Speech Enhancement and Deep Learning Speaker Separation

Separation, Identification, and Enhancement of a Conversational Partner in a Cocktail Party Environment

ES-23-SPA-10-1070

Aalborg University
Institute of Electronic Systems

Institute of Electronic Systems
Aalborg University
https://www.es.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
Speech Enhancement and Deep Learning Speaker Separation

Theme:
Speech Enhancement, Speech Separation, Deep Learning, Minimum Overlap-Gap Algorithm

Project Period:
Spring 2023

Project Group:
ES-23-SPA-10-1070

Participant(s):
Jonas Kronborg Busk
Morten Andersen
Rasmus Nielsen

Supervisor(s):
Jesper Jensen
Poul Hoang
Zheng-Hua Tan

Copies: 0

Page Numbers: 6

Date of Completion:
June 2, 2023

Abstract:

This report looks into a possible solution to the cocktail party effect, which depicts an environment with multiple different conversations, background music, and other sources of noise. The goal is to explore possible solutions for a speech enhancement system consisting of a speech separation, speaker ranking, and speech enhancement stage. This system would ideally be capable of isolating the user's conversational partner.
The foundation of the solution is based on the newly proposed Minimum Overlap-Gap algorithm for speaker ranking and enhancement. However, potential speech separation stages remain largely unexplored. This report investigates a single-microphone setup using deep learning.
Different state-of-the-art network architectures are explored, and two are chosen for further investigation. These are Convolutional TasNet and Dual-Path Recurrent Neural Network. The networks are trained and tested in 2-, 3- and 4-speaker scenarios. Possible improvement techniques are also explored.
Several models showed potential for enhancing the target speaker's voice.

Preface

The report comprises three principal sections. The initial section entails a comprehensive literature review of state-of-the-art neural networks employed in speech separation. Chapters 3 to 5 depict the practical implementation and development of the complete speech enhancement system. The final chapters encompass the evaluation and outcomes of the developed models.

Appendix A contains examples of loss graphs resulting from training the models.

Jonas Kronborg Busk
jbusk18@student.aau.dk

Morten Andersen
moande18@student.aau.dk

Rasmus Nielsen
rniel18@student.aau.dk

Aalborg Universitet, June 2, 2023

Contents

1 Introduction

Advancements in technology have given humanity access to advanced tools, medicine, entertainment, and much more. This has increased our standard of living tremendously over the years and has made it possible to ease or nearly remove reductions in bodily functions using prosthetics or devices such as hearing aids. The progression of hearing aids is a great example for showing this advancement clearly. They started out as simple horns, which do not amplify the sound but simply funnel the sound from a larger area into the ear [1]. These progressed into analog hearing aids using vacuum tubes and with the invention of the transistor the size shrunk, such that they could fit in an ear. However, these hearing aids only amplified the signal to a higher sound level and applied no other signal processing to reduce noise or single out signals of interest. This type of selected amplification, called speech enhancement, is of great interest due to the fact that hearing loss does not only affect a person's sound perception but also their ability to discriminate speech signals [2]. Speech enhancement has progressed over the years, with hearing aids being able to adapt to different listening environments and improving speech discrimination for users [1]. However, the problem known as the cocktail party problem remains unsolved. The cocktail party problem revolves around the ability to single out a signal of interest in an acoustic environment of multiple competing speakers, music, reverberation, and noise [3].

The cocktail party problem is regarded as one of the most challenging situations any speech enhancement system may encounter, due to the complexity of the acoustic environment. These systems aim to help the user by identifying and amplifying speech signals of interest while attenuating competing speakers and noise. This speech enhancement system can be used in applications such as hearing assistive devices (HADs) and speaker-phone systems. However, even with the most recent state-of-the-art, this remains a difficult problem [4].

The traditional methods for speech enhancement depend on the microphone setup. A single microphone speech enhancement system involves the estimation of temporal statistics of the conversational partner. A multiple microphone setup enables the use of beamformers, where the direction-of-arrival (DOA) and/or spatiotemporal statistics are typically used [4]. However, multiple speakers pose a big challenge for these traditional methods, as the competing speakers are often acoustically indistinguishable without additional information. For example, current DOA estimators such as steered-response power phase transform (SRP-PHAT), maximum likelihood, and deep learning-based DOAs are not able to robustly handle multiple speakers without a priori information on the conversational partner's location or voice activity [4].

Recent advancements in this area attempt to increase the robustness without the use of extensive data collection, while also attempting to keep the computational cost and latency minimal, such that a feasible implementation in hearing aids or other devices would be possible. The minimum overlap-gap (MOG) algorithm, by Poul Hoang et al., introduces a method for calculating the likelihood of possible candidate speakers talking to the user given separated audio channels for both the user and candidate speakers [4]. This algorithm relays on the turn-taking nature of

a conversation, where only one conversational partner is speaking at a time, and the other is silent, and the number of overlaps and gaps in sentences are minimal as it is considered rude or awkward [5]. The MOG algorithm is of interest due to the fact that it uses no additional sensors as previous algorithms do. The work done by Poul Hoang et al. was mostly focused on the ranking and enhancement [4]. Therefore, the potential designs for better-performing speech separation systems for such a speech enhancement system remain largely unexplored.
However, this still leaves the problem of separating the recorded audio mix into separate speech channels for each of the candidate speakers and the user. In this area, most recently, deep neural networks have provided advancements in accuracy, computational complexity, and latency. Extensive research has been conducted on the case of single-microphone speech separation, which is used as the foundation of this project.
Given these recent advancements, the following section will investigate how speech enhancement systems can be constructed, lay out the foundational concept for deep neural networks for speech separation, and investigate the state-of-the-art of single microphone speech separation.

2 Technical Analysis

This chapter lays out a basic formulation of the problem and a possible architecture of a speech enhancement system. This is followed by an investigation into the concepts of a neural network, including the basic functional blocks, training methods, and evaluation metrics. Afterward, state-of-the-art is examined for single-microphone speech separation. At the end of the chapter, the scope of the project is described.

2.1 Project Formulation

In this project, the speech enhancement will be based on the system infrastructure presented by Poul Hoang et al. [4]. This speech enhancement system consists of 3 functional units: separation, ranking, and enhancement. This system can be seen in figure 2.1.

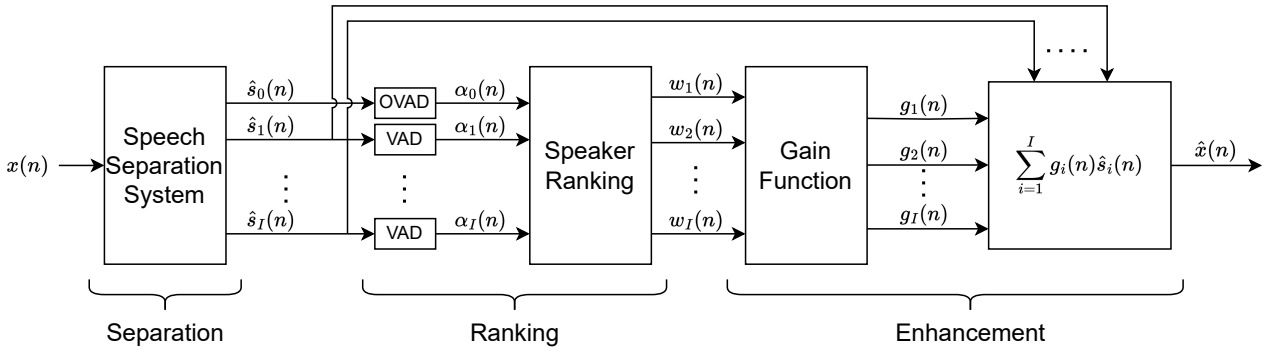


Figure 2.1: The functional blocks of a speech enhancement system [4].
OVAD produces the Voice Activity Detection (VAD) for the own voice (user), while the other VADs are for the candidate speakers.

The separation stage aims to separate the input signal or signals into I amount of channels, where each channel contains a candidate speaker and I is the number of candidate speakers. The candidate speakers can be defined as the conversational partner (target) and the competing speakers. Methods for this include beamforming or deep neural networks (DNN) such as TasNet [6, 7].

The ranking stage aims to rank the candidate speakers according to the likelihood of that speaker being the conversational partner for the user, also known as the target speaker. An algorithm for this ranking system is provided by Poul Hoang et al. called the minimum overlap-gap (MOG) algorithm [4]. The MOG algorithm has the advantage of not using additional sensors like electroencephalography (EEG) or eye tracking cameras, besides microphones. It uses the presence of the user's own voice to identify the target speaker. This algorithm exploits the turn-taking nature of conversations and attempts to rank the candidate speakers using speech overlaps and speech gaps between the user and candidate speakers.

The ranking stage of the MOG algorithm pairs the voice activity detection outputs of the user with the estimated target speaker by looking at the differences in the voice activities and chooses

the candidate speaker with the minimum amount of overlaps and gaps in the voice activities [4]. The enhancement stage outputs a linear combination of the separated speakers, where the speech channels are weighted according to the probability of a channel being the target channel. Another simple way to do the enhancement is to use a fixed gain for the candidate speaker with the highest likelihood of being the target speaker and attenuate the other channels.
The problem of interest in this project concerns the separation stage. The focal point will be on a single-microphone setup, where the input is a mix of the user and different candidate speakers. For single-microphone setups, the state-of-the-art uses deep neural networks for speech separation. The next section will therefore look into the basic building blocks of a neural network.

2.2 Layers of a Neural Network

To build a foundation for understanding the state-of-the-art in speech separation, an introduction to neural networks will be given. It will start off with the basic building blocks, such as fully connected layers and activation functions. It will then move on to more complex concepts such as convolutional layers, recurrent neural networks, and normalization functions.

2.2.1 Fully connected layer

A fully connected layer is the classical type of neural network. The layers are made with a number of input neurons that are connected to a number of output neurons. Every input neuron is connected to all output neurons. The connections between neurons apply a weight to the neuron value, which are the values learned by the algorithm. Each layer typically has a bias value, which is also learned. Layers can be stacked. The first layer is the input layer, the last layer is the output layer, and all layers in between are hidden layers. A network containing at least one hidden layer is called a deep neural network. The amount of neurons in a layer is referred to as the width of the layer. An example of a neural network with one hidden layer can be seen in figure 2.2.

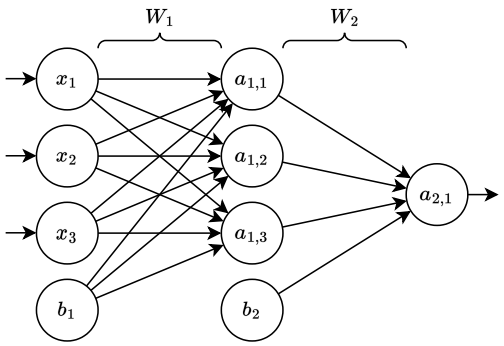


Figure 2.2: A fully connected layer with input layer x, a hidden layer, and one output layer.

Each layer is dependent on the previous layer, and therefore the final network consists of nested functions. When the values through the network are calculated the operation is called forward

propagation. Forward propagation can be defined using matrix multiplication as shown in equation 2.1.

$$g(x) = f^{(L)}(B^{(L)} + W^{(L)} f^{(L-1)}(B^{(L-1)} + W^{(L-1)} \dots f^{(1)}(B^{(1)} + W^{(1)} x) \dots)) \tag{2.1}$$

where:

x	is input vector
L	is the number of layers
W	is a matrix of weights
B	is a vector of biases
f	is the activation function
$g(x)$	is the map from the input x to the output y of the neural network

2.2.2 Activation functions

Activation functions are used to add non-linearity to the neural network. This ensures that the model will be able to learn a more complex task. If no activations are used, the network will be entirely linear. This is no different than using a linear regression model.

Sigmoid

A sigmoid activation layer is one of the first fixed layer activations to be used since its introduction in 1988 [8]. The mathematical representation is shown in equation 2.2 and the graph in figure 2.3.

$$f_{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

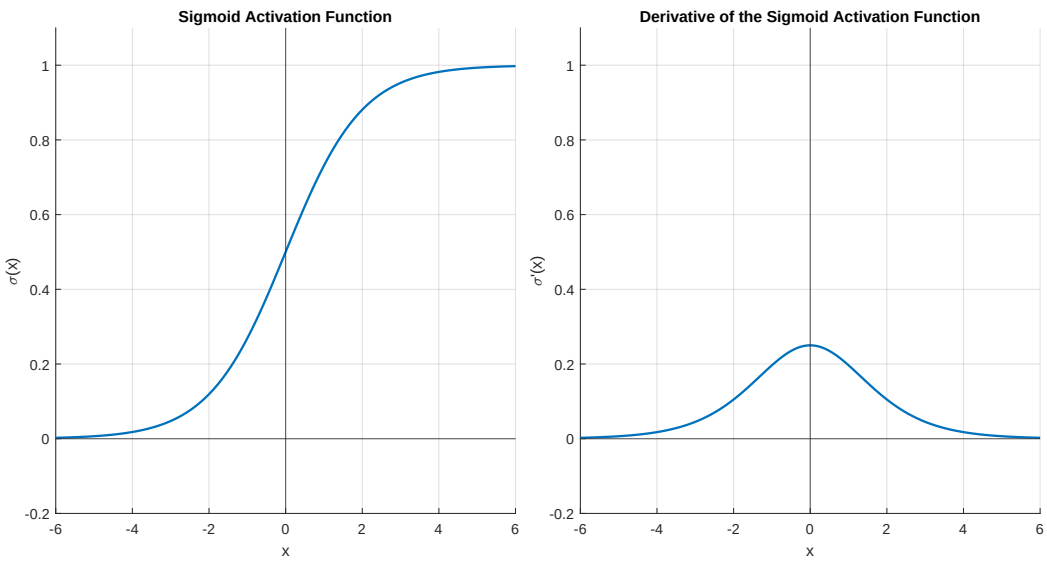


Figure 2.3: The sigmoid function shown on the right, and the derivative on the right.

Sigmoid activation has a problem when used in deep networks. Since the gradient has a maximum value of 0.25, the gradient will decrease when propagating through the layers when doing backward propagation [9]. This is called the *vanishing gradient problem*. This problem increases with the increasing depth of a neural network. A possible solution to this problem is using a rectified activation function such as the rectified linear unit (ReLU).

ReLU, LReLU, and PReLU

ReLU was suggested as a solution to the vanishing gradient problem that the sigmoid activation has. It sets the neuron activation as the largest value of either zero or the input x. This means the negative values are discarded and set to zero. The math can be seen in equation 2.3, where $a = 0$ in this instance. A graph of ReLU can be seen in figure 2.4.

$$f_{ReLU}(x) = \begin{cases} x & \text{for } x > 0 \\ a \cdot x & \text{for } x \leq 0 \end{cases} \tag{2.3}$$

$$f'_{ReLU}(x) = \begin{cases} 1 & \text{for } x > 0 \\ a & \text{for } x \leq 0 \end{cases} \tag{2.4}$$

When calculating the gradients in a neural network using backpropagation, the derivative of ReLu results in equation 2.4. The area where ReLu is zero can pose a problem. The learned bias can become so small that the neuron never fires, and thus the neuron becomes 'dead' [10]. This can affect upwards of 40% of the neurons [10]. This can be mitigated by changing the slope of the negative side of the ReLu function, to a low value. In this way, the negative input will still influence the gradient, but not as much as a positive input. This activation is called a leaky ReLU (LReLU), which is described in equation 2.3, where a is a small constant value.

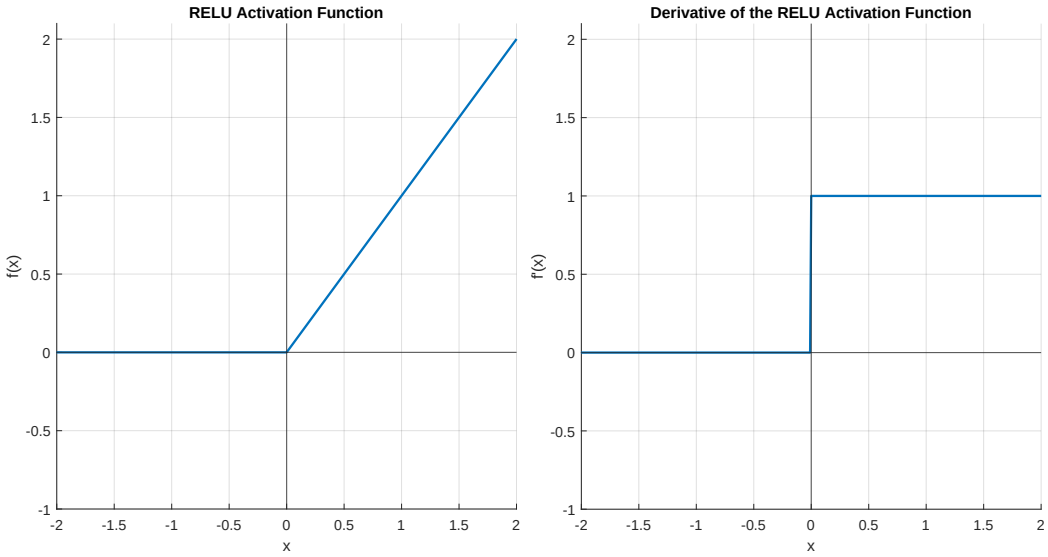


Figure 2.4: The RELU function is shown on the right, and the derivative on the right.

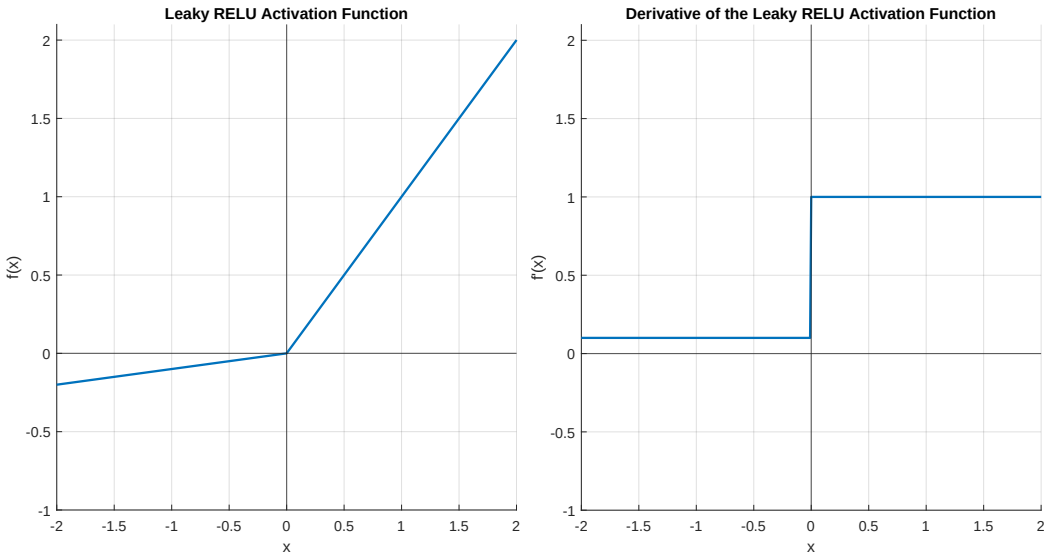


Figure 2.5: The Leaky RELU function shown on the right, and the derivative on the right.

A parameterized version of the ReLU is introduced, where a is a learnable parameter, and the activation is called PReLU. When updating the a parameter, the gradient is found by using equation 2.5.

$$\frac{\partial f(x)}{\partial a} = \begin{cases} 0 & \text{for } x > 0 \\ x & \text{for } x \leq 0 \end{cases} \tag{2.5}$$

2.2.3 Convolutional layer

The convolutional layers use the convolution operation. Here the learnable parameter is a matrix called a 'kernel' that is convoluted with the layer's input. Convolutional networks apply the same kernel to the whole input at a specified stride, if the kernel has a smaller size than the input. This results in parameter sharing across convolutional operations. Thus the convolutional layer is performing learnable feature extraction. Due to this parameter sharing, the convolutional layer can be more memory efficient, compared to fully connected layers, as fewer parameters must be stored. The feature-based representation typically reduces the dimensionality compared to the input. Thus the result is a smaller feature representation. Feature refers to the result of applying the same filter to the whole dataset, thereby differing from a fully connected layer, where unique weights and biases are used for each neuron. Subsequent operations on the feature representation can be more efficient as this representation is reduced in size. Multiple convolutions with different kernels/filters can be applied on the same array or matrix, thus making more channels. A one-dimensional convolutional layer (Conv1D) works as shown in figure 2.6.

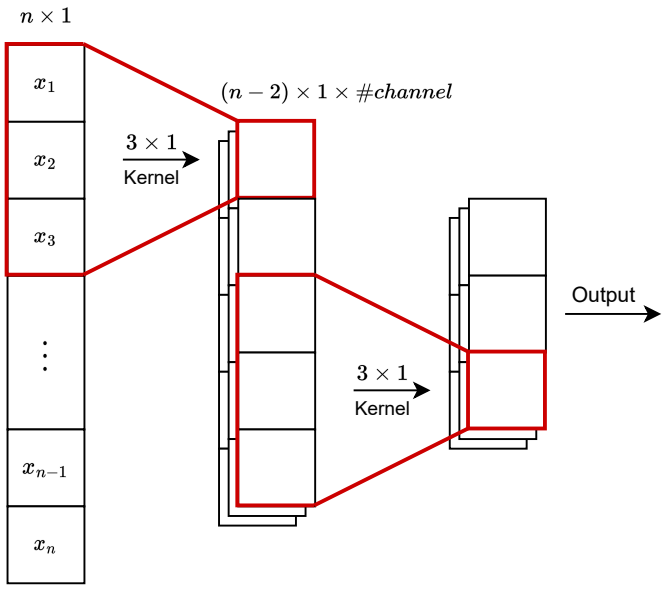


Figure 2.6: Visual representation of a 1D convolutional network with 3 layers, using a 3×1 kernel.

The convolution can be configured with a few options, namely stride and dilation. Stride describes how many entries the kernel shift for each operation. The default behavior is to shift one entry, but with a stride of two, it will be every other entry, as shown in figure 2.7. Dilation changes the spacing between the used entries, thereby expanding the receptive field. With a dilation factor of two, every other entry will be used. This can be seen on figure 2.8

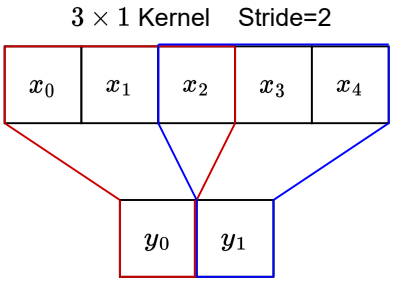


Figure 2.7: 1D convolution with kernel size 3×1 and stride is 2.

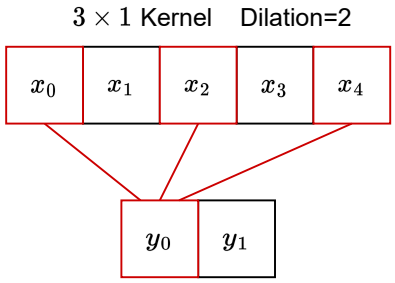


Figure 2.8: 1D convolution with kernel size 3×1 and dilation factor 2.

Training a convolutional layer using back propagation results in a gradient dependent on the subsequent layers' input gradient and the convolutional layer's input. These variables are combined using convolution to obtain the gradient for each variable in the kernel (Equation 2.6). The gradient of the loss with respect to the biases can be seen in equation 2.7.

$$\frac{\partial L}{\partial k} = x * \frac{\partial L}{\partial Z} \tag{2.6}$$

$$\frac{\partial L}{\partial b} = \sum \frac{\partial L}{\partial Z} \tag{2.7}$$

Pooling layer

Pooling layers can be used to reduce the dimension of a convolutional neural network. There are two different methods of pooling. The first one is average pooling. It calculates an average of all points in the kernel. A visual representation can be seen in figure 2.9.

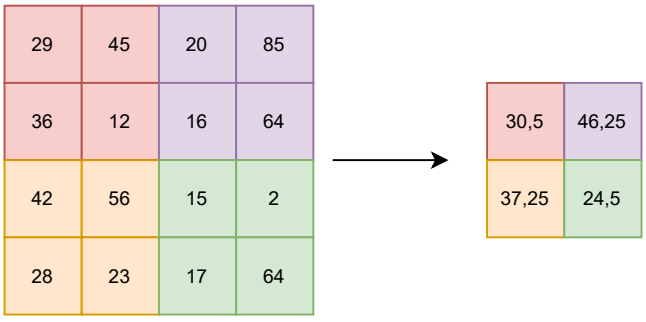


Figure 2.9: Shows how average pooling uses the average value of the 4 points to represent them.

This way of representing the data points is not great. As an example, if we have two different kernels with approximately the same mean but very different variances, they will end up being represented almost the same. Average pooling is smoothing the features. Another method called max pooling can be used, to alleviate this. This method uses the largest number in the kernel

to represent all the values in the kernel. This is better at keeping prominent features since no smoothing is done. A visual example is shown in figure 2.10.

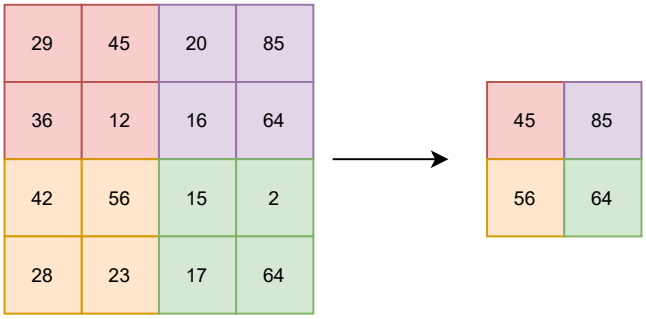


Figure 2.10: Shows how max pooling takes the largest value to represent the 4 points.

When a pooling layer is applied some information loss is occurring, but it is a trade-off between computational complexity and accuracy.

1×1 Convolution

When performing expensive operations, it can be computationally efficient to first reduce the channels. The number of channels can be reduced using a 1×1 convolution layer. A 1×1 convolution layer can also be used to increase the number of channels. The size of the convolutional layer does not change, only the channels since a $1 \times 1 \times n_{channels}$ kernel is used. It can be seen as a form of pooling layer just across the channels instead of the width/height of the data. By using a 1×1 before a larger kernel, like a 5×5, a reduction in computations is achieved. Figure 2.11 shows how a dataset with n channels can be changed into a set with k channels.

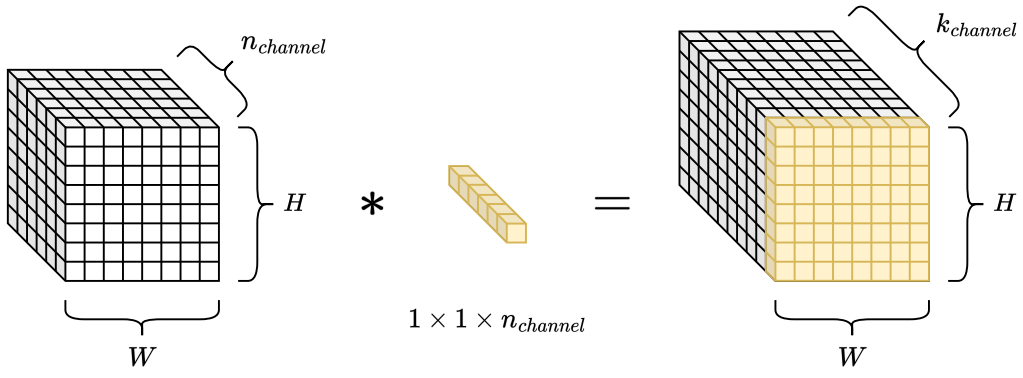


Figure 2.11: A 1×1 convolution used on a dataset of height H, width W and n channels, changing it to a set with k channels.

2.2.4 Recurrent Neural Network

Recurrent neural networks (RNN) are a type of neural network specialized for processing sequential data [11]. RNNs are able to generalize patterns across different sequence lengths and across different positions in time. RNNs achieve this by sharing parameters (Weights and biases) across several time stamps. This parameter sharing is done by using a feedback loop. The architecture of the feedback differs between implementations and the choice of specific RNNs. However, a common implementation is using equation 2.8 where the neuron has a feedback loop from its output back to itself, where it is weighted on the next timestamp.

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}; \theta) \tag{2.8}$$

Where:

$h^{(t)}$ is the output of the neuron at timestamp t
$x^{(t)}$ is input to the neuron at timestamp t
θ is the parameters of that neuron, typically a set of weights and biases

Due to this feedback loop, RNNs are capable of having a dynamic receptive field, while networks such as convolutional networks have fixed receptive fields [12]. The receptive field refers to the number of input variables that affect the output [13]. In convolutional networks, the kernel has a fixed size and therefore only a fixed amount of inputs can affect the output. However, in RNNs the weighting matrix of the feedback loop controls how previous samples affect the output. As this weighting matrix is a learned parameter, the network is capable of optimizing its own receptive field, thereby achieving a dynamic receptive field.
The described RNN is a causal RNN as it only uses values from past samples. However, in some applications, the performance of the neural network might increase significantly if it has access to the whole input sequence, both in the past and future. These applications include speech recognition, where the correct interpretation of the current phoneme may depend on the next phonemes due to co-articulation or even the next few words because of linguistic dependencies [11].
To incorporate future data points, a bidirectional RNN (bRNN) can be used. A bRNN combines two RNNs, one that moves forward in time (Starting at the beginning of the sequence) and one that moves backward in time (Starting at the end of the sequence). A diagram of a bRNN network can be seen in figure 2.12. As stated previously, using two RNNs starting from each end enables the output to depend on both the past and the future without specifying a window size, which enables the dynamic receptive field.

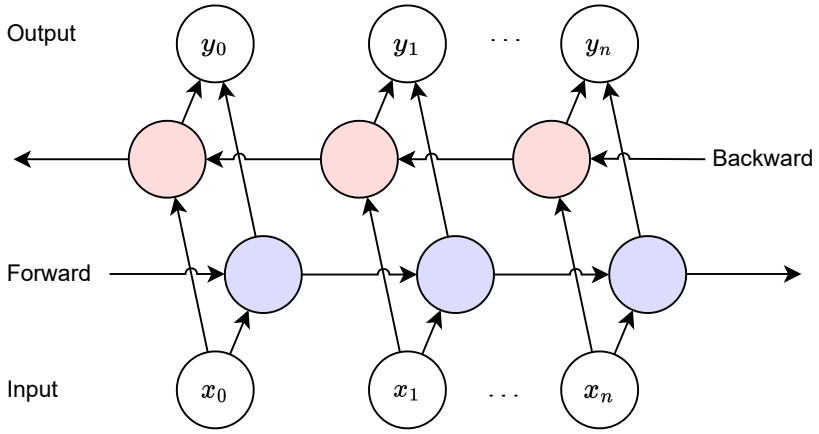
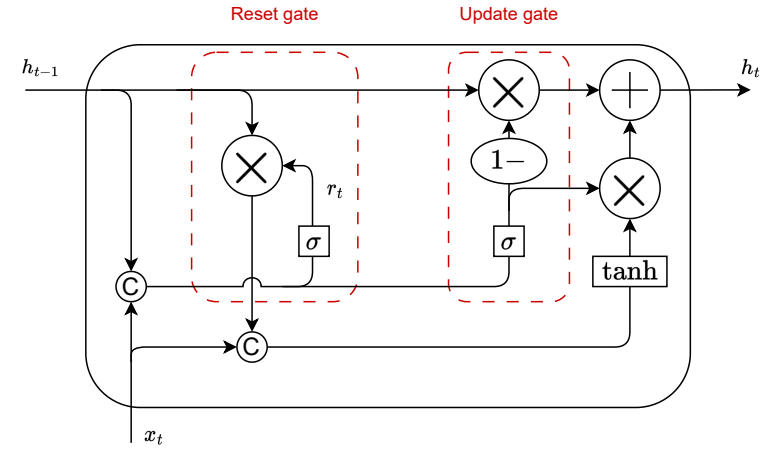
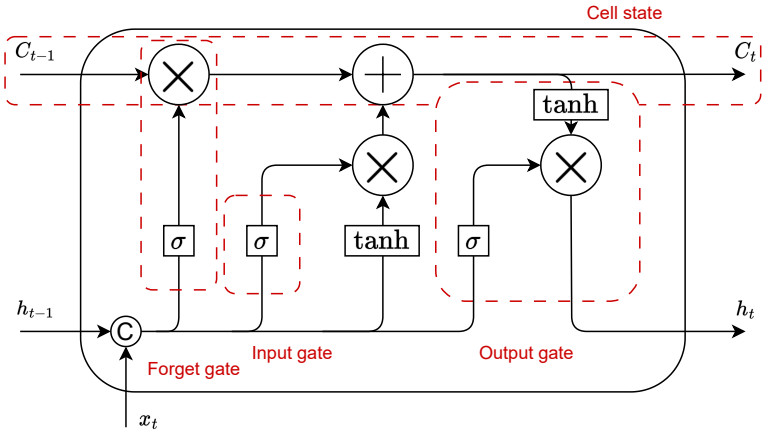


Figure 2.12: Shows the structure of a bidirectional RNN consisting of two sub-RNN. One that propagates the information forward in time (blue) and one that propagates information backward in time (red).

As with classic DNNs, RNNs also have the problems of vanishing or exploding gradients. With classic DNNs, a careful selection of weights and biases can in theory solve this problem as the weights and biases can be different. However, with RNNs the same weights are multiplied together many times. This results in the fact that the gradient will eventually explode or vanish depending on whether the value of the weight is above or below 1. This results in long-term dependencies being difficult for RNNs to learn as the gradient of the long-term dependencies will be exponentially lower than short-term dependencies.
There are multiple strategies or methods to try and mitigate this problem of learning long-term dependencies. One way to mitigate this problem is the use of multiple timescales, one that handles the short-term dependencies and one that handles the long-term dependencies. Another way is the use of skip connections, where an additional delay is added to the output. This results in the gradients diminishing as a function of $\frac{\tau}{d}$ instead of τ, where τ is the number of timesteps and d is the additional delay.
Another way of mitigating this problem is the use of a gated RNN architecture. These architectures include the gated recurrent unit (GRU) and the long short-term memory (LSTM) unit. The LSTM unit consists of 3 gates to control the flow of information. These are the input gate, the forget gate, and the output gate [11]. The architecture of an LSTM unit can be seen in figure 2.13b. Using these gates it is possible to produce paths where the gradients can flow for long duration, allowing the neural network to be more capable of learning long-term dependencies. It is also possible to make the weights conditioned on context rather than fixed as in a classic RNN. Due to variable weight, the time scale can also be changed dynamically.
GRU is a simplified version of the LSTM [11]. Instead of having 3 gates, it only has two. These are the update gate and the reset gate. The architecture of a GRU can be seen in figure 2.13a. The update gate controls how much information to save and the reset gate controls how much information to forget.



(a) Gated Recurrent Unit.



(b) Long Short Term Memory.

Figure 2.13: Shows two architectures for gated recurrent neural networks. 'C' refers to the concatenation of signals. σ and *tanh* are the sigmoid and *tanh* activation function.

2.2.5 Transformers

In 2017, a new approach emerged for processing sequential data [14]. These are known as transformers. Transformers make use of a concept called attention. Attention is a method for relating information at other time stamps to the current time stamp. Attention has been used previously, but often in combination with RNNs. Transformers rely solely on attention-based mechanisms and therefore process the whole input sequence all at once. This eliminates the constraints of sequential data processing as in the RNNs, which enables high levels of parallelization and removes the vanishing gradient problem in finding long-term dependencies. The attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and outputs are vectors [14]. The query (q), keys (k), and values (v) are constructed from the input by multiplying it with its corresponding learned matrix, i.e. $q_i = x_i \cdot W_Q$, $K_i = x_i \cdot W_K$, $v_i = x_i \cdot W_V$.

The original transformer made use of scaled-dot product attention [14]. This attention function

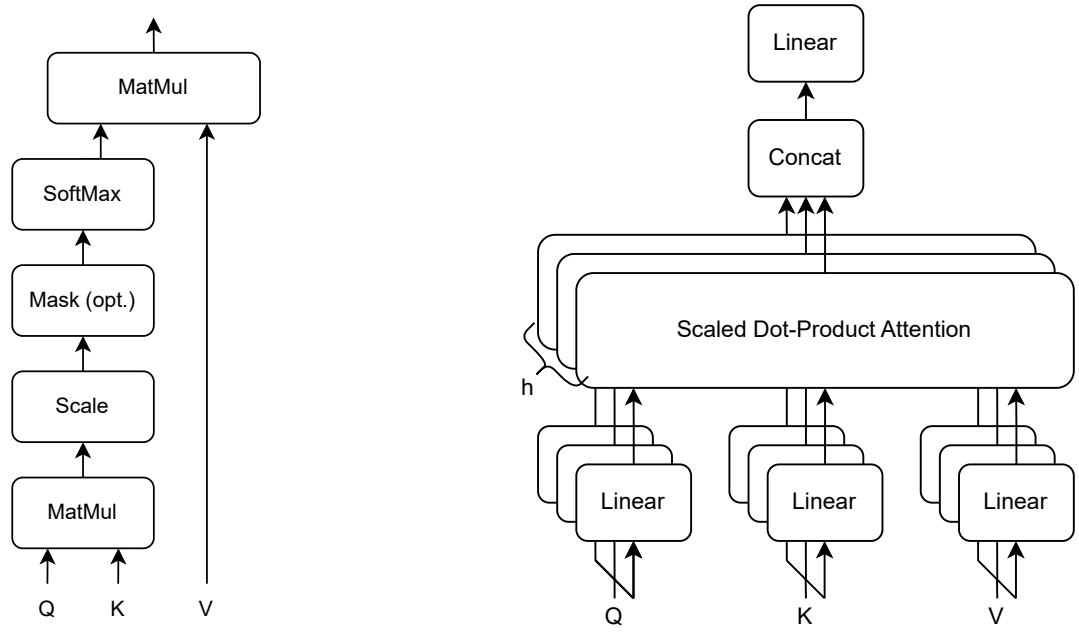
is computed as a weighted sum of the values, where the weights are computed as a function of the query with the corresponding key. In practice, the queries, keys, and values are packed into a corresponding matrix Q, K, and V. In equation 2.9, the calculated for scaled-dot product attention is seen using matrix computation.

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right) \cdot V \tag{2.9}$$

Where:

d_k is the dimension of the input keys

The scaling of d_k is done to counteract the small gradient of the softmax function when the magnitude of the dot product is high which is theorized to happen for large values of d_k [14].
The attention mechanism can be expanded to multi-head attention. In multi-head attention, the queries, keys, and values are linearly projected h times with different learned linear projections to d_q, d_k, and d_v dimensions respectively (The dimension of the queries and keys are kept equal as they are multiplied together in the scaled dot-product attention function, i.e. $d_q = d_k$). The scale-dot product attention is then applied in parallel to the projected queries, keys, and values. The results of this are then concatenated and projected again to achieve the final values.
The computation of the scaled-dot product and the multi-attention is illustrated in figure 2.14.



(a) Scaled Dot-Product Attention **(b)** Multi-head Attention

Figure 2.14: Shows the functional diagram of scaled dot-product attention and multi-head attention [14]. In the scaled dot-product attention MatMul referees to matrix multiplication and scaling is done by $1/\sqrt{d_k}$. In multi-head attention, the inputs Q, K, and V are first linearly projected into h dimension (Linear) before the scaled dot-product attention is applied.

The multi-head attention is the primary building block in the transformer architecture presented in [14]. This architecture is based upon an encoder/decoder structure. The encoder employs multi-head attention followed by two linear layers with a ReLU activation function between them. After each step, an add and normalization operation is performed. The decoder uses masked multi-head attention as the first operation. The next step uses the output of the encoder as the queries and keys in the next step. The output of the previous step is used as the values. This is followed by a linear layer as in the encoder. Add and normalization is used after each step. As the transformer architecture makes no use of convolution or recurrent networks, it requires a positional encoding to make use of the sequence order. The positional encoding can use sinusoidal functions. The complete architecture can be seen in figure 2.15.

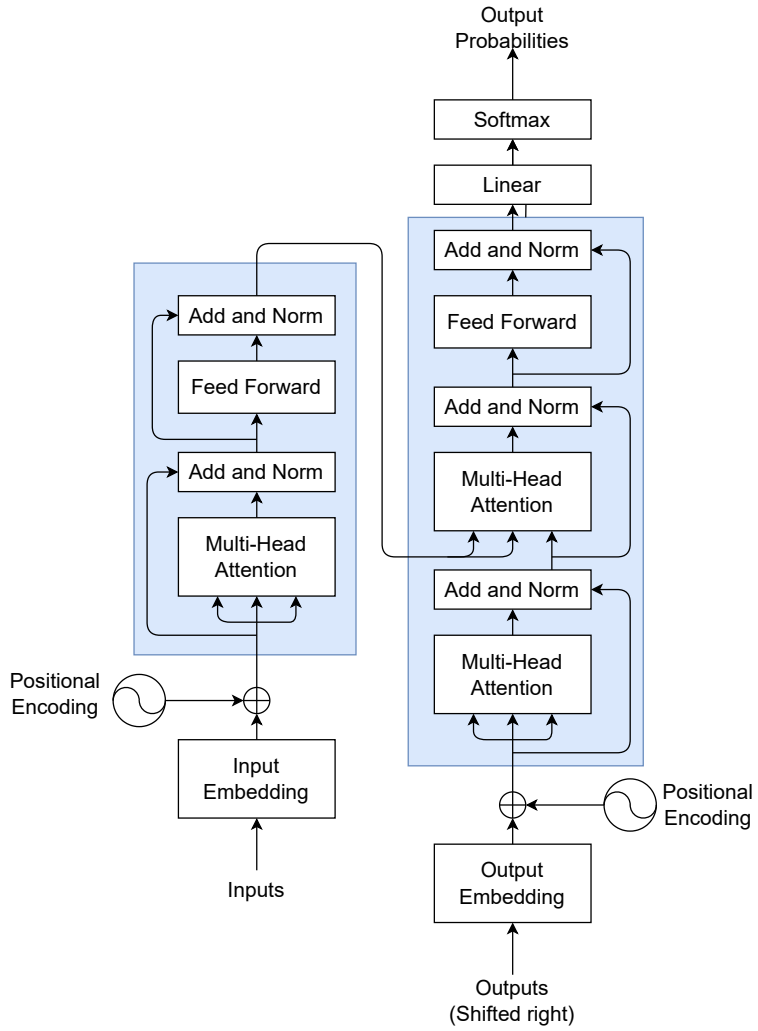


Figure 2.15: Shows the architecture of the original transformer using an encoder/decoder setup [14].

Transformers, based solely on attention, have proven to outperform RNNs with attention at tasks such as language translation and simultaneously allowed for a higher level of parallelization [14]. This is also currently the method used for achieving state-of-the-art performance in speech separation algorithms [15, 16].

2.2.6 Backwards Probagation

The weights and biases can be adjusted to improve the model with a method called backward propagation. This is an algorithm that, in combination with optimization methods, finds the set of weights and biases that optimizes the objective function. Different objective functions can be used depending on the application including MSE, cross-entropy, and absolute error. For each batch of training data, the loss is calculated and used for the correction of the parameters. It is not possible to change the activation functions ($f(\cdot)$) directly as they depend on the weights, biases, and the previous activations (Equation 2.10). Having this in mind, it is possible to look at how the loss changes when the parameters are changed a bit and then adjust the parameters until the objective function is optimized. Backward propagation computes the gradient of the objective function with respect to each parameter using the chain rule. For each layer in the network three partial derivatives have to be computed, namely the change in the loss with respect to the weights (Equation 2.11), the bias (Equation 2.12), and the input of the layer (Equation 2.13). This can be extended to larger and more complex network structures and different loss functions.

$$A_2 = f(W_1^T A_1 + b_1) \tag{2.10}$$

$$\frac{\partial C}{\partial W_1} = \frac{\partial C}{\partial A_2}\frac{\partial A_2}{\partial f}\frac{\partial f}{\partial W_1} \tag{2.11}$$

$$\frac{\partial C}{\partial b_1} = \frac{\partial C}{\partial A_2}\frac{\partial A_2}{\partial f}\frac{\partial f}{\partial b_1} \tag{2.12}$$

$$\frac{\partial C}{\partial A_1} = \frac{\partial C}{\partial A_2}\frac{\partial A_2}{\partial f}\frac{\partial f}{\partial A_1} \tag{2.13}$$

The partial derivatives are used in an optimization algorithm to update the parameters recursively using the training set and the associated labels. When all the available training data has been passed forward and backward through the network, it is called an epoch. It is common to go through a number of epochs, to train the network more, and thus get a better model. The samples are scrambled between epochs to get different sample combinations in the batches. Several different optimizers can be used to update the parameters in an iterative way. These include gradient descent (GD), stochastic gradient descent (SGD), mini-batch gradient descent, ADAM etc.

Gradient descent (GD) is a relatively efficient optimization method and uses equation 2.14 to update the parameters, where θ is the parameters, and $J(\theta)$ is the sum of the losses for a batch. Here i is the batch number and goes from 1 to the number of batches. When i is equal to the number of batches one epoch has occurred.

$$\theta_{(i+1)} = \theta_{(i)} - \mu \cdot \nabla J(\theta_{(i)}) \tag{2.14}$$

Since the parameters are updated once after seeing the whole dataset, meaning that there is one batch for each epoch, the gradient will typically be large which can result in the estimated

optimum hovering over and missing the optimum. A solution to this problem is to use the same updating rule as in GD but update the parameters more frequently using SGD where the parameters are updated after seeing each data point. However, this can result in noisy steps that go away from the optimum because it is influenced by every sample. To mitigate this the dataset can be split into mini-batches. Using mini-batches the large gradient from the GD method and the gradient noise from the SGD method can be avoided.
Another widely used optimizer is the "Adaptive Moment Estimation" -method or ADAM. This method calculates different adaptive learning rates for the different parameters using estimates of the mean and variance of the gradients [17]. ADAM uses five steps (equation 2.15 through 2.19) to update the parameter, where equation 2.15 and 2.16 computes an estimation of the first and second moment of the gradient, equation 2.17 and 2.18 is used to correct for the introduced bias and equation 2.19 updates the parameters using the previous parameter.

$$m_{(t)} = \beta_1 m_{(t-1)} + (1 - \beta_1)\, \nabla(C_{(t-1)}) \tag{2.15}$$

$$v_{(t)} = \beta_2 v_{(t-1)} + (1 - \beta_2)\, (\nabla(C_{(t-1)}))^2 \tag{2.16}$$

$$\hat{m}_{(t)} = \frac{m_{(t)}}{1 - \beta_1^t} \tag{2.17}$$

$$\hat{v}_{(t)} = \frac{v_{(t)}}{1 - \beta_2^t} \tag{2.18}$$

$$\theta_{(t)} = \theta_{(t-1)} - \alpha \frac{\hat{m}_{(t)}}{\sqrt{\hat{v}_{(t)}} + \epsilon} \tag{2.19}$$

ADAM updates the exponential moving average of the gradient and the gradient squared where β_1 and β_2 sets the exponential decay of the moving averages [17]. The scalar ϵ in equation 2.19 ensures that the denominator will not become zero and is small, typical $\epsilon = 10^{-8}$.
ADAM is a simple and efficient gradient-based optimizer, that performs well with large datasets and high-dimensional parameter spaces. It combines several features from other adaptive optimizers making it perform well with sparse gradients and non-stationary objectives. In conclusion, the ADAM algorithm is robust and is one of the most used optimizers for backpropagation in neural networks [17, 18].

2.2.7 Normalization Functions

When working with neural networks, it is possible to overtrain the network. This results in the model overfitting the training data and thereby removing the generalization of the model. Regularizes and Normalizations can alleviate this problem by making the model more generalized. The aforementioned dropout and zoneout work as regularizes, and modify how or when the neurons fire. Normalization layers were first introduced with batch normalization, which improved training time and accuracy. It has since sparked a number of other papers

looking into other methods of normalization. These are presented in the following sections.

Batch Normalization

Batch Normalization (BN) normalizes neuron values, by subtracting the batch mean form each value and dividing by the standard deviation. Two learnable parameters γ and β are introduced to make the model more adaptable [19]. γ scales the output and thus changes the standard deviation. β shifts the mean. Equation 2.20 and 2.21 shows the process.

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \tag{2.20}$$

$$y_i = \gamma \hat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i) \tag{2.21}$$

The mean and standard deviation statistics are calculated for each neuron across different batches. The batch size will have an influence on the statistics. Figure 2.16 shows how the inter-batch statistics are calculated.

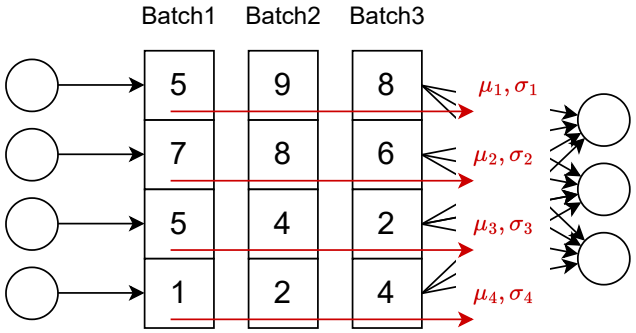


Figure 2.16: Statistics are calculated from the output of the same neuron across the batch.

To calculate the backward pass the following equations 2.22, 2.23 and 2.24 are used. Where m is the number of neurons in a layer.

$$\frac{\partial C}{\partial x_i} = \frac{\partial C}{\partial \hat{x}_i} \cdot \frac{1}{\sqrt{\sigma^2 + \epsilon}} + \frac{\partial C}{\partial \sigma^2} \cdot \frac{2(x_i - \mu)}{m} + \frac{\partial C}{\partial \mu} \cdot \frac{1}{m} \tag{2.22}$$

$$\frac{\partial C}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial C}{\partial y_i} \cdot \hat{x}_i \tag{2.23}$$

$$\frac{\partial C}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial C}{\partial y_i} \tag{2.24}$$

Batch normalization makes it possible to use larger learning rates [20]. It's also a regularizer, eliminating the need for dropout in some cases. However, it also comes with some negatives. Since it's statistics based, it needs a larger batch size when training, to approximate the mean and variance more precisely [19]. When using batch normalization with RNNs separate

hyperparameters γ and β are needed for each time step [19]. Using other forms of normalizations for RNNs might be more appropriate.

Weight Normalization
Following the problems present with BN, Weight Normalization (WN) was introduced as an alternative. WN does not use statistics over a batch like BN, so it can be used with RNNs and LSTM [21]. Weight normalization re-parameterizes each weight vector w, in terms of a parameter vector v and a scalar parameter g. Stochastic gradient descent is then done with respect to these parameters instead. The calculation is shown in equation 2.25.

$$w = \frac{g}{||v||}v \tag{2.25}$$

The two parameters can be decoupled, thus speeding up the convergence of stochastic gradient descent [21]. The gradient of the cost C differentiated with respect to v and g gives the following equations.

$$\nabla_g C = \frac{\nabla_w C \cdot v}{||v||} \tag{2.26}$$

$$\nabla_v C = \frac{g}{||v||}\nabla_w C - \frac{g\nabla_g C}{||v||^2}v \tag{2.27}$$

Layer Normalization
Layer normalization (LN) is another alternative to batch normalization, as it can be used on RNNs and LSTMs [22]. Layer normalization is done across the layer values instead of batch values. For an RNN, it corresponds to doing normalization for each time step. Layer normalization is calculated the same way as Batch Normalization using equation 2.20 and 2.21. The mean and standard deviation are calculated using values from all neurons in a single layer. An example is shown in figure 2.17.

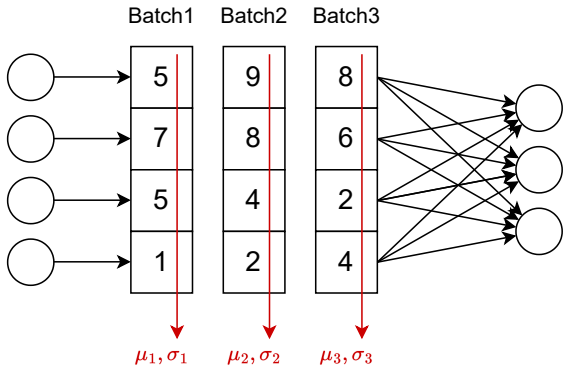


Figure 2.17: Layer Normalization uses statistics from neuron values in a single mini-batch.

Layer normalization is effective at stabilizing the hidden state dynamics in recurrent networks, and can speed up training [22].

Group Normalization
Group normalization is a modified version of Layer normalization. Instead of using all neuron values in a layer, they are split into groups that share the parameters μ, σ, γ and β. Additionally, a hyper-parameter *num_groups* is introduced to describe the number of neurons in a group [19].

2.3 Training Objective and Evaluation Metrics

To train a neural network, it is required to define a loss function or training objective. This function is used as the feedback for the network and quantifies how well the network is performing. Recently, the scale-invariant signal-to-noise-ratio (SI-SNR) is the commonly used training objective for speech separation networks [6, 7]. The SI-SNR is defined by the three equations seen in 2.28, 2.29, and 2.30. The higher this SI-SNR score is, the better the system is performing.

$$s_{target} = \frac{\langle \hat{s}, s \rangle s}{||s||^2} \tag{2.28}$$

$$e_{noise} = \hat{s} - s_{target} \tag{2.29}$$

$$\text{SI-SNR} = 10 \log_{10} \frac{||s_{target}||^2}{||e_{noise}||^2} \tag{2.30}$$

Where:

$\langle \hat{s}, s \rangle$ is the inner product between $\hat{s}$ and s
$\hat{s}$ is the estimated source
s is the original clean source

Besides SI-SNR, signal-to-distortion ratio (SDR) is often used as a metric of performance and reported alongside SI-SNR. The SDR is calculated using equation 2.31 [23].

$$\text{SDR} = 10 \log_{10} \frac{||s||^2}{||s - \hat{s}||^2} \tag{2.31}$$

When SI-SNR and SDR are reported as performance metrics it is often done using the improvement in SI-SNR or SDR compared to the original mix. The improvement is given by equation 2.32 and 2.33 [24].

$$\text{SI-SNRi} = \text{SI-SNR}(\hat{s}, s) - \text{SI-SNR}(x, s) \tag{2.32}$$

$$\text{SDRi} = \text{SDR}(\hat{s}, s) - \text{SDR}(x, s) \tag{2.33}$$

Where:

x is the mixed speech signal

A problem appears if one attempts to train a neural network for single-channel speech separation using only a loss function. This problem is that the ordering or permutation of the output vector containing the estimated masks $\hat{m}_i$, is unknown. This is known as the label permutation problem [25], i.e. assigning a speaker to a corresponding output. The permutation invariant training (PIT) solves this by calculating the loss for all permutations of the output vector and then choosing the one with the lowest loss score [25]. However, the output-speaker assignment might change across frames, so an improvement to PIT was made. This improvement is called utterance-level PIT (uPIT) and attempts to trace speakers across consecutive frames [25]. It does this by calculating the loss score for a whole utterance assuming that all output frames use the same permutation. The permutation with the lowest loss score is then chosen and used for the whole utterance.

The methods, which have been mentioned up until now, are measures of the quality of the estimated source signal. However, there are other metrics on which to judge a specific implementation. These include the number of trainable parameters, memory cost, required giga floating point operations per second (GFLOPs) for each second of computation, and latency. These parameters are of increased importance when the intended application is small battery-powered devices with limited computational power, such as hearing aids. The acceptable latency of a hearing aid is not dependent on technological progress, while the limitation of memory and computational power is a moving boundary due to progress in hearing aid design. It is therefore of interest to investigate what the acceptable latency is in a hearing aid. The acceptable latency has been found to be dependent on the level of hearing loss, where a greater hearing loss allows for a longer tolerable latency [26]. Quantitatively, results from Goehring et al. indicate that a delay of up to 20 ms is not expected to exceed tolerance limits for the average listener with hearing loss [26].

The aforementioned evaluation metrics are commonly used to evaluate the performance of a separation stage. However, in this project, a speech enhancement system is built. An example of such a speech enhancement system is the MOG algorithm which is evaluated using extended short-time objective intelligibility (ESTOI) and perceptual evaluation of speech quality (PESQ) [4]. ESTOI is the extended version of STOI which is a measure used to quantify the intelligibility or quality of speech signals. The STOI measure operates by comparing two speech signals: a reference signal and a degraded signal. The reference signal typically represents the original, undistorted speech, while the degraded signal represents a version of the speech that has been subjected to some kind of distortion or processing, such as noise, filtering, or compression. The STOI measure calculates the intelligibility of the degraded signal by analyzing the short-term spectrogram of both the reference and degraded signals. STOI produces a value between 0 and 1, where 0 indicates a complete lack of intelligibility and 1 represents perfect intelligibility. Higher STOI scores indicate better speech quality and greater intelligibility [27]. ESTOI incorporates additional features to improve its accuracy and robustness.
PESQ is a measure evaluating the perceived quality of speech signals. It assesses the overall quality of a degraded speech signal compared to a reference signal. PESQ takes into account

various perceptual features of speech, including loudness, temporal masking, and frequency-dependent masking. It models the human auditory system to predict the perceived quality of the degraded speech signal [28]. The PESQ measure has a range from -0.5 to 4.5, where a higher score corresponds to higher speech quality.

2.4 Speech Separation using DNNs: State-of-the-Art

The problem of separating speech into multiple channels given a single channel input can be formulated as the estimation of C sources s_i given the discrete waveform mixture $x(n)$. This is formulated mathematically in equation 2.34, where the sources are linearly mixed to obtain the recorded audio.

$$x(n) = \sum_{i=1}^{C} s_i(n) \tag{2.34}$$

Different methods for estimating s_i using deep neural networks have emerged. They can be classified by the method and the domain of estimation. In the past, the estimation has been done in the time-frequency (T-F) domain by attempting to estimate the magnitude spectrograms of the source using the spectrogram of the input mixture. The input spectrogram is estimated using the short-time Fourier transform (STFT) [29]. The estimation of the source spectrograms can be done either directly or using masking. As the name suggests, direct estimation directly estimates the source spectrograms from the input spectrogram using nonlinear regression, where the clean source spectrograms are used as training data [7]. The other method uses the magnitude spectrogram to estimate a weighting function (mask) for each source, which is then multiplied onto the bins of the input spectrogram to recover the source spectrograms. In both methods, the actual sound signal is recovered using the inverse STFT. Over the recent years, deep neural networks have improved the performance of mask estimation, which was the most commonly used method prior to the introduction of separation in time-domain [7]. An example of an architecture for time-frequency mask estimation can be seen in figure 2.18.

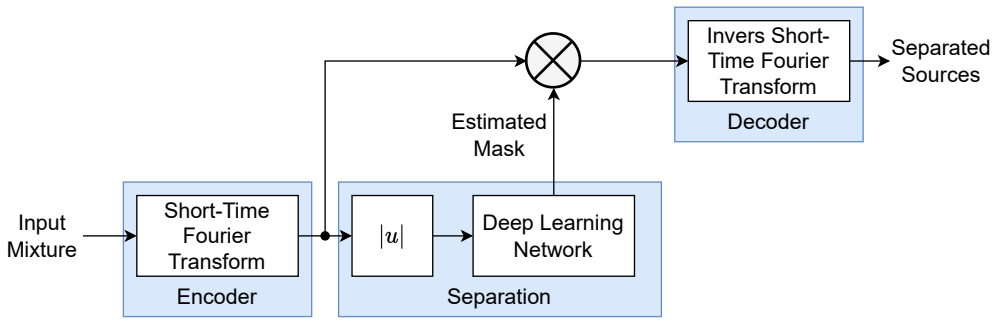


Figure 2.18: Speech separation with T-F mask estimation using deep learning [30].

The latest method for speech separation avoids the use of spectrograms by doing the separation directly in time domain using direct or mask estimation. This approach avoids several limitations of the spectrogram approach and is therefore of great interest. By doing the separation directly

in the time domain the calculation of a spectrogram is avoided and thereby computational cost is reduced. Avoiding the use of a spectrogram also avoids the separation of magnitude and phase, removing the need for phase reconstruction, which has been a limiting factor for the quality of the reconstruction in time-frequency systems [7].
The limitations of time-frequency analysis are formally described by the Gabor limit [31]. This Gabor limit describes the performance trade-off between frequency and time resolution in linear transformations like the STFT. As a result, time-frequency-based speech separation systems based on magnitude spectrograms are constrained in their ability to achieve both low latency and high-frequency resolution [32].
The quality of time-domain approaches has surpassed that of time-frequency masking and is therefore the current state of the art. The following sections will look closer into the methods of time-domain separation.

2.4.1 TasNet

The time-domain masking approach was first introduced in the time-domain audio separation network (TasNet) using a deep autoencoder architecture with a (b)LSTM network [6]. The major contribution was the use of a 1-D convolution to encode the speech signal using a set of learned basis functions and 1-D deconvolution to decode the resulting weighted signal. The combination of an encoder and decoder is called an autoencoder. The autoencoder essentially replaces the STFT and ISTFT of the time-frequency masking approach. The autoencoder architecture has been the basis of nearly all developments since but with replacements for the separation stage. Following the encoder, the mask is estimated using a separation network and multiplied with the encoded signal. The result is then passed to the decoder. This architecture can be seen in figure 2.19. In this approach, a bLSTM was used in the non-causal approach and an LSTM was used in the causal approach.

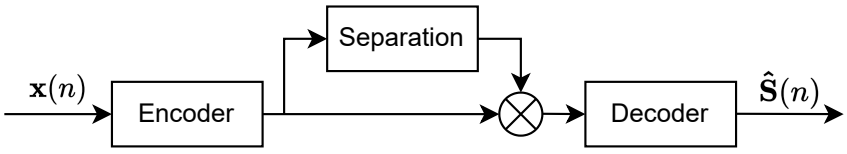


Figure 2.19: The autoencoder architecture used in most state-of-the-art approaches for speech separation.

The original TasNet was then surpassed by an iteration of TasNet where the (b)LSTM block was replaced by a temporal convolutional network (TCN) [7]. The convolutional TasNet (Conv-TasNet) uses a series of stacked 1-D dilated convolutional blocks with skip connections. The architecture allows the network to model long-term dependencies while maintaining a model size of $5.1 \cdot 10^6$ trainable parameters, which is small compared to other approaches. The Conv-TasNet architecture can be seen in figure 2.20. A causal and a non-causal version of Conv-TasNet are presented with the difference being in the normalization method used. For the non-causal approach global layer normalization (gLN) is used while cumulative layer normalization (cLN) is used in the causal approach. Using the causal implementation results in a 4.7 dB reduction

of SI-SNRi compared to the non-causal approach. The causal approach of Conv-TasNet is actually surpassed by the performance of the causal LSTM-TasNet but only by 0.2 dB in SI-SNR. However, the model size of the Conv-Tasnet is $\approx 1/6$ of the LSTM-TasNet. The Conv-TasNet is the latest causal approach. The non-causal approach has been surpassed by newer implementations.

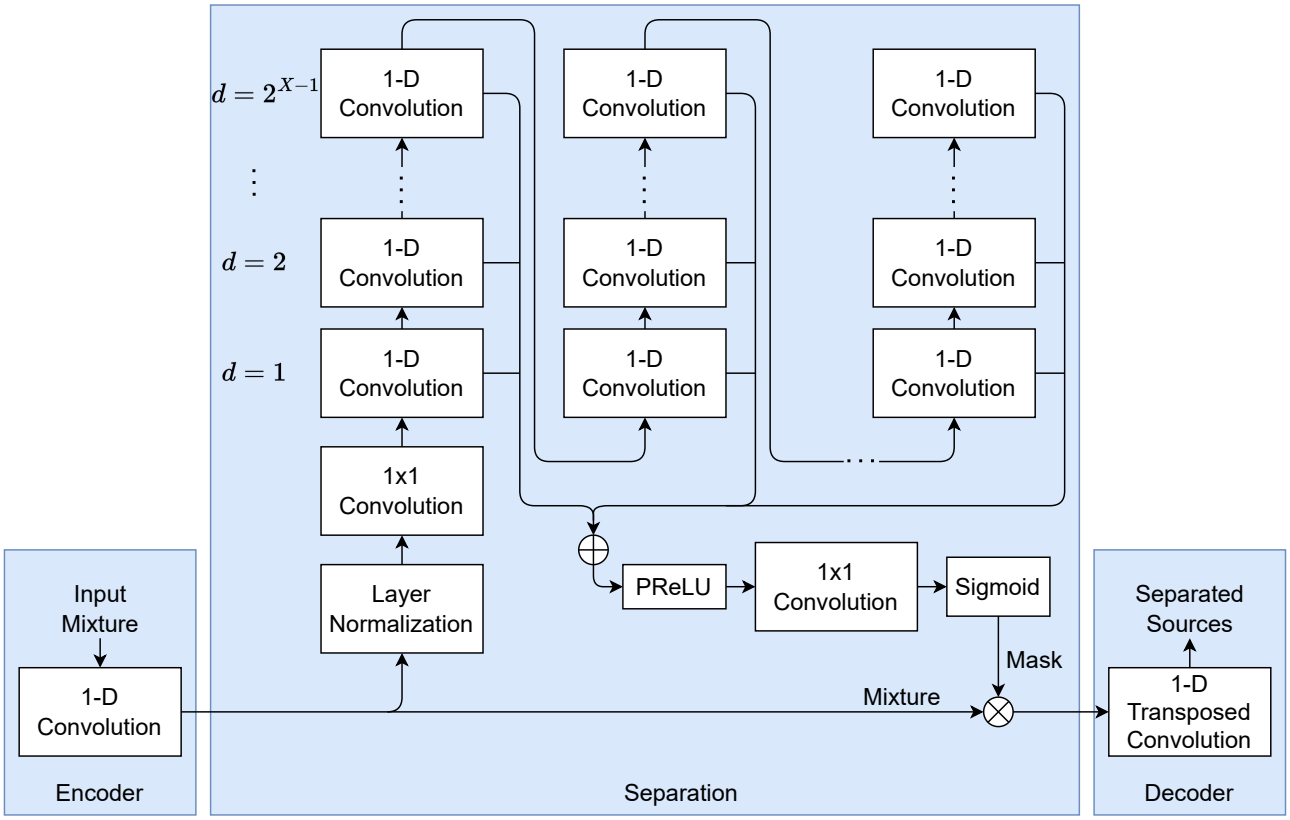


Figure 2.20: The architecture of the convolutional TasNet using stacked 1-D dilated convolutional blocks with skip connections [7].

2.4.2 Dual-Path Recurrent Neural Network

After Conv-TasNet, new approaches swayed from the use of TCNs and returned to the use of RNNs. The reason for this was the introduction of the dual-path RNN (DPRNN) [12]. The DPRNN uses the TasNet autoencoder architecture but uses a dual-path RNN as the separation stage. DPRNN attempts to effectively model extremely long sequences with a dynamic receptive field without the issues regarding optimization as in traditional RNNs. The main idea is the use of two networks, one for handling short-term dependencies and one for long-term dependencies. DPRNN splits the input sequence into smaller chunks with potential overlap. The chunks are a collection of frames. These frames are generated using a certain window size and a certain overlap. The idea is then to perform both intra-chunk and inter-chunk modeling for local and global dependencies, respectively. This modeling is done with two bRNNs. These blocks of dual-path RNNs can be stacked to increase the depth of the network. The result of the final DPRNN block is converted into a sequential output by an overlap-add algorithm on the chunks. The architecture of the DPRNN can be seen in figure 2.21. The idea of using a dual-path

architecture to have distinct networks modeling local and global dependencies proved successful. The implementation in the literature used 6 DPRNN blocks consisting of bRNNs with 128 hidden units in each direction [12]. The best performance was achieved with a frame size of 2 samples, a chunk size of 250 frames, and an overlap between chunks of 50 %. Using this setup it was possible to achieve an SI-SNR of 18.8 dB with a model size of only $2.6 \cdot 10^6$ trainable parameters, which is both a better performance than previous implementations and a smaller model size.

Yi Luo et al. suggest an approach for online processing using the DPRNN network [12]. By using a uni-directional RNN for the inter-chunk modeling, the delay is only dependent on chunk size. In the original implementation, a chunk size of 250 frames was used, where each frame was 2 samples and with a 50 % overlap. They employed a sampling rate of 8 kHz. The first chunk would require 251 samples which results in a delay of 31 ms. The performance of such a network was not tested.

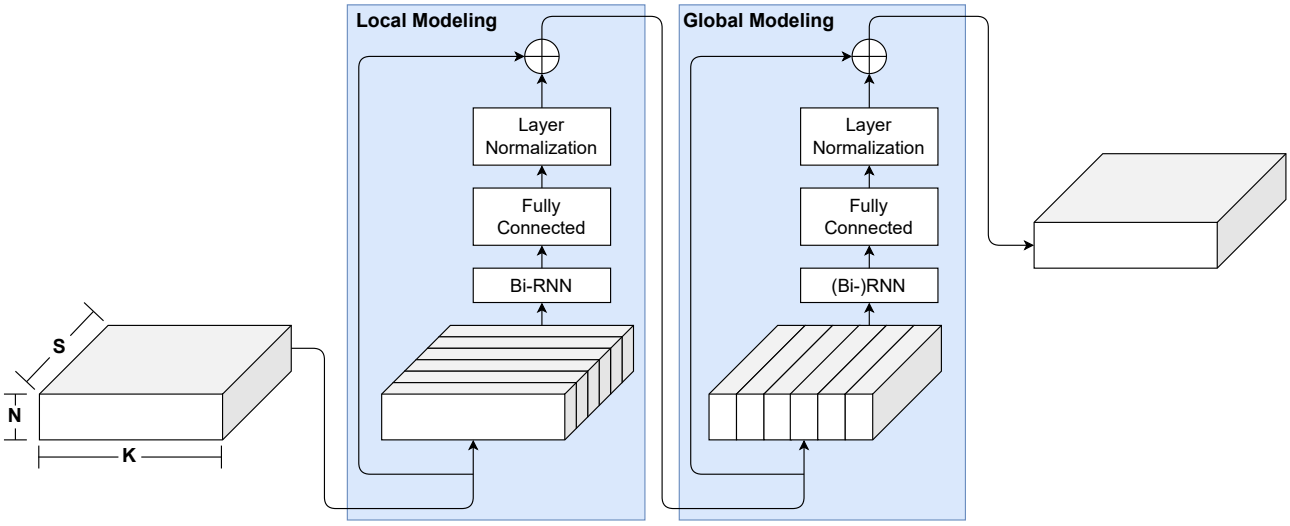


Figure 2.21: The architecture of the dual-path recurrent neural network [12]. N refers to the number of filters in the encoder, K refers to the chunk size, and S is the resulting amount of chunks given the input size.

2.4.3 Dual-Path Transformer Network

Inspired by the success of the dual-path architecture, the dual-path transformer network (DPTNet) was developed [33]. DPTNet is the first DNN for speech separation which introduces direct context-aware modeling using a modified transformer. DPTNet uses the dual-path architecture but replaces the bRNNs with modified transformers. The transformer used in DPTNet is the encoder part of the architecture presented in section 2.2.5 consisting of multi-head attention blocks and a linear feedforward block. The transformer block is modified by replacing the first linear layer with a recurrent layer. This enables the network to utilize the order information in a sequence without the use of positional encoding as in the original transformer [14, 33]. These transformer blocks can be stacked as in the DPRNN. The output of the last inter-chunk transformer is used as the input to a 2-D convolution which learns a mask for each source. The overlap-add method and two 1-D convolutions are used on the chunks to form the

output mask. The complete architecture can be seen in figure 2.22. The transformer-based dual-path network outperforms the first DPRNN based only on RNNs as it achieves an SI-SNR of 20.2 dB which is a 1.4 dB improvement compared to DPRNN. It achieves this performance with a model size of $2.7 \cdot 10^6$ trainable parameters which is an increase of $0.1 \cdot 10^6$ compared to DPRNN.

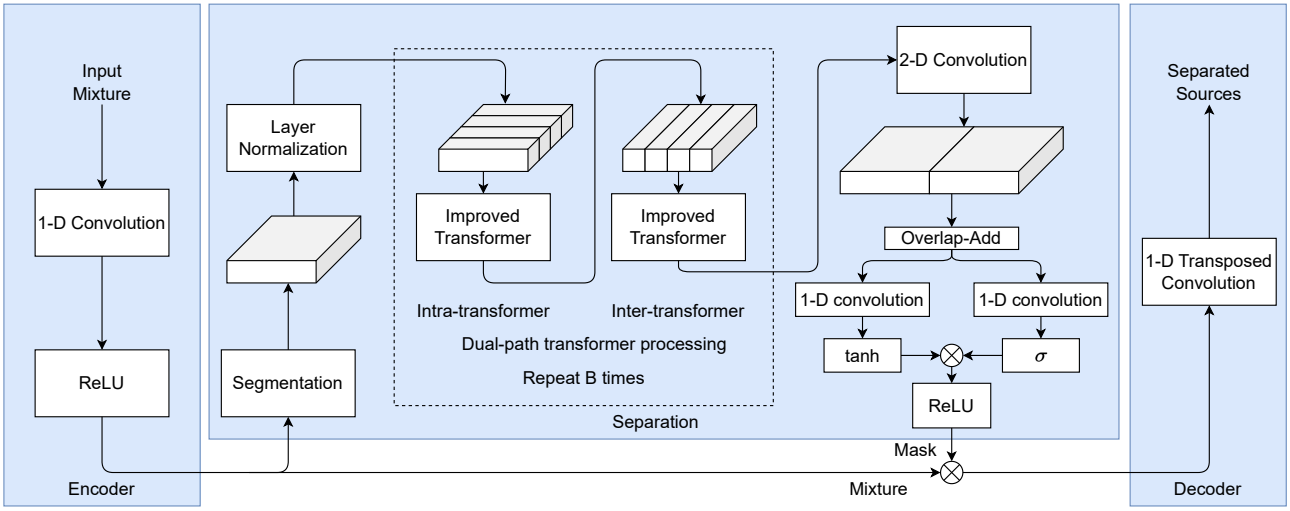


Figure 2.22: The architecture of the dual-path transformer network [33].

2.4.4 Wavesplit

The next contribution deviates from past literature and from the use of the common autoencoder architecture. This new network is called Wavesplit and is based on convolutional layers and deep clustering [34]. It, therefore, makes no use of masking as other methods have done in the time domain. Traditionally, deep clustering has been done in the time-frequency domain, however, Wavesplit operates in the time domain directly. Wavesplit uses two stacks: a speaker stack and a separation stack. The architecture of the stacks is based on the convolutional separation network in [7]. The clustering is done between the two stacks and is based on a k-means approach. The speaker stack produces N latent vectors representing each speaker, where N is the number of speakers. This is used as the input for the clustering algorithm which produces N number of centroids. The centroids and the original speech mix are then used in the separation stack to produce the separated speech channels. The two stacks are trained jointly. The speaker stack is optimized to have small intra-speaker distances and large inter-speaker distances, i.e. optimizing for easy separation using a clustering approach. The separation stage is optimized, similarly to time domain masking approaches, using SDR to minimize reconstruction error. Wavesplit showed state-of-the-art performance on the wall street journal set (WSJ0-2Mix) achieving an SI-SNR of 21.0 dB. The paper also attempted to train the algorithm on separating maternal and fetal heart rates and achieved an SI-SNR of 12.3 dB (Conv-TasNet achieved an SI-SNR of 11.4 dB). This highlights the perspective for the use of these algorithms in other separation tasks. The model size of wavesplit was not reported by the authors, but a later paper gave an estimated size of $42.5 \cdot 10^6$ trainable parameters and another paper reported a size of $29 \cdot 10^6$ trainable parameters [16, 15].

Beyond showing a new approach to speech separation in the time domain, wavesplit also contributed to a possible improvement in the training of speech separation networks. Up until this paper, separation benchmarks like the WSJ0-2Mix was created by summing specific clean mixtures with specific gain, which generates a finite amount of mixtures [34]. Instead, wavesplit proposes the use of dynamic mixing. In dynamic mixing, the speech mixes are created during training time, by sampling random windows of training recordings and summing them with random gain. Similar approaches have been used in music separation. The use of dynamic mixing shows systematic improvements. Wavesplit improved its SI-SNR score by 1.2 dB when using dynamic mixing.

2.4.5 Sandglasset

The next contribution is the sandglasset network [16]. Sandglasset marks a return to the autoencoder and dual-path architecture popularized by TasNet and DPRNN. Sandglasset uses the dual-path structure with a combination of RNN and attention-based layers in each block. Whereas DPTNet used transformers for both the local and global modeling, sandglasset only uses an attention layer for global dependencies and instead uses an RNN layer for local dependencies. The novel contribution of sandglasset is the use of continuous down- and upsampling in each block. The downsampling is performed after the local RNN and the upsampling is performed after the global attention layer. Inspired by the varying granularity of speech (e.g. phenomes, syllables, words), the down- and upsampling is varied from block to block. In the first half, the signal frames are successively downsampled into shorter feature sequences of larger segments in coarser time scales, i.e. high granularity and high-level features. In the last half, the high-level features are upsampled into longer feature sequences of smaller segments in finer time scales, i.e. high granularity and low-level features. The upsampling is done with the inverse granularity of the downsampling, i.e. each downsampling of factor B in the first half has a corresponding upsampling of factor B in the last half. This gives rise to the sandglass shape and is highlighted in figure 2.23 where the architecture of sandglasset is shown. A residual connection is added between blocks of equal granularity to preserve information and avoid vanishing gradient issues. These residual connections are critical for the quality of signal reconstruction [16]. The implementation of sandglasset in [16] uses 6 dual-path blocks and achieves an SI-SNR of 21.0 dB which is identical to wavesplit without dynamic mixing. However, it achieves this performance with a model size of only $2.3 \cdot 10^6$ trainable parameters.

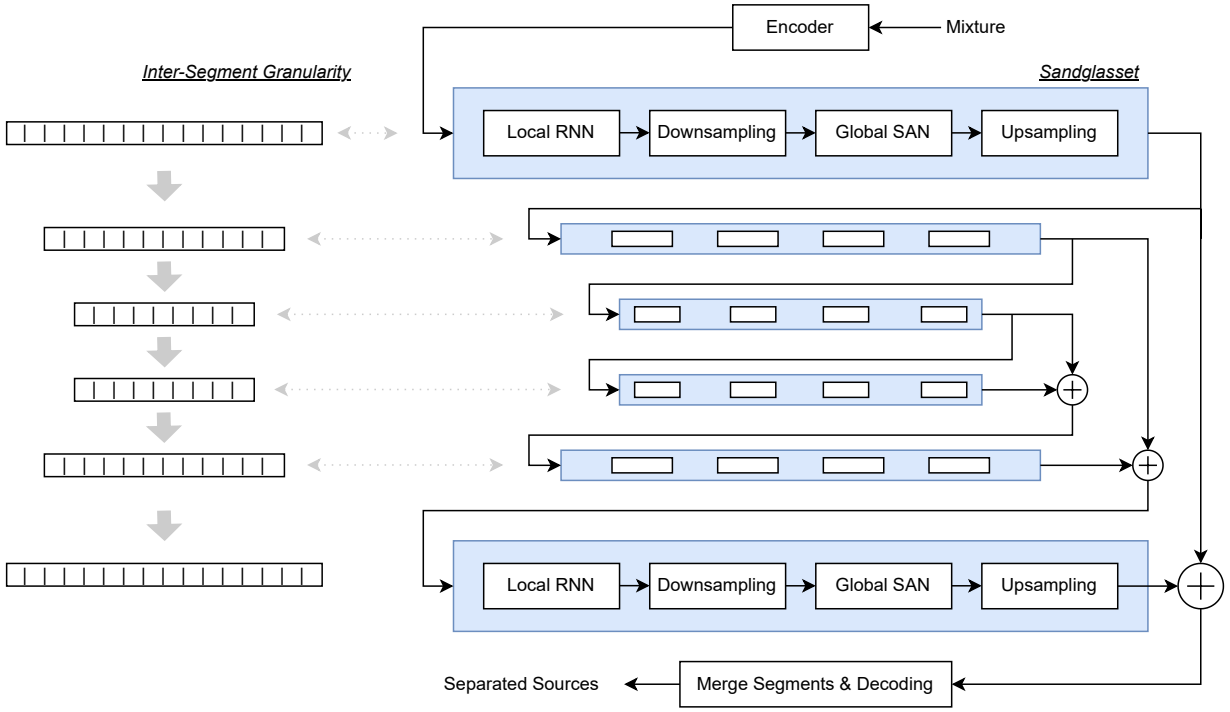


Figure 2.23: The architecture of the sandglasset network [16].

2.4.6 Separation Transformer

DPTNet and Sandglasset have shown the effectiveness of the dual path structure in combination with transformers. However, both of these implementations use RNNs somewhere in their architecture. The separation transformer (SepFormer) avoids the use of RNNs completely and thereby enables the high-level parallelization possible in the transformer structure [15]. SepFormer uses the same dual-path structure as DPTNet or Sandglasset, however, it uses transformer blocks for modeling both local and global dependencies. The dual-path structure mitigates the quadratic computation cost of the transformers as it works on chunks instead of the whole input sequence. To allow for the utilization of order information in the input sequence, SepFormer includes sinusoidal positional encoding as in the original transformer [14]. The architecture can be seen in figure 2.24. Using two dual-path blocks, with 8 parallel multi-head attention blocks, a chunk size of 250 with a 50% overlap between them, and a stride of 8 in the encoder (Essentially downsampling by a factor of 8), SepFormer achieves an SI-SNR score of 20.4 dB with a model size of $26 \cdot 10^6$ trainable parameters. This is not a new state-of-the-art score and is performance comparable to that of the DPTNet, but it is achieved with a downsampling factor of 8. However, when dynamic mixing is utilized, SepFormer achieves an SI-SNR score of 23.3 dB which is 0.1 dB better than wavesplit and is, therefore, a new state-of-the-art performance. However, due to possible parallelization, the forward propagation and training time are improved compared to DPRNN and DPTNet. The memory usage of the SepFormer is also lower than that of the DPRNN and DPTNet. The comparison of these parameters between SepFormer, DPRNN, DPTNet, and Wavesplit can be seen in the paper by Cem Subakan et al [15]. While the performance increase is limited, SepFormer highlights some of the advantages of transformers compared to the traditional RNN approach.

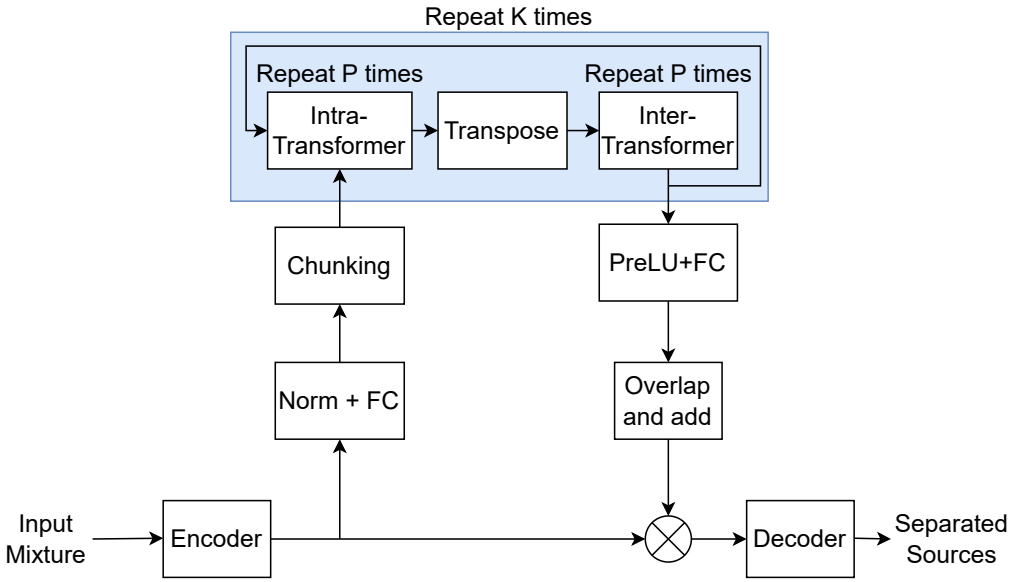


Figure 2.24: The architecture of the Sepformer network [15].

2.4.7 Performance Summary of Separation Networks

The performance and metrics of the aforementioned DNNs are summarized in table 2.1 using model size, SI-SNRi, and SDRi. The methods are split into non-causal and causal.

Table 2.1: Model size, SI-SDR, and SDR improvements (dB) for different methods validated on the WSJ0-2Mix dataset using 2 speakers. These are ordered chronologically in their respective categories. * estimated by [16]. ** reported by [15].

Method	Model size (10^6)	SI-SNRi (dB)	SDRi (dB)
Non-Causal Methods			
BLSTM-TasNet	23.6	13.2	13.6
Conv-TasNet-gLN	5.1	15.3	15.6
DPRNN	2.6	18.8	19.0
DPTNet	2.7	20.2	20.6
Wavesplit	42.5*/29**	21.0	21.2
Wavesplit + DM	42.5*/29**	22.2	22.3
Sandglass	**2.3**	21.0	21.2
SepFormer	26	20.4	20.5
SepFormer + DM	26	22.3	22.4
SFSRNet	59	22.0	22.1
SFSRNet + DM	59	**24.0**	**24.1**
Causal Methods			
LSTM-TasNet	32.0	**10.8**	**11.2**
Conv-TasNet-cLN	**5.1**	10.6	11.0

The aforementioned DNNs are described in an attempt to build a foundation for understanding state-of-the-art in speech separation research. The contributions mentioned are evaluated to be the major established steps in speech separation research. The exception to this is SFSRNet which has not been seen mentioned in other articles reviewed by the authors. It is mentioned anyway as it highlights an interesting approach, where the speech separation is done on a downsampled input and then reconstructed using the estimated sources and the original input. This approach is interesting as it can be used to lower the computational complexity of a network without significant performance drops, which is relevant for causal real-time implementations. More recent research into speech separation exists as it is an area of major activity. However, the ramifications and importance of these works are yet to be established. It is not within the scope of the project to review this research and therefore only the more thoroughly established steps are mentioned as part of the state-of-the-art.

2.5 Scope of Project

In this project, the goal is to develop a speech enhancement system capable of enhancing the target speaker from the user's point-of-view, i.e. a potential solution for the cocktail party problem. All other speakers will therefore be seen as noise, which is to be attenuated. This system will use the MOG algorithm as a core component. The contribution of the MOG paper was focused on the ranking and enhancement that MOG could provide. A solution to speech separation was presented using beamforming. However, potential designs using a better-performing separation system for such a speech enhancement system remain largely unexplored. This area will be explored by combing the MOG algorithm with a state-of-the-art speech separation system using deep neural networks.

As this project will focus on constructing a proof-of-concept for a speech enhancement system using state-the-art speech separation, several assumptions and limitations are made to simplify the system. These assumptions cover things that are deemed nonessential for showing such a proof-of-concept. First, it is assumed that the user is only in a conversation with a single target. Next, it is assumed that the user's speech signal can be identified by other systems, e.g. one can imagine a system parallel to the speech separation stage, which uses beamforming to single out the user's voice. The speech signal recovered this way, will then be used as input to the ranking stage and as the basis (reference) for comparison in the MOG algorithm. The separated channels, from the separation stage, will then be used as the candidate speakers. One of these candidate speakers will contain the user's own voice. This signal will be ranked as the least likely target speaker by the MOG algorithm as it will have a high amount of overlaps and gaps. In development, the reference voice for the ranking stage will be supplied by the user's clean signal before mixing. This system architecture is seen in figure 2.25.

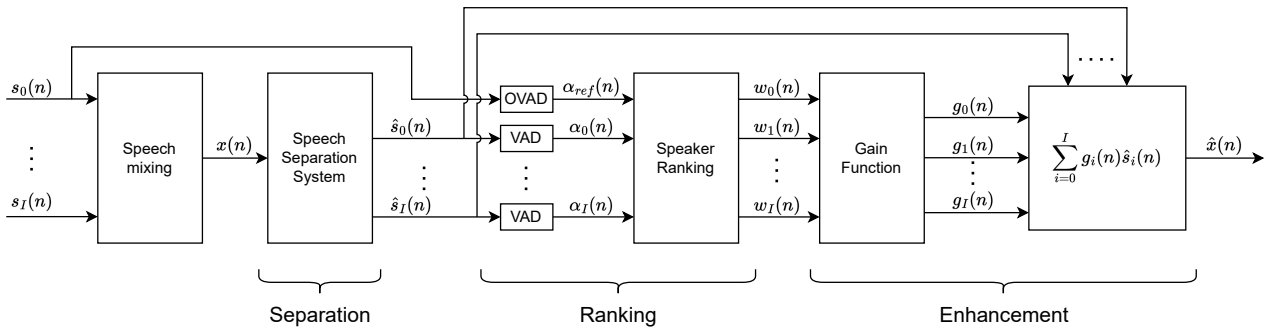


Figure 2.25: Shows the modified system diagram with a skip-connection for the user's voice, which is used as the reference signal for the ranking stage.

A limitation is put on the number of different scenarios tested. Three scenarios will be evaluated in this project, which is a 2-, 3-, and 4-speaker scenario. The 2-speaker scenario will not contain the user's own voice while the others will. This choice is made such that a 2-speaker scenario can contain a competing speaker and a target speaker. 2- and 3-speaker scenarios are well explored in the literature using a standard dataset and therefore provide a basis for comparison. The performance of state-of-the-art neural networks for speech separation is largely unexplored for more speakers than three. However, a 4-speaker scenario is added as this can contain two separate one-on-one conversations, which provides a realistic use-case for the system.
Another limitation is the number of speech separation models investigated. Conv-TasNet and DPRNN are the models which will be investigated in this project. These models are selected as Conv-TasNet has a causal implementation and DPRNN has a suggested causal implementation. This functionality is of interest as documenting the performance of these causal systems will provide insight into the feasibility of such a speech enhancement system in practical products. From this point-of-view other attributes of these models are also worth highlighting. Conv-TasNet showed low forward pass time when compared to other models (0.02 ms for a 2 ms frame in Conv-TasNet). While for DPRNN the forward pass time is around 120 ms for a 4 s input segment. The numbers of forward pass time are hard to compare as they are often done on different GPUs and measured in different ways, but these numbers suggest a performance that makes real-time implementations feasible. These two architectures also report some of the smallest model sizes when compared to other networks presented as state-of-the-art. The delay of these two networks is also within the tolerance limits for the average person with hearing loss (See section 2.3). For Conv-TasNet, the delay is 2 ms, while for DPRNN it would be 31 ms given the suggested implementation by the authors. Conv-TasNet is also one of the most represented networks in literature and can therefore act as a benchmark for developed networks.
Concepts that add to the quality of training are also of interest, as these in general add to the performance at no extra cost as they are only used doing training time. These include dynamic mixing and uPIT. Dynamic mixing increases the generalization of the resulting model and uPIT solves the permutation problem of the output channels (The order of the output channels).

2.6 Partial Conclusion

In the previous sections, we looked into the problem of speech enhancement and its subproblems. The problem consists of the prediction of the target speaker, attenuation of competing speakers, and amplification of the target speaker. An architecture for the possible solution to this problem is presented and consists of 3 major components: The separation stage, the ranking stage, and the enhancement stage. A possible solution to the ranking and enhancement problem was presented by Poul et al. while potential designs for the speech separation are largely unexplored [4]. Therefore the development of the separation stage is the focal point. Deep neural networks are the state-of-the-art for the separation of competing speakers in a single-channel setup. To lay a foundation for understanding deep neural networks and by extension, the state-of-the-art, the basic building blocks and functionality of neural networks are described. Using this foundation, the state-of-the-art in speech separation is presented with a focus on the established research. Several networks of interest have been identified. These are Conv-TasNet and DPRNN, Sandglasset. Two relevant concepts for the improvement of training have been identified, which are dynamic mixing and uPIT.

The following chapters will describe the datasets which will be used for training the neural networks and performance evaluation. Following this, the development of the speech separation stage based on Conv-TasNet and DPRNN is described. Following this, the MOG algorithm is implemented to perform speaker ranking and enhancement. The complete system is then made and evaluated, using both the non-causal and causal speech separation stage, in three scenarios. These are a 2-, 3-, and 4-speaker acoustic scene.

3 Speech Data sets for Training and Evaluation

In this section, the datasets used for training and evaluation are described. The two datasets used are an utterance-based dataset and a conversation-based dataset [35]. The utterance dataset is the widely used WSJ0 set [35]. The conversation dataset used is the "Task dialog by native-Danish talkers in Danish and English in both quiet and noise" (TDNDT) collected by Anna Sørensen et al. and a synthetic conversation based on the WSJ0 set [36]. For the WSJ0 predefined mixes of the dataset is available for 2-speaker and 3-speaker configurations, which are widely used as benchmarks in the literature. However, for testing on a 4-speaker configuration, a new mix has to be made and therefore a dynamic mix algorithm is developed in section 3.3 based on the dynamic mix used in wavesplit [34]. But first, a detailed description of the two datasets is given in the following sections.

3.1 Utterance Dataset

The WSJ0 dataset will be used as the utterance dataset. This is a dataset widely used in literature to train and evaluate speech separation networks. This dataset contains a large set of people reading short sections from the Wall Street Journal news. These are of varying length between 2 s and 44 s. Two predefined mixes are widely used. These are a 2-speaker mix and a 3-speaker mix generated using scripts made by John R. Hershey et al. [37]. These scripts generate training data, validation data, and test data, which consists of 20000, 5000, and 3000 clips of mixed speech, respectively. The training set and validation set are generated from a set of 12776 single utterance clips from 101 unique speakers. The test set is generated from a set of 1857 single utterance clips from 22 unique speakers. The test set speakers are not present in the training and validation clips, thereby making them unseen to a network under test. The settings are the same for the 2-speaker and 3-speaker sets, i.e. same size and subset are used. Each utterance is amplified/attenuated using a gain between -5 dB and 5 dB before they are mixed. For a 4-speaker configuration, a custom dataset has to be made as no standard predefined mix is available. Two versions of a 4-speaker mix are made. The first is a simple approach, that utilizes the existing 2-speaker mix, by mixing two random 2-speaker mixes to form a 4-speaker mix. The advantage is the simplicity of generating this mix, however, it does half the size of the dataset. A second approach is therefore developed, which is based on dynamic mixing [34]. This is described in detail in section 3.3.

3.2 Conversation Dataset

Two conversation sets are used. One is based on a real conversation between two people from the TDNDT dataset, while one is a synthetically generated set based on the WSJ0 set. The TDNDT set contains 19 pairs of unique talkers. For each recording, the pairs are seated in separate booths and are supplied with a picture, and by talking the pair has to find 10 differences between their

two pictures. The pairs are made up of man-man, woman-woman, and woman-man with 11, 4, and 4 unique pairs for each group respectively. Each talker wore a Shure WH20 microphone along with a pair of Sennheiser HD650 headphones. 12 recordings are available for each pair except for pair 12, which has 11 recordings, for a total of 227 recordings. These recordings are in 4 categories. 3 recordings in Danish, 3 recordings in Danish with background babbel noise, 3 recordings in English, and 3 recordings in English with background babbel noise. For each recording, 4 channels are provided. Channel 1 contains talker 1, channel 2 contains talker 2, channel 3 contains the operator, and channel 4 contains the mix of the three previous channels and the noise if used. Only the first two channels are used to generate datasets.
A 4-speaker test set is generated by combining two different pairs. A 3-speaker test set is generated by combining a pair with a single person from a different pair.
A synthetic conversation data set has been made using the WSJ0 training dataset. This is done in a turn-taking manner, where audio from two random speakers is cut and mixed in such a way that the structure mimics a real conversation. An analysis of the turn length, pause and overlap length between two conversational partners from the TDNDT dataset is used to generate such a synthetic dataset.

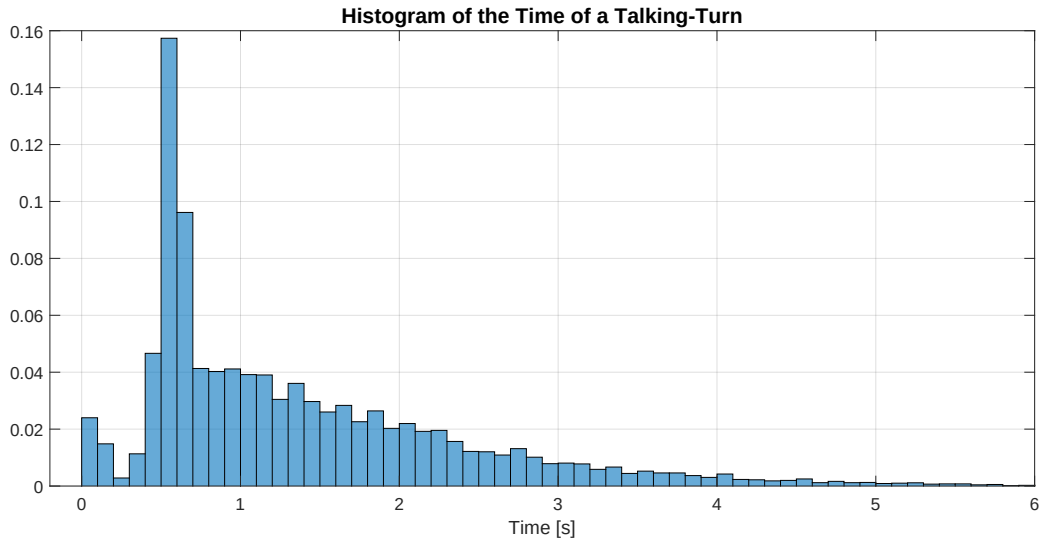


Figure 3.1: Histogram of the turn length.

It can be seen in figure 3.1 that there are a high amount of "Talking-Turns" with a length of 0.5 s to 0.6 s. This is happening because of the structure of the task the speaker is presented with. A lot of the responses from one conversational partner to the other are just a short verification if a part of the pictures is matching or not.

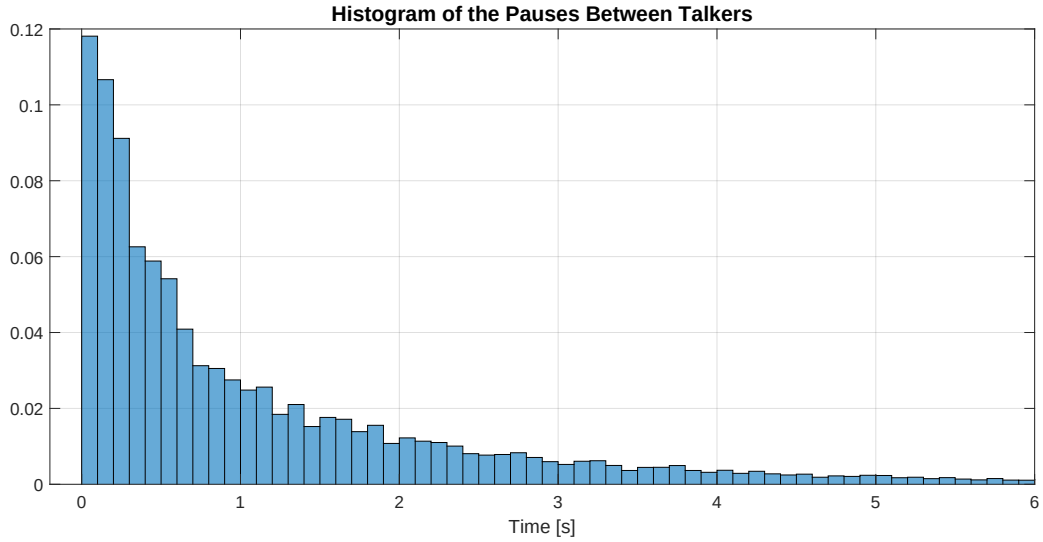


Figure 3.2: Histogram of the pause between talkers.

The pauses between speakers follow an exponential distribution, which makes sense when the conversations are turn-taking, where long silent pauses between turns can be awkward and uncomfortable [5]. This is what can be seen in figure 3.2

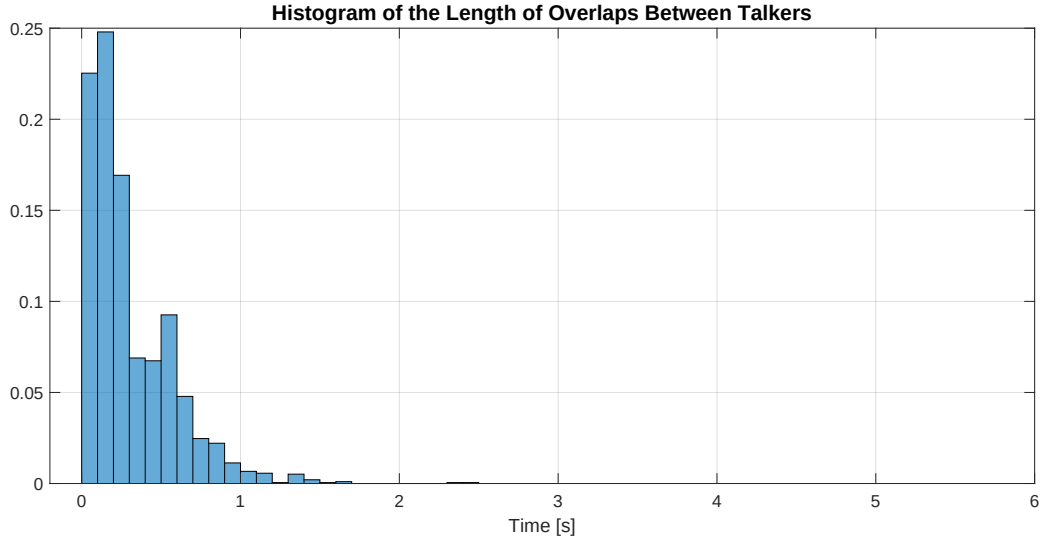


Figure 3.3: Histogram of the overlaps between conversational partners.

In figure 3.3 the distribution of the overlaps between two conversational partners can be seen. The are two types of overlaps in this dataset. The first type of overlap is happening when a speaker is answering just before the conversational partner is finished talking. The second type of overlap happens when a speaker is breaking the turn-taking structure by talking while the conversational partner is in the middle of a sentence.
These statistics are used to generate the synthetic WSJ0 conversation dataset with a more realistic structure. The conversations are then mixed in the same way as the TDNDT dataset to construct 3-speaker and 4-speaker mixes.

3.3 Dynamic Mixing

As mentioned in the wavesplit section 2.4.4, dynamic mixing can increase the performance of the separation network. By modifying the audio clips and mixing them at random during runtime, the amount of training data can be expanded. The idea is that this will better generalize the model as it has to function on a larger variety of inputs.

Multiple techniques are used to create the mixes. For each training epoch, a new mix is generated. Each audio clip is boosted or attenuated by a uniformly distributed random value between $\pm 5\,\mathrm{dB}$, to emulate people speaking louder or from further away. This is identical to the mixing performed in the creation of the WSJ02-mix. The next step is to speed up/down the clips between $\pm 5\,\%$ of the original speed. This is done by resampling the clips and, playing them at the original sampling frequency. In addition to changing the speed the speaker's pitch is also adjusted.

Sandglasset mentions the use of mixtures with the same speaker but different utterances, meaning that a single speaker's own voice can overlap with itself [16]. They observed that the network was bad at distinguishing between people with similar voice timbre. The mixing of same-speaker utterances will be referred to as self-mix. When the training of the sandglasset network converged, a new training set is fed to the network. This dataset included $50\,\%$ self-mixed mixtures. Self-mixing has been included in dynamic mixing. Here it is possible to choose the ratio of self-mixed in the generated dataset. The self-mix operation is done by manipulating a pointer matrix containing unique indices. These indices point to a unique 4-second utterance from the available data. The indices are reshaped into a pointer matrix with a size of $(N_{columns}/channels) \times channels$. Each entry represents a unique 4-second utterance, while each row constitutes the utterances used to generate a mix. Figure 3.4 shows the pointer matrix, where c indicates the number of utterances in a mix and b is the total amount of mixes generated.

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} & \dots & a_{0,c} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,c} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,c} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{b,0} & a_{b,1} & a_{b,2} & a_{b,3} & \dots & a_{b,c} \end{bmatrix}$$

Figure 3.4: Pointer matrix containing indices of the columns in the utterance matrix. The pointer matrix has shape $(num\_files/speaker\_channels) \times channels$.

When the pointer matrix is constructed, it is filled row by row. This results in all the utterances from speaker 1 being slotted in row by row. So without shuffling, the mixes would mostly contain the same speaker. However, it is also possible to exploit this fact, such that a part of the dataset contains selfmixes. To accomplish this, the columns get locked together in pairs thus forming a matrix of shape $channels/2 \times (num\_files/channels) \times 2$, as shown in figure 3.5. Permutations are made in the two first dimensions, to shuffle the combinations of pairs.

$$\begin{bmatrix} \begin{bmatrix} a_{0,0} & a_{0,1} \end{bmatrix} & \begin{bmatrix} a_{0,2} & a_{0,3} \end{bmatrix} & \dots & \begin{bmatrix} a_{0,c-1} & a_{0,c} \end{bmatrix} \\ \begin{bmatrix} a_{1,0} & a_{1,1} \end{bmatrix} & \begin{bmatrix} a_{1,2} & a_{1,3} \end{bmatrix} & \dots & \begin{bmatrix} a_{1,c-1} & a_{1,c} \end{bmatrix} \\ \begin{bmatrix} a_{2,0} & a_{2,1} \end{bmatrix} & \begin{bmatrix} a_{2,2} & a_{2,3} \end{bmatrix} & \dots & \begin{bmatrix} a_{2,c-1} & a_{2,c} \end{bmatrix} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \begin{bmatrix} a_{b,0} & a_{b,1} \end{bmatrix} & \begin{bmatrix} a_{b,2} & a_{b,3} \end{bmatrix} & \dots & \begin{bmatrix} a_{b,c-1} & a_{b,c} \end{bmatrix} \end{bmatrix}$$

Figure 3.5: The entries get paired across the columns. This is to ensure that two utterance from the same person is mixed together.

Afterward, some of the pairs are *"locked"* so they don't change. The number of locked pairs is chosen based on the amount of selfmix. So if 50 % selfmix is wanted, 50 % of the rows are locked. For the columns, it is always the first one, as seen in figure 3.6. The remaining entries are shuffled around at random, and the locked pairs are added back. The indices in this matrix are now used to select the corresponding unique utterance and combine these to create mixes.

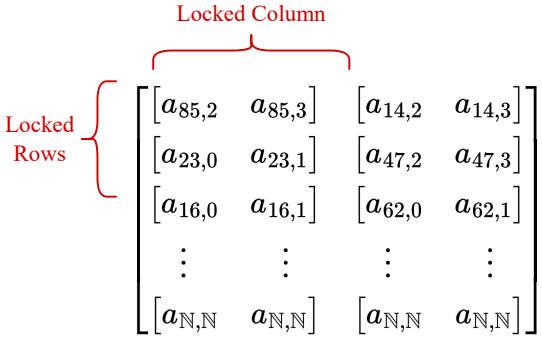


Figure 3.6: Example of a four-speaker mix. The first column is always locked. The locked rows are chosen based on the amount of self-mix wanted.

To further increase the robustness of the model, noise is added to the mix. It can be either Gaussian/white noise or babbel noise. White noise can emulate microphone noise, while babbel noise is used to emulate a "cocktail party" environment. Babbel noise is added by randomly choosing a segment with a fitting length from a pre-made mix. The pre-made mix is a 6-speaker mix constructed from training utterances. The volume is lowered according to the wanted SNR. The dynamic mix is done before each epoch to generate a new training mix. This reduces the possibility of overfitting, as training samples are unlikely to repeat. The validation set is created by running dynamic mix on a part of the complete dataset before training. The rest is used for creating training samples during runtime. Dynamic mixing will be referred to as DM in the following sections. The default settings for dynamic mix are no noise and selfmix at 5 %. Unless otherwise specified these settings are used.

4 Implementation of DNN Speech Separation Models

In this chapter, the DNNs for speech separation will be developed, trained, and evaluated using the WSJ0 utterance dataset. Two models are worked on within this section: Convolutional TasNet and DPRNN. The convolutional TasNet has a publically available model implementation supplied by Yi Luo et al. which is used in this project [7]. Both the offline and online convolutional TasNet is evaluated first to ensure that the framework build for training and testing is able to produce results similar to that reported in the literature. This includes data processing, SI-SNR calculations, and uPIT. The complete test includes 2-speaker, 3-speaker, and 4-speaker models with or without dynamic mix. Following this, a development of the DPRNN model is described in section 4.2. The offline model follows the description supplied by Yi Luo et al. but the model is further expanded to include a novel online variation [12]. This is followed by training and evaluation of the offline and online DPRNN. The DPRNN is tested in similar ways to the convolutional TasNet.
The networks using the WSJ mix are trained for 100 epochs, while the models using dynamic mixing are trained for 200 epochs and 400 epochs. All models are trained with uPIT, a gradient clipping of 5 using an L_2-norm, a sampling rate of 8 kHz, and a learning rate of $1.0 \cdot 10^{-3}$. An early stopper is used to prevent overfitting on models not using dynamic mix. This early stopper stops the training if the performance of the network has not improved in 10 consecutive epochs. For models using dynamic mix an early stopper was not used, as the overfitting is limited by the use of dynamic mix. Inputs to the models are normalized using the L_2-norm.

4.1 Experimental tests of Convolutional TasNet

The architecture of the TasNet model is configured to match the best-performing configuration found in the paper by Yi Luo et al. [7]. These are 512 encoder filters with a length of 16 samples, 128 channels in the bottleneck layer, 512 channels in each convolutional layer, 128 channels in skip connections, a kernel size of 3 in all convolutional layers, 8 convolutional blocks in each stack with 3 stacks in total. The normalization method is varied between gLN and cLN for a non-causal implementation and a causal implementation, respectively.
All of the test scenarios are summarized in table 4.1.

Table 4.1: Shows the scores in SI-SNRi and parameters of different variations of TasNet trained on the WSJ dataset. A different batch size stems from the use of different GPUs.

TasNet	Val Score	Test Score	Epochs	Causal	DM	Model Size	Batch Size
2-spk TasNet	15.8 dB	15.1 dB	100	×	×	$5.1 \cdot 10^6$	6
	11.4 dB	10.1 dB	100	✓	×	$5.1 \cdot 10^6$	6
	12.2 dB	10.6 dB	200	×	✓	$5.1 \cdot 10^6$	6
	12.2 dB	11.1 dB	200	✓	✓	$5.1 \cdot 10^6$	8
3-spk TasNet	12.9 dB	11.6 dB	100	×	×	$5.2 \cdot 10^6$	12
	8.4 dB	7.0 dB	80	✓	×	$5.2 \cdot 10^6$	6
4-spk TasNet	9.3 dB	7.8 dB	100	×	×	$5.2 \cdot 10^6$	6
	6.1 dB	5.0 dB	28	✓	×	$5.2 \cdot 10^6$	6
	10.3 dB	7.3 dB	200	×	✓	$5.2 \cdot 10^6$	6
	8.8 dB	6.6 dB	200	✓	✓	$5.2 \cdot 10^6$	6
	12.11 dB	9.3 dB	400	×	✓	$5.2 \cdot 10^6$	12
	9.0 dB	6.7 dB	400	✓	✓	$5.2 \cdot 10^6$	8

For the 2-speaker non-causal TasNet, a max average SI-SNRi of 15.8 dB on the validation set is achieved. This translates to an average SI-SNRi of 15.1 dB on the test set. Yi Luo et al. report a test SI-SNRi of 15.3 dB. This results in a difference of 0.2 dB.
A small difference is also observed in the causal implementation. It achieves a maximum average SI-SNRi of 11.4 dB on the validation set. This translates to an SI-SNRi of 10.1 dB on the test set. This is a difference of 0.5 dB compared to the test results reported by Yi Luo et al. [7]. This difference in performance can originate from the batch size used as this metric was not specified by Yi Luo et al. For these results, an Nvidia T4 GPU with a batch size of 6 was used for training. To investigate whether this hypothesis is true, a training loop was conducted on an Nvidia A40 GPU with a batch size of 20. The 2-spk non-causal network achieved an SI-SNRi of 15.0 dB on the test set. These results support the argument, that the difference can, in part, originate from the batch size used. One also has to account for the potential differences in the construction of the framework or dataset.
To further investigate the differences between the implemented network and the literature, an analysis of the basis functions in the encoder and decoder. This analysis was also performed by Yi Luo et al. The encoder and decoder are analyzed by normalizing each filter (basis functions) using the L_∞-norm and then sorting them using unweighted pair group method with arithmetic mean (UPGMA) [7, 38]. The discrete Fourier transform is applied to investigate the frequency content of these filters. The results of performing these operations on the non-causal 2-speaker TasNet can be seen in figure 4.1.

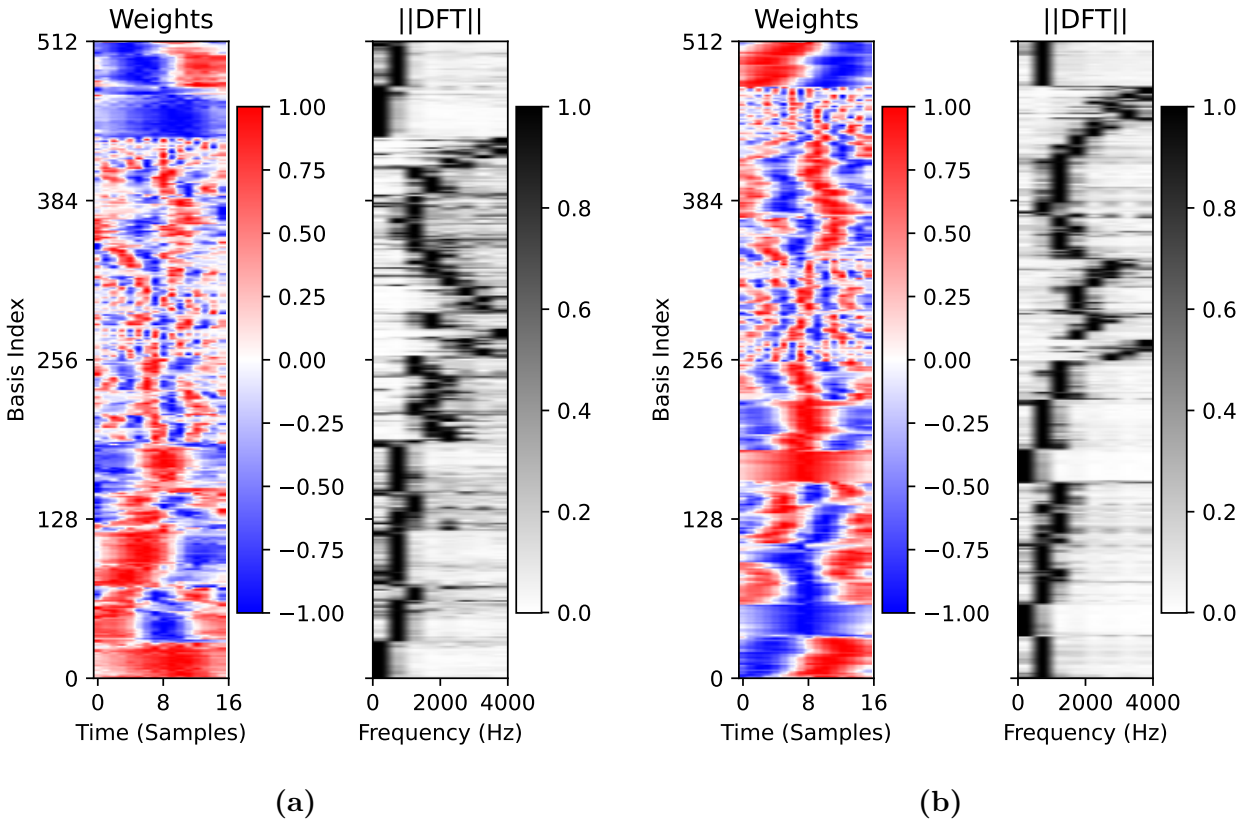


Figure 4.1: Shows the 2-speaker non-causal TasNet encoder's (a) & decoder's (b) normalized weights sorted by their Euclidean similarity using UPGMA and the discrete Fourier transform of the sorted weights [38].

It can be observed from these figures, that the frequency response of most filters resembles band-pass filters with low center frequencies. It can also be observed that a number of filters have similar magnitude responses, but have different phase responses. This can be seen in the time-domain representation of the filters, where these filters have similar shapes but are circular time-shifted versions of each other. The overall structure of the weights is quite different from that reported by Yi Luo et al. but the overall conclusion is similar.
This analysis in combination with the small performance difference, highlights that there are some small differences between the framework developed for training models for speech separation and the framework used by Yi Luo et al.
The 2-speaker noncausal model is the best-performing model overall for TasNet, thus it has been used to visualize the audio waveforms of an example speech separation. A mix with two 4-second utterances has been put through the separation network. Figure 4.2 shows the original utterances, the mix, and the separated channels, of the example mix.

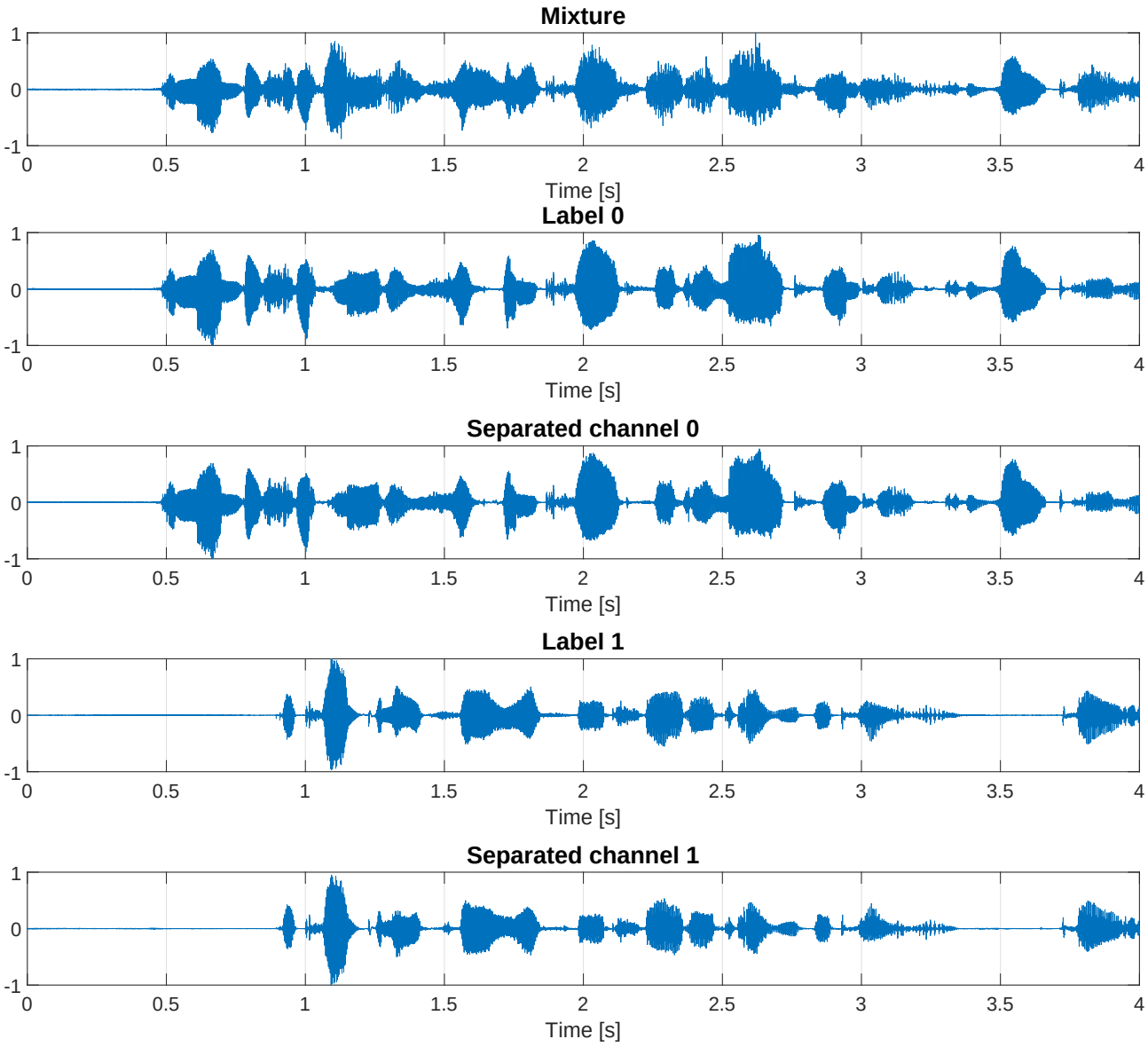


Figure 4.2: An example speech separation. The original utterances are shown as Label.

In order to evaluate the performance of the developed dynamic mix, the 2-speaker TasNet model has been trained with different dynamic mix settings. The first version was trained with only the option that randomly adjusts the individual speakers' volume, the next was trained with random volume change and resampling of the utterances. Lastly, TasNet was trained with every option where selfmix was set to 5 %. A histogram with all the test utterances SI-SNRi's is shown in figure 4.3. The model with only volume change shows two distinct bumps, with a lot of the utterance SI-SNRi being very low or even negative. When adding resampling to dynamic mix the best-performing utterances have a lower SI-SNRi, but the variance is much smaller, and thus the model becomes more generalized. Adding selfmix and gaussian noise to the mix does not seem to produce a noticeable performance increase.

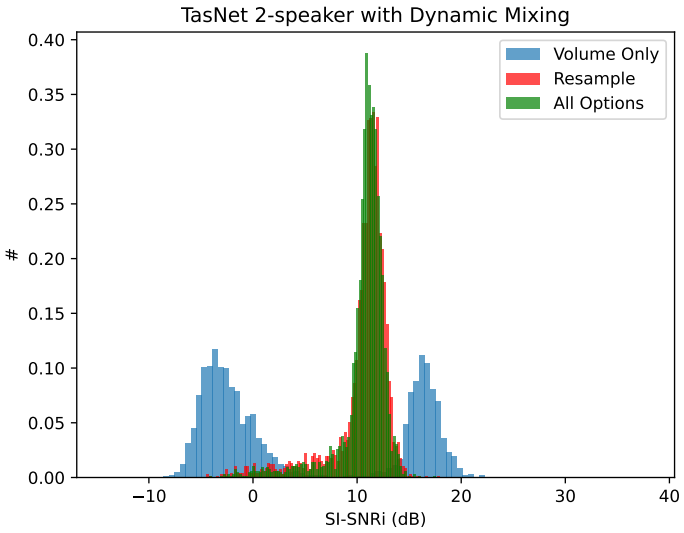


Figure 4.3: Test scores for 3 different versions of dynamic mix used to train TasNet.

Dynamic mix has been used to train the 2-speaker TasNet model. The non-causal version achieved a maximum average SI-SNRi of 12.2 dB on the validation set and an SI-SNRi of 10.6 dB on the test set. The causal version achieved a maximum average SI-SNRi of 12.2 dB on the validation set, while it achieved a SI-SNRi of 11.1 dB on the test set.
The effects of the dynamic mix have also been investigated on the 4-speaker TasNet model. Figure 4.4 shows a histogram of all SI-SNRi scores for the test utterances, in the 4-speaker non-causal model both with and without dynamic mix. The figure shows that dynamic mix reduces the SI-SNRi variance, and thus the model should be more robust. This matches the expected result. The same test has been done for 4-speaker causal TasNet, as shown in figure 4.5. The histogram shows that dynamic mix enhances the performance of the model.

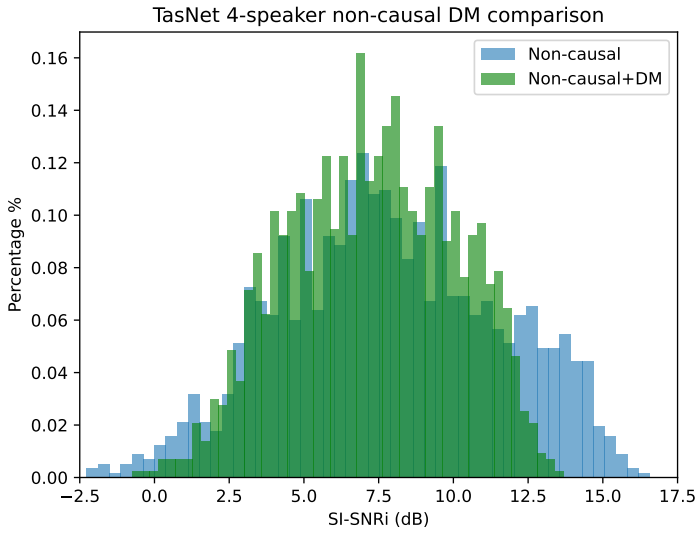


Figure 4.4: Histogram over the performance of non-causal TasNet, with and without dynamic mix.

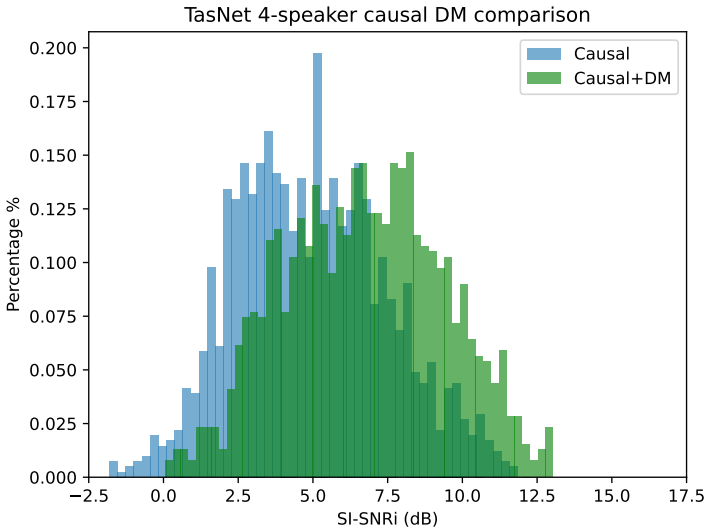


Figure 4.5: Histogram over the performance of causal TasNet, with and without dynamic mix.

It was also attempted to train the networks using the option of including gaussian noise, however, this resulted in the network low pass filtering the output. This was seen as undesired as it does not remove the noise and impacts the resulting separated channels negatively. Further investigation into the addition of noise to the input using dynamic was not conducted.
Overall dynamic mix shows some increase in robustness, but the average SI-SNRi suffers from it. The Sepformer and Wavesplit papers report a performance increase of 2 dB and 1 dB respectively [15, 34]. This is not the case for most of the test utterances as shown in table 4.1, except for 2-speaker causal with dynamic mix. However, looking at the validation SI-SNRi a performance increase is present.

4.2 Development of DPRNN

An alternative separation network is developed to investigate the effect of the speech separation performance on the complete system performance. This separation network is based on the dual-path recurrent neural network (DPRNN). In contrast to the convolutional TasNet, a model for the DPRNN is not supplied by the authors. The model is therefore developed in the following section. This is based on the paper by Yi Luo et al. [12]. The model is expanded to include an online processing version.

4.2.1 Basic DPRNN

The architecture of the DPRNN is fundamentally the same as TasNet. It consists of an autoencoder using 1-D convolutions and a separation network. However, the architecture of the separation network is the differentiating part. The novel idea of DPRNN is the use of a dual path to effectively model short-term and long-term dependencies. To perform this modeling it is required to first split the output of the encoder into chunks using a segmentation scheme. The modeling is done using stacks consisting of an intra-chunk RNN and an inter-chunk RNN. Following the modeling, a mask is then estimated for each speaker. The segmentation is then inverted using an overlap-add scheme. The result of the overlap-add can then be multiplied with the encoder output and decoded to estimate the separated channels. The development can therefore be boiled down to the development of four blocks: segmentation, modeling, mask estimation, and overlap-add.

Segmentation

The function of the segmentation scheme is to split the encoded basis functions into chunks, each consisting of a set number of basis functions with a given overlap. The encoded basis functions are a sequential 2-D matrix with shape $\mathbf{W} \in \mathbb{R}^{N \times L}$, where N is the number of filters in the encoder. The segmentation transforms $\mathbf{W}$ into a 3-D tensor with shape $\mathbf{T} \in \mathbb{R}^{N \times K \times S}$, where K is the chunk size. The first and last chunk is zero-padded to ensure that every sample in $\mathbf{W}$ appears in K/P chunks, where P is the stride. Yi Luo et al. used 50 % overlap in their implementation, which means that $P = K/2$, which results in every sample appearing in 2 chunks. The chunk size and the kernel size of the encoder are the two tuneable hyperparameters, that control the information held in each chunk. The segmentation is illustrated in figure 4.6.

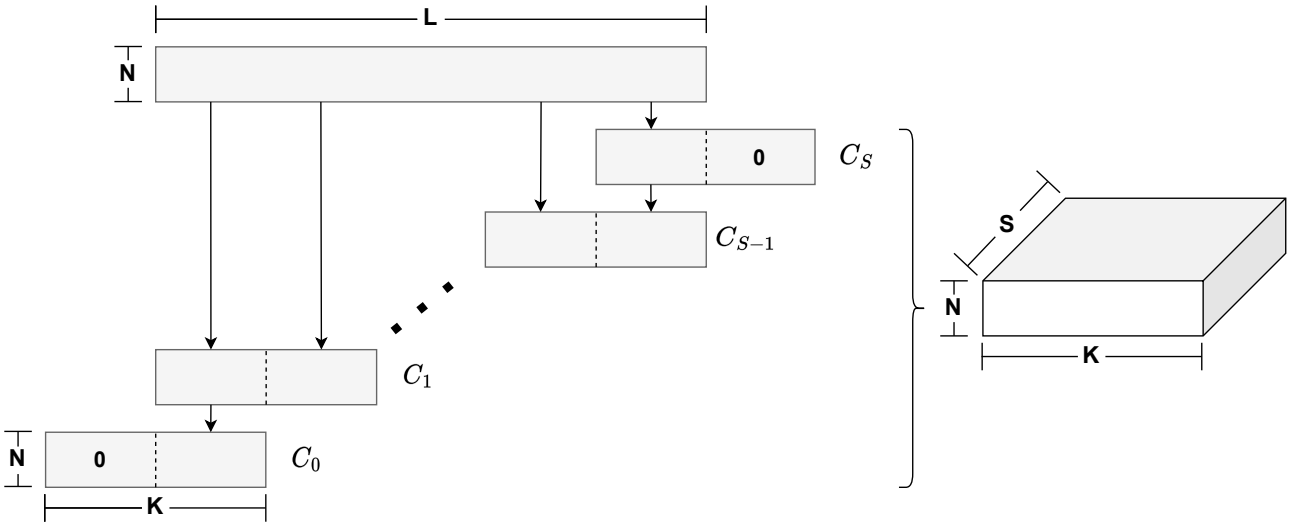


Figure 4.6: Shows the segmentation scheme with an 50 % overlap between chunks. The first and last chunk are zero-padded.

Overlap-add

The focus will now be directed towards the overlap-add algorithm, as this algorithm is used to invert the segmentation [12]. The basic functionality of the overlap-add algorithm is the summation of all samples in the chunks originating from the same sample in **W**. Practically, this is done by first removing the zero-padding and by collecting every pair of chunks into two different tensors. Two distinct tensors are needed as the overlap in the segmentation algorithm is 50 %. The resulting tensors will each contain half of the chunk with each index containing the samples correlated to the original sample. These two tensors can then be added together to form a 2-D matrix with the same shape as **W**, i.e. the original input to the segmentation algorithm. This algorithm is illustrated in figure 4.7

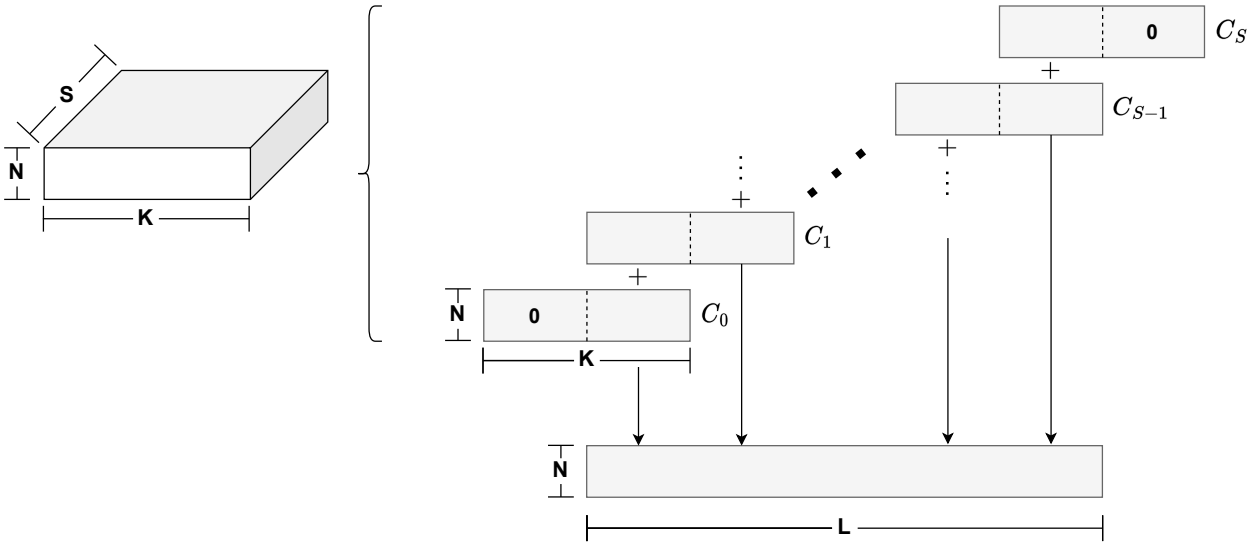


Figure 4.7: Shows the overlap-add algorithm which is used to invert the segmentation. The zero-padding at the start and end are discarded.

Modeling

After segmentation, the resulting tensor $\mathbf{T}$ is ready for modeling and separation. The 3-D tensor $\mathbf{T}$ is used as the input for a stack of dual-path blocks. Each dual path block consists of an intra-chunk and an inter-chunk modeling step. In the DPRNN these processing steps are done using bidirectional LSTMs [12]. The first step is the intra-chunk processing which is applied to the second dimension of $\mathbf{T} \in \mathbb{R}^{N \times K \times S}$, i.e. the processing is applied locally within each chunk. This step is described in equation 4.1.

$$\mathbf{U} = f(\mathbf{T}[:, :, i]), i = 1,...,S \tag{4.1}$$

Where:

$\mathbf{U} \in \mathbb{R}^{H \times K \times S}$	is the output
$f(\cdot)$	is the mapping function of the intra-chunk LSTM
$\mathbf{T}[:, :, i] \in \mathbb{R}^{N \times K}$	is the input sequence from chunk i

The output of the intra-chunk processing is $\mathbf{U} \in \mathbb{R}^{H \times K \times S}$, where H is the number of hidden units in the LSTM. A fully connected layer with bias is used to convert $\mathbf{U}$ into the input shape of $\mathbf{T}$. Layer normalization (LN) is then applied to increase the generalization ability of the network. This layer normalization is described in section 2.2.7, but with the mean and variance calculated using the complete feature space of $\mathbf{T}$, i.e. variance and mean are calculated over N, K, and S. The last step of the intra-chunk processing is a residual connection between the output of the layer normalization and the input. These steps are described in equation 4.2.

$$\hat{\mathbf{T}} = \mathbf{T} + N_l(W \cdot \mathbf{U} + b) \tag{4.2}$$

Where:

$\hat{\mathbf{T}} \in \mathbb{R}^{N \times K \times S}$	is the output of the intra-chunk processing
$\mathbf{T} \in \mathbb{R}^{N \times K \times S}$	is the input of the intra-chunk processing
$N_l(\cdot)$	is the layer normalization function
W	is the weight tensor of the fully connected layer
$\mathbf{U} \in \mathbb{R}^{H \times K \times S}$	is the output of the LSTM processing step
b	is the bias of the fully connected layer

The tensor $\hat{\mathbf{T}}$ is used as the input for the inter-chunk processing step, which follows the intra-chunk processing. The inter-chunk processing step is similar to that of the intra-chunk. The difference is which sequence is used as the input to the LSTM. The inter-chunk processing is applied to the last dimension of $\hat{\mathbf{T}} \in \mathbb{R}^{N \times K \times S}$, i.e. the processing is applied globally across all chunks in distinct time steps. This step is described in equation 4.3.

$$\mathbf{V} = h(\hat{\mathbf{T}}[:, i, :]), i = 1,...,K \tag{4.3}$$

Where:

$\mathbf{V} \in \mathbb{R}^{H \times K \times S}$	is the output
$h(\cdot)$	is the mapping function of the inter-chunk LSTM
$\hat{\mathbf{T}}[:, i, :] \in \mathbb{R}^{N \times S}$	is the input sequence of one time step i from all chunks

As with the intra-chunk processing step, the LSTM is followed by a fully connected layer, layer normalization, and a residual connection. This is shown in equation 4.4.

$$\hat{\mathbf{V}} = \hat{\mathbf{T}} + N_l(W \cdot \mathbf{V} + b) \tag{4.4}$$

Where:

$\hat{\mathbf{V}} \in \mathbb{R}^{N \times K \times S}$	is the output of the inter-chunk processing
$\hat{\mathbf{T}} \in \mathbb{R}^{N \times K \times S}$	is the input of the inter-chunk processing
$N_l(\cdot)$	is the layer normalization function
W	is the weight tensor of the fully connected layer
$\mathbf{V} \in \mathbb{R}^{H \times K \times S}$	is the output of the LSTM processing step
b	is the bias of the fully connected layer

The output of the inter-chunk processing is used as the input for the next dual-path block. B number of dual-path blocks form the stack, which results in the dual-path block processing being applied B times. After the stack, the last output is passed to the mask estimation step. The architecture of a dual-path processing block is shown in figure 4.8.

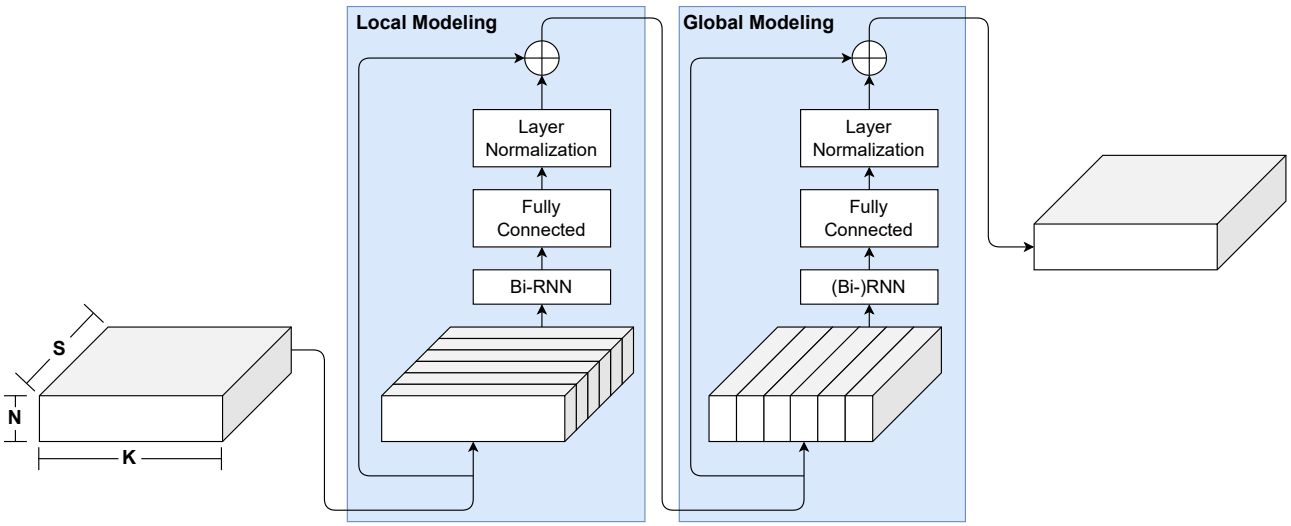


Figure 4.8: Shows the architecture of the dual-path processing block [12].

Mask Estimation
The output of the last dual-path block is used to estimate masks. Yi Luo et al. do not state how this mask estimation is done in DPRNN [12]. We, therefore, choose to base the mask estimation on the DPTNet as this paper explicitly states how the masks are estimated [33]. In DPTNet they use a 2-D convolutional layer to estimate the masks, which is preceded by a PReLU function. This 2-D convolutional layer will take the final output of a dual-path processing block as input and transform that into n number of same-sized masks, where n is the number of speakers. These masks are then used as the input for the overlap-add algorithm before they are multiplied together with the original encoded signal. This separated encoded signal is then decoded using the decoder and the final separated output is achieved. It is worth mentioning that all the masks are transformed using the same decoder, i.e. the decoder is shared and not unique.

4.2.2 DPRNN for Online Processing
The original DPRNN will be modified to allow for an online processing version. This is done to investigate the possible quality of a complete real-time speech enhancement system. Yi Luo et al. suggest a method for making the DPRNN capable of online processing [12]. The suggestion involves changing the inter-chunk from a bidirectional LSTM to a unidirectional LSTM. The minimal latency of the system is then defined by the chunk size K [12]. However, this does not alter the use of the global layer normalization. Without changing this, each output chunk would still depend on other chunks in the past and future. The normalization method will therefore be changed, such that each chunk will be independent of future chunks.
It is decided that the normalization technique will be based on the cumulative layer normalization (cLN) introduced in the convolutional TasNet [7]. The cLN is based on layer normalization but in time steps, i.e. the cLN accumulates statistics from past frames. As more and more frames are processed by the separation system, cLN would approximate global layer normalization. This is an advantage to the other option of simply performing layer normalization using individual chunks because cLN preserves the information between past chunks, which is lost in a chunk-based layer normalization.
In the convolutional TasNet, cLN is used on a 2-D encoded frame, however, in DPRNN the idea is to accumulate mean and variance over chunks, which are 3-D. The equations for the original cLN are therefore expanded to allow for 3-D inputs are seen in 4.5, 4.6, and 4.7.

$$cLN(f_s) = \frac{f_s - E[f_{t\leq s}]}{\sqrt{Var[f_{t\leq s}] + \epsilon}} \odot \gamma + \beta \tag{4.5}$$

$$E[f_{t\leq s}] = \frac{1}{NKs} \sum_{NKs} f_{t\leq s} \tag{4.6}$$

$$Var[f_{t\leq s}] = \frac{1}{NKs} \sum_{NKs} (f_{t\leq s} - E[f_{t\leq s}])^2 \tag{4.7}$$

The use of cLN and unidirectional LSTM in the inter-chunk processing ensures that the DPRNN is causal on the chunk level. This results in the latency being dependent only on the number of samples in a chunk, i.e. chunk size K, and the sampling rate.

4.3 Experimental tests of DPRNN

An architecture for an offline and online DPRNN has now been developed. The hyperparameters are tuned according to the results presented by Yi Luo et al. [12]. Following these results, the number of stacks has been set to 6 (The number of dual path processing blocks), the encoder dimension has been set to 64, and the hidden neurons in the RNNs have been set to 128. The number of speakers, chunk size, and the number of samples in a frame is varied across implementations. Yi Luo et al. found that the best chunk size and the number of samples in a frame are 250 basis functions and 2 samples respectively both with 50 % overlap [12]. The overlap will not be changed. However, the chunk size and number of samples in a frame are used in two flavors: 250 basis functions in a chunk with 2 samples in a frame or 100 basis functions in a chunk with 16 samples in a frame. The latter should perform worse than the first, 2.8 dB measured using SI-SNRi [12]. However, the training time is also increased by a large margin. On an Nvidia A40, an offline DPRNN setup with 100 basis functions and 16 samples completes 100 epochs of training in approximately 14 hours, while an offline DPRNN setup with 250 basis functions and 2 samples completes 100 epochs of training in approximately 4 days and 3 hours. This increase in training time originates from the fact that the setup using a chunk size of 250 and 2 samples per frame produces more chunks on a 4-second segment, than the other setup. This means an increased computation and memory requirement which significantly increases training time. Therefore, the testing is done by first testing the setup using 100 basis functions and 16 samples to ensure correct functionality before the test is expanded to complete a setup with 250 basis functions and 2 samples.
The test results are summarized in table 4.2.

Table 4.2: Shows the scores in SI-SNRi and parameters of different variations of DPRNN. A different batch size stems from the use of different GPUs.

DPRNN	Val Score	Test Score	Epochs	Causal	DM	Model Size	Batch Size
C100-2spk	16.4 dB	16.0 dB	100	×	×	$2.6 \cdot 10^6$	16
	14.9 dB	13.9 dB	100	✓	×	$1.9 \cdot 10^6$	16
C250-2spk	18.4 dB	17.7 dB	82	×	×	$2.6 \cdot 10^6$	6
	15.5 dB	15.1 dB	92	✓	×	$1.9 \cdot 10^6$	6
	17.7 dB	17.2 dB	200	×	✓	$2.6 \cdot 10^6$	6
	14.8 dB	14.0 dB	84	✓	✓	$1.9 \cdot 10^6$	6
C250-3spk	14.5 dB	13.8 dB	100	×	×	$2.6 \cdot 10^6$	6
	14.6 dB	13.5 dB	98	✓	×	$1.9 \cdot 10^6$	6
C250-4spk	11.1 dB	9.4 dB	96	×	×	$2.6 \cdot 10^6$	6
	8.5 dB	6.9 dB	100	✓	×	$2.0 \cdot 10^6$	6
	11.3 dB	9.6 dB	200	×	✓	$2.6 \cdot 10^6$	6
	10.2 dB	7.6 dB	200	✓	✓	$2.0 \cdot 10^6$	6
	13.1 dB	10.1 dB	400	×	✓	$2.6 \cdot 10^6$	6
	11.1 dB	8.3 dB	400	✓	✓	$2.0 \cdot 10^6$	6

The first test of an offline DPRNN with a setup of 100 basis functions in a chunk and 16 samples in a frame is trained for 100 epochs on an Nvidia A10 with a batch size of 16. The trained model achieved a maximum average SI-SNRi of 16.4 dB on the validation set. This translates to an average SI-SNRi of 16.0 dB on the WSJ02 test data. This is exactly in line with the performance reported by Yi Luo et al., where they reported an SI-SNRi of 16.0 dB [12].
As the first setup produces the expected values, the next model is tested with a chunk size of 250 basis functions and a frame size of 2 samples. The model is trained for 100 epochs with a batch size of 6. The training was stopped at epoch 82 by the early stopper due to no increase in the performance during the last 10 epochs. The trained model achieved a maximum average SI-SNRi of 18.4 dB on the validation set. This translates to an average SI-SNRi of 17.7 dB on the WSJ02 test set. This is 1.1 dB lower than the performance of 18.8 dB reported by Yi Luo et al. but is still better performing than the version using a chunk size of 100 basis functions [12]. An example of this speech separation performed by this model is seen in figure 4.9.

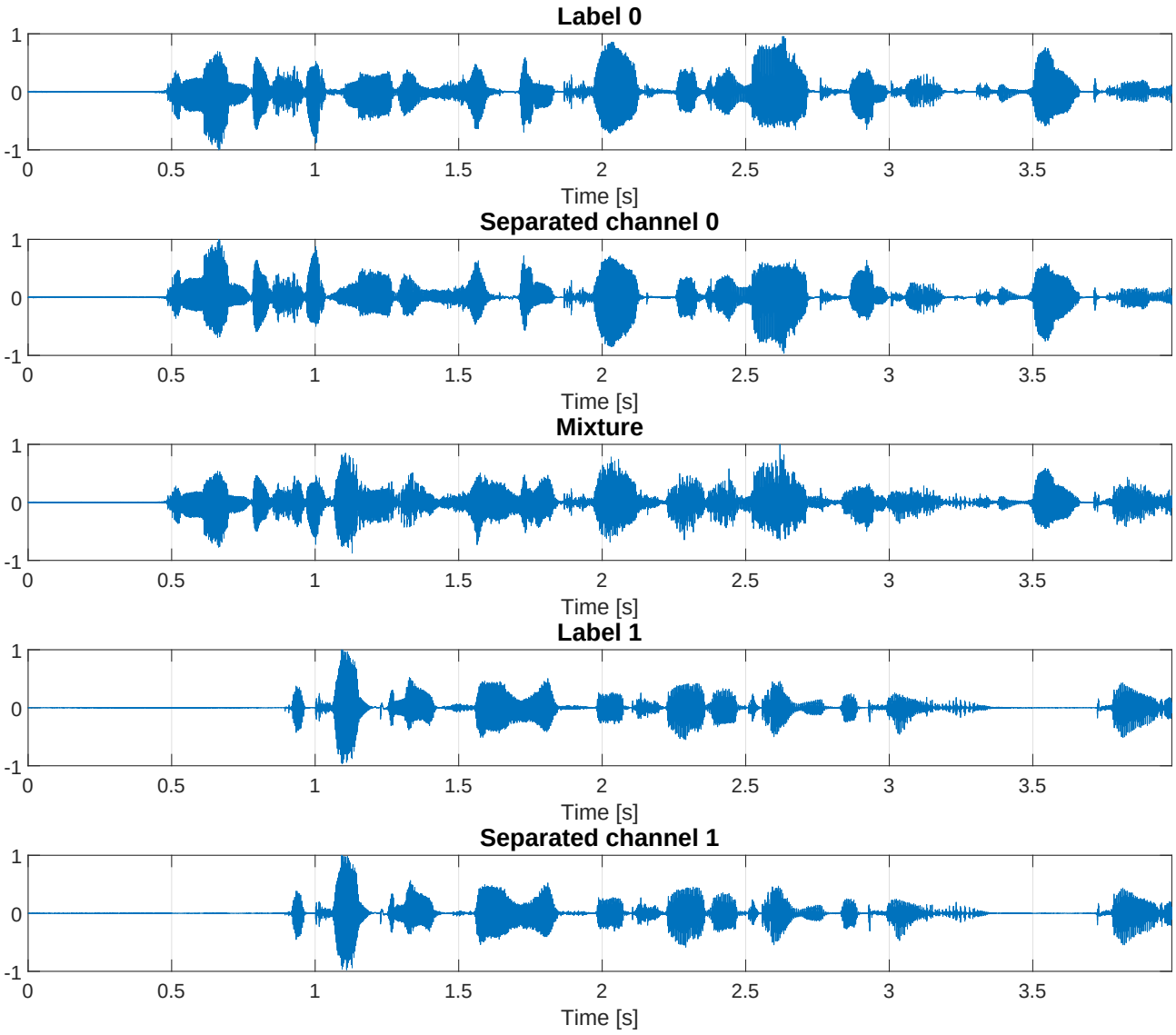


Figure 4.9: Shows an example of the separation performed by the 2-speaker non-causal DPRNN

Beyond the points mentioned in the discussion of the performance discrepancy between the TasNet framework and the literature, there might also be a discrepancy due to small differences in the implementation of the DPRNN architecture. As stated in the development of the DPRNN the method for mask estimation is unknown and the approach from DPTNet was therefore used. Due to this implementation difference, some performance difference is expected.
Two online variants are tested. They are equal to the offline variants tested, only differing on the details described in section 4.2.2. The online DPRNN with a chunk size of 100 achieves a maximum average SI-SNRi of 14.9 dB on the validation set. This translates to an average SI-SNRi of 13.9 dB on the test set. The online DPRNN with a chunk size of 250 achieves a maximum average SI-SNRi of 15.5 dB. This translates to an average SI-SNRi of 15.1 dB on the test set.

The online models are also smaller than their offline counterparts, mainly due to the elimination of a bidirectional LSTM in the inter-chunk modeling step. Their model size is $1.9 \cdot 10^6$ trainable parameters. While there is a performance drop between the offline and online versions, the drop is lower than that of the TasNet implementation. The drop is 2.6 dB for the DPRNN with a chunk size of 250, while for TasNet the drop is 5.0 dB.
For the 4-speaker variants of the DPRNN, the highest performance of 9.6 dB on the test set was reached by the non-causal model trained using dynamic mix. The same model trained using the modified WSJ set achieved an SI-SNRi of 9.4 dB on the test set. Dynamic mixing was also able to improve the performance of the causal model, where it achieved an SI-SNRi of 7.6 dB, which is an improvement of 0.7 dB compared to the model trained on the modified WSJ mix. The model size of the 4-speaker models was also increased slightly, due to the increased size of the mask estimation step.

4.4 Partial Conclusion

In this chapter, two models have been trained and tested for speech development. One is TasNet, which was based on a publicly available code repository, while the other one, DPRNN, was developed. Both an offline and online DPRNN model was developed. The online version has a latency depending only on the chunk size K and the sampling rate. These models were both trained and tested in two speaker variants and four speaker variants. Two training methods were employed, one using the predefined WSJ02-mix and the other using a dynamically generated training set. Small differences in the results were observed when compared to the performance reported in the literature. It was concluded that these originate from different batch sizes and different methods of handling varying segment lengths in the WSJ02-mix. For the DPRNN, differences in the implementation can also be the cause of these differences as not all implementation details were specified by Yi Luo et al. [12]. However, even given these differences, the same general conclusion was reached with respect to best-performing hyperparameters and autoencoder properties.
DPRNN outperformed TasNet in all categories, both causal, non-causal, 2-speaker, 3-speaker, and 4-speaker setups. This is believed to be due to the local and global modeling architecture of the DPRNN and the dynamic receptive field of the LSTMs. For the 2-speaker variant, offline DPRNN was able to achieve an SI-SNRi of 17.7 dB, an SI-SNRi of 13.8 dB for the 3-speaker variant and an SI-SNRi of 9.6 dB for the 4-speaker variant. These models have a size of $2.6 \cdot 10^6$ trainable parameters. All test results are seen in table 4.1 for TasNet and in table 4.2 for DPRNN.

5 Speaker Ranking & Enhancement

In this section, the development of the speaker ranking- and enhancement stage will be discussed. A block diagram of the system can be seen in figure 5.1.

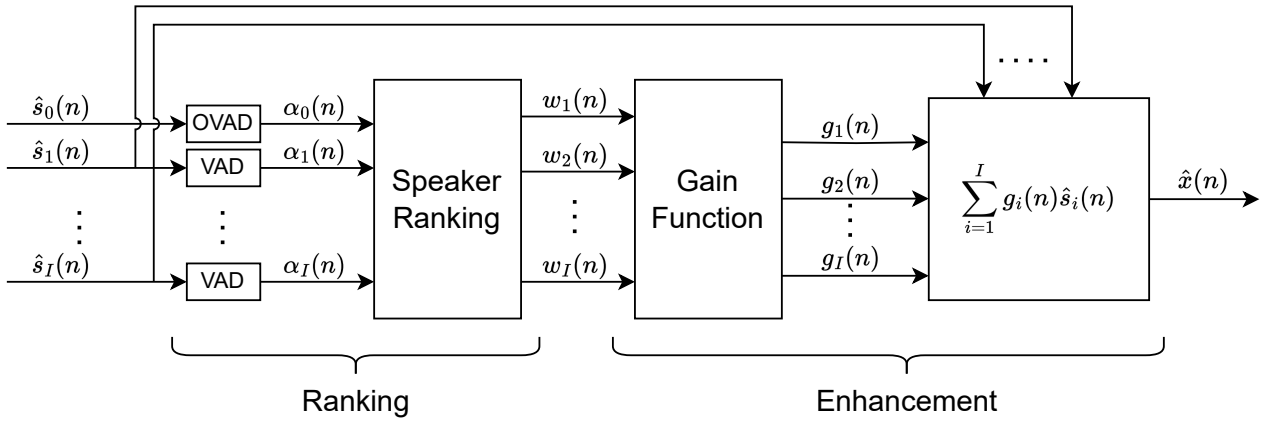


Figure 5.1: The combined ranking and enhancement system propose by Poul Hoang et al.[4].

The speaker ranking stage consists of two systems, first the separated audio signals ($\hat{s}_i$) are passed through a VAD system to indicate then the speech and silence are present. These decisions (a_i) are then fed to the "Minimum Overlap-Gap" algorithm where the overlaps and gaps between the VAD decisions of two channels are used to decide which channels are having a conversation with each other. The VAD system used in this project is the "rVAD" system [39]. This algorithm uses a two-stage denoising followed by a voice activity detection stage. The first denoising stage uses pitch as a speech indicator to remove high-energy noise segments detected by a posterior signal-to-noise ratio (SNR) weighted energy difference. The second denoising stage uses noise estimation to perform speech enhancement in an attempt to remove stationary noise. The final stage is a VAD, which uses the pitch estimation of the previous step to group the frames into pitch segments. The a posterior SNR weighted energy difference is then calculated for these segments and used to evaluate the VAD decision. The pitch estimation is a computationally expensive process and therefore a second rVAD is presented which replaces pitch estimation with spectral flatness [39]. This method is called rVAD-fast and is significantly faster ($\approx$ 10 times), which is an advantage when dealing with a large data set or running the VAD system on low-resource devices. This is the reason the second version is used as the VAD system.

In the end, the ranking (w_i) is used in the enhancement stage to attenuate the channels with (g_i) according to their probability of containing the target speaker.

5.1 Minimum Overlap-Gap Algorithm

The MOG algorithm relies on the assumption that a conversation between two persons is turn-taking, meaning that while one person is talking, the other one is silent and vice versa. In more technical terms it can be defined as the difference in the voice activities that should be maximized meaning that the amount of gaps and overlaps of the voice activities is minimized, for a turn-taking conversation to happen. Knowing this the channel with the target speaker can be defined as the channel with the largest sum of squared error (SSE) for a given frame (5.1).

$$\hat{i} = \arg \max_{i \in \{1,\dots I\}} \sum_{k=n-N+1}^{n} (\alpha_0(k) - \alpha_i(k))^2 \tag{5.1}$$

where:

α_0	is the VAD signal of the user
α_i	is the VAD signal of the i^{th} channel
N	is the number of samples in a frame
$\hat{i}$	is the channel index of the estimated target speaker
I	is the number of candidate speakers

The size of the frame (N) has a significant influence on the performance of the MOG algorithm as the variance of the sum of squared differences will increase as the frame size decreases [4]. In the enhancement stage, the output is based on a hard decision. This can have a detrimental effect when misclassification occurs. This is due to the fact that the MOG can switch instantaneously between candidate speakers. This problem can be overcome by using BMOG.

Bayesian MOG

The Bayesian MOG (BMOG) relies on the fact that the distributions of the SSE between the user and the target, and the user and a competing speaker are different. The posterior probability can be calculated for each hypothesis by computing the likelihood functions. The distributions of the SSEs can be estimated as a beta-binomial distribution with the distribution function 5.2.

$$p(\phi_i|\gamma,\beta,N) = \frac{N!}{\phi_i!(N-\phi_i)!} \cdot \frac{B(\phi_i+\gamma, N-\phi_i+\beta)}{B(\gamma,\beta)} \tag{5.2}$$

$$B(\alpha,\mu) = \frac{\Gamma(\alpha)\cdot\Gamma(\mu)}{\Gamma(\alpha+\mu)} \tag{5.3}$$

Where:

N	is the number of samples in a frame
ϕ_i	is the SSE of the i^{th} candidate speaker
γ , β	are shaping parameters
$B(\,\cdot\,,\cdot\,)$	is the Beta-function (5.3)
$\Gamma(\,\cdot\,)$	is the Gamma-function

The shaping parameters are different for the distribution of the User/Target and User/Competing speaker denoted as γ_t, β_t for the User/Target and γ_v, β_v for the User/Competing speaker. Equation 5.2 suffers from instability, as factorials will explode in size beyond what is practical, for a relatively small input. Therefore it can be advantageous to do the calculations in the logarithmic domain using the log-gamma function, thus transforming multiplications into additions.

$$\ln(p(\phi_i|\gamma,\beta,N)) = k1 - k2 \tag{5.4}$$

$$k1 = \ln\Gamma(N+1) + \ln\Gamma(\phi_i+\gamma) + \ln\Gamma(N-\phi_i+\beta) + \ln\Gamma(\gamma+\beta) \tag{5.5}$$

$$k2 = \ln\Gamma(\phi_i+1) + \ln\Gamma(N-\phi_i+1) + \ln\Gamma(\gamma) + \ln\Gamma(\beta) + \ln\Gamma(\gamma+N+\beta) \tag{5.6}$$

The parameters γ and β are estimated using maximum likelihood estimation, for different integration times. A power model 5.7 is then fitted to each shaping parameter given the maximum likelihood estimates, such that each power model represents a shaping parameter as a function of the integration time e.g. $\tilde{\gamma}(T_{int},\hat{a},\hat{b})$.

$$f(T_{int}) = a \cdot {T_{int}}^{b} \tag{5.7}$$

Table 5.1: Power model parameters for approximating shaping parameters in the beta-binomial distributions [4].

	$\tilde{\gamma}_t(\cdot,\hat{a},\hat{b})$	$\tilde{\beta}_t(\cdot,\hat{a},\hat{b})$	$\tilde{\gamma}_v(\cdot,\hat{a},\hat{b})$	$\tilde{\beta}_v(\cdot,\hat{a},\hat{b})$
$\hat{a}$	2.5091	0.8522	0.7736	0.8057
$\hat{b}$	0.7879	0.7817	0.9727	0.9681

The posterior probability now be calculated using Bayes theorem (5.8).

$$P(\mathcal{H}_i|\phi_1,...,\phi_I) = \frac{P(\mathcal{H}_i) \cdot p_{\Phi_1,...,\Phi_I}(\phi_1,...,\phi_I|\mathcal{H}_i)}{p_{\Phi_1,...,\Phi_I}(\phi_1,...,\phi_I)} \tag{5.8}$$

$$p_{\Phi_1,...,\Phi_I}(\phi_1,...,\phi_I|\mathcal{H}_i) = p_{\Phi_i}(\phi_i|\gamma_t,\beta_t,N) \prod_{j\in\mathcal{I}\setminus i} p_{\Phi_j}(\phi_j|\gamma_v,\beta_v,N) \tag{5.9}$$

$$p_{\Phi_1,...,\Phi_I}(\phi_1,...,\phi_I) = \sum_{k=1}^{I} P(\mathcal{H}_k) \cdot p_{\Phi_k}(\phi_k|\gamma_t,\beta_t,N) \prod_{l\in\mathcal{I}\setminus k} p_{\Phi_l}(\phi_l|\gamma_v,\beta_v,N) \tag{5.10}$$

where:

$P(\mathcal{H}_i)$	is the prior probability of the conversational partner being channel i.
$p_{\Phi_1,...,\Phi_I}(\phi_1,...,\phi_I\|\mathcal{H}_i)$	is the likelihood function conditioned on $\mathcal{H}_i$. Equation 5.9.
$p_{\Phi_1,...,\Phi_I}(\phi_1,...,\phi_I)$	is the marginal. Equation 5.10.
$\mathcal{I}$	is the set $\mathcal{I} = \{1,...,I\}$.
$\mathcal{I} \setminus i$	is the set $\mathcal{I}$ excluding i.

For stability reasons, the above calculations are done in the logarithmic domain.

5.2 Test on Conversation Dataset

The distributions of the SSEs of the user/target and user/competing speaker are made using the shaping parameters approximated using equation 5.7. The dataset used for making the histogram is the TDNDT dataset [36]. Since the MOG algorithm was originally developed where the input signals were sampled at 16 kHz but the signal from the separation stage is sampled at 8 kHz the development of this MOG algorithm is done using input signals sampled at 8 kHz. When comparing the histograms from figure 5.2 and 5.3 with those in [4] it can be seen that the sample rate does not change the distributions. This makes sense as it is the structure of the conversations that makes the distributions.

These histograms are made using an integration time of $T_{int} = 10\,\text{s}$. This integration time is used as it is the one with a significant improvement in ESTOI and PESQ compared to the state-of-the-art methods, and an integration time of 10 s performs better at a higher number of candidate speakers than an integration time of 5 s as a higher integration time will decrease the dispersion of the distributions, thus making the distribution of the user/target and the user/competing speaker more separated resulting in higher classification accuracy. But a too large integration time will result in a slow reaction time and therefore misclassifications can occur. It can be seen that the estimated PDF approximates the distribution of the SSEs. This matches the results published by Poul Hoang et al. [4].

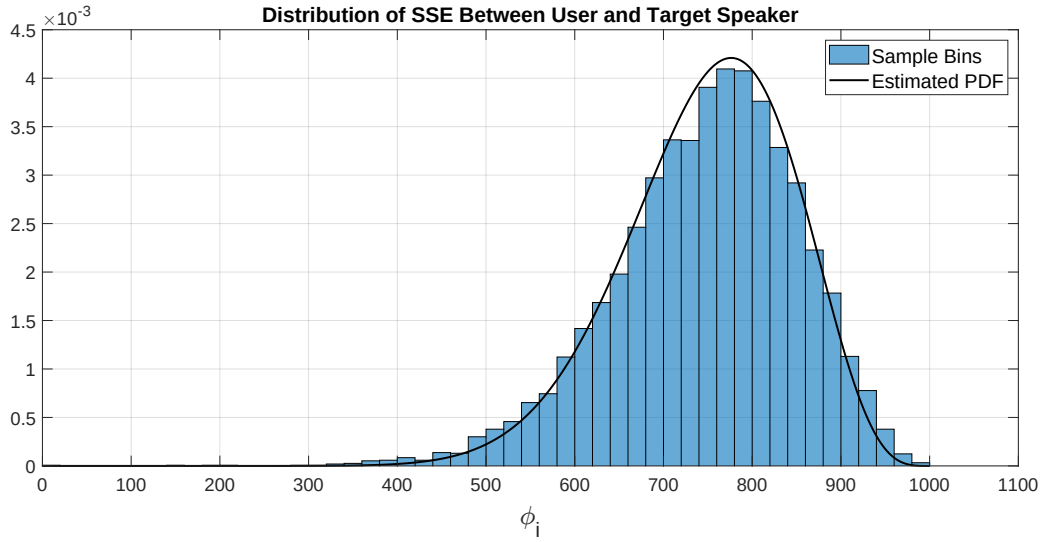


Figure 5.2: Distribution of the SSE between the user and target speaker and the fitted beta-binomial distribution.

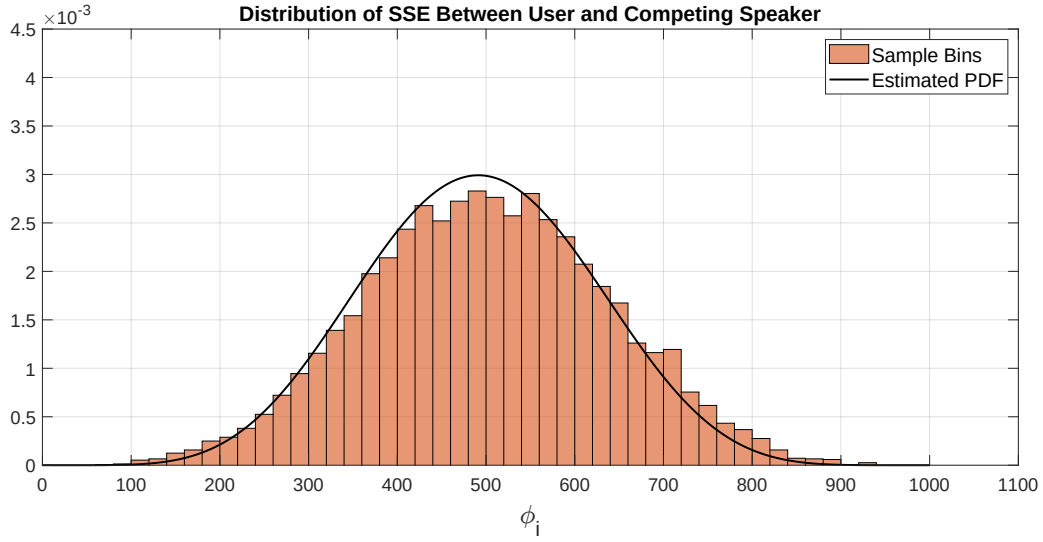


Figure 5.3: Distribution of the SSE between user and competing speakers and the fitted beta-binomial distribution.

A scenario with two conversations can be seen in figure 5.4 and 5.5. It can be seen in figure 5.4 that the MOG estimates that "User" is in a conversation with "Candidate speaker 1" most of the time. It can also be seen that at the time stamp of 190 seconds to 230 seconds there are a lot of switching between "Candidate speaker 1" and "Candidate speaker 3" as the target speaker. This is an example of where the BMOG algorithm can improve.

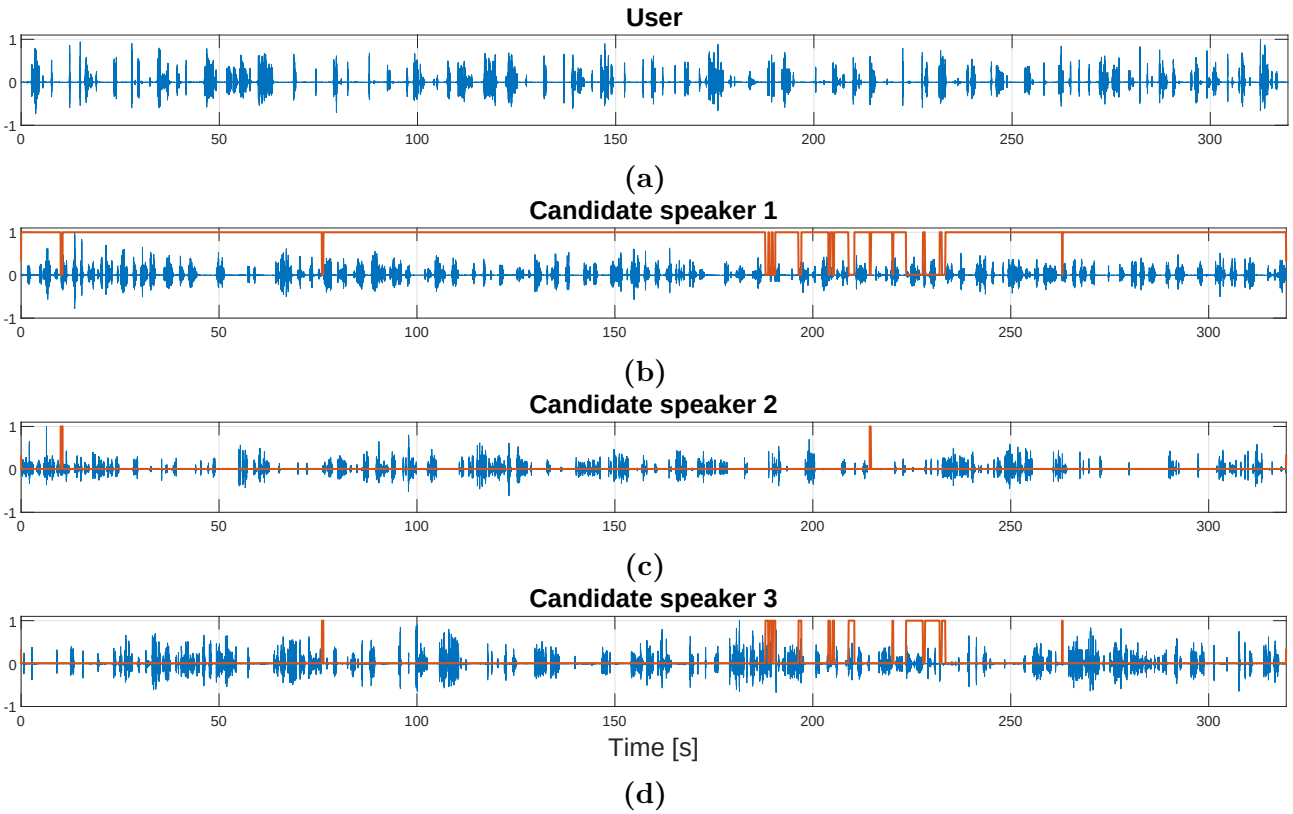


Figure 5.4: Audio signals from 1 user and 3 candidate speakers (Blue) and the corresponding weights (Orange) calculated using the MOG algorithm, plotted for a user and three candidate speakers.

In figure 5.5 the BMOG also estimates that "User" is having a turn-taking conversation with "Candidate speaker 1". It can be seen that the unwanted switching that the MOG algorithm produces in the channel where "Candidate speaker 1" is placed is now reduced using the BMOG algorithm. The switching is still present, but it is more smooth and "Candidate speaker 1" is not fully muted as it is using the MOG algorithm which will result in a more smooth transition.

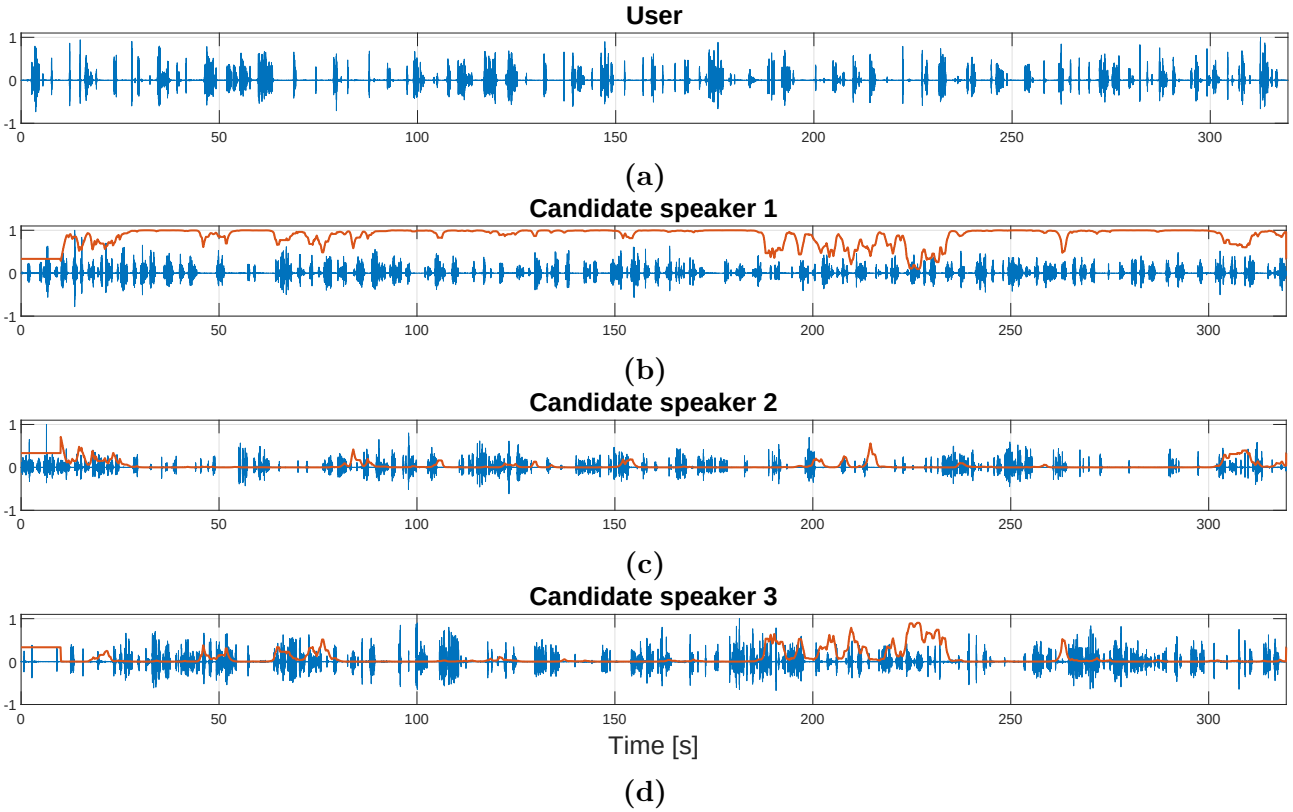


Figure 5.5: Audio signals from 1 user and 3 candidate speakers (Blue) and the corresponding weights (Orange) calculated using the BMOG algorithm, plotted for a user and three candidate speakers.

For evaluating the speech ranking and enhancement performance ESTOI and PESQ will be used as it is done by Poul Hoang et al. [4], where a random conversation and Nc random competing speakers are used for each realization. This is done for an integration time T_{int} of 10 s and for 1 to 10 competing speakers. The minimum gain for the MOG is 0.01 and 0 for the BMOG. The results are averaged over 100 realizations. The evaluation is done to do a comparison with the results in the paper. The results of the evaluations can be seen on figure 5.6 and 5.7.

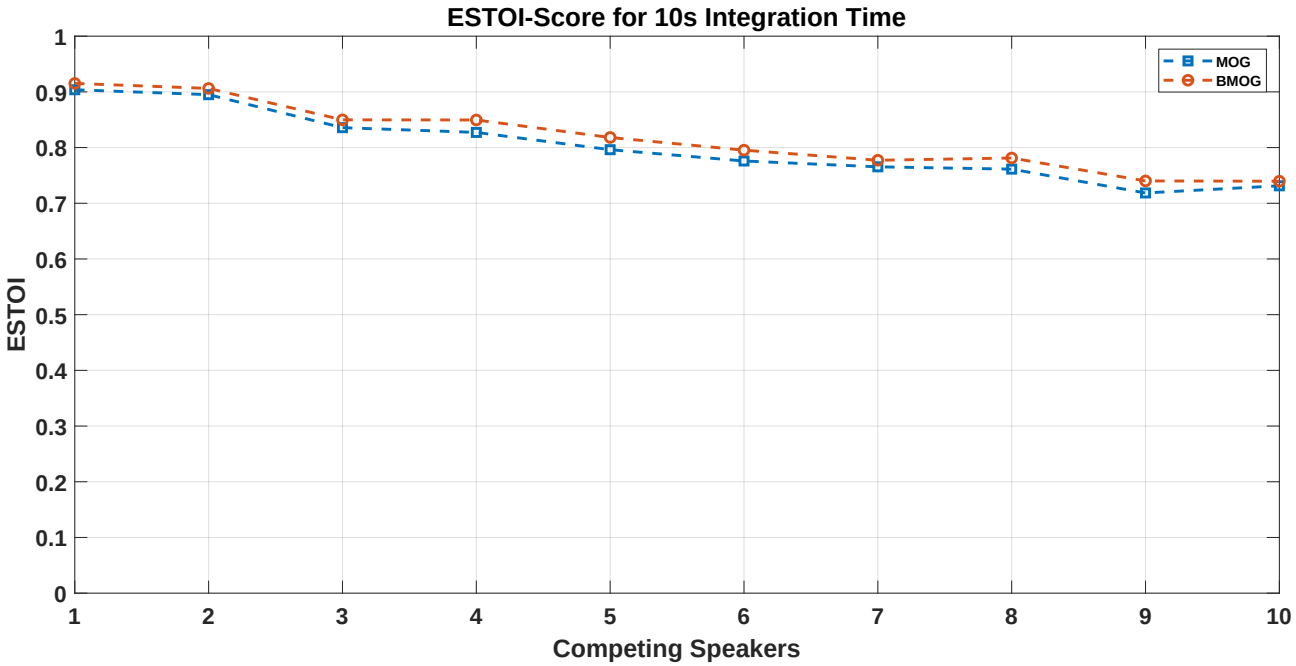


Figure 5.6: Averaged ESTOI as a function of the number of competing speakers.

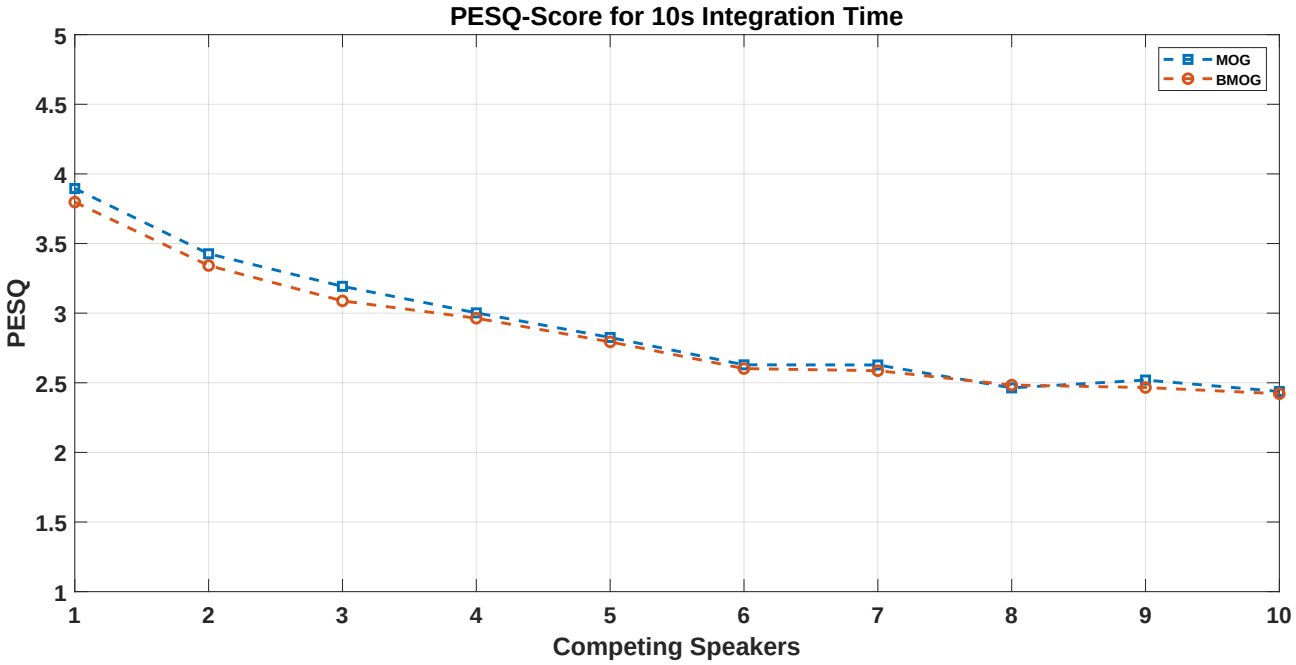


Figure 5.7: Averaged PESQ as a function of the amount of competing speakers.

It can be seen that the scores in the figures above (Figure 5.6 and 5.7) are similar to the results presented by Poul Hoang et al. Some variation will occur as the speakers are randomly selected.

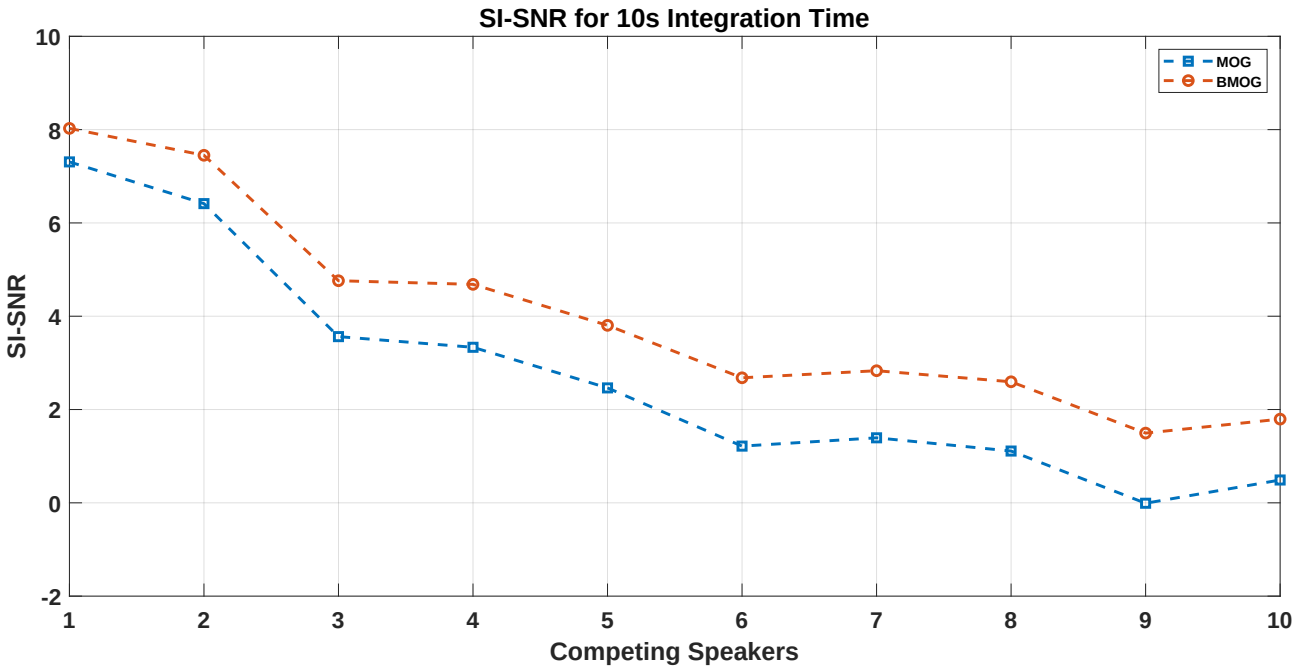


Figure 5.8: Average SI-SNR of the output as a function of the number of competing speakers.

The SI-SNR of the ranking and enhancement system has also been evaluated. It can be seen that the MOG has slightly worse performance that BMOG for the SI-SNR measure.
Due to the fact that the BMOG algorithm is better at handling misclassification and has a better SI-SNR when compared with the MOG, it is used for speaker ranking in the full system.

6 Evaluation of the Speech Enhancement System

In this chapter, the complete speech enhancement system is evaluated. The complete system is constructed by combining a speech separation model from chapter 4.2 and the BMOG algorithm. The speech separation models have been trained using the WSJ dataset due to the limited size of the TDNDT. It is also shown in this chapter that directly training on the TDNDT has been unsuccessful. The full system can be seen in figure 6.1.

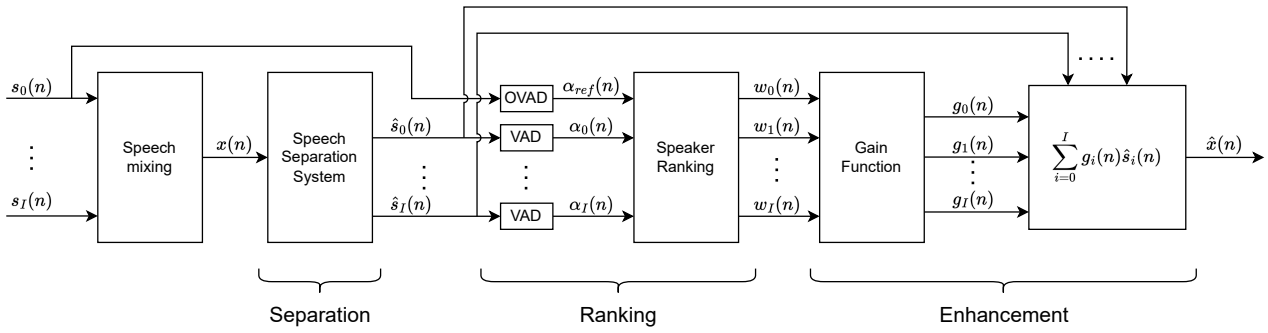
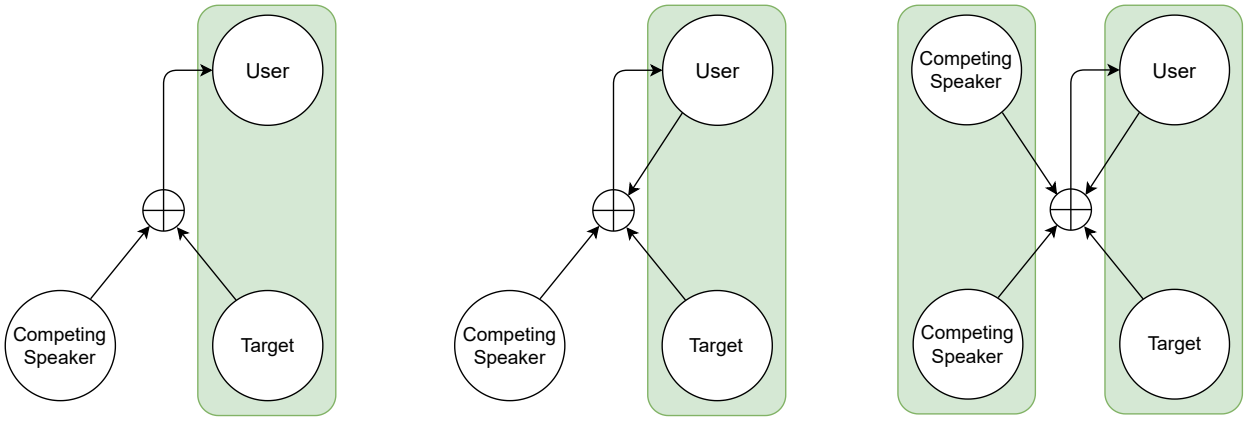


Figure 6.1: Full enhancement system, with separation stage, ranking stage, and enhancement state.

A 2-, 3-, and 4-speaker version of TasNet and DPRNN is evaluated using both utterance and conversation datasets in scenarios which can be seen in figure 6.2a, 6.2b and 6.2c.



(a) Two speaker scenario.

(b) Three speaker scenario with a target, a competing speaker and the user's own voice.

(c) Four speaker scenario with a target, user's own voice, and two competing speakers having a separate conversation.

Figure 6.2: The three conversation scenario that the results will be based on. Green boxes show conversational partners.

An ablation study is conducted on the best-performing system to evaluate the potential areas of improvement. This involves removing certain components of the system to understand the contribution of the component to the full system.
Methods involving transfer learning are used in an attempt to improve the performance of the speech separation stage. Following this, an analysis of the results is given.

6.1 Results

In the following tests, the best 2-, 3-, and 4-speaker TasNet and DPRNN models in both causal and non-causal versions have been used as the speech separation stage, i.e. pre-trained models trained using the WSJ0 dataset. These systems have been tested on the TDNDT dataset (Both English and Danish sound clips). Following this, the same systems have been tested using the synthetic WSJ0 conversation dataset in an attempt to isolate the effect of the change from utterance to conversation and the effect of changing the test settings from being matched, in the synthetic WSJ0 conversation dataset, to being unmatched, in the TDNDT dataset. The performance is measured at two points. The first point is at the output of the separation stage, as these channels are the ones used by the ranking system. The quality of the separated channels is measured using SI-SNR and SI-SNRi. All SI-SNR scores are relative to the unprocessed input mix. SI-SNRi is measured in comparison to the SI-SNR of the input mix. The average performance, the average of the worst-performing channel, and the average of the best-performing channels are shown. The second measuring point is at the output, i.e. after the separated channels have been weighted by the ranking system. At this point, the quality is measured using SI-SNR and ESTOI. These numbers are measured using all samples after 10 s, as this is the point where BMOG starts functioning. All models have been tested with 200 randomly constructed sound clips adhering to the acoustic scene described in figure 6.2.
The results for the synthetic dataset are seen in table 6.1 for TasNet and in table 6.4 for DPRNN. The results for the Danish conversation set are seen in table 6.2 and 6.3 for TasNet and in table 6.5 and 6.6 for DPRNN. The results will be discussed in section 6.4.

Table 6.1: Shows the results of testing different TasNet model on the synthetic WSJ0 conversation set. All results are an average of 200 forward passes. High and low SI-SNR refers to the average of the best and worst channels respectively. Causal implementations are marked with a "C".

Model	SI-SNR	SI-SNRi	High SI-SNR	Low SI-SNR	Output SI-SNR	ESTOI
2 spk	13.6	13.6	15.2	12.0	8.5	0.85
2 spk C	6.0	6.0	7.8	4.3	3.6	0.82
3 spk	7.4	10.7	14.2	1.8	3.1	0.79
3 spk C	3.1	6.3	9.5	-3.4	2.3	0.78
4 spk DM	3.4	8.8	11.6	-5.6	-4.8	0.65
4 spk C DM	-0.6	4.6	7.9	-11.2	-4	0.63

Table 6.2: Shows the results of testing different TasNet model on the English TDNDT dataset. All results are an average of 200 forward passes. High and low SI-SNR refers to the average of the best and worst channels respectively. Causal implementations are marked with a "C".

Model	SI-SNR	SI-SNRi	High SI-SNR	Low SI-SNR	Output SI-SNR	ESTOI
2 spk	4.6	4.6	8.5	0.7	-0.6	0.68
2 spk C	2.6	2.6	6.7	-1.6	-3.3	0.65
3 spk	-2.6	1.6	3.6	-9.5	-8.8	0.52
3 spk C	-4.2	0.2	1.6	-10.6	-6.4	0.54
4 spk DM	-1.1	5.8	6.4	-8.8	-12.3	0.44
4 spk C DM	-3	3.5	3.3	-9.7	-8.5	0.44

Table 6.3: Shows the results of testing different TasNet model on the Danish TDNDT dataset. All results are an average of 200 forward passes. High and low SI-SNR refers to the average of the best and worst channels respectively. Causal implementations are marked with a "C".

Model	SI-SNR	SI-SNRi	High SI-SNR	Low SI-SNR	Output SI-SNR	ESTOI
2 spk	4.9	4.9	8.3	1.5	-0.3	0.71
2 spk C	2.4	2.4	7.1	-2.4	-4.3	0.66
3 spk	-2.7	1.6	4.4	-10.2	-8.7	0.54
3 spk C	-3.9	0.3	2	-10.7	-6.8	0.53
4 spk DM	-1.4	5.1	5.7	-8.9	-11.5	0.45
4 spk C DM	-3.3	3.3	3.2	-9.8	-8.2	0.46

Table 6.4: Shows the results of testing different DPRNN model on the synthetic WSJ0 conversation set. All results are an average of 200 forward passes. High and low SI-SNR refers to the average of the best and worst channels respectively. Causal implementations are marked with a "C".

Model	SI-SNR	SI-SNRi	High SI-SNR	Low SI-SNR	Output SI-SNR	ESTOI
2 spk	15.4	15.4	17.0	13.7	8.4	0.87
2 spk C	11.1	11.1	12.7	9.6	8.0	0.87
3 spk	7.7	11	13.2	3.3	2.4	0.81
3 spk C	8	11.2	14.4	2.8	3.3	0.81
4 spk DM	5.2	10.4	12.8	-2	-3.3	0.67
4 spk C DM	1.8	6.9	9.8	-8.4	-5.8	0.63

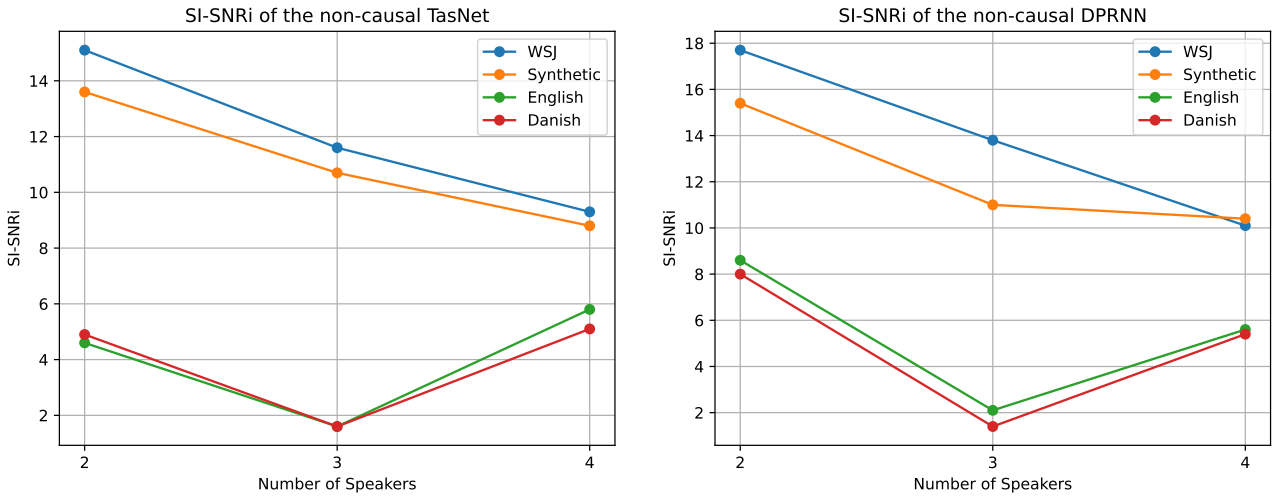
Table 6.5: Shows the results of testing different DPRNN model on the English TDNDT dataset. All results are an average of 200 forward passes. High and low SI-SNR refers to the average of the best and worst channels respectively. Causal implementations are marked with a "C".

Model	SI-SNR	SI-SNRi	High SI-SNR	Low SI-SNR	Output SI-SNR	ESTOI
2 spk	8.6	8.6	12.8	4.5	2.8	0.73
2 spk C	1.4	1.4	7.0	-4.2	0.2	0.72
3 spk	-2.2	2.1	3.5	-8.5	-10.2	0.58
3 spk C	-2	2.2	5.6	-11.1	-10.9	0.51
4 spk DM	-0.9	5.6	6.3	-9.5	-11.6	0.43
4 spk C DM	-3.6	3	3.8	-12.1	-12.9	0.44

Table 6.6: Shows the results of testing different DPRNN model on the Danish TDNDT dataset. All results are an average of 200 forward passes. High and low SI-SNR refers to the average of the best and worst channels respectively. Causal implementations are marked with a "C".

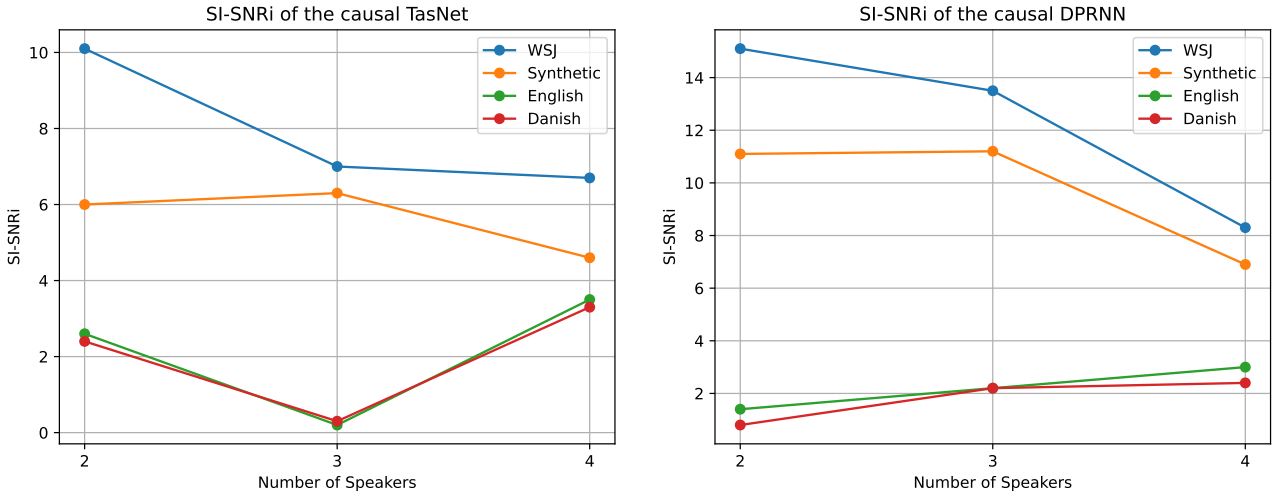
Model	SI-SNR	SI-SNRi	High SI-SNR	Low SI-SNR	Output SI-SNR	ESTOI
2 spk	8.0	8.0	11.9	4.1	3.7	0.75
2 spk C	0.8	0.8	6.2	-4.5	0.0	0.74
3 spk	-3	1.4	2.5	-9.1	-10.7	0.57
3 spk C	-2.2	2.2	5.4	-11	-10.9	0.53
4 spk DM	-1.1	5.4	6	-9.6	-10.9	0.46
4 spk C DM	-4	2.4	2.5	-11.8	-12.2	0.46

To highlight the performance change between the WSJ0 utterance test, the WSJ0 conversation test, and the Danish conversation set, figure 6.3 shows the non-causal model performance and figure 6.3 shows the performance of the causal models.



(a) Non-causal TasNet. **(b)** Non-causal DPRNN.

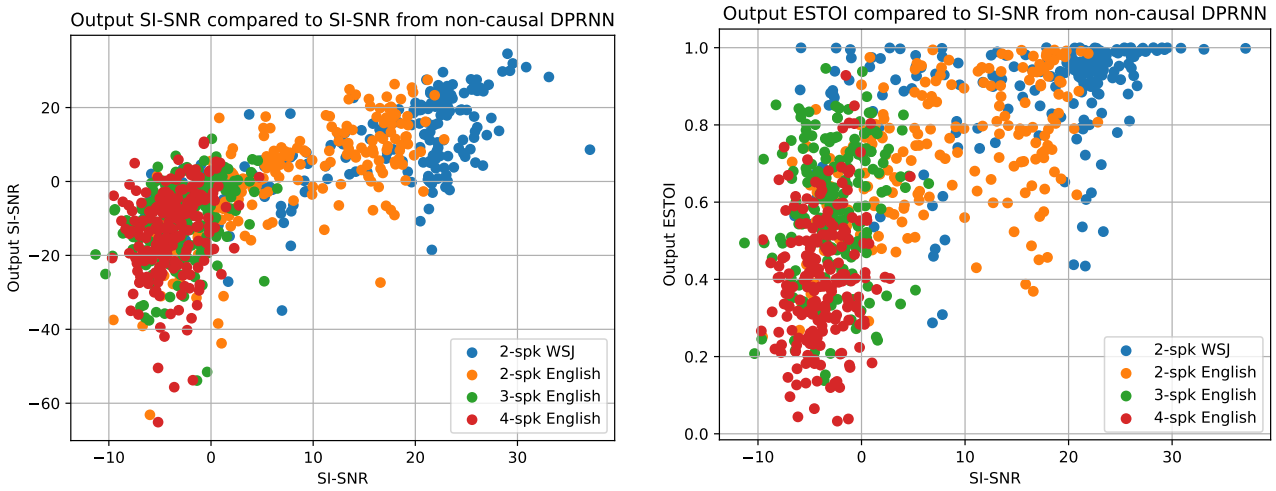
Figure 6.3: Shows the performance in SI-SNRi of the non-causal models of TasNet and DPRNN across different test sets. 2-, and 3-speaker are trained on predefined WSJ mixes, while 4-speaker models are trained using dynamic mix.



(a) Causal TasNet. **(b)** Causal DPRNN.

Figure 6.4: Shows the performance of the causal models of TasNet and DPRNN across different test datasets. 2-, and 3-speaker are trained on predefined WSJ mixes, while 4-speaker models are trained using dynamic mix.

The correlation between SI-SNR of the separated channels compared to the resulting ESTOI on the output is also investigated. The result of this can be seen in figure 6.5.



(a) BMOG SI-SNR.

(b) ESTOI.

Figure 6.5: Shows the SI-SNR compared to ESTOI scores and SI-SNR compared to output SI-SNR scores of different models based on the non-causal DPRNN.

6.2 Ablation Study

An Ablation study is performed on the full enhancement system using the best-performing separation network, which is the non-causal DPRNN. The system can be split into two sub-systems, which means the ablation study is done in two parts. The first part is a test of the speech separation given an ideal speaker ranking and enhancement, i.e. the speaker ranking always chooses the channel from the speech separation which best approximates the target. The second part is a study of the ranking stage given an ideal speech separation. This part has already been investigated as a part of the (B)MOG algorithm testing and verification using the TDNDT dataset (Section 5.2). The ESTOI performance for 2-, 3- and 4-speaker from figure 5.6 are reused for this study. The part with ideal ranking uses the same frame rate as the BMOG part of the system. For each update, all the separated channels are compared to the label of the target speaker, to see which channel has the most fitting samples. The ESTOI scores for the two sub-systems and the full system can be seen in figure 6.6.

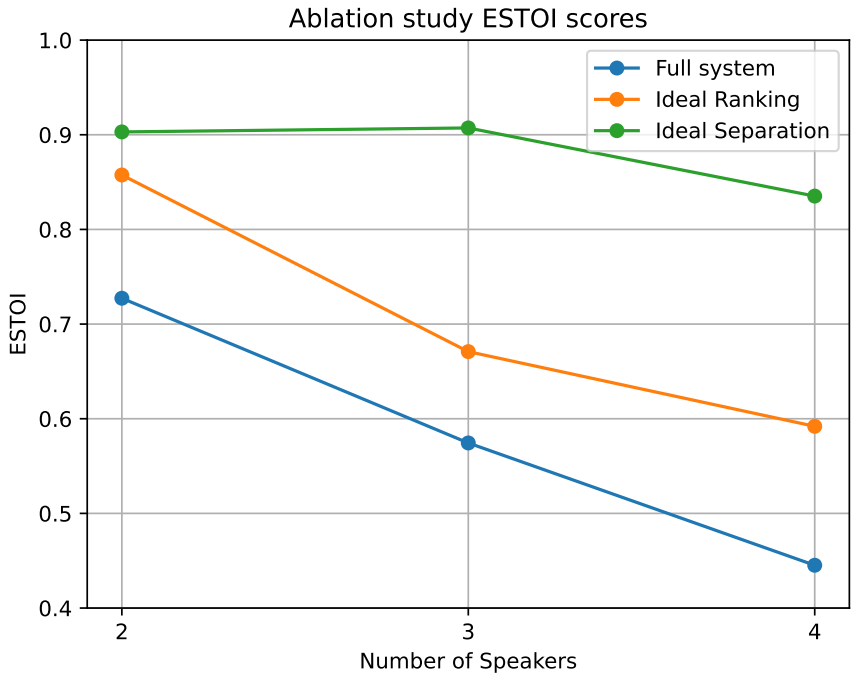


Figure 6.6: ESTOI scores for 2-, 3- and 4-speaker scenarios for the full system, a system with ideal ranking stage, and a system with ideal speech separation.

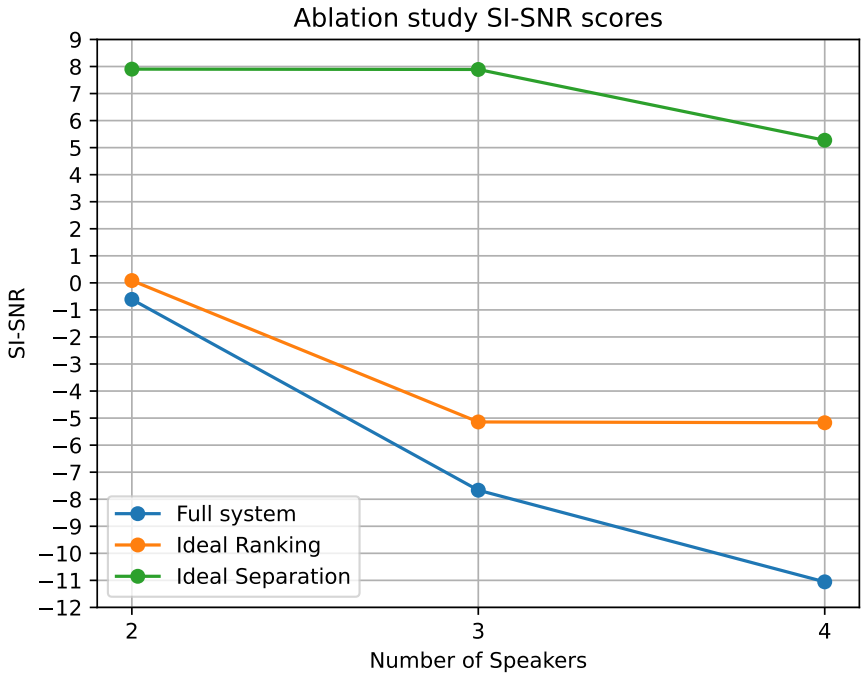


Figure 6.7: SI-SNR scores for 2-, 3- and 4-speaker scenarios for the full system, a system with ideal ranking stage, and a system with ideal speech separation.

6.3 Transfer Learning Study

Two transfer learning schemes are used to recover parts of the performance drop when moving from the WSJ datasets to the TDNDT dataset. Both approaches split the TDNDT set into two subsets, one subset for training containing 14 pairs and one set containing 5 pairs for testing. The training set contains 8 pairs of man-man, 3 pairs of woman-woman, and 3 pairs of man-woman for a total of 167 recordings. The test set contains 3 pairs of man-man, 1 pair of woman-woman, and 1 pair of man-woman for a total of 60 recordings. The first approach uses a naive method, where the DPRNN non-causal model is further trained on the training set followed by tests. The second approach uses the same rVAD as in the speaker ranking step to identify sections of speech in the training set. All sections shorter than 2 s are discarded as these mostly contain single words or simple answering phrases. The rest are then combined using the dynamic mix algorithm described in section 3.3 to form an utterance dataset comparable to WSJ. The test set is unaltered. Both approaches are compared to training a new model using the same dataset. The results of these two approaches are seen in table 6.7, where they have been compared to the original model on which the transfer learning is based (WSJ U) and new models trained using the same dataset. The transfer learned models have been trained for 40 epochs, while the new models were trained for up to 100 epochs or until stopped by the early stopping algorithm. The training settings are equal to those used for the training of the speech separation models (See chapter 4). The only difference is that a learning rate of $1 \cdot 10^{-5}$ was used for the models trained with transfer learning (Previously $1 \cdot 10^{-3}$). The tests are conducted using 200 forward passes with random combinations of the speakers available in the test set.

Table 6.7: Shows the results of testing different transfer learning schemes compared to the original WSJ-trained model. All results use the 2-speaker non-causal DPRNN. U refers to an utterance dataset and C refers to a conversation dataset. TL refers to transfer learned from WSJ U.

Model	SI-SNR	SI-SNRi	High SI-SNR	Low SI-SNR	Output SI-SNR	ESTOI
WSJ U	11.6	11.6	15.5	7.7	7.9	0.80
TDNDT C	5.2	5.2	5.3	5.1	0.9	0.70
TL TDNDT C	14.7	14.7	14.8	14.5	9.5	0.84
TDNDT U	0.1	0.1	0.3	0.0	-14.4	0.65
TL TDNDT U	12.1	12.1	12.5	11.6	6.0	0.82

6.4 Analysis of Results

The evaluation of the complete enhancement system is based on three scenarios, which are shown in figure 6.2. Two different conversation datasets have been evaluated in these scenarios. These are the synthetic WSJ conversation dataset and the TDNDT dataset (See section 3). The TDNDT dataset has been subdivided into 2 smaller datasets, one where the conversations are in Danish and one where it is in English.

The first dataset evaluated was the synthetic WSJ0 conversation dataset. This dataset is included as this, while synthetic, provides nearly matched conditions to the conditions on which the networks have been trained. The benefit of this becomes clear when looking at the performance drop of the SI-SNRi in figure 6.3 and 6.4. For the non-causal models, the drop in performance does not exceed 2.8 dB, and in the case of the 4-speaker model, the performance becomes nearly identical. This is hypothesized to originate from the use of dynamic mixing on the 4-speaker models. For the causal models, a more inconsistent performance drop is observed but does not exceed 4 dB. The SI-SNR of the resulting separated channels is on average all above 1.8 dB except for the 4-speaker causal TasNet model. The output SI-SNR is on average positive for all 2- and 3-speaker models evaluated on the synthetic WSJ conversation dataset. Combined with an average ESTOI above 0.78 these models are deemed capable of single out the target and attenuating competing speakers.

However, this is not the case for the 4-speaker models, where the output SI-SNR is on average negative. This is likely due to variance in SI-SNR of different channels as highlighted by the high SI-SNR and the low SI-SNR scores. A single bad-performing channel can make the speaker ranking nearly impossible. Both if the target speaker slips into multiple channels, but also if the bad-performing channel is the target. However, even for these 4-speaker models, the ESTOI score does not drop below 0.63.

Moving from the matched conditions of the synthetic WSJ conversation dataset to the unmatched conditions of the TDNDT dataset. This dataset is completely new to the model, with new speakers, in new languages, and with real conversations. The first observation is the far larger performance drop compared to the synthetic WSJ conversation dataset or the utterance WSJ dataset. In the worst case, the SI-SNRi performance drops to 0.2 dB. A performance drop of approximately 10 dB in SI-SNRi is observed for 2-speaker models. However, even with these performance drops, the 2-speaker non-causal DPRNN is able to achieve an output SI-SNR of 2.8 dB and 3.7 dB for English and Danish TDNDT data, respectively. These models also score an average ESTOI of 0.73 and 0.75 respectively. For the 2-speaker causal DPRNN, the output SI-SNR is approximate 0 dB. For all other models, the output SI-SNR is negative, as the models are not able, on average, to single out the target speaker and attenuate competing speakers. It is observed that the difference between the high SI-SNR and the low SI-SNR channels is larger for these models.

Three further observations are made. The first is that the performance difference between the conversation in English and Danish is marginal and essentially non-existent. This implies that the models are language-independent, which matches what is reported in literature [40].

The second observation is the performance increase of the 4-speaker model compared to the

2- and 3-speaker models. In all cases except the non-causal DPRNN, the 4-speaker model is able to achieve the largest SI-SNRi score. Similar to the increased performance of the 4-speaker model trained on the synthetic WSJ dataset, this is hypothesized to originate from the use of dynamic mixing.

The third observation is the negative output SI-SNR even given an average positive SI-SNR on the separated channels. This might originate from the fact that even a single bad-performing channel can cause difficulties for the subsequent systems. This bad-performing channel can contain leakage or cross-talking from other channels, which might be picked up by the VAD system as voice activity. This would then cause errors in the resulting probability of being the target speaker calculated by the BMOG system. This chain effect would convert a small performance degradation on the separated channels to a large performance degradation on the output of the full system.

In an attempt to find a correlation between the SI-SNR of the separated channels and the ESTOI on the output of the system, the ESTOI and SI-SNR values for different models using the English TDNDT dataset and the synthetic WSJ dataset are plotted in figure 6.5. From figure 6.5a, it can be observed that a higher SI-SNR on separated channels leads to a higher SI-SNR on the output. However, it is observed that the effect diminishes as the SI-SNR increases. From figure 6.5b, it can be observed that the general trend is that a higher SI-SNR on the separated channels leads to a higher ESTOI output. However, the variance is large, which results in the fact that a high SI-SNR does not necessarily guarantee a high ESTOI.

An ablation study was conducted. The results of this are seen in figure 6.6. From this figure, it is observed that an ideal ranking system is able to boost the performance of the system, but that an ideal separation stage results in a far larger performance increase for 3-speaker and 4-speaker models. For the 2-speaker model, the performance increase is more similar, but an ideal separation still results in a better performance.

Due to the result of the ablation study, it was attempted to improve the performance of speech separation with the use of transfer learning with the non-causal 2-speaker DPRNN model as the base model. The results are seen in table 6.7. The results highlight the difficulties of training on the Danish TDNDT dataset directly as the SI-SNRi performance of the new models 'Danish C' and 'Danish U' are worse than the performance of the model trained using the WSJ set. However, the transfer learning using the WSJ-trained model proved successful. Training on the conversation set proved to provide the best performance increase. This model achieved an average output SI-SNR of 9.5 dB on TDNDT set. This model also score an average ESTOI of 0.84 which is an overall improvement compared to both the original WSJ-trained model and the model trained on the conversation set. The other approach uses an utterance version of the TDNDT dataset. The model trained using transfer learning was able to achieve an average output SI-SNR of 6.0 dB on TDNDT set. This model also score an average ESTOI of 0.82 which is also an improvement compared to both the model trained only on the utterance TDNDT set. However, it is not an improvement compared to the original WSJ-trained model when measured on output SI-SNR, but it does improve the ESTOI by 0.2.

7 Conclusion

In this project, it was attempted to develop a novel speech enhancement system consisting of 3 functional blocks: Speech separation, speaker ranking, and speech enhancement. This architecture is suggested by Poul Hoang et al. [4]. The functional idea is based on a hearing aid user in a cocktail party environment, i.e. multiple competing speakers. The system would ideally be capable of attenuating competing speakers and amplifying the target speaker of the user. This system is developed in 3 versions: 2-speakers, 3-speakers, and 4-speakers. The main contribution of this project is the exploration of using state-of-the-art neural networks for speech separation and using these to create a complete speech enhancement system that can potentially solve the cocktail party problem.

Based on a review of the state-of-the-art speech separation using deep neural networks, two models were chosen for implementation. These are the convolutional TasNet and the DPRNN [7, 12]. Doing the implementation of these models, a novel online version of the DPRNN was developed using cumulative layer normalization and unidirectional LSTMs. These models achieved a SI-SNRi of 15.1 dB and 13.5 dB on the test set for a 2-speaker and 3-speaker model respectively using the matching WSJ0 dataset.

For speaker ranking and speech enhancement, the Minimum Overlap-Gap (MOG) algorithm was implemented.

For the evaluation of the speech enhancement system, 2 different conversation datasets were used. One is a synthetic conversation set based on the WSJ0 dataset. The other is the "Task dialog by native-Danish talkers in Danish and English in both quiet and noise" (TDNDT) conversation set, which contains real-world conversations [36]. The synthetic conversation set was generated such that statistics of the pause, overlap and sentence length match that of the TDNDT.

The results show functional results on the synthetic dataset for 2- and 3-speaker separation models for both Conv-TasNet and DPRNN. These models are functional in the sense that they all achieve an output SI-SNR of above 0 dB, which means that the target speaker has been amplified compared to the competing speakers. The worst performing model was the 3-speaker causal Conv-TasNet which achieved an average SI-SNR on the output of 2.3 dB compared to the target label with an average ESTOI of 0.78. This was achieved with an average input SI-SNR of -3.2 dB compared to the target label. The best-performing model was the 2-speaker non-causal DPRNN which achieved an average SI-SNR on the output of 8.4 dB compared to the target label with an average ESTOI of 0.87. This was achieved with an average input SI-SNR of 0 dB compared to the target label.

Conducting tests on the TDNDT, the results degraded significantly compared to the synthetic dataset. The degradation is hypothesized to originate mainly from the fact that the TDNDT set contains unseen speakers with different conditions regarding the acoustic environment and the speakers. However, functioning systems were still achieved. The 2-speaker non-causal DPRNN achieved an average SI-SNR on the output of 2.8 dB compared to the target label with an average ESTOI of 0.73 using English speech. This was achieved with an average input SI-SNR

of 0 dB compared to the target label. The same model achieved an average SI-SNR on the output of 3.7 dB compared to the target label with an average ESTOI of 0.75 using Danish speech. This was achieved with an average input SI-SNR of 0 dB compared to the target label. An ablation study was conducted on the complete speech enhancement system. This study highlighted that improvement in the speech separation quality has a greater potential for performance increases than the speaker ranking system. Two methods for improving speech separation were investigated and showed potential for increasing the performance of the system. These are the use of dynamic mixing in the training process or the use of transfer learning on the speech separation models to introduce the new conditions of the acoustic environment in the TDNDT set.

8 Discussion

In this project, we were able to show a proof of concept for a novel speech enhancement system based on a deep neural network and the MOG algorithm. Using this speech enhancement, we were able to achieve results that on average show that the 2- and 3-speaker models are working using the synthetic WSJ0 set. Working in the sense, that output SI-SNR is above 2.3 dB while having an ESTOI score above 0.78. While these results are on a synthetic dataset, the dataset was generated such that the pause length, sentence length, and overlap length match that of the TDNDT set, such that it more closely resembles a real conversation. However, even with these extra measures to ensure better matching conditions between the synthetic WSJ and the TDNDT set, we still observe a bigger drop in performance when applying the WSJ-trained models to the TDNDT set compared to applying the same models to the synthetic WSJ set. The performance drop is likely due to the difference in testing conditions, which are new to the model. The effect of these differences on performance drop is largely unknown. However, we will mention some differences, which are believed to be of larger impact.
First, the speakers are completely different and while the model itself is largely language-independent, it might not be talking style independent. The TDNDT set is comprised of native Danish people, which means they have a Danish accent when speaking English. This can be compared to the WSJ0 set, where the speakers are native American and therefore have a different accent. Furthermore, the WSJ0 speakers are articulated and monotone in their speaking, i.e. they ensure that every word and sound is easily heard. This can be compared to the TDNDT set, where the conversation is more relaxed and informal, such that words and sounds are not equally expressed, i.e. more dynamic. The structure of a conversation is completely missing from the utterance dataset as these are single sentences by individual people. However, as mentioned, the effect of the structural differences was attempted to be isolated as they were included in the synthetic WSJ, where no large performance drops were observed, even with it using unseen speakers. However, one thing the synthetic WSJ set did not include is the possibility of different speaking tempi. This is one thing that was included with the dynamic mix algorithm. It was able to achieve smaller performance drops than expected on the 4-speaker model. It was even able to have a smaller performance drop than 2-speaker and 3-speaker models.
However, even with these differences between datasets, we were able to show a functioning speech enhancement on the TDNDT set using the 2-speaker DPRNN, both causal and non-causal. These were capable of achieving higher than 0 dB SI-SNR and 0.72 ESTOI. While the 2-speaker scenario is not the most complex or interesting scenario, it can be argued that it still contains use cases as algorithms for removing a specific voice do exist. This means, that it would be possible to first remove the user's voice, leaving only the competing speakers. This signal can then be passed through speech enhancement to detect and amplify the target speaker [41].
Furthermore, we would argue that, while the 3-speaker and 4-speaker enhancement systems do not achieve an average SI-SNR of 0 dB or higher, they still do contain data points with an SI-SNR value above 0 dB and an ESTOI of over 0.7 These are rare, but they do highlight that

this task is possible for a speech enhancement system. This can be observed in figure 6.5.
Putting these things together, we would argue that we have demonstrated a proof of concept for the novel end-to-end speech enhancement system presented in this project. However, as mentioned, there is room for improvement. An ablation study was conducted to find the areas with the most room for improvement. Looking at figure 6.6 and 6.7, it is observed that the speech separation has the largest room for improvement. While an improved ranking system would also yield performance benefits, these are largely small compared to the potential performance yield of improved speech separation. We will, therefore, now discuss ideas for improvement of the speech separation stage.

Improvement using Dynamic Mix
The first idea for improvement is the use of a dynamic mix for all models. As mentioned earlier, it is hypothesized that the use of different speech tempi and the fact that input mixes are created anew every epoch would increase the generalization ability of a model. It is hypothesized that this is why we see the 4-speaker model outperform the 2-speaker and 3-speaker models when measuring SI-SNRi.

Improvement using Transfer Learning
The second idea is the use of transfer learning. This was shortly explored in the results section in table 6.7, where transfer learning was applied to the non-causal 2-speaker DPRNN model. The results of the transfer learning show increase in performance when trained on the TDNDT dataset. Another approach was attempted using an utterance TDNDT created using rVAD. However, this training method turned out to make the network perform slightly worse. The reason for this is unknown at the time of writing. No immediate differences were found between this and the WSJ0 set, besides the previously mentioned. It could be interesting to solve this problem, as it might highlight what makes training on the WSJ0 set work.
Worth mentioning, is the fact that training a completely new model using either of these approaches (Conversation TDNDT or utterances TDNDT) was not able to achieve the same performance as the original model trained using WSJ. The reason for this failure might be due to the limited size of this dataset, resulting in the network simply overfitting the training data and is therefore unable to generalize to unseen cases.
However, the overall findings, suggest that the approach using transfer learning with the TDNDT can be expanded to 3- and 4-speaker models to achieve an increase in performance.

Improvement by Expanding the DNN Architecture
The third idea for improvement is the simple idea of simply using larger models for 3-speaker and 4-speaker models. This could minimize the performance drop we observe when we increase the number of speakers. This might require a new hyperparameter tuning for each number of speakers. We would argue that this is a logical step, as the 3-speaker and 4-speaker scenarios are more complex situations and might require a more complex model. These larger models would then require more memory and time to train.

Future Work

These areas of improvement are seen as possible next steps within the scope of this project. However, looking further ahead, we would also like to present ideas for additions to the model and new areas of investigation. These additions could include the possibility of noise removal. One could look into the area of deep neural networks for denoising and expanding speech enhancement to include these capabilities. This could potentially also include the ability to remove room reverberation. This would provide a more complete solution for the cocktail party problem.

It might also be interesting to look into completely new approaches for the task of speech enhancement. An idea could be to train a network for detecting conversational partners and placing these into a common channel. This essentially boils the problem down to separating two speakers instead directly separating 3, 4, or more speakers. This idea is mainly inspired by the fact that the network was observed to place conversational partners in the same channel when it was unable to separate them. This behavior might originate from the fact that the turn-taking behavior of a conversation creates a continuous stream of speech. This could resemble the continuous stream of speech from each utterance in the WSJ0 training set, thereby leading the network to believe that the conversation originates from one person. This behavior was not consistent but might be exploitable and trainable.

Another idea is to include tasks currently done by the speaker ranking system into the neural network. This could be either formulated as a complete end-to-end speech enhancement system or simply include smaller tasks, such as performing the task of the VAD system. Such a system might be able to eliminate the sections of voice activity generated by cross-talking between channels introduced by the separation stage. This is wanted as the cross-talk can result in misclassifications by the VAD system. A more robust VAD system might improve this problem with cross-talk. A possible solution could be to generate the VAD decisions using a second deep neural network after the separation stage-trained using clean conversation data without cross-talk.

Another interesting area for exploration is the use of a multi-microphone setup in speech separation. These systems could use the microphones already present in systems using beam-forming. Multi-microphone setups do exist for the use of speech dereverberation such as the Scene-Agnostic Multi-Microphone Speech Dereverberation system presented by Yochai Yemini et al. [42]. This system takes inspiration from deep neural networks for image detection using multiple images from different positions. Yochai Yemini et al. drew a parallel between this and spectrograms from different microphones and were able to apply the same theories to successfully reverberate speech signals using multiple microphones. This could potentially be used to construct a speech separation stage capable of using a multi-microphone setup.

Bibliography

[1] Max E. Valentinuzzi. *Hearing Aid History: From Ear Trumpets to Digital Technology*. URL: https://www.embs.org/pulse/articles/hearing-aid-history-from-ear-trumpets-to-digital-technology/?fbclid=IwAR34xl6X2w6ni2zPK%20uPMyucVrGzCm5edaT_N1Htab7UyynVQxPOXczIQlIk (visited on 02/21/2023).

[2] JR. John S. Turner and John H. Per-Lee. *Auditory Dysfunction: Hearing Loss*. URL: https://www.ncbi.nlm.nih.gov/books/NBK225/ (visited on 02/21/2023).

[3] Mark A. Bee and Christophe Micheyl. "The Cocktail Party Problem: What Is It? How Can It Be Solved? And Why Should Animal Behaviorists Study It?" In: *Journal of Comparative Psychology* (2008).

[4] Poul Hoang, Zheng-Hua Tan, Jan Mark De Haan, and Jesper Jensen. "The Minimum Overlap-Gap Algorithm for Speech Enhancement". In: *IEEE Access* (2022).

[5] Harvey Sacks, Emanuel A. Schegloff, and Gail Jefferson. "A Simple Systematic for the Organisation of Turn Taking in Conversation". In: *Language* (1974).

[6] Yi Luo and Nima Mesgarani. "TasNet: Time-Domain Audio Separation Network for Real-Time Single-Channel Speech Separation". In: (2018).

[7] Yi Luo and Nima Mesgarani. "Conv-TasNet: Surpassing Ideal Time-Frequency Magnitude Masking for Speech Separation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2019).

[8] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. "A survey on modern trainable activation functions". In: *Neural Network 138* (2021).

[9] Tina Jacob - KDnuggets. *Vanishing Gradient Problem Explained*. URL: https://www.kdnuggets.com/2022/02/vanishing-gradient-problem.html (visited on 02/28/2023).

[10] Standford University. *CS231n Convolutional Neural Networks for Visual Recognition*. URL: https://cs231n.github.io/neural-networks-1/#actfun (visited on 03/03/2023).

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[12] Yi Luo, Zhuo Chen, and Takuya Yoshioka. "Dual-Path RNN: Efficient Long Sequence Modeling for Time-Domain Single-Channel Speech Separation". In: *arXiv* (2020).

[13] André Araujo, Wade Norris, and Jack Sim. *Computing Receptive Fields of Convolutional Neural Networks*. URL: https://distill.pub/2019/computing-receptive-fields/ (visited on 03/01/2023).

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[15] Cem Subakan, Mirco Ravanelli, Samuele Cornell, Mirko Bronzi, and Jianyuan Zhong. "ATTENTION IS ALL YOU NEED IN SPEECH SEPARATION". In: *arXiv* (2021).

[16] Max W. Y. Lam, Jun Wang, Dan Su, and Dong Yu. "SANDGLASSET: A LIGHT MULTI-GRANULARITY SELF-ATTENTIVE NETWORK FOR TIME-DOMAIN SPEECH SEPARATION". In: *arXiv* (2021).

[17] Diederik P. Kingma and Jimmy Lei Ba. "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION". In: *International Conference on Learning Representations* (2015).

[18] Dokkyun Yi, Jaehyun Ahn, and Sangmin Ji. "An Effective Optimization Method for Machine Learning Based on ADAM". In: *Applied sciences* (2020).

[19] Towards Data Science. *Different Normalization Layers in Deep Learning*. URL: `https://towardsdatascience.com/different-normalization-layers-in-deep-learning-1a7214ff71d6` (visited on 03/13/2023).

[20] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning* (2015).

[21] Tim Salimans and Diederik P. Kingma. "Weight Normalization: A Simple Reparameterizationto Accelerate Training of Deep Neural Networks". In: *arXiv* (2016).

[22] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: *arXiv* (2016).

[23] Peter Ochieng. "DEEP NEURAL NETWORK TECHNIQUES FOR MONAURAL SPEECH ENHANCEMENT: STATE OF THE ART ANALYSIS". In: *arXiv* (2022).

[24] Yinhao Xu, Jian Zhou, Liang Tao, and Hon Keung Kwan. "Multi-Scale Feature Fusion Transformer Network for End-to-End Single Channel Speech Separation". In: *arXiv* (2022).

[25] Morten Kolbæk, Dong Yu, Zheng-Hua Tan, and Jesper Jensen. "Multitalker Speech Separation With Utterance-Level Permutation Invariant Training of Deep Recurrent Neural Networks". In: *IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 25, NO. 10* (2017).

[26] Tobias Goehring, Josie L. Chapman, Stefan Bleeck, and Jessica J. M. Monaghan. "Tolerable delay for speech production and perception: effects of hearing ability and experience with hearing aids". In: *International Journal of Audiology* (2018).

[27] Cees H. Taal, Richard C. Hendriks, Richard Heusdens, and Jesper Jensen. "A SHORT-TIME OBJECTIVE INTELLIGIBILITY MEASURE FOR TIME-FREQUENCY WEIGHTED NOISY SPEECH". In: *IEEE International Conference on Acoustics, Speech and Signal Processing* (2010).

[28] A. W. RIX, J.G. Beerends, M.P. Hollier, and A.P. Hekstra. "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings* (2001).

[29] DeLiang Wang. "Time–Frequency Masking for Speech Separation and Its Potential for Hearing Aid Design". In: *Trends in Amplification* (2008).

[30] MathWorks. *Cocktail Party Source Separation Using Deep Learning Networks*. URL: https://se.mathworks.com/help/deeplearning/ug/cocktail-party-source-separation-using-deep-learning-networks.html. (accessed: 27.02.2023).

[31] Jacob N. Oppenheim and Marcelo O. Magnasco. "Human Time-Frequency Acuity Beats the Fourier Uncertainty Principle". In: *American Physical Society* (2013).

[32] Alfian Wijayakusuma, Davin Reinaldo Gozali, Anthony Widjaja, and Hanry Ham. "Implementation of Real-Time Speech Separation Model Using Time-Domain Audio Separation (TasNet) and Dual-Path Recurrent Neural Network (DPRNN)". In: *ScienceDirect: Procedia Computer Science* (2020).

[33] Jingjing Chen, Qirong Mao, and Dong Liu. "Dual-Path Transformer Network: Direct Context-Aware Modeling for End-to-End Monaural Speech Separation". In: *arXiv* (2020).

[34] Neil Zeghidour and David Grangier. "Wavesplit: End-to-End Speech Separation by Speaker Clustering". In: *IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 29* (2021).

[35] John S. Garofolo, David Graff, Doug Paul, and David Pallett. *CSR-I (WSJ0) Complete*. URL: https://catalog.ldc.upenn.edu/LDC93s6an (visited on 05/24/2023).

[36] Anna Josefine Sørensen, Michal Fereczkowski, and Ewen N MacDonald. *Task dialog by native-Danish talkers in Danish and English in both quiet and noise.* Last accessed 10 May 2023. 2018. URL: https://doi.org/10.5281/zenodo.1204951.

[37] John R. Hershey, Jonathan Le Roux, Shinji Watanabe, and Bret Harsham. *Single-Channel Multi-Speaker Separation using Deep Clustering*. URL: https://www.merl.com/demos/deep-clustering (visited on 05/24/2023).

[38] Robert R. Sokal and Charles D. Michener. *A Statistical Method For Evaluating Systematic Relationships*. Vol. 28. The University of Kansas Science Bulletin, 1958, pp. 1409–1438.

[39] Zheng-Hua Tan, Achintya kr. Sarkar, and Najim Dehak. "rVAD: An Unsupervised Segment-Based Robust Voice Activity Detection Method". In: *ScienceDirect: Computer Speech & Language 59* (2019).

[40] Dong Yu, Morten Kolbæk, Zheng-Hua Tan, and Jesper Jensen. "Permutation invariant training of deep models for speaker-independent multi-talker speech separation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2017).

[41] Poul Hoang, Zheng-Hua Tan, Thomas Lunner, Jan Mark de Haan, and Jesper Jensen. "Maximum Likelihood Estimation of the Interference-Plus-Noise Cross Power Spectral Density Matrix for Own Voice Retrieval". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020).

[42] Yochai Yemini, Ethan Fetaya, Haggai Maron, and Sharon Gannot. "Scene-Agnostic Multi-Microphone Speech Dereverberation". In: *arXivg* (2020).

Appendices

A Example Loss graphs

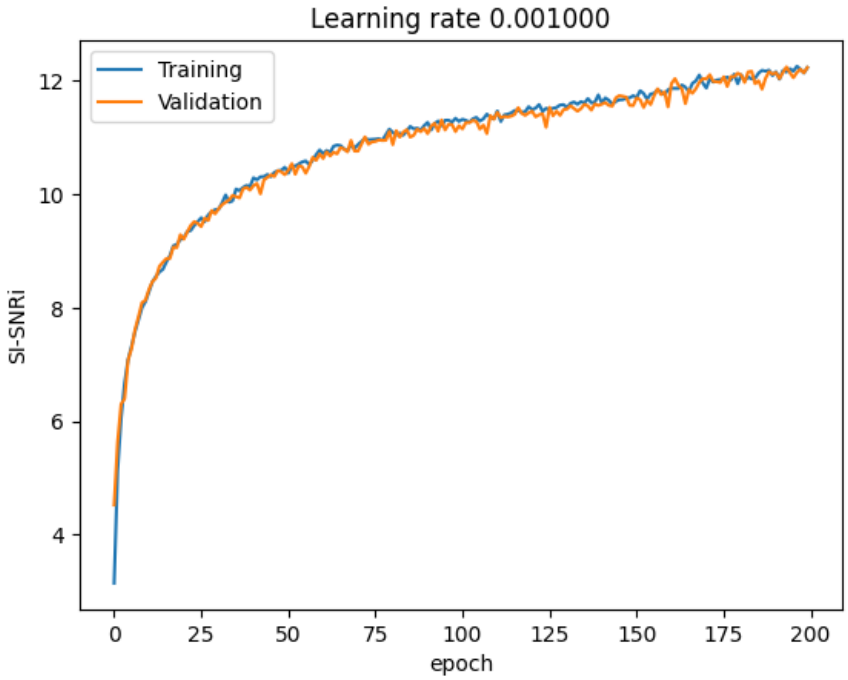


Figure A.1: Causal TasNet 2-speaker with Dynamic Mix.

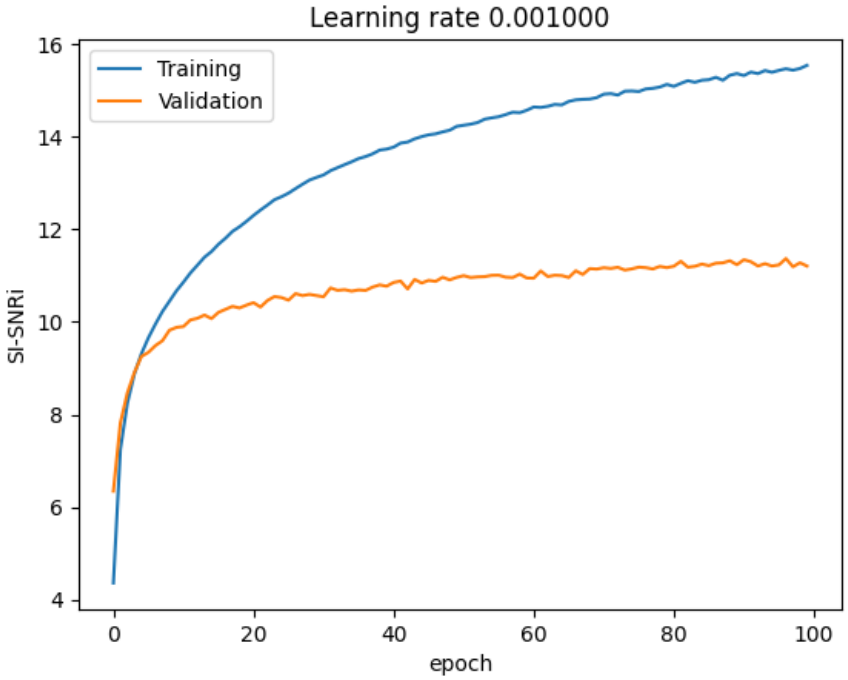


Figure A.2: Causal TasNet 2-speaker.

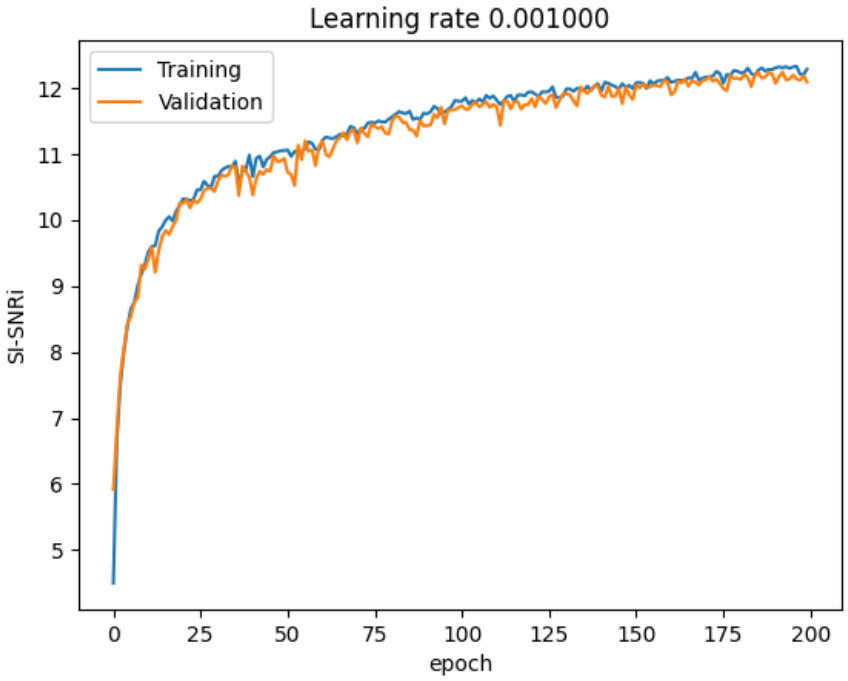


Figure A.3: Non-causal TasNet 2-speaker with Dynamic Mix.

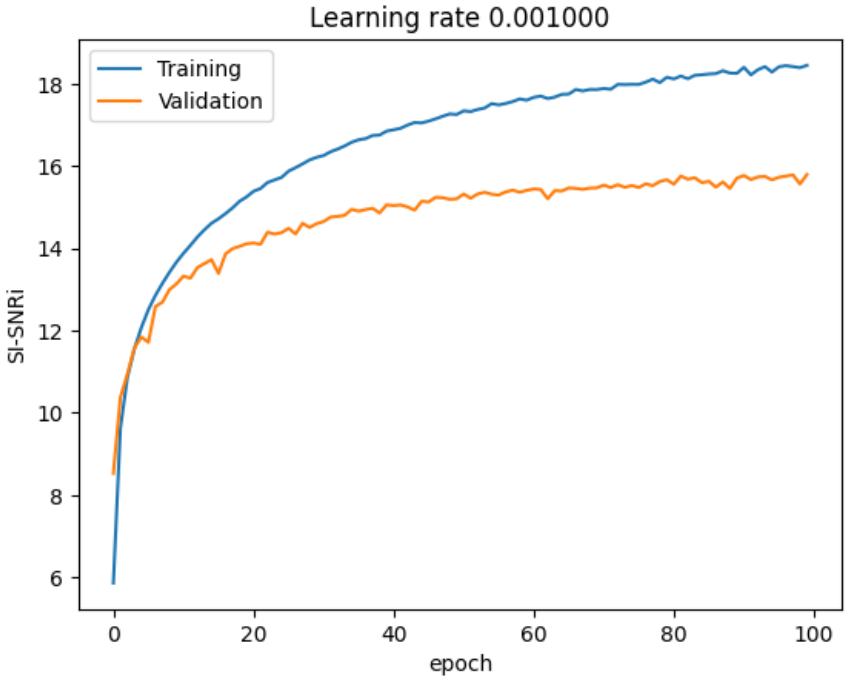


Figure A.4: Non-causal TasNet 2-speaker.

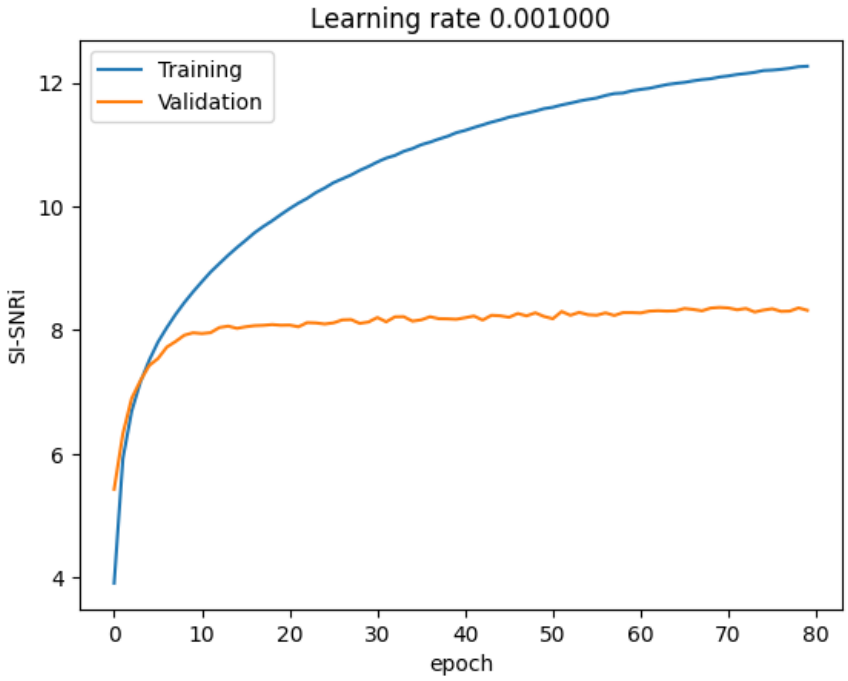


Figure A.5: Causal TasNet 3-speaker.

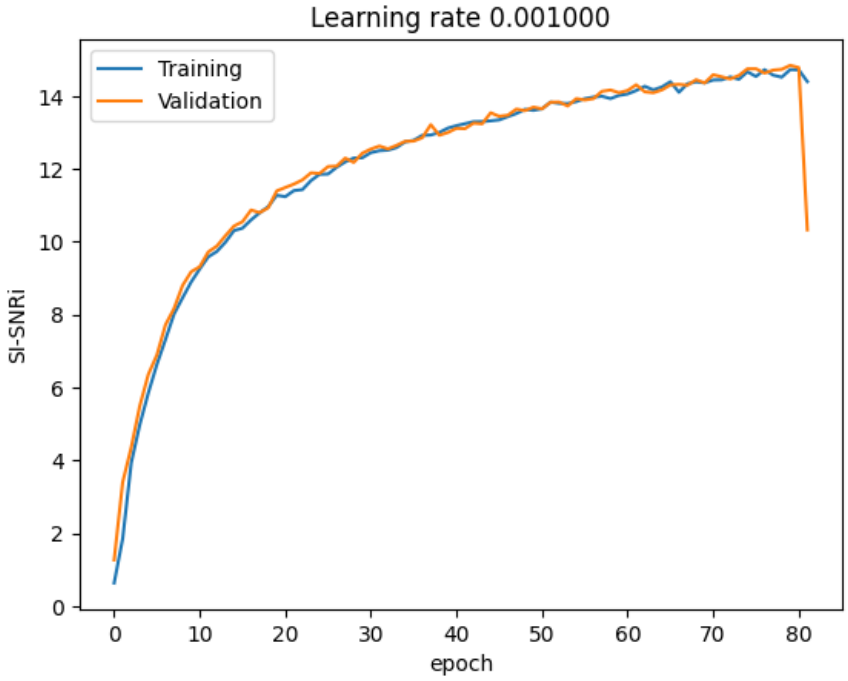


Figure A.6: Causal DPRNN 2-speaker with Dynamic Mix.

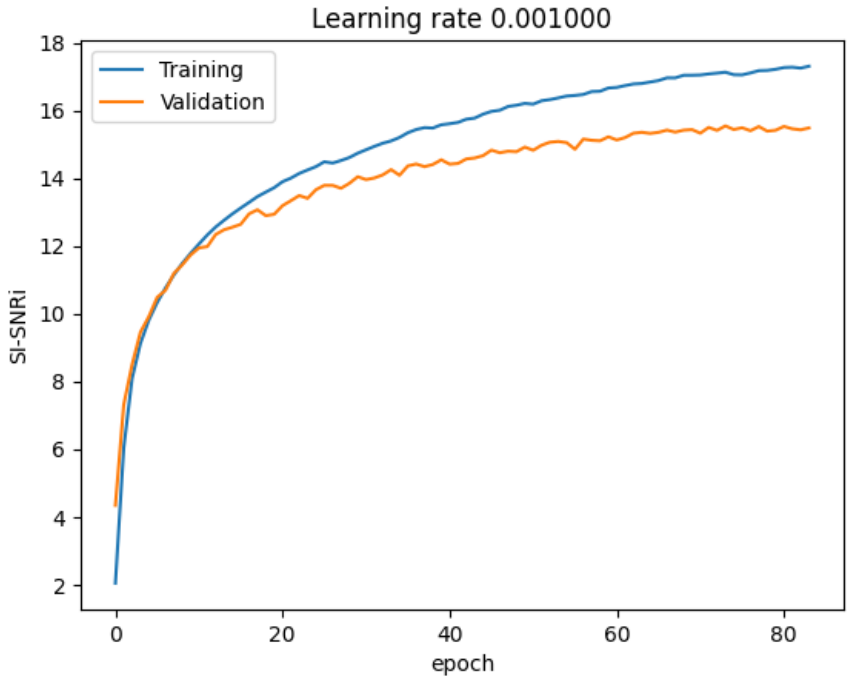


Figure A.7: Causal DPRNN 2-speaker.

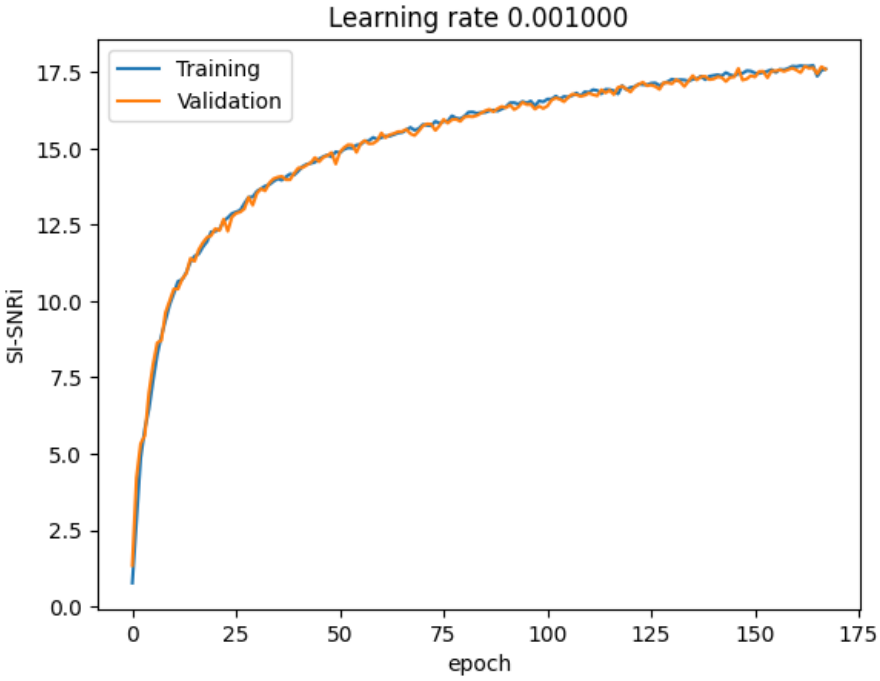


Figure A.8: Non-causal DPRNN 2-speaker with Dynamic Mix.

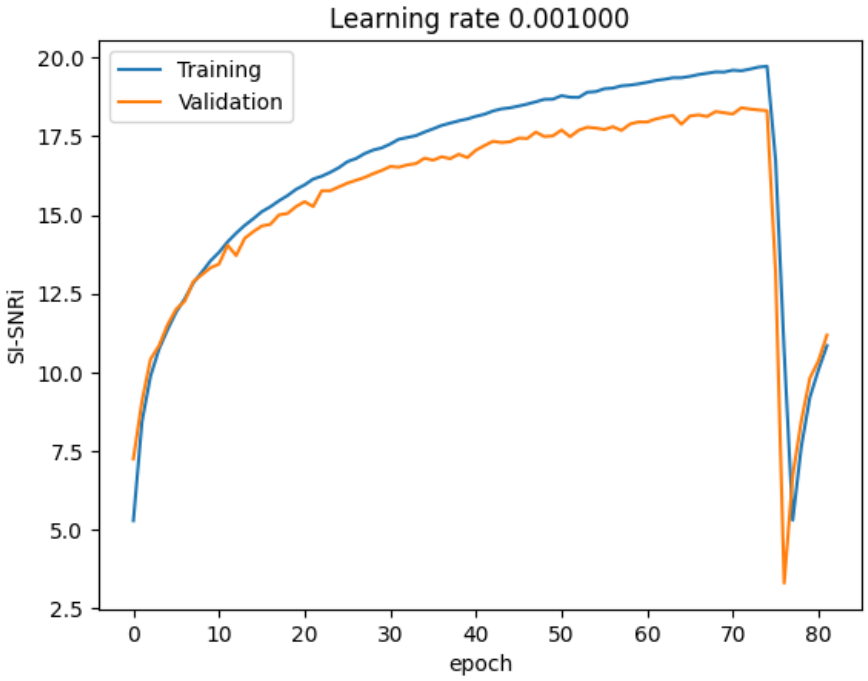


Figure A.9: Non-causal DPRNN 2-speaker.

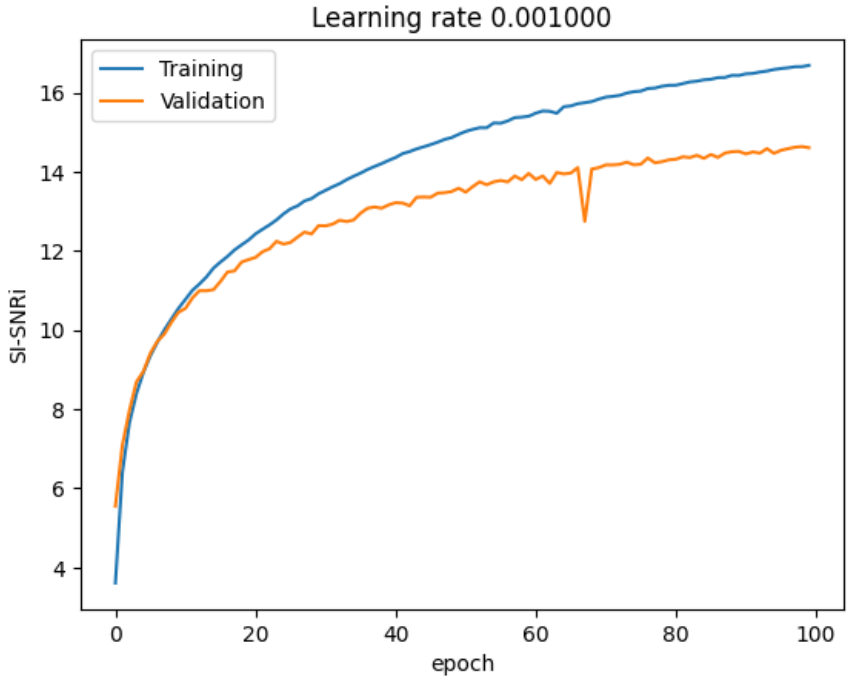


Figure A.10: Causal DPRNN 3-speaker.

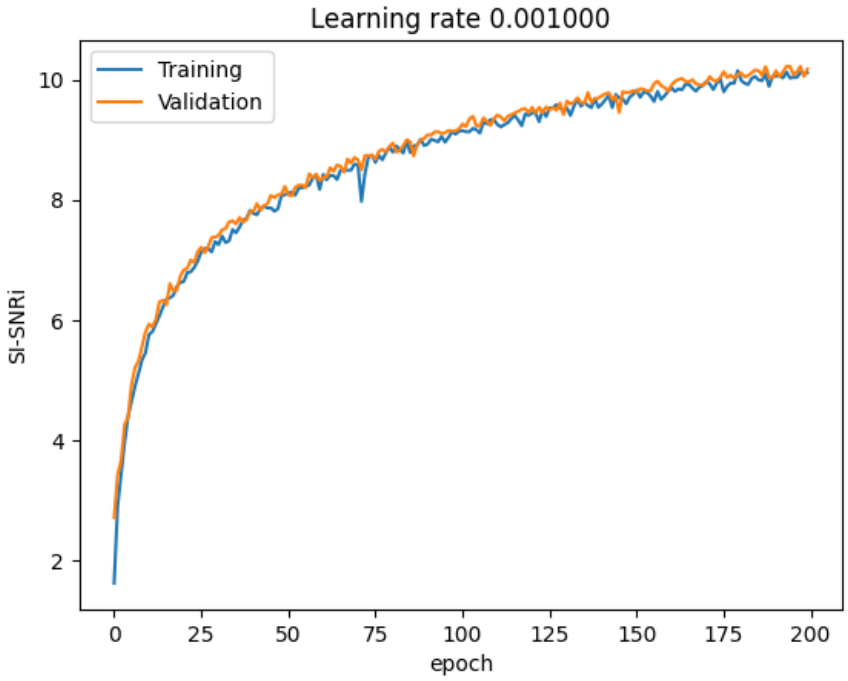


Figure A.11: Causal DPRNN 4-speaker with Dynamic Mix.

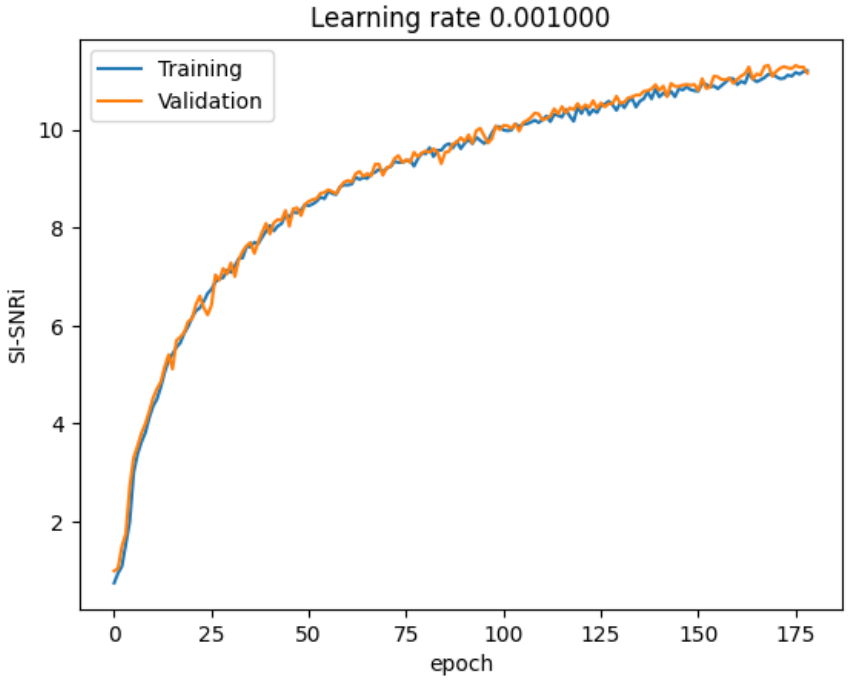


Figure A.12: Non-causal DPRNN 4-speaker with Dynamic Mix.