using Adaptive Filters

One way to estimate the acoustic echo in order to remove it later is by using an adaptive filter. The flow chart in figure 2.1 describes this structure of such filters.

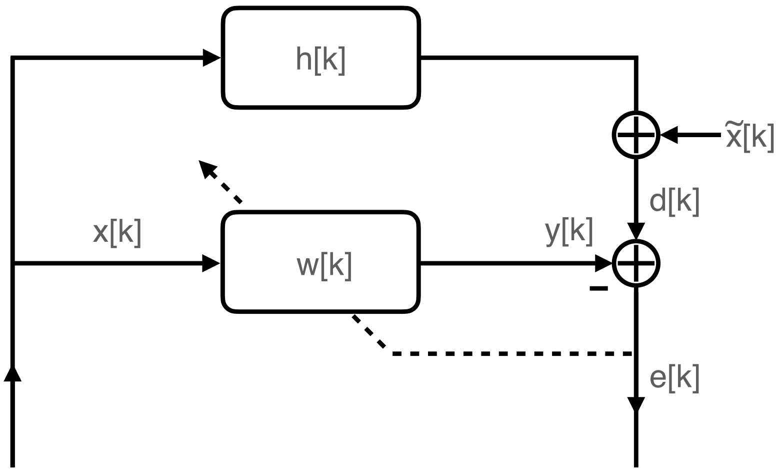

Figure 2.1: Structure of an adaptive filter

The far-end loudspeaker signal $x[k]$ is convolved with the plant description $h[k]$ and the near-end speaker signal $\tilde{x}[k]$ is added forming the signal $d[k]$.

$$d[k] = x[k] * h[k] + \tilde{x}[k] \tag{2.1}$$

The aim is to minimize the error signal $e[k]$, ideally for it to be equal to the signal $\tilde{x}$. This is done by estimating the plant through a filter described by $w[k]$. The filter coefficients $w[k]$ are identified using adaptive methods, therefore also changing over time. Using this filter the signal $y[k]$ is constructed:

$$y[k] = w[k] * x[k] \tag{2.2}$$

The error signal, which is also the signal which gets transmitted, therefore is:

$$e[k] = d[k] - y[k] \tag{2.3}$$

The plant description $h[k]$ cannot be assumed to be the same over time, but rather can also change with time. Furthermore, it is possible that the signal $\tilde{x}[k]$ is not only the near-end speaker signal, but it can be noisy speech or a noise source only. Situation like these make the prediction of $w[k]$ harder to impossible.

Chapter 3

Deep Neural Network Model Components

Neural networks can be used to approximate any derivable function, which maps an input value to an output value. This is achieved by connecting various units, known as neurons. A neural network is loosely modeled after the human brain and its synapses. The simplest neural network is a perceptron, consisting of a single neuron, which weights the input, adds a bias, and applies a (nonlinear) activation.

$$out = \mathrm{f}\left(w^T \cdot x + b\right) \tag{3.1}$$

Depending on the use case this simple structure is expanded and adapted to several input, hidden, and output neurons.
The corresponding weights are determined in a data driven way. By presenting the network example inputs and the desired outputs these weights get adjusted during the training process via a process called backpropagation, which is further described in chapter 4.1.

In order to learn temporal context from the features, a recurrent structure is needed. When the temporal context is important, it is necessary to take the previous information into account. Therefore, the network needs to memorize past information. Of course, the network cannot keep all previous information in memory. The amount of time steps the network keeps track of past information has significance during the learning process. Traditionally, recurrent cells suffer from gradient problems, when computing the backpropagation. Basically, the error signal either vanishes or diverges. This is due to the gradients getting smaller the further back the gradient is computed in the network. To overcome this, alternative structures were developed.

The long short-term memory (LSTM) cell was first proposed in 1997 by Sepp Hochreiter and Jürgen Schmidhuber [1] as a way to keep the error flow constant. An alternative to the LSTM cell was proposed in 2014 [2] in form of the gated-recurrent unit (GRU).

3.1 Long Short-Term Memory (LSTM) Cell

A LSTM cell has three main structural parts, the input gate, output gate, and forget gate in addition to a memory cell c_t and hidden state h_t. The three gates - input gate i_t, output gate o_t, and forget gate f_t - are used to control the information flow from and to the cell, therefor controlling when to update the two states c_t and h_t. The main idea is that using those gates the gradient can be passed along unchanged, therefore avoiding the vanishing gradient problem. The gates are described by the following expressions:

$$i_t = \sigma\left(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}\right) \tag{3.2}$$

$$f_t = \sigma\left(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}\right) \tag{3.3}$$

$$o_t = \sigma\left(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}\right) \tag{3.4}$$

All three gates have a sigmoid activation function σ and appropriate weight matrices W as well as bias vectors b. Because of the usage of sigmoid activation the output of all gates is a value between 0 and 1. For a value close to 0 the gate is closed and little information can pass. The closer the value to 1, the more information can pass the gate. Intuitively, the function of the input gate can be understood to regulate the activations into the cell state c_t, the forget gate can reset the cell state and the output gate, and the output gate controls the flow of information to the rest of the network. Looking at each gate individually, it is apparent that they are structurally the same to a single neuron (see equation 3.1) with multiple inputs.

In order to compute the cell state c_t, g_t has to be computed.

$$g_t = \tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}\right) \tag{3.5}$$

The cell state c_t is updated as follows, where $\odot$ is the Hadamard product:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{3.6}$$

In order to update the hidden state one performs the following:

$$h_t = o_t \odot \tanh\left(c_t\right) \quad , \tag{3.7}$$

which means that the output gate o_t controls when the hidden state gets updated. The hidden state h_t is also passed to the rest of the network.

The structure of a LSTM cell is pictured in figure 3.1. In the concatenation blocks the appropriate weight matrices and biases are applied to the input x_t and the previous hidden state h_{t-1} (see equations 3.2 till 3.5).

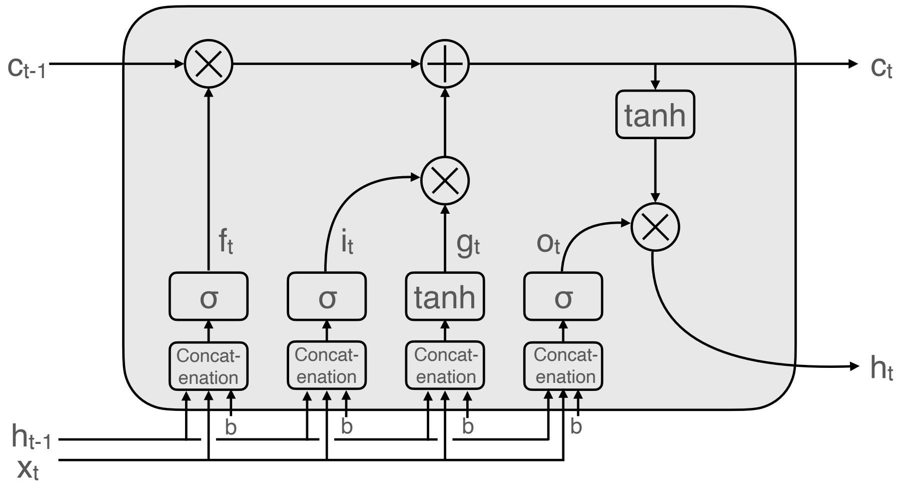

Figure 3.1: LSTM Cell

3.2 Gated Recurrent Unit (GRU)

While a LSTM cell has both, a hidden and a cell state, the GRU only has a hidden state. It therefore has fewer parameters than a LSTM cell.

In order to compute the cell update the state of the two gates, the update gate z_t and reset gate r_t have to be computed.

$$z_t = \sigma\left(W_{in}x_t + b_{in} + W_{hz}h_{t-1} + b_{hz}\right) \tag{3.8}$$

$$r_t = \sigma\left(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}\right) \tag{3.9}$$

The update gate z_t performs the same function as the forget and input gate of an LSTM cell (see chapter 3.1). The reset gate r_t decides how much past information to forget. The updated hidden state can be expressed as follows:

$$n_t = \tanh\left(W_{in}x_t + b_{in} + r_t \odot \left(W_{hn}h_{t-1} + b_{hn}\right)\right) \tag{3.10}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot n_t, \tag{3.11}$$

where $\odot$ is the Hadamard product.

The GRU cell can be visualized as seen in figure 3.2.

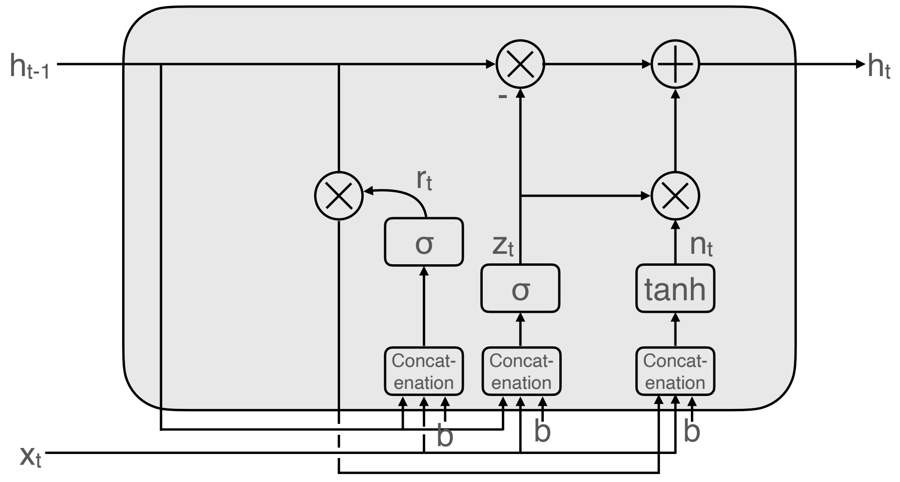

Figure 3.2: GRU Cell

3.3 Model Features

The input features of the network are the loudspeaker signal spectrogram, the spectrogram of the residuum of the acoustic echo canceller, the volume setting of the device, and the processing delay. The signals have a sampling frequency of 16 kHz. The Fourier transform length is set to 32 ms, corresponding to 512 FFT-bins, as it is the same used by the device. The volume setting can be an integer value between 1 and 12. A volume setting of 12 applies a broad band gain of 0 dB. For each lower setting the gain is decreased by -3 dB. So for a volume setting of 10 the equivalent gain would be -6 dB. The processing delay is the number of frames, which could not be processed in time. The spectrograms have a hop size of $16ms$. For the calculation of the spectrogram a hann like window with a length of $32ms$ is used (see figure 3.3). The window is of the same shape as the one used by the ID5 device, having the shape of a half-wave sine.

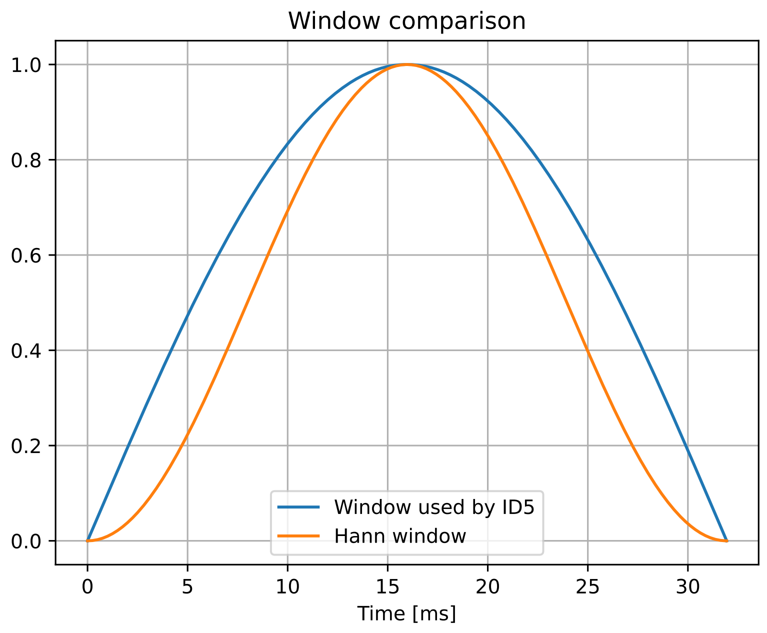

Figure 3.3: Hann like window

The magnitudes of the loudspeaker spectrogram Fx, the microphone spectrogram Fz_{in}, and the AEC prediction Fhx_{fg}, are then transformed to logarithmic spectrograms.

$$Lx = \log_{10}(Fx) \tag{3.12}$$
$$Lz_{in} = \log_{10}(Fz_{in}) \tag{3.13}$$
$$Lhx_{fg} = \log_{10}(Fhx_{fg}) \tag{3.14}$$

The resulting spectrogram Lx is one of the features, with the final feature, the residuum of the acoustic echo canceller being computed as follows.

$$Le = Lhx_{fg} - Lz_{in} \tag{3.15}$$

The used spectrogramms represent a corresponding time span of 10 seconds long audio samples.
In order to reduce the feature size, the spectral resolution of the spectrogram features can be reduced by using a filter bank.

3.3.1 Filter Bank

Since the computational expense of the network is of importance a way to reduce the feature size is proposed. By applying a filter bank the spectral resolution can be reduced.

Since the design of the filter bank is motivated by human perception, the information loss is kept to a minimum. The mel scale [3] was obtained through listener experiments. In order to get the mel value given a frequency the following formula can be used:

$$m = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{3.16}$$

With ever-increasing frequency the resolution of the human hearing worsens. This can be taken advantage of by designing a filter bank which maps frequency bins to mel bins. Because of the worse resolution for higher frequencies more frequency bins get mapped to one mel bin as can be seen in figure 3.4.

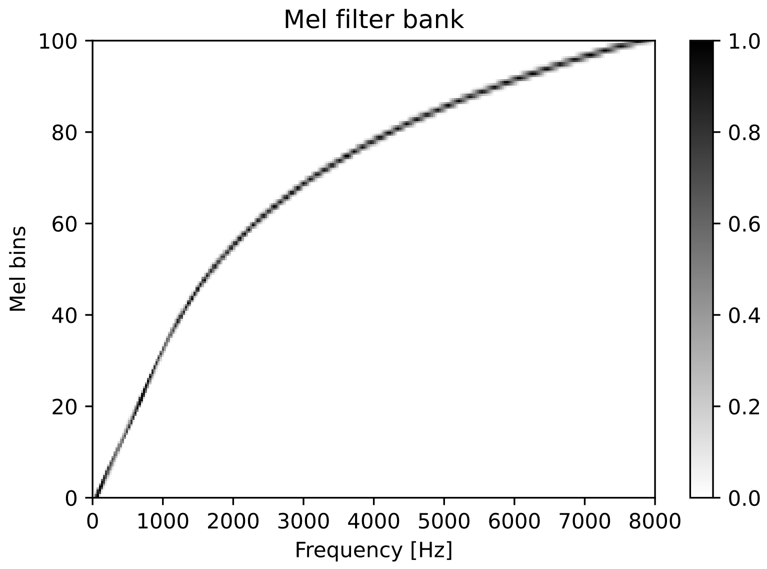

Figure 3.4: Mel filter bank

The choice of a filter bank mapping to 100 mel bins is motivated by [4], as the authors achieved the best results estimating a mask for filtering a target signal using this amount of bins. The resulting input feature spectra now have 100 mel bins compared to 257 frequency bins for the logarithmic magnitude spectra, therefore reducing the feature size significantly.

Chapter 4

Training

The model weights are learned in a supervised way. Therefore, the training features are presented to the network. The output is then compared to the ideal output and the weights adjusted to better match this target. The performance of the network depends on the network architecture and the learned weights. Therefore, it is important to choose the right cost function and teach the network in a sensible way.

4.1 Backpropagation

During backpropagation the network output is compared to the desired output. In order to compare how far off the network prediction differs from it's supposed output a cost function $\mathfrak{L}$ is used (see chapter 5). The weights of the network layer $\mathbf{W}$ are then adjusted using the derivation of the cost funciton and the learning rate η. The size of the weight matrix $\mathbf{W}$ is dependent on the number of neurons in the layer. For a single neuron layer this matrix becomes a vector $\mathbf{w}$. The following formulas can be extended for more neurons, but are written down for the single neuron case.

$$\Delta\mathbf{w} = -\eta\frac{\partial\mathfrak{L}}{\partial\mathbf{w}} \tag{4.1}$$

Using this update rule, the error will get minimized, i.e. the cost function decreases. If the number of layers is increased the chain rule needs to be applied. Therefore, the cost functions' partial derivatives for all individual weights for each sample can be expressed as:

$$\frac{\partial\mathfrak{L}(\mathbf{w})}{\partial\mathbf{w}_{ji}} = \frac{\partial\mathfrak{L}(\mathbf{w})}{\partial a_j}\cdot\frac{\partial a_j}{\partial\mathbf{w}_{ji}} = \delta_j\frac{\partial a_j}{\partial\mathbf{w}_{ji}} \tag{4.2}$$

The parameter δ_j is the sensitivity of the k-th unit and is a measure how much the overall cost is affected by this unit's activation. Starting at the output layer the sensitivity is computed for all units in said layer. Then the update rule for the last hidden to output weights is given by:

$$\Delta \mathbf{w}_{kj} = \eta \delta_k \frac{\partial a_k}{\partial \mathbf{w}_{kj}} \tag{4.3}$$

Next the sensitivities δ_m for all the hidden layers need to be computed.

$$\delta_m = \frac{\partial \mathfrak{L}_n(\mathbf{w})}{\partial a_m} = \sum_{k=1}^{K} \frac{\partial \mathfrak{L}_n(\mathbf{w})}{\partial a_k} \cdot \frac{\partial a_k}{\partial a_m} = \sum_{k=1}^{K} \delta_k \frac{\partial a_k}{\partial a_m} \tag{4.4}$$

This process needs to be repeated for each following layer to compute all weight updates $\nabla_{\mathbf{w}} \mathfrak{L}$.

The algorithm can be summarized in three steps:

1. Evaluation of the cost function for a given sample - forward pass
2. Computing the derivation of said function for each layer starting form the end - backpropagation
3. Updating the weights

4.2 Optimization

The last step of the backpropagation algorithm is to update the weights (see chapter 4.1). Once the partial derivatives of the cost function $\mathfrak{L}$ are computed the weight update for the i-th sample is:

$$\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} - \eta \nabla_{\mathbf{w}} \mathfrak{L}(\mathbf{w}^{(i)}) \tag{4.5}$$

The parameter η is the learning rate and is used to weight the update step. Using a smaller q learning rate, the weights will take longer to converge, while using a bigger learning rate might result in divergence. Instead of calculating the gradient for all samples, which is called batch gradient, it is more efficient to compute the gradient for only one or a couple of samples. Such algorithms are the Stochastic Gradient Descent (SGD) and mini-batch methods.

The vanilla gradient descent update rule as seen in equation 4.5 is a valid choice, but more sophisticated methods have been developed.

4.2.1 Gradient Descent with Momentum

The gradient descent with momentum tries to find the optimal parameter set Θ_t (see [5]). Therefore, the following algorithm can be used. The choice for α and β depends on the training objective. The default value for α in TensorFlow [6] is 0.01. The constant β is chosen between 0 and 1. For β closer to 1, the past gradient information is contributing to the parameter update more.

Algorithm 1 Gradient Descent with Momentum

```
1: $\Theta_t$, $m_t$                        ▷ Initialize the parameter vector and the first momentum vector
2: while $\Theta_t$ is not converged do
3:     $t \leftarrow t + 1$
4:     $g_t \leftarrow \nabla_\Theta f_t(\Theta_{t-1})$            ▷ Calculate the current gradient
5:     $m_t \leftarrow \beta \cdot m_{t-1} + (1 - \beta) \cdot g_t$      ▷ Update the first momentum vector
6:     $\hat{m}_t \leftarrow \frac{m_t}{1-\beta^t}$       ▷ Calculate the bias-corrected first momentum vector
7:     $\Theta_t \leftarrow \Theta_{t-1} - \alpha \cdot \hat{m}$                ▷ Update the parameters
8: end while
```

Compared to the standard gradient descent algorithm, introducing the momentum term has the advantage that higher step sizes α are possible as the algorithm is more stable against oscillating values for the result. This behavior can be observed, if the optimization surface is steeper in one dimension compared to the others. In addition, the algorithm converges faster.

4.2.2 Root Mean Square Propagation

Another optimization algorithm is the Root Mean Square Propagation (RMSprop) [7]. It uses the second momentum. In [6] the default values for α is 0.001. β is again chosen between 0 and 1.

Algorithm 2 RMSprop

```
1: $\Theta_t$, $v_t$                       ▷ Initialize the parameter vector and the second momentum vector
2: while $\Theta_t$ is not converged do
3:     $t \leftarrow t + 1$
4:     $g_t \leftarrow \nabla_\Theta f_t(\Theta_{t-1})$            ▷ Calculate the current gradient
5:     $v_t \leftarrow \beta \cdot v_{t-1} + (1 - \beta) \cdot g_t^2$    ▷ Update the second momentum vector
6:     $\hat{v}_t \leftarrow \frac{v_t}{1-\beta^t}$      ▷ Calculate the bias-corrected second momentum vector
7:     $\Theta_t \leftarrow \Theta_{t-1} - \alpha \cdot \frac{1}{\sqrt{\hat{v}_t}+\epsilon}$           ▷ Update the parameters
8: end while
```

4.2.3 Adaptive Moment Estimation

The Adaptive Moment Estimation (ADAM) optimizer [8] is a combination of the gradient descent with momentum and RMSprop algorithms. By combining both optimizers ADAM makes use of both the first and second moment, as the first moment normalized by the second moment defines the direction of the update. In [6] the default values for α, β_1, and β_2 are 0.001, 0.9, and 0.999 respectively.

Algorithm 3 ADAM

```
1: $\Theta_t$, $m_t$, $v_t$    ▷ Initialize the parameter vector, the first momentum vector, and the second momentum vector
2: **while** $\Theta_t$ is not converged **do**
3:     $t \leftarrow t + 1$
4:     $g_t \leftarrow \nabla_\Theta f_t(\Theta_{t-1})$    ▷ Calculate the current gradient
5:     $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$    ▷ Update the first momentum vector
6:     $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$    ▷ Update the second momentum vector
7:     $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$    ▷ Calculate the bias-corrected first momentum vector
8:     $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$    ▷ Calculate the bias-corrected second momentum vector
9:     $\Theta_t \leftarrow \Theta_{t-1} - \alpha \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$    ▷ Update the parameters
10: **end while**
```

4.2.4 Regularization

In general, it is best practice to apply regularization to the training of a neural network for example in order to overcome overfitting. Two of the most common ways to introduce regularization are $L1$ and $L2$ regularization. For $L1$ the following expression gets added to the weight update: $\lambda||\mathbf{w}||_1$, with λ being the regularization parameter controlling the amount. The resulting weight vector will be sparse as opposed to the $L2$ regularization. There the expression $\lambda||\mathbf{w}||_2^2$ gets added.

The weight decay implementation for ADAM in TensorFlow [6] employs a decoupling the weight decay from the optimization step [9].

Chapter 5

Cost Functions

The cost function should always be chosen according to the task at hand. The cost function takes the target signal spectrogram $S_{k,n}$ and the predicted signal spectrogram $\hat{S}_{k,n}$ as input. The spectrogram $\hat{S}_{k,n}$ is computed from the residuum of the acoustic echo cancellor Fe as:

$$\hat{S}_{k,n} = p_{est} \cdot Fe_{k,n} \tag{5.1}$$

While the network architecture does not take the phase information into account findings in [10] suggest that a combination of magnitude-only and phase aware parts of the cost function yield better results.
With the binary cross entropy a loss is presented, where the network output, i.e. the estimated spectral mask, is directly compared to its optimum.
Finding the optimal cost function for the task is discussed in chapter 8.2.

5.1 Signal Ratio based Loss Functions

The signal to noise ratio (SNR) and signal to distortion ratio (SDR) are two metrics to compare the desired signal to the noise level. When comparing the two complex spectrograms $S_{k,n}$ and $\hat{S}_{k,n}$, the difference of the two, i.e. $\hat{S}_{k,n} - S_{k,n}$ can be seen as the noise. The objective is to minimize this term or to maximize the SDR, therefore this metric can be used as a cost function.

$$\mathfrak{L}_{SDR} = -\frac{\log_{10} \sum_{k,n} S_{k,n}^2}{\log_{10} \sum_{k,n} \left| \hat{S}_{k,n} - S_{k,n} \right|^2} \tag{5.2}$$

Additionally, minimizing the negative SNR is also a valid choice as a loss.

$$\mathfrak{L}_{SNR} = -\frac{\log_{10} \sum_{k,n} |S_{k,n}|^2}{\log_{10} \sum_{k,n} \left| |\hat{S}_{k,n}| - |S_{k,n}| \right|^2} \tag{5.3}$$

Lastly, a combination of the SDR and SNR loss is tried. As noted in [10] a linear combination of a magnitude only $\mathfrak{L}_{SNR}$ and a phase aware $\mathfrak{L}_{SDR}$ loss function can yield better results. The two individual cost functions are mixed together using the factor α to control the contribution of each function. By combining equation 5.3 and 5.2 a new cost function can be defined as:

$$\mathfrak{L}_{SNR+SDR} = (1-\alpha) \cdot \mathfrak{L}_{SNR} + \alpha \cdot \mathfrak{L}_{SDR} \tag{5.4}$$

For the value α a value of 0.5 was found to be optimal in [10], therefore both losses are contributing in equal measures. This result was obtained by performing a grid search with values for α ranging from 0 to 1 in 0.1 steps.

5.2 Combined Loss

A loss function as described in [11] is implemented. The loss consists of two parts. The first part considers the compressed magnitude and the phase, which is than weighted and combined with the complementary weighted second compressed magnitude only part.

$$\mathfrak{L}_{Comb} = \alpha \sum_{k,n} \left| |S_{k,n}|^c e^{j\varphi s_{k,n}} - |\hat{S}_{k,n}|^c e^{j\varphi \hat{s}_{k,n}} \right|^2 + (1-\alpha) \sum_{k,n} \left| |S_{k,n}|^c - |\hat{S}_{k,n}|^c \right|^2 \tag{5.5}$$

The values for α and c are set to 0.3 as suggested in [11]. For α a grid search as described in chapter 5.1 is performed in [11]. In [11] it is also observed, that by putting more emphasis on the complex part by increasing α a steep performance drop is visible.

5.3 Binary Cross Entropy

Instead of comparing spectrograms, for the binary cross entropy the estimated filter mask gets compared to the ideal one. Therefore, the ideal filter mask p_{opt} is computed out of the power spectrogram of the target signal $P_{k,n} = |S_{k,n}|^2$ and the power spectrogram of the residuum of the acoustic echo canceller $Pe_{k,n} = |Fe_{k,n}|^2$, where $Fe_{k,n}$ is the residuum of the acoustic echo canceller, as:

$$p_{opt} = \frac{P_{k,n}}{Pe_{k,n}} \tag{5.6}$$

Using the two masks p_{opt} and p_{est}, the predicted post filter mask, the binary cross entropy can be calculated as:

$$\mathfrak{L}_{Binary} = -\left(p_{opt} \cdot \log\left(p_{est}\right) + \left(1 - p_{opt}\right) \cdot \log\left(1 - p_{est}\right)\right) \tag{5.7}$$

Intuitively it makes sense to use this cost function for the estimation of a spectral mask, as this loss tries to make a binary decision to whether applying the mask or not.

Chapter 6

Evaluation Metrics

In order to evaluate the model a number of different metrics are used. Two metrics, the Signal to Distortion Ratio (SDR) and Echo Return Loss Enhancement (ERLE), are based on signal ratios, while the other metrics take the human perception into account. Finally, with AECMOS a neural network based method is presented.

6.1 Signal to Distortion Ratio

Like the name suggests the signal to distortion ratio (SDR) defines a ratio between the desired signal and the distortion. It should be noted that the definition of the SDR used differs from the one used in chapter 5.1 for the cost function. Therefore, first the residuum F_e is calculated by subtracting the acoustic echo canceller prediction spectrum F_{hx_fg} from the microphone signal spectrum F_{z_in} and applying the predicted post filter mask P_{est}.

$$F_e = (F_{z\_in} - F_{hx\_fg}) \cdot P_{est} \tag{6.1}$$

In order to compute the unwanted distortion the target signal spectrum Y is subtracted from the residuum and the logarithmic power spectrum is calculated:

$$P_n = 10 \cdot \log_{10} \|F_e - Y\|^2 , \tag{6.2}$$

where k is the frequency index and t is the time index. The same is done for the spectrum of the clean target signal.

$$P_s = 10 \cdot \log_{10} \|Y\|^2 \tag{6.3}$$

Taking advantage of the logarithmic calculation rules the ratio can be expressed as follows:

$$SDR = P_s - P_n \tag{6.4}$$

6.2 Echo Return Loss Enhancement

The echo return loss enhancement (ERLE) is the second ratio considered for evaluation of the model. It measures the signal loss, which comes back as an echo. The higher the ERLE, the smaller the amount of returned echo is. Therefore, a high value is desired. First the residuum is calculated the same as for the SDR (chapter 6.1).

$$F_e = (F_{z\_in} - F_{hx\_fg}) \cdot P_{est} \tag{6.5}$$

For this signal the power spectrum is computed:

$$P_e = 10 \cdot \log_{10} \|F_e\|^2 \tag{6.6}$$

The same is done for the microphone signal.

$$P_{z_{in}} = 10 \cdot \log_{10} \|F_{z_{in}}\|^2 \tag{6.7}$$

The ERLE is then calculated as:

$$ERLE = P_{z_{in}} - P_e \tag{6.8}$$

6.3 Short-Time Objective Intelligibility Measure

The short-time objective intelligibility measure (STOI) tries to measure the speech intelligibility of an enhanced signal [12]. The method compares the clean target signal with the enhanced signal, i.e. the residuum with the applied post filter. First both signals are brought into the Fourier domain. Frames without active speech are removed from the spectra. A one-third octave band filter bank is applied to both signals. Next the short time temporal envelope of the two signals is computed. The enhanced signal is also normalized and clipped in order to compensate level differences. STOI is the then defined as the correlation coefficient of those two signal streams. The STOI is expressed as a single number between zero and one.

6.4 Perceptual Speech Quality Measure

The perceptual speech quality measure (PESQ) [13] is a proven metric to evaluate the perceived speech quality. It is also defined as a ITU-T norm [14]. Just like STOI (chapter 6.3) the metric again needs the clean reference and the enhanced signal for calculation. First both signal levels are aligned to a standard listening level. Thereafter, they are filtered to simulate a standard telephony handset. After aligning the signals temporally a psychoacoustic model is applied. During the transform frequency and gain variations are equalized. For the metric two parameters are derived and are mapped to a prediction of subjective mean opinion score (MOS) (see table 6.1).

rating	label
5	Excellent
4	Good
3	Fair
2	Poor
1	Bad

Table 6.1: Mean opinion score scale

6.5 Speech Quality Assessment Metric for Echo Impairment (AECMOS)

Especially PESQ (chapter 6.4) and STOI (chapter 6.3) are metrics not designed for the task at hand, i.e. estimating the quality of the acoustic echo canceller, but are still widely used in the literature. Therefore, a new metric is proposed in the form of AECMOS [15], which evaluates the enhanced signal in terms of echo and degradation from other sources such as noise, distortions, etc. For both measures mean opinion scores (MOS) are predicted. The method is based on a deep neural network, which takes the near-end microphone signal, far-end signal, and the enhanced microphone signal as input. Therefore, AECMOS is different to the other metrics presented as it takes no reference signal as input.

The training data, as for every neural network approach, is of great importance. The model was trained on 64013 samples with a mean lenght of 8.2 seconds. With 45.6% the near-end only scenario takes the majority of the scenario split, with 26.7% far-end only and 27.7% double talk forming the rest. As for the enhancement methods, 17 neural network based AEC models from the ICASSP 2021 AEC Challenge [16] were used. The labeling process is described in [17], as a crowdsourcing approach is needed to deal with the large amount of data. For each audio sample 5 human ratings were collected.

It was shown in [15] that the AECMOS metric better alignes with the human perception than PESQ and ERLE, in form of AECMOS degredation and AECMOS echo respectively.

Chapter 7

Datasets

The data a neural network is trained with is one of the most important decisions when designing a system taking advantage of neural networks. In order for the system to handle as many situations as possible, it is important to show a wide array of possible scenarios to the network.

During the design process two different datasets were considered. One of the datasets consists of a corpus of recorded data. In order to show a more diverse training set to the network a synthetic dataset was developed. This approach considered certain characteristics of the ID5 intercom system, like the nonlinearities of the loudspeaker. Since it is not necessary to make recordings, more room acoustic scenarios are considered, with the aim to improve the generalization capabilities of the system.

In order to take advantage of parallelization data generators were implemented using the TensorFlow framework (see [6]).

7.1 Recorded Dataset

For the pre-recorded data corpus different scenarios were recorded. The intercom system ID5 from Commend International [1] was placed on a table in a conference room and a Genelec 8010A loudspeaker [2] was placed in front of it, as can be seen in figure 7.1.

[1]`https://www.commend.com`; accessed 28/08/2022

[2]`https://www.genelec.com`; accessed 28/08/2022

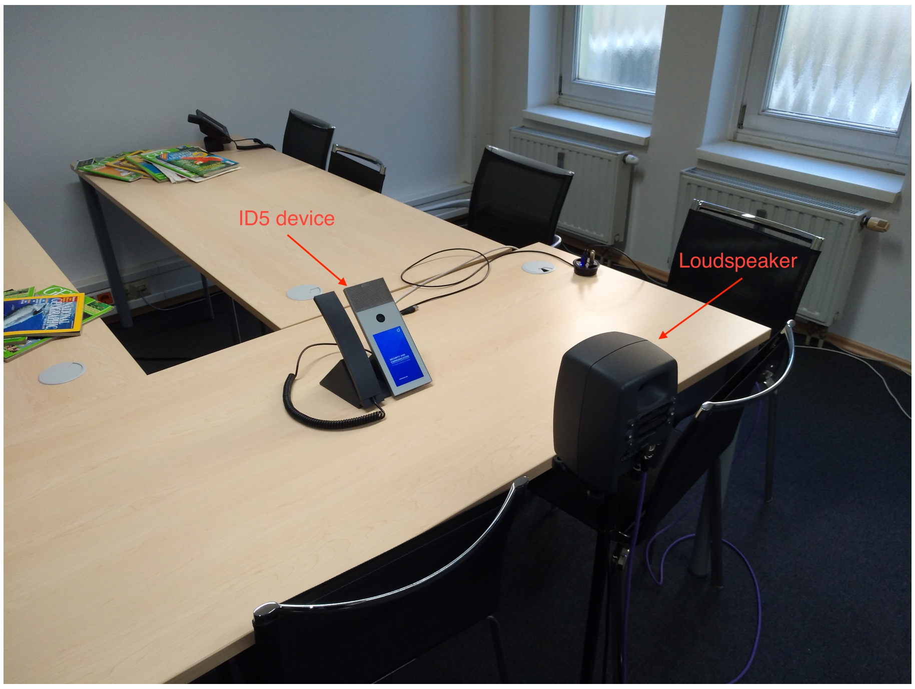

Figure 7.1: Recording Setup

The room was a typical office room with the following dimensions.

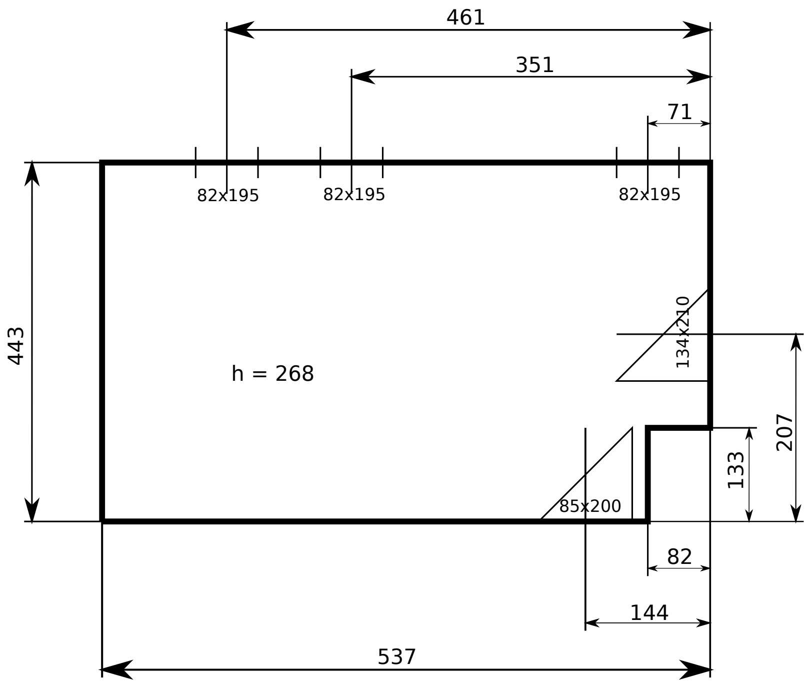

Figure 7.2: Recording Room, measurements in [cm]

The room was filled with typical office furniture and has the frequency dependent reverberation time according to figure 7.3. The measurement was performed according to the

RT30 norm [18]. The RT30 is defined as the time it takes for the signal to decay from -5 to -35 dB compared to the initial value. The resulting time gets then multiplied by the factor two.

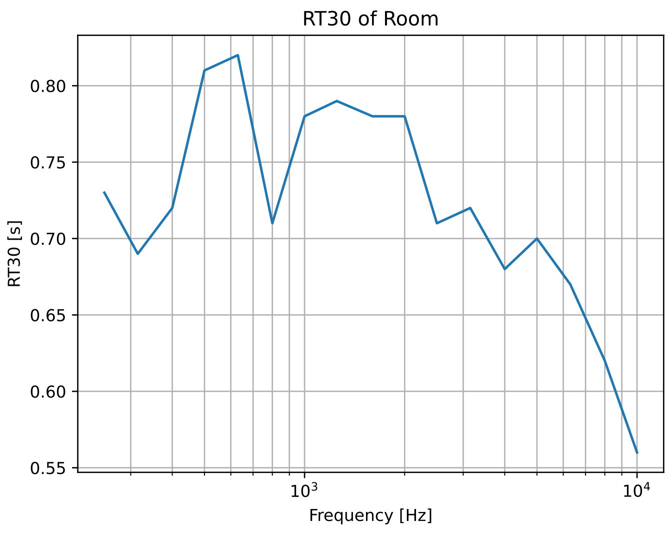

Figure 7.3: RT30 of the room for different frequencies

Two different loudspeaker distances, 50 and 100 cm, in order to consider different near-end speaker positions, were recorded. On the intercom system the loudspeaker volume setting was dialed to 8, 10, and 12. Changing the volume setting and distance is important, since the acoustic echo cancellar (AEC) shows different performance characteristics. For example the loudspeaker nonlinearities are more pronounced for higher volume settings. Also, it is easier for the AEC to predict the correct coefficients if the near-end signal energy is high compared to the far-end signal. For each scenario, far-end - only the internal loudspeaker of the intercom is playing - and near-end - only the loudspeaker in front of the device is active - were independently recorded using the intercom build in microphones. 20 different speaker signals were used for each scenario and far-end/near-end signal respectively. In order to build the dataset all possible combinations were calculated. Therefore, for each scenario 16 speakers were chosen for training whereas 4 were used for the validation set. Then the far-end and near-end speaker signals were combined in a way as to all possible combinations were being considered. Importantly far-end and near-end speaker were chosen to be not the same. The delay between the far-end loudspeaker signal and the far-end microphone signal was varied randomly, modeled by a normal distribution with a mean value of 30 ms and a standard deviation of 6 ms. This delay is a result of the processing of the audio signals by the intercom system. The resulting 30 second audio clips are cut into 10 second non overlapping segments. In the first 10 second segments the acoustic echo cancellar cannot be assumed to be stable, therefore also training the network with these examples is important. In order to simulate this situation the learned coefficients of the acoustic echo cancellar are reset for every 100 samples.

The size of the resulting dataset can be seen in table 7.1.

data type	training data/hours	validation data/hours
far-end only	2.4	0.6
near-end only	2.4	0.6
mixed	115.2	7.2

Table 7.1: Amount of training and validation data in hours

7.2 Synthetic Data Generator

7.2.1 Deep Noise Suppression Dataset

The synthetic data generator is using the DNS Challenge dataset (see [19]). The dataset contains clean speech files, noisey data files files and room impulse responses among other scenarios which were not considered for data augmentation. The clean speech files contain 500 hours of speech read by 2150 speakers. Every clip is 31 seconds long.
The noise data contains 70000 files, which are 10 seconds long. 60000 clips were obtained by analyzing YouTube videos. They contain 150 different noise classes. Speech activity detection was used in order to remove clips with any kind of speech. The noise files were deemed relevant to voice over IP (VOIP) applications.
As part of the dataset a number of different room impulse responses are provided. The room impulse responses were recorded by Microsoft with RT60 ranging from 300 ms to 1300 ms. For data augmentation only shorter room impulse responses were considered, as they are more relevant in office settings.

7.2.2 Hammerstein Model

In order to model the nonlineraties of the loudspeaker a Hammerstein model was used. The sweep response was measured using the same setup as in figure 7.1. To quantify the distortion caused by the nonlinearities of the loudspeaker the total harmonic distortion (THD) can be computed. The nth order distortion can be calculated, given the root mean square (RMS) value of the nth harmonic measured P_n and the RMS of the signal P, as:

$$THD_n = \frac{P_n}{P} \tag{7.1}$$

order	THD [%]
1	3.3
2	2.4
3	0.6
4	0.6
5	0.5

Table 7.2: THD for different orders

Then the THD up to order six is computed:

$$THD = \sqrt{THD_1^2 + THD_2^2 + THD_3^2 + THD_4^2 + THD_5^2}, \tag{7.2}$$

which results in a value of 4.2% for the device loudspeaker.

The calculation of the Hammerstein kernels was implemented according to [20], [21]. The sweep used is pictured in 7.4 and starts with a frequency f_1 of $50Hz$ going up till $24kHz$ (f_2). The duration is $10s$.

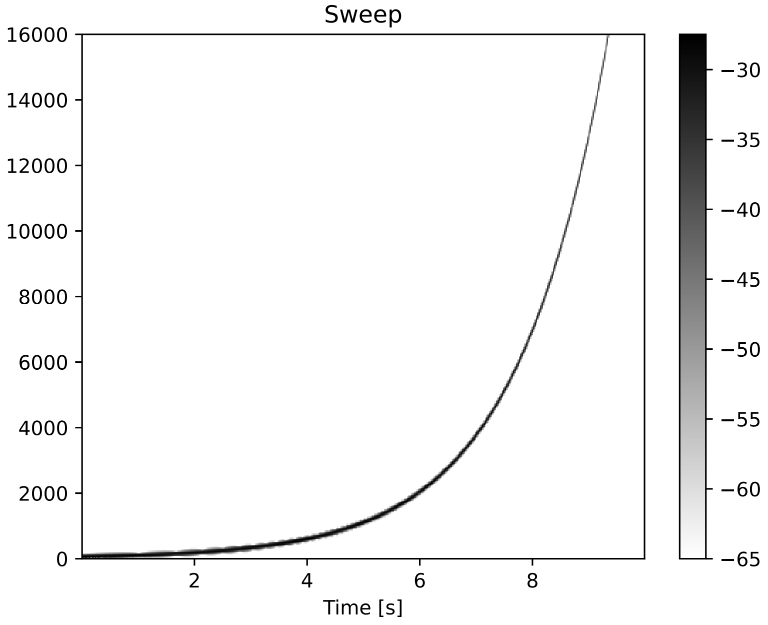

Figure 7.4: Sweep used to identify the system

Using these properties a factor L can be computed as:

$$L = \frac{1}{f_1} \cdot \text{round}\left(10 \frac{f_1}{\log\left(\frac{f_2}{f_1}\right)}\right) \tag{7.3}$$

Therefore, the exponential swept-sine $s(t)$ can be written down as:

$$s(t) = sin\left(2\pi f_1 L\left[\exp\left(\frac{t}{L}\right) - 1\right]\right) \tag{7.4}$$

Next the inverse sweep is calculated according to:

$$\tilde{s}(t) = \frac{f_1}{L}\exp\left(-\frac{t}{L}\right)s(-t) \tag{7.5}$$

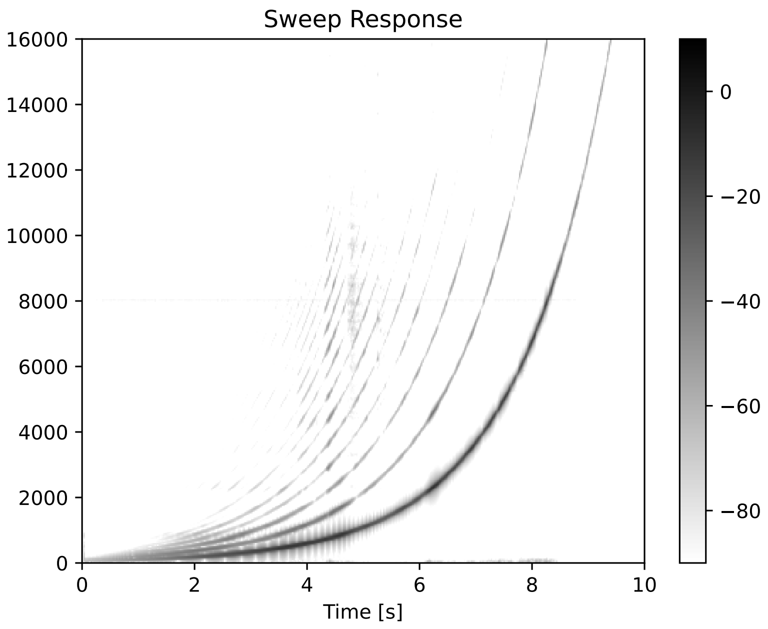

Figure 7.5: Sweep response of the device under test

Using the sweep response $y(t)$ (figure 7.5) the system response $h(t)$ can be computed, in the time domain, via a deconvolution.

$$\tilde{h}(t) = y(t) * \tilde{s}(t) \tag{7.6}$$

The normalisation factor ν is calculated as:

$$\nu = \max(s(t) * \tilde{s}(t)) \tag{7.7}$$

Then the system response is:

$$h(t) = \frac{\tilde{h}(t)}{\nu} \tag{7.8}$$

Next the linear part, as well as the higher order parts of the response must be identified.

The start time of the first M harmonics can be computed as:

$$dt = L \cdot \log(m) \cdot fs, \quad \text{with } m = 1...M \tag{7.9}$$

The chosen length of the individual kernels is $40ms$. Finally, the identified kernels are transformed into volterra kernels using matrix multiplication. The Hammerstein kernels in frequency domain are stacked in the matrix H and the transformation matrix is defined as:

$$A_{n,m} = \begin{cases} \frac{(-1)^{2n+\frac{1-m}{2}}}{2^{n-1}} \binom{n}{\frac{n-m}{2}}, & \text{for } n \geq m \text{ and } (n+m) \text{ is even} \\ 0, & \text{else} \end{cases} \tag{7.10}$$

The volterra kernels are then computed as:

$$G_{m\times m} = (A^T_{n\times m})^{-1} \cdot H_{n\times m}. \tag{7.11}$$

The result in the time domain h_m can be seen in figure 7.6.

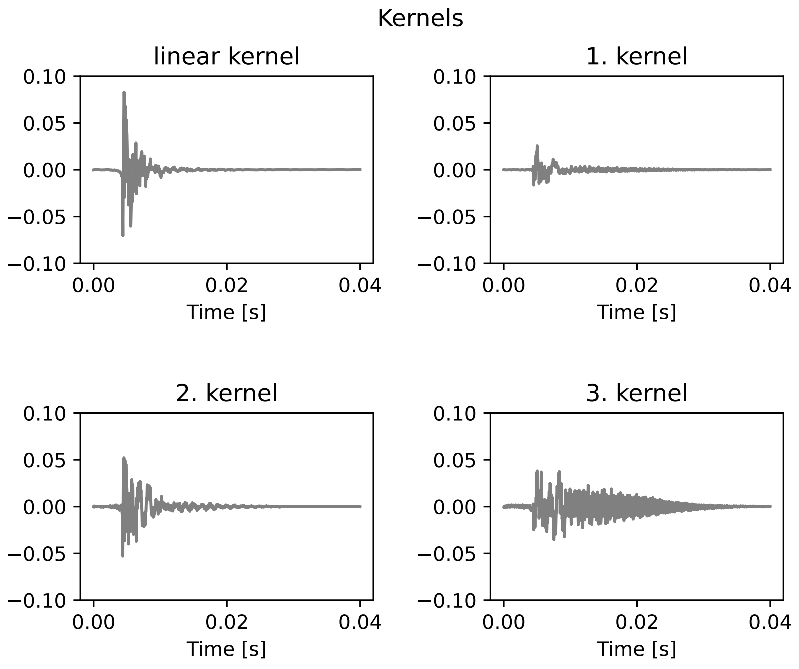

Figure 7.6: Identified Hammerstein kernels up to the third harmonic

Using the identified kernels the sweep response can be reconstructed using convolution. First the linear kernel is convolved with the sweep. Next both the sweep and the kernel are up sampled. The up sampled sweep is squared and then convolved with the up sampled kernel. The down sampled result is added to the linear part. For higher order kernels the

up sampling as well as the power factor change accordingly.
Figure 7.7 sums up the proccess in form of a flow chart.

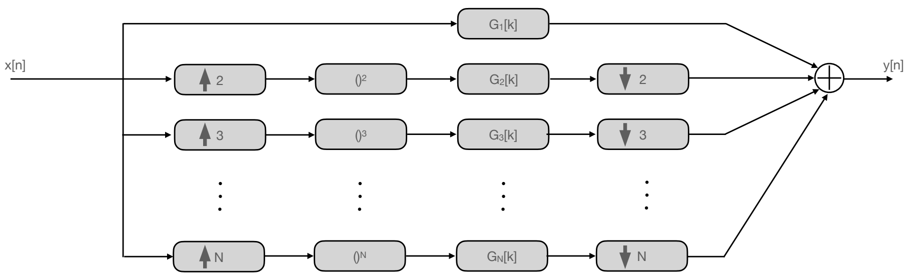

Figure 7.7: Reconstructed signal y[n] out of the signal x[n]

This can be done for the sweep (see figure 7.8) as well as other signals.

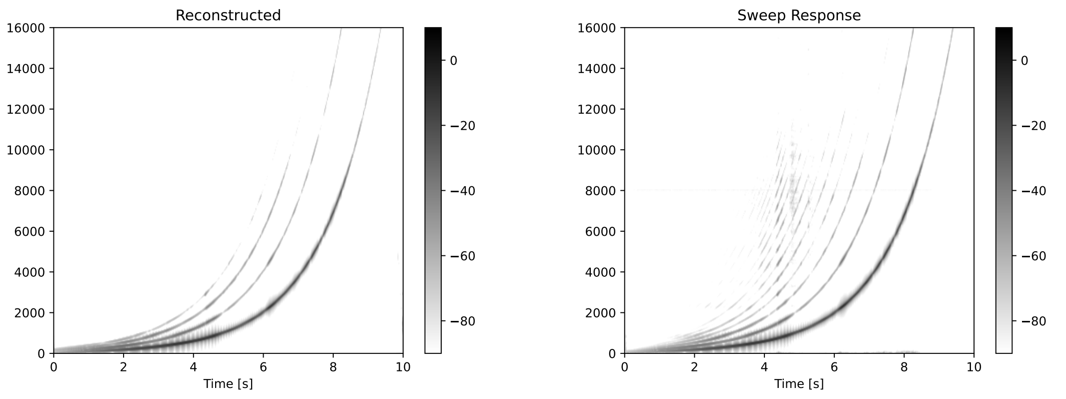

Figure 7.8: Reconstructed sweep response (left figure) and measured sweep response (right figure)

The linear part of the response can be reconstructed well (see figure 7.8), as the linear part approaches the one found in the response. The first three harmonics can be modeled quite well, as the reconstruction looks fairly similar to the recorded sweep response.

7.2.3 Signal Flow of the Data Generator

First the root mean squared (RMS) value of the clean far-end speaker signal is corrected to the value of the signal as received over the network. Next, a noise signal is mixed with the far-end signal, considering a pre-defined signal-to-noise ratio (SNR). The resulting signal also represents the loudspeaker signal over the network. In order to simulate the processing delay this signal is delayed. The non-delayed signal is used to build the loudspeaker signal. After simulating the playback processing module, given a specific loudspeaker gain setting, the nonlinearities of the loudspeaker are considered by simulating them using a Hammerstein model (see chapter 7.2.2). Finally, a room random room impulse response is imprinted on the resulting signal.

The clean near-end speaker signal is first convolved with a random room impulse response, albeit being the same used for the loudspeaker path. Next the signal's RMS is corrected in order to consider typical speaker distances in relation to the device. The microphone characteristic of the device is also considered and imprinted on the signal. The resulting near-end is mixed with a noise signal. A pre-defined SNR is thereby being considered.

In order to create realistic loudspeaker and microphone signals noise is added to both. For the microphone signal white noise with -80 dB full scale (dBFS), whereas the filtered white noise is added to the loudspeaker signal. The second order low pass filter has a cut-off frequency of 20 Hz and the rms value is also -80 dBFS.

The described data augmentation steps can be seen in figure 7.9. The signal processing of the device is simulated by the audio core executable, among other things applying the microphone equalizer and calculating the prediction of the acoustic echo cancellar.

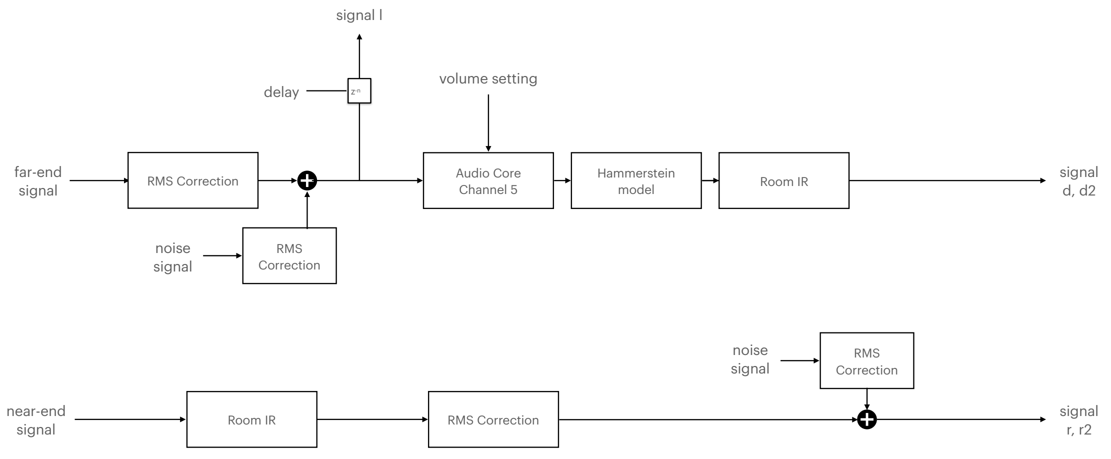

Figure 7.9: Data augmentation flow chart

In order to get the inputs of the neural network the signals are once more processed by the audio core (figure 7.10). The near-end microphone signal r is used as the target signal.

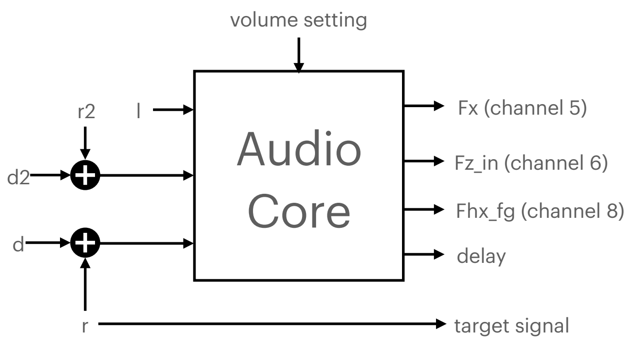

Figure 7.10: Input and output of audio core

The values of the RMS of the various signal, the delay, and the SNR are chosen according to measurements from the recorded data as well as values observed in the real world use case.

The microphone RMS value is modeled using a Gaussian distribution. The mean value of the distribution is -41.0 dB and the standard deviation is 2.7 dB. As can be seen in figure 7.11 the recorded data follows a multimodal distribution, as for the recorded data only two microphone distances, 50 and 100 cm, were considered. Since it cannot be assumed that those are the only speaker to microphone distances, modelling the data with the Gaussian distribution makes sense.

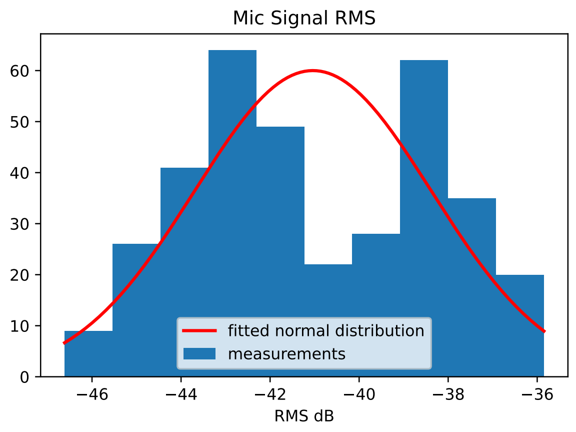

Figure 7.11: Distribution of the RMS of the near-end microphone signals of the recorded dataset

The RMS of the far-end signal is modeled using a normal distribution. A mean value of -23.6 dB and a standard deviation of 2.5 dB could be observed. The distribution of the RMS of the far-end microphone signals is shown in figure 7.12.

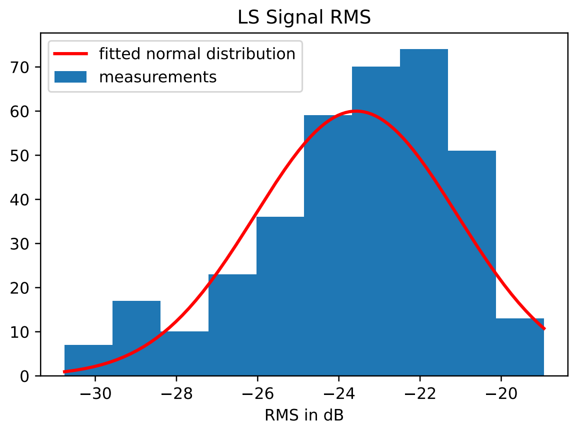

Figure 7.12: Distribution of the RMS of the far-end loudspeaker signals of the recorded dataset

For the SNRs uniform distribution between 0 and 25 dB is used. The choice is motivated by observation of real world data. For the distribution of the volume setting of the device the higher volumes are heavily favored, as for 95 % of the data the volume is set between 8 and 12, for 4 % between 5 and 7. Far-end and near-end only examples make up 10 % each of the dataset, whereas the majority, i.e. 80 %, are double talk scenarios.

Finally, the processing delay is also drawn from a normal distribution with a mean value of 30 ms and a standard deviation of 6 ms. The distribution is cut off at 0 and 100 ms. Again, these values were chosen from observation of the device in real world usage.

Altogether 330 hours of trainings and 10 hours of validation data were created, with 60% of the data having applied noise.

7.2.4 Verification of the Synthetic Data

In order to verify that the process of the synthetic data generation the validation dataset was recreated using the processing pipeline for the synthetic data. The result is the same amount of data as in table 7.1, with the same far-end and near-end speaker signal as the recorded dataset. Those signals are processed in the same way as described in chapter 7.2, except the near-end microphone signal. For this signal two sweep measurements were performed, with the same recording set-up as described in chapter 7.1, for the two recording distances. Then the steps of convolving the near-end speaker signals with a room impulse response and the microphone characteristic (see chapter 7.2) were replaced with the convolution of the sweep responses for 50 and 100 centimeter recording distances respectively. The resulting dataset was compared to the recorded dataset, by evaluating the metrics described in chapter 6. The post-processing filter was ignored for the measurements as only the audio core signals are used.

	recorded data		synthetic data	
metric	mean	σ	mean	σ
SDR [dB]	-2.3	8.9	-0.4	8.7
ERLE [dB]	8.6	4.9	9.6	5.2
PESQ [MOS]	1.6	0.4	1.7	0.5
STOI	0.7	0.3	0.7	0.3
AECMOS ECHO [MOS]	3.1	0.8	3.0	0.8
AECMOS DEG [MOS]	2.7	0.9	2.6	0.9

Table 7.3: Comparison between synthetic and recorded data

As can be seen in table 7.3 the recorded and synthetic data are not too far off. For all the non signal ratio based metrics, i.e. metrics which are not the SDR or ERLE, the results look very promising as on average they are only one decimal point off with a similar standard deviation. For the SDR and ERLE, the results differ more significantly, but there the standard deviation is also higher. All in all, the results for the synthetic data look promising enough to further evaluate their usage as training data.

Chapter 8

Post Filter Neural Network Model

Now it is time to put all the different blocks together. Out of the different neural network building blocks described in chapter 3 four different models are defined. Moreover, out of the cost function defined in chapter 5 it is necessary to find a suitable one.

The four different model architectures are compared in terms of their performance and computational complexity. The first model (model 1) takes the three input signals and performs a short time Fourier transform (STFT). Next it applies a filter bank to the input feature spectra (see figure 8.1). Using this filter bank the 257 Fourier bins are mapped to 100 mel bins. The logarithmic perception of frequency by the human hearing system motivates the transform to logarithmic spectra. Next the model uses a linear layer. This layer applies a `tanh` activation function to the weighted layer input.

$$out = \tanh\left(w^T \cdot x + b\right) \tag{8.1}$$

The features passed to this layer are the logarithmic loudspeaker STFT spectrum as well as the difference between the logarithmic AEC prediction and logarithmic microphone STFT spectra. Moreover, the delay index and volume setting of the device are passed as integer values. The linear layer uses 100 neurons with the output passed to a GRU layer with the same number of neurons. This layer is followed by another dense layer. In order to map the resulting mel spectrogram to a Fourier spectrogram one an inverse filter bank is added as a last step.

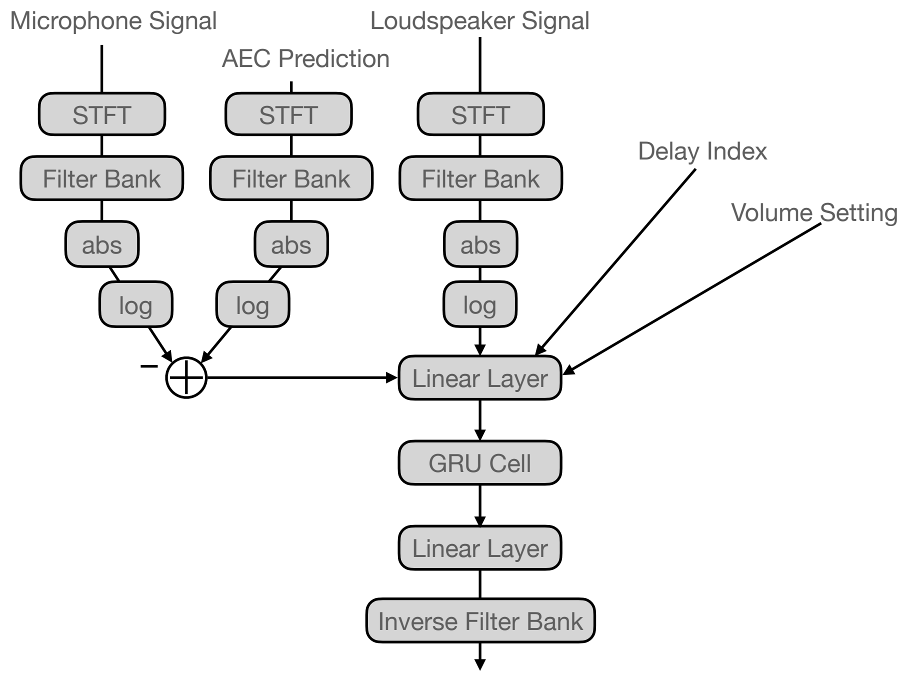

Figure 8.1: Proposed model architecture using a filter bank (model 1)

The second model (model 2) does not use a filter bank. The linear layer and GRU both use 128 neurons. The architecture is depicted in figure 8.2.

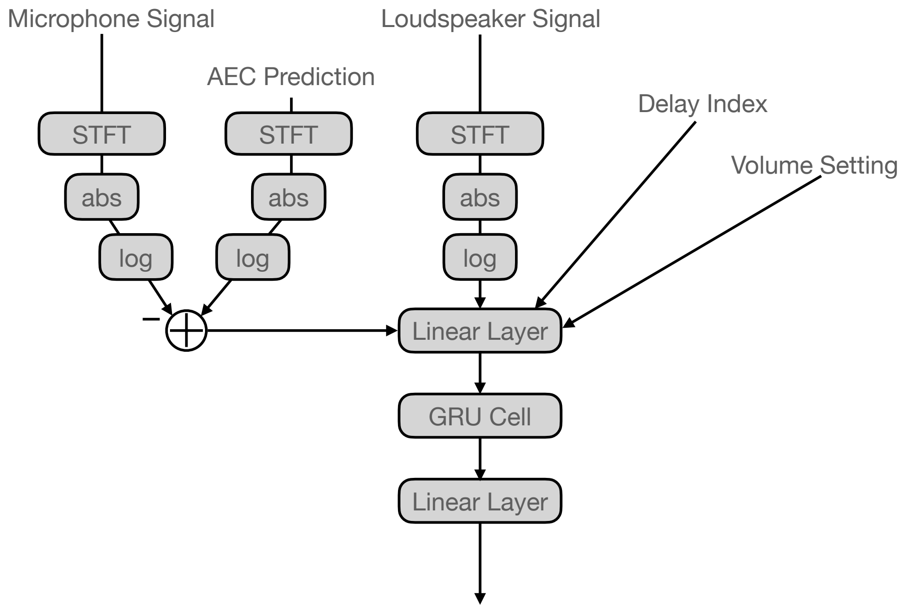

Figure 8.2: Proposed model architecture without a filter bank (model 2)

For the third model (figure 8.3) the GRU cell of model 2 was replaced by a LSTM cell with 128 neurons.

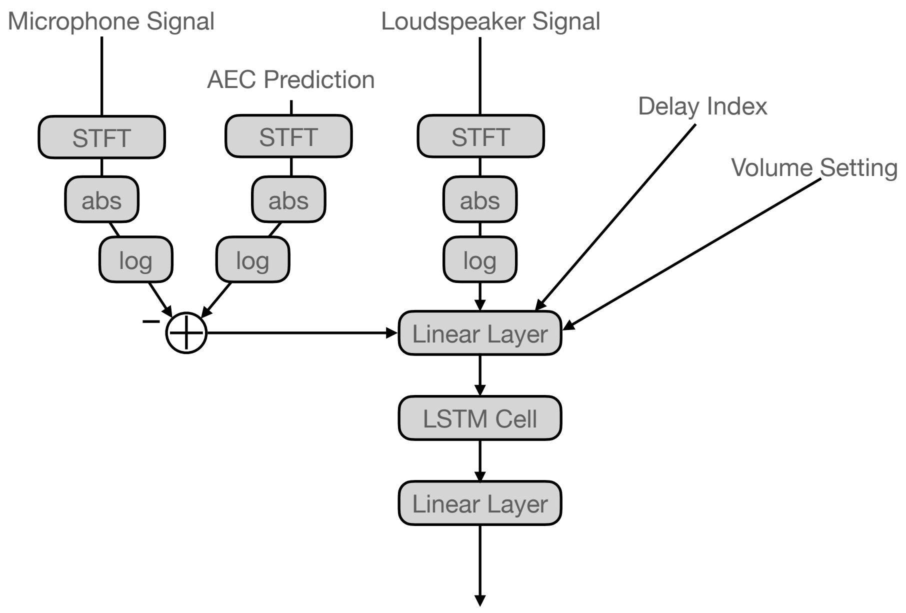

Figure 8.3: Proposed model architecture with a LSTM cell (model 3)

Finally, model 4 (figure 8.4) is composed of a LSTM cell with 100 neurons as well as filter banks for the input feature spectrograms.

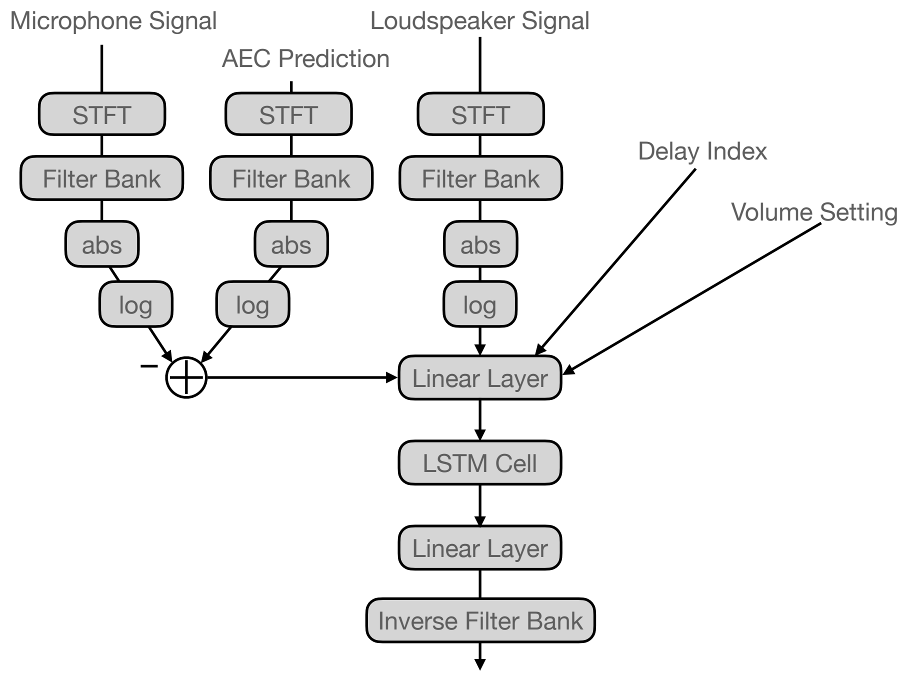

Figure 8.4: Proposed model architecture with a LSTM cell and filter bank (model 4)

8.1 Optimizers used for training

All the cost functions were optimized using the ADAM optimizer implementation of TensorFlow [6] with a learning rate of 10^{-3}. For the combined loss (equation 5.5) the ADAM optimizer with decaying learning rate was used. The initial learning rate was set to 10^{-3}, with the decay being $0.1 \cdot 10^{-3}$ per epoch. The values for β were set to their default values $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $epsilon = 10^{-6}$.

8.2 Cost Function Selection

The different cost functions are compared. In order to do so, all parameters except the cost function are fixed. The model used is model 2 as described in figure 8.2. The recorded training and validation data (see chapter 7.1) are used, and all models are trained for 30 epochs. The obtained models are evaluated on metrics, the perceptual speech quality measure (PESQ), echo return loss enhancement (ERLE), and signal to distortion ratio (SDR), as described in chapter 6.

Cost Function	PESQ	ERLE	SDR
$\mathfrak{L}_{Binary}$	2.00	12.78	9.40
$\mathfrak{L}_{SNR}$	2.00	12.24	9.20
$\mathfrak{L}_{SDR}$	1.80	12.49	9.10
$\mathfrak{L}_{SNR+SDR}$	1.90	12.27	9.00
$\mathfrak{L}_{Comb}$	2.22	12.93	8.86

Table 8.1: Comparison of different cost functions

While all models show quite similar performance when the SDR and ERLE is observed, the PESQ score differs significantly between the different cost functions. As is seen in table 8.1 the combined loss achieves the highest PESQ score. Therefore, this cost function is used for all the following training runs. These findings line up with [11] and [10].

8.3 Model Architecture Selection

For the different models validation metrics (see chapter 6) are evaluated after 300 training epochs. The results are summed up in table 8.2. Additionally, the CPU usage was measured on the device. The results are compared to the baseline system, where no post filter was acitve.

Model	PESQ [MOS]	ERLE [dB]	SDR [dB]
Baseline	1.81	8.46	-0.75
Model 1	2.30	14.13	8.63
Model 2	2.29	14.44	8.89
Model 3	2.30	14.45	8.89
Model 4	2.29	14.19	8.69
Model	**AECMOS DEG [MOS]**	**AECMOS ECHO [MOS]**	**STOI**
Baseline	2.65	3.12	0.70
Model 1	2.55	4.40	0.75
Model 2	2.67	4.42	0.76
Model 3	2.69	4.43	0.76
Model 4	2.57	4.40	0.75
Model	**# Parameters**	**CPU usage [%]**	
Baseline	-	55	
Model 1	91000 (+51400)	66	
Model 2	198401	69	
Model 3	230913	72	
Model 4	110800 (+51400)	65	

Table 8.2: Model architecture comparison

In terms of the metrics the models perform similarly, while all showing significant improvements over the baseline system. Especially the main goal, improving the echo suppression, works well as can be seen by the high AECMOS Echo scores. While models 2 and 3 increase the AECMOS Deg score compared to the baseline, models 1 and 4, i.e. the models which include a filter-bank have a lower score meaning that the near-end signal is more destroyed when compared to the baseline. However, this is not visible when looking at the PESQ score. Here all models improve by about the same value, the same as the STOI. Also, for the ERLE and SDR all models gain about 6 dB and 9 dB respectively, when compared to the baseline system. The results for all models justify the inclusion of the filter in the device.

The number of parameters can be dramatically reduced by using a filter bank. It should be noted, that in order to take advantage the filter bank needs to be implemented efficiently on the device as in the worst case 52400 parameters are added. Since the transformation matrix is sparse this number can be reduced, therefor offering the lowest computational complexity, while maintaining an equally good performance to a model without the filter bank. Unfortunately on the device this would need further changes in the implementation as can be seen by the similar CPU utilization between the models with and without the filter-bank.

8.4 Final Model

Because of the limited computational capabilities of the device model 4 is implemented, taking advantage of the feature reduction introduced by the filter bank.

Still it needs to be investigated which training data is better suited. As described in chapter 7.2.4 the synthetic data is modelled quite well, as the recorded training data can be recreated using the synthetic data generation process with the performance of the baseline system being quite similar (see table 7.3 in chapter 7.2.4). Due to practicality the recorded training data (see chapter 7.1) was recorded in the same room. This can be fatal, as different rooms have different acoustic properties. The synthetic training data (see chapter 7.2) has the advantage that different rooms are considered. Moreover, it is important that the scene of both parties of the call are retained, meaning that environment noise, like a tram passing, should be transmitted. This is also considered in the synthetic training data.

Model 4 was trained with both the synthetic and recorded training data. The cross validation of those models trained for 300 epochs can be seen in table 8.3.

	ERLE [dB]				SDR [dB]			
data type	synth. training		rec. training		synth. training		rec. training	
	mean	σ	mean	σ	mean	σ	mean	σ
synth. validation	14.9	7.1	19.6	9.3	6.1	4.5	3.0	2.4
rec. validation	13.9	6.8	16.1	8.2	5.2	6.1	7.9	3.7
	PESQ [MOS]				**STOI**			
data type	synth. training		rec. training		synth. training		rec. training	
	mean	σ	mean	σ	mean	σ	mean	σ
synth. validation	1.5	0.4	1.3	0.3	0.7	0.2	0.6	0.2
rec. validation	1.7	0.5	2.0	0.6	0.7	0.3	0.7	0.3
	AECMOS ECHO [MOS]				**AECMOS DEG [MOS]**			
data type	synth. training		rec. training		synth. training		rec. training	
	mean	σ	mean	σ	mean	σ	mean	σ
synth. validation	3.7	0.5	3.9	0.5	2.7	1.0	2.6	1.0
rec. validation	3.6	0.6	4.1	0.4	2.4	0.8	2.4	0.9

Table 8.3: Comparison of the model trained with rec. and synth. data evaluated on synthetic and recorded data

The model trained on the recorded training data outperformed the model trained with the synthetic data on every metric on the recorded validation dataset. But also on the synthetic validation data the model trained on the recorded data on the ERLE and the AECMOS Echo score. Yet the model trained on the synthetic data offers better perceived speech quality, as both the PESQ and AECMOS DEG are higher compared to the model trained on the recorded data. Since training the model for more epochs resulted in better echo suppression, extending the training epochs past 300 for the model trained on the synthetic data would make sense. Compared to the model trained on the recorded data, the one trained on the synthetic data has to adapt to many room acoustic settings and more complex scenarios like noisy speech. Therefore, for a fairer comparison the number of training epochs should be increased for the model trained on synthetic data.

Chapter 9

Conclusion

State of the art neural network architectures can achieve the goal of better residual echo suppression by augmenting an existing system.

For the model architecture different models were considered. To reduce the footprint on the CPU a filter bank, which design was motivated by the human hearing, was implemented. The number of parameters of those models could be reduced dramatically, while retaining similar performance characteristics on the evaluation metrics. Through the application of the filter bank the headroom of the resulting system is large enough to deploy the proposed filter on the device.

As for the training data first the recorded data was compared to a recreation using measured transfer functions of the device. The results looked promising enough to also train the model on a synthetic data corpus. For this training data different room acoustics could be considered. Moreover, it is possible to introduce noise to the training data. Unfortunately, the model trained with said data could not reach the performance of the model trained on the recorded data. It would still make sense to further investigate the usage of the synthetic data. As the corpus is larger and more diverse acoustic situations make it harder for the system to adapt the model will probably need more training time compared to the one trained on the recorded data. Comparing the models after the same number of epochs is therefore not a fair comparison. Therefor, for now the model was trained on the recorded data.

The proposed model outperformed the baseline system such that it will be implemented on the device and be pushed out as a software update in the future. Especially, the echo suppression could be improved dramatically. But also the near-end signal could be retained naturally. This is true for all volume settings of the device, but especially for the high settings, where the existing system struggled.

As could be shown, the inclusion of a neural network based approach can improve the system. Therefore, one can assume that by replacing the rest of the system by a neural network would make sense. As more system resources are freed up the model could be bigger. Moreover, the training data generation would be made significantly easier as the existing system would not be needed to be emulated. It could even be based on the same architecture. This would also result in a more efficient way to role out future updates to the system. Now when changing anything in the signal processing chain before the post filter will result in the need to create new training data and the training of a new network.

Bibliography

[1] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural computation*, vol. 9, pp. 1735–80, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735.

[2] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, *On the properties of neural machine translation: Encoder-decoder approaches*, 2014. DOI: 10.48550/ARXIV.1409.1259. [Online]. Available: https://arxiv.org/abs/1409.1259.

[3] V. Stevens, “Newman, 1937 stevens ss, volkmann j., newman eb”, *A scale for the measurement of the psychological magnitude pitch, Journal of the Acoustical Society of America*, vol. 8, pp. 185–190, 1937.

[4] F. Weninger, J. R. Hershey, J. Le Roux, and B. Schuller, “Discriminatively trained recurrent neural networks for single-channel speech separation”, in *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2014, pp. 577–581. DOI: 10.1109/GlobalSIP.2014.7032183.

[5] N. Qian, “On the momentum term in gradient descent learning algorithms”, *Neural Networks*, vol. 12, no. 1, pp. 145–151, 1999, ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(98)00116-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608098001166.

[6] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[7] S. Ruder, *An overview of gradient descent optimization algorithms*, 2016. DOI: 10.48550/ARXIV.1609.04747. [Online]. Available: https://arxiv.org/abs/1609.04747.

[8] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. DOI: 10.48550/ARXIV.1412.6980. [Online]. Available: https://arxiv.org/abs/1412.6980.

[9] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2017. DOI: 10.48550/ARXIV.1711.05101. [Online]. Available: https://arxiv.org/abs/1711.05101.

[10] S. Braun and I. Tashev, “A consolidated view of loss functions for supervised deep learning-based speech enhancement”, in *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, IEEE, 2021, pp. 72–76.

[11] S. Braun and I. Tashev, “Data augmentation and loss normalization for deep noise suppression”, in *International Conference on Speech and Computer*, Springer, 2020, pp. 79–86.

[12] C. Taal, R. Hendriks, R. Heusdens, and J. Jensen, “An algorithm for intelligibility prediction of time–frequency weighted noisy speech”, *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 19, pp. 2125–2136, Oct. 2011. DOI: `10.1109/TASL.2011.2114881`.

[13] A. Rix, J. Beerends, M. Hollier, and A. Hekstra, “Perceptual evaluation of speech quality (pesq)-a new method for speech quality assessment of telephone networks and codecs”, in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*, vol. 2, 2001, 749–752 vol.2. DOI: `10.1109/ICASSP.2001.941023`.

[14] *Wideband extension to recommendation p.862 for the assessment of wideband telephone networks and speech codecs*, ITU-T, 2005.

[15] M. Purin, S. Sootla, M. Sponza, A. Saabas, and R. Cutler, *Aecmos: A speech quality assessment metric for echo impairment.* 2021. arXiv: `2110.03010v3 [eess.AS]`.

[16] K. Sridhar, R. Cutler, A. Saabas, *et al.*, *Icassp 2021 acoustic echo cancellation challenge: Datasets, testing framework, and results*, 2020. DOI: `10.48550/ARXIV.2009.04972`. [Online]. Available: `https://arxiv.org/abs/2009.04972`.

[17] R. Cutler, B. Naderi, M. Loide, S. Sootla, and A. Saabas, *Crowdsourcing approach for subjective evaluation of echo impairment*, 2020. DOI: `10.48550/ARXIV.2010.13063`. [Online]. Available: `https://arxiv.org/abs/2010.13063`.

[18] *Acoustics - measurement of room acoustic parameters - part 2: Reverberation time in ordinary rooms (iso 3382-2:2008); german version en iso 3382-2:2008*, DIN EN ISO, 2008-09.

[19] C. K. Reddy, V. Gopal, R. Cutler, *et al.*, “The interspeech 2020 deep noise suppression challenge: Datasets, subjective testing framework, and challenge results”, in *INTERSPEECH*, 2020.

[20] A. Novak, L. Simon, F. Kadlec, and P. Lotton, “Nonlinear system identification using exponential swept-sine signal”, *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 8, pp. 2220–2229, 2009.

[21] A. Novak, P. Lotton, and L. Simon, “Synchronized swept-sine: Theory, application, and implementation”, *Journal of the Audio Engineering Society*, vol. 63, no. 10, pp. 786–798, 2015.