UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

FAKULTÄT FÜR
ELEKTROTECHNIK,
INFORMATIK UND
MATHEMATIK

Robust multi-channel speech recognition with neural network supported statistical beamforming

Von der Fakultät für Elektrotechnik, Informatik und Mathematik der Universität Paderborn

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegte Dissertation

von

M.Sc. Jahn Heymann

Erster Gutachter: Prof. Dr.-Ing. Haeb-Umbach
Zweiter Gutachter: Priv.-Doz. Dr. rer. nat. Ralf Schlüter

Tag der mündlichen Prüfung: 02.12.2020

Paderborn 2020
Diss. EIM-E/355

UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft

FAKULTÄT FÜR
ELEKTROTECHNIK,
INFORMATIK UND
MATHEMATIK

Acknowledgement

First and foremost I would like to thank my supervisor. Reinhold, you convinced me to quit my job and pursue a PhD instead, which, in retrospective, was absolutely the right thing to do at that time. During my more than five years as a member of your group you provided me with great freedom to explore ideas but also with the necessary guidance. Whenever I encountered an issue or needed advice you were there to help (and you showed great patience waiting for me to finally finish this work). Thank you.

Further I want to express my gratitude to my fellow team members, especially you, Lukas, but also Janek, Christoph, Jens, Thomas and Jörg. Without our fruitful discussions and your input this work would not have been possible. And for sure it would not have been so much fun without the regular coffee breaks and the conference visits.

I also want to thank the DFG for funding most of the work, Google for the two very educational and interesting internships and an additional year of funding, the PC^2 for providing the necessary computational resources and my sister Britta for proof reading this work.

Finally a big thank you goes out to my wife, Lisa, and my family for all your support and pushing me to finally finish this work.

Abstract

Automatic speech recognition is a crucial component for voice centric human-machine interfaces and has seen large improvements in terms of accuracy in recent years. These gains were largely driven by the switch to models based on deep neural networks which are able to exploit vast amounts of training data. But despite all improvements, recognition of far-field speech in noisy environments remains challenging.

In these far-field scenarios, spatial cues can help to improve the recognition performance by amplifying the target speech signal and dampening other interfering sources. Commonly this is achieved by classical signal processing and so-called beamforming which exploits time differences between the individual signals of multiple microphones to amplify signals from a certain region of space while suppressing signal from other directions. In this work, we leverage powerful neural networks which have been applied successfully to speech recognition and also to the signal processing component. But instead of replacing one model with another, we combine them and support statistical beamformers with a neural network to get the best of both worlds. We show that this setup can reduce the recognition error rates by more than half on two benchmarking datasets compared to a single-channel baseline. We then successively extend this system by reducing the latency to a frame-online operating mode, removing the need for simulated parallel data and optimizing the signal processing component jointly with the acoustic model.

Zusammenfassung

Die automatische Spracherkennung ist eine wichtige Komponente für eine sprachgestützte Mensch-Maschine Kommunikation und hat wurde in den letzten Jahre signifikant verbessert. Diese Verbesserungen sind größtenteils auf die Verwendung von tiefen Neuronalen Netzwerken zurückzuführen, welche in der Lage sind, große Menge an Trainingsdaten zu nutzen und davon zu profitieren. Dennoch ist die Erkennung von Sprache aus größeren Entfernungen vor allem in lauten Umgebungen immer noch problematisch und fehlerbehaftet.

In diesen Fernfeld-Szenarien können räumliche Merkmale die Erkennungsleistung steigern, indem sie die Sprache des zu erkennenden Sprechers verstärken und andere Umgebungsgeräusche dämpfen. In der Regel wird hierzu klassische Signalverarbeitung in Form des so genannten statistischen Beamformings verwendet. Dieses nutzt die zeitlichen Differenzen zwischen den einzelnen Signalen bei einer Aufnahme mit mehreren Mikrofonen, um Signale aus einer bestimmten räumlichen Richtung zu verstärken und die aus anderen Richtungen zu dämpfen.

In dieser Arbeit wird die Nutzung leistungsstarker Neuronaler Netzwerke, welche bereits erfolgreich in der Spracherkennung angewendet werden, auch für die Signalverarbeitungskomponente erforscht. Anstatt jedoch nur ein Modell durch ein anderes zu ersetzen, werde diese kombiniert und statistisches Beamforming mit Neuronalen Netzen unterstützt, um das Beste aus beiden Welten zu vereinen. Wir zeigen, dass diese Kombination die Erkennungsfehler, verglichen mit der einkanaligen Baseline, auf zwei Testdatensätzen mehr als halbiert. Darauf aufbauend, erweitern wir dieses System Schritt für Schritt, indem wir die Latenz auf ein Frame reduzieren, die Abhängigkeit von parallelen simulierten Daten entfernen und die Signalverarbeitungskomponente gemeinsam mit der Spracherkennungskomponente trainieren.

Contents

1 Introduction

In April 2011 Apple introduced Siri, the first commercially available digital personal assistant with a voice interface. Three years later, Amazon presented the Amazon Echo, the first smart speaker. A smart speaker is a device which allows the user to interact with a digital personal assistant from anywhere in a room just by using his voice. This new product category quickly gained traction and other major tech companies like Google (Google Home) and Apple (HomePod) released similar devices in the following years. By the end of 2020, 75 % of the (english speaking) households will own at least one smart speaker according to a recent report[1]. Apart from smart speakers, other devices such as televisions are now also adapting a voice centric interface.

One key enabling technology for such devices is a solid automatic speech recognition (ASR) system. ASR aims at recognizing the spoken words from an audio recording. While the first ASR systems in the 1960s were only able to recognize a few standalone words (e.g. the IBM Shoebox[2]), technological advances and a statistical modeling view allowed for systems usable for dictation tasks in the late 1990s. Supported by the drastically increased computational power and a vast amount of transcribed audio data, the usage of deep neural networks (DNNs) finally lead to accuracies high enough for a voice centric interface.

The other important technology is speech signal processing or speech enhancement. ASR becomes much more challenging once the distance between the speaker and the microphone increases. In a typical smart speaker use case, the speaker is located a few meters away from the device which then not only captures the desired speech signal, but also the signal of other sound emitting sources and reverberations. The reverberations have a smearing effect on the signal and the other sound sources introduce diverse non-stationary noise which cannot be removed easily. Those two effects have a detrimental impact on the performance of an ASR system. Speech signal processing can compensate for these interferences up to a certain extent and significantly reduces the error rates of an ASR system in a far-field scenario.

An especially effective processing technique for far-field scenarios is spatial filtering or acoustic beamforming. In fact, it has been considered as a front-end processing technique for ASR for many years. As early as 1990 Compernolle et al. showed that significant word error rate (WER) improvements are achievable by acoustic beamforming [Com+90]. Today, almost all smart speakers have multiple microphones – a prerequisite for beamforming which is based on the time differences between the

[1] https://about.ads.microsoft.com/en-us/insights/2019-voice-report
[2] https://en.wikipedia.org/wiki/IBM_Shoebox

recorded signal caused by different lengths of the sound propagation paths.

Classically, speech signal processing is based on signal statistics and distributions to model these statistics. However, to keep those models tractable, simplifying assumptions like temporal independence are made which might not hold in practice and result in a rather crude approximation of the real underlying distribution. On the other hand, recent progress in acoustic modeling shows that DNNs are very well suited to handle and classify speech signals. In recent years, a tendency to replace statistical models with a DNN can be observed. And indeed, many of those publications demonstrate oftentimes drastically improved performance by doing so.

So should statistical models for speech signal processing be just abandoned and replaced by neural networks? In this thesis we argue for a different approach. Instead of replacing one with the other, we combine them in an attempt to preserve the best of both worlds: the classification accuracy of a neural network and the generalization qualities of a statistical model. We especially focus on beamforming with the goal to exploit spatial information to improve the accuracy of an ASR system in challenging environments.

This work is structured as follows. In the first three chapters, we provide the necessary theoretical background and review existing approaches. In particular, Ch. 2 quickly recaps the basic building blocks of a neural network and then focuses on the gradient computation required to train the model. The next chapter (Ch. 3) then explains the foundations of an ASR and discusses ways to utilize multi-channel data for the presented model. This then brings us to classical speech signal processing in Ch. 4 where we first model the physical signal and then, based on this model, focus on statistical beamforming and dereverberation. That concludes the background part.

Next, we present the two datasets which will be used in this thesis to evaluate proposed models. Results from the literature are summarized and serve as a baseline for comparison in later chapters. Ch. 6 outlines the main contributions of this work and shows how it relates to the publications which have been made over the course of the PhD program.

The second half of this thesis then presents these contributions and discusses them in detail. Starting with the main contribution, a neural network supported beamformer, in Ch. 7, we extend this idea in the following chapters. In Ch. 8 we present a low-latency variant of the system and Ch. 9 discusses a way to train it in an unsupervised fashion. Finally, in Ch. 10 we remove the distinction between signal processing and speech recognition with a joint model and analyze it thoroughly. We opt for directly evaluating the presented model and discuss the results in each of the chapters individually rather than presenting a single big evaluation.

At the end of each chapter we give a short summary of the main takeaways and conclude this thesis with an overall summary in Ch. 11.

2 Neural networks

The first forms of a neural network (NN) were already described in the 1950s (e.g. [Ros58]) and first promising results were reported on various tasks in the late 1980s and 1990s (see [Sch15] Sec. 5.6 onwards for numerous examples). However, NNs were mainly a research topic while applications were dominated by statistical modeling based approaches with strong priors and hand crafted features. This changed dramatically with the ability to train *deep* networks, i.e., networks that consists of several layers and can easily have millions of parameters. Another key factor enabling the use of NNs was the availability of large amounts of data and drastically increased computational power, especially with the use of powerful graphics processing units (GPUs) or more recently also specialized hardware [WB+19]. The probably final breakthrough was achieved in 2012 when a NN-based system [KSH12] won the Large Scale Visual Recognition Challenge [Rus+15] by a large margin.

NNs are particularly good at classifying high dimensional sensory data, outperforming all previous approaches in applications like image recognition. In this work we will also make extensive use of NNs, specifically to process and classify speech signals. Due to their current popularity, new network architectures, applications and also methods are published nearly on a daily basis. In this chapter we therefore only shortly introduce the very basics of NNs and the most common layers in 2.1. Otherwise, we solely focus on the training of NNs with backpropagation (Sec. 2.2.2), especially with the concept of computational graphs which allow for automatic differentiation (Sec. 2.2.3). With the help of the Wirtinger calculus (Sec. 2.2.4) this will enable us to backpropagate through complex-valued signal processing algorithms later in this work.

A more detailed overview of the history of NNs and current developments is given in [Sch15]. For further details and technical background we refer the reader to recent books (e.g. [GBC16]).

2.1 Layers

Layers can be viewed as the building blocks of a NN. For example, a typical network for ASR consists of 5 – 10 consecutive layers, but the number of layers might also well exceed 100 for application such as image recognition [He+16a]. The following reviews the three basic types of layers which we will use for our networks in this work.

2.1.1 Fully connected layers

The fully connected (or feedforward (FF) or dense) layer might be considered as the most basic but also the most general layer. Its naming originates from the fact that each input is connected to each output with a learnable weight. It consists of an affine transformation with learnable parameters followed by a non-linearity. Given an input vector $\mathbf{x}$, the output $\mathbf{y}$ of the layer is computed as

$$\mathbf{y} = \mathrm{f}(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{2.1}$$

with the non-linearity $\mathrm{f}(\cdot)$, the learnable projection (or weight) matrix $\mathbf{W}$ and the likewise learnable bias $\mathbf{b}$. Typical non-linearities include the tanh function family (including sigmoid) and nowadays mostly rectified linear units (ReLUs) and derived variants. The latter ones promise a better convergence due to their mostly linear operating point which helps gradient propagation [NH10].

2.1.2 Convolutional layers

For data where features are independent of the spatial location they appear in (e.g. for images), sharing the weights for different parts of the input leads to a more effective use of the parameters [GBC16]. Convolutional neural networks (CNNs) perform an N-dimensional convolution on the input with learnable filters. The input, filters and output can have multiple channels. For sensory input this can, e.g., correspond to the color channels of an image. Each filter has the same number of channels as the input whereas the number of channels of the output is determined by the number of filters, i.e., each filter produces its own output, also referred to as feature map. More specifically, the 2-dimensional convolutional layer we will use in this work computes C_{out} two dimensional feature maps $\mathbf{Y}_{c_{\text{out}}}$ from an input[1] $\mathbf{A} \in \mathbb{R}^{C_{\text{in}} \times X \times Y}$ with C_{in} channels using learnable filters $\mathbf{W}_{c_{\text{out}}} \in \mathbb{R}^{C_{\text{in}} \times F_x \times F_y}$ as[2]

$$\mathbf{Y}_{c_{\text{out}}}(i,j) = \sum_{c=0}^{C_{\text{in}}-1} \sum_{x=0}^{F_x-1} \sum_{y=0}^{F_y-1} \mathbf{X}(c, i+x, j+y)\mathbf{W}_{c_{\text{out}}}(c,x,y). \tag{2.2}$$

The outputs $\mathbf{Y}_{c_{\text{out}}}$ for each filter are then stacked to obtain the final output $\mathbf{Y} \in \mathbb{R}^{C_{\text{out}} \times X \times Y}$. For dimensionality reduction it is also possible to use striding, i.e., to advance the filter with a step size > 1 along one or more dimensions, effectively subsampling the input.

2.1.3 Recurrent layers

For time series like speech signals, the value at the current time step is usually correlated with values from past time steps and the output does not only depend on the current input but also on previous inputs. Recurrent layers model this dependency by recursively

[1] In this work we pad the input with zeros such that the resulting feature map has the same size as the input.

[2] Although it is called convolution, the layer actually computes a cross-correlation. However, since the filters are learned, one can also interpret it as a convolution learning a flipped filter.

conditioning the output at one time step on (a projection) of the previous time step. Given a sequence of input vectors $\mathbf{x}_{0:T}$, the output vector $\mathbf{y}_t$ at time step t of a recurrent layer with parameters $\boldsymbol{\Theta}$ is computed as

$$\mathbf{y}(t) = \mathcal{F}\left(\mathbf{x}(t), \mathbf{y}(t-1); \boldsymbol{\Theta}\right), \tag{2.3}$$

whereas the most basic recurrent neural networks (RNNs) compute

$$\mathbf{y}(t) = \gamma\left(\mathbf{W}_x \mathbf{x}(t) + \mathbf{W}_y \mathbf{y}(t-1) + \mathbf{b}\right) \tag{2.4}$$

with some non-linearity γ.

Since the same parameter matrix $\mathbf{W}_y$ is used for all time steps, these layers especially suffer from what is known as the vanishing or exploding gradient problem [Hoc91].

This problem can be alleviated using more complex architectures which provide a more direct path for the gradient to propagate. One effective and also popular architecture are long short-term memory (LSTM) networks [HS97] which compute

$$\begin{aligned}
\mathbf{i}(t) &= \sigma\left(\mathbf{W}_{xi}\mathbf{x}(t) + \mathbf{W}_{yi}\mathbf{y}(t-1) + \mathbf{b}_i\right) \\
\mathbf{f}(t) &= \sigma\left(\mathbf{W}_{xf}\mathbf{x}(t) + \mathbf{W}_{yf}\mathbf{y}(t-1) + \mathbf{b}_f\right) \\
\mathbf{o}(t) &= \sigma\left(\mathbf{W}_{xo}\mathbf{x}(t) + \mathbf{W}_{yo}\mathbf{y}(t-1) + \mathbf{b}_o\right) \\
\mathbf{g}(t) &= \tanh\left(\mathbf{W}_{xg}\mathbf{x}(t) + \mathbf{W}_{yg}\mathbf{y}(t-1) + \mathbf{b}_g\right) \\
\mathbf{c}(t) &= \mathbf{f}(t) \circ \mathbf{c}(t-1) + \mathbf{i}(t) \circ \mathbf{g}(t) \\
\mathbf{h}(t) &= \mathbf{o}(t) \circ \tanh(\mathbf{c}(t))
\end{aligned} \tag{2.5}$$

and will be used extensively in this work. Recurrent neural networks can work in an uni-directional or bi-directional fashion. While the former can only exploit past input information, the latter consists of two recurrent networks where one operates on the reversed feature sequence. This allows to also take into account future context but introduces the requirement to already have the complete time series available for processing.

2.2 Training

NNs are parametric models whose many parameters are *learned* from data points during an explicit training phase. The by far most common and also very generic approach to train a NN is by using stochastic gradient descent (SGD). In the following we will first describe the update rule for the parameters and then discuss in detail how the necessary gradients can be computed efficiently.

2.2.1 Update rule

In the supervised setup, consider a dataset $\mathcal{D}$ of N data points $(\mathbf{x}_n, \mathbf{y}_n)$, a NN with parameters $\boldsymbol{\theta}$ that estimates $\hat{\mathbf{y}}_n = \mathcal{F}(\mathbf{x}_n; \boldsymbol{\theta})$ and a per-example loss function $\mathcal{L}_{\text{example}}(\hat{\mathbf{y}}_n, \mathbf{y}_n)$ which allows to estimate the overall loss $\mathcal{L}_{\text{total}}(\mathcal{D}; \boldsymbol{\theta})$ as

$$\mathcal{L}_{\text{total}}(\mathcal{D}; \boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}_{\text{example}}(\mathcal{F}(\mathbf{x}_n; \boldsymbol{\theta}), \mathbf{y}_n). \tag{2.6}$$

Besides the popular per-example loss functions like the mean squared error (MSE) for, e.g., regression tasks and the cross-entropy (CE) for classification tasks, there are numerous task specific loss functions. One example in the ASR context is the connectionist temporal classification (CTC) loss which will be discussed in Sec. 3.1.2. SGD aims to minimize the overall loss $\mathcal{L}_{total}$ by iteratively first estimating a gradient on a small random subset $\mathcal{B} \subset \mathcal{D}$ of the data called (mini-) batch

$$\frac{\partial \mathcal{L}_{\text{total}}(\mathcal{D};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \approx \frac{\partial \mathcal{L}_{\text{total}}(\mathcal{B};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{1}{|\mathcal{B}|} \sum_{(\mathbf{x}_m,\mathbf{y}_m)\in\mathcal{B}} \frac{\partial \mathcal{L}_{\text{example}}(\mathcal{F}(\mathbf{x}_m;\boldsymbol{\theta}),\mathbf{y}_m)}{\partial \boldsymbol{\theta}} \tag{2.7}$$

and then updating the parameters with a learning rate μ

$$\boldsymbol{\theta}^{(\text{new})} = \boldsymbol{\theta}^{(\text{old})} - \mu \frac{\partial \mathcal{L}_{\text{total}}(\mathcal{B};\boldsymbol{\theta}^{(\text{old})})}{\partial \boldsymbol{\theta}^{(\text{old})}}. \tag{2.8}$$

For the scope of this thesis, the important point to realize is that in order to train a NN, we need the gradient of a differentiable loss function w.r.t. the parameters of the network. There is a multitude of literature on SGD and its convergence properties and we refer the reader to this for more details. A good starting point with a focus on deep learning is Sec. 4.3+ in [GBC16]. In this work, we will mostly use a modified update rule called ADAM [KB14] which also takes into account adaptive estimates of lower-order moments to modify the effective learning rate for each parameter.

2.2.2 Backpropagation

For NNs the gradient w.r.t. to each individual parameter can be calculated efficiently by exploiting the chain rule as we will see in the following.

First, consider the scalar chain rule. Given N nested functions[3] $f_1, f_2, \ldots, f_N : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ parameterized by some θ_i, $y = f_N(h_{N-1};\theta_{N-1})$, $h_{i+1} = f_{i+1}(h_i;\theta_i)$ and $h_0 = x$ with $x, y \in \mathbb{R}$, the derivative $\frac{\partial y}{\partial \theta_i}$ can be written as

$$\frac{\partial y}{\partial \theta_i} = \frac{\partial y}{\partial h_{N-1}} \frac{\partial h_{N-1}}{\partial h_{N-2}} \cdots \frac{\partial h_{i+1}}{\partial \theta_i} = \frac{\partial y}{\partial h_{i+1}} \frac{\partial h_{i+1}}{\partial \theta_i}. \tag{2.9}$$

In order to calculate $\frac{\partial y}{\partial \theta_{i-1}}$ we can reuse the result for $\frac{\partial y}{\partial h_{i+1}}$ already calculated for $\frac{\partial y}{\partial \theta_i}$ by exploiting the chain rule

$$\frac{\partial y}{\partial \theta_{i-1}} = \frac{\partial y}{\partial h_{N-1}} \frac{\partial h_{N-1}}{\partial h_{N-2}} \cdots \frac{\partial h_{i+1}}{\partial h_i} \frac{\partial h_i}{\partial \theta_{i-1}} = \frac{\partial y}{\partial h_{i+1}} \frac{\partial h_{i+1}}{\partial h_i} \frac{\partial h_i}{\partial \theta_{i-1}}. \tag{2.10}$$

Looking at the layers briefly described in the previous section, it becomes apparent that NNs operate on vectors and matrices rather than scalar values. To generalize the operations, we will consider tensors here. A tensor is D-dimensional grid of scalar values. Special cases include a vector ($D = 1$) or a matrix ($D = 2$). A D-dimensional vector $\mathbf{i} \in \mathbb{Z}^D$ is used to index the individual elements within the grid. For a matrix, e.g., the vector $\mathbf{i} = [i_1, i_2]$ indexes the element in the i_1-th row and i_2-th column. To obtain a

[3]The functions must be at least point-wise differentiable w.r.t. to their inputs

chain rule similar to the scalar case, we have to introduce a generalized Jacobian and a generalized inner product.

Given a tensor

$$\mathbf{A} \in \mathbb{R}^{N_1^{(A)} \times N_2^{(A)} \times \ldots N_{D_a}^{(A)}} \tag{2.11}$$

and a function f mapping it to a tensor

$$\mathbf{B} \in \mathbb{R}^{N_1^{(B)} \times N_2^{(B)} \times \ldots N_{D_b}^{(B)}} \tag{2.12}$$

with $\mathbf{B} = f(\mathbf{A})$, we define the generalized Jacobian $\frac{\hat{\partial}\mathbf{B}}{\hat{\partial}\mathbf{A}}$ as a generalized matrix with the dimensionality

$$\left[N_1^{(B)} \times N_2^{(B)} \times \ldots N_{D_b}^{(B)}\right] \times \left[N_1^{(A)} \times N_2^{(A)} \times \ldots N_{D_a}^{(A)}\right]. \tag{2.13}$$

Note that there are two groups and we use $\hat{\partial}$ to distinguish the generalized form from the partial derivative which is denoted using ∂. The first group corresponds to what can be interpreted as the rows of a generalized matrix and has the same dimension as $\mathbf{B}$, whereas the second group corresponds to the columns of a generalized matrix and has the same dimension as $\mathbf{A}$. This generalized matrix is indexed by two vectors $\mathbf{i} \in \mathbb{Z}^{D_a}$ and $\mathbf{j} \in \mathbb{Z}^{D_b}$ and its elements are

$$\left(\frac{\hat{\partial}\mathbf{B}}{\hat{\partial}\mathbf{A}}\right)_{\mathbf{j},\mathbf{i}} = \frac{\partial\left(\mathbf{B}\right)_{\mathbf{j}}}{\partial\left(\mathbf{A}\right)_{\mathbf{i}}} \tag{2.14}$$

Note that $\left(\mathbf{B}\right)_{\mathbf{j}}$ and $\left(\mathbf{A}\right)_{\mathbf{i}}$ are scalar values.

We can also define a product between generalized matrices

$$\mathbf{X} \in \mathbb{R}^{\left[N_1^{(X_1)} \times \cdots \times N_{D_{x_1}}^{(X_1)}\right] \times \left[N_1^{(X_2)} \times \cdots \times N_{D_{x_2}}^{(X_2)}\right]} \tag{2.15}$$

and

$$\mathbf{Y} \in \mathbb{R}^{\left[N_1^{(Y_1)} \times \cdots \times N_{D_{y1}}^{(Y_1)}\right] \times \left[N_1^{(Y_2)} \times \cdots \times N_{D_{y2}}^{(Y_2)}\right]} \tag{2.16}$$

as

$$\mathbf{XY} = \sum_{\mathbf{j}} \mathbf{X}_{:,\mathbf{j}} \mathbf{Y}_{\mathbf{j},:} = \mathbf{Z} \tag{2.17}$$

with

$$\mathbf{Z} \in \mathbb{R}^{\left[N_1^{(X_1)} \times \cdots \times N_{D_{x_1}}^{(X_1)}\right] \times \left[N_1^{(Y_2)} \times \cdots \times N_{D_{y2}}^{(Y_2)}\right]}. \tag{2.18}$$

Note that the dimension of the second group of the left operand must match the dimension of the first group of the right operand, i.e., $D_{x_2} = D_{y_1}$ and $N_k^{(X_2)} = N_k^{(Y_1)}$ $\forall k \in [1, D_{x_2}]$. For the special case where $D_{x_1} = D_{x_2} = D_{y_1} = D_{y_2} = 1$, i.e., where $\mathbf{X}$ and $\mathbf{Y}$ are matrices, this operation corresponds to the matrix multiplication. Further, a generalized scalar can be represented as $x \in \mathbb{R}^{[\,]\times[\,]}$, a generalized row vector as

$\mathbf{x} \in \mathbb{R}^{\left[N_1^{(X_1)} \times \cdots \times N_{D_{x_1}}^{(X_1)}\right] \times \left[\ \right]}$ and a generalized column vector as $\mathbf{x} \in \mathbb{R}^{\left[\ \right] \times \left[N_1^{(X_2)} \times \cdots \times N_{D_{x_2}}^{(X_2)}\right]}$. The later two can be again seen as tensors.

With these definitions, we can now formulate the chain rule for tensors. Suppose we have the functions[4]

$$f_1, f_2, \ldots, f_K : \mathbb{R}^{N_1^{(f_k)} \times \cdots \times N_{D_{f_{k_{\text{in}}}}}^{(f_k)}} \to \mathbb{R}^{N_1^{(f_k)} \times \cdots \times N_{D_{f_{k_{\text{out}}}}}^{(f_k)}}, D_{f_{k_{\text{out}}}} = D_{f_{(k+1)_{\text{in}}}} \tag{2.19}$$

with $\mathbf{y} = f_K(\mathbf{h}_{K-1})$, $\mathbf{k}_{i+1} = f_{i+1}(\mathbf{h}_i)$ and a scalar $J = f_J(\mathbf{y})$ resulting from a loss function f_J. The generalized Jacobian $\frac{\hat{\partial} J}{\hat{\partial} \mathbf{h}_i}$ can be calculated as

$$\frac{\hat{\partial} J}{\hat{\partial} \mathbf{h}_i} = \frac{\hat{\partial} J}{\hat{\partial} \mathbf{y}} \frac{\hat{\partial} \mathbf{y}}{\hat{\partial} \mathbf{h}_{K-1}} \frac{\hat{\partial} \mathbf{h}_{K-1}}{\hat{\partial} \mathbf{h}_{K-2}} \cdots \frac{\hat{\partial} \mathbf{h}_{i+1}}{\hat{\partial} \mathbf{h}_i} = \frac{\hat{\partial} J}{\hat{\partial} \mathbf{h}_{i+1}} \frac{\hat{\partial} \mathbf{h}_{i+1}}{\hat{\partial} \mathbf{h}_i}. \tag{2.20}$$

Note that the dimensions are

$$\frac{\hat{\partial} J}{\hat{\partial} \mathbf{h}_i} \in \mathbb{R}^{\left[\ \right] \times \left[N_1^{(f_i)} \times \cdots \times N_{D_{f_{i_{\text{out}}}}}^{(f_i)}\right]}, \tag{2.21}$$

$$\frac{\hat{\partial} J}{\hat{\partial} \mathbf{h}_{i+1}} \in \mathbb{R}^{\left[\ \right] \times \left[N_1^{(f_{i+1})} \times \cdots \times N_{D_{f_{i+1_{\text{out}}}}}^{(f_{i+1})}\right]}, \tag{2.22}$$

and

$$\frac{\hat{\partial} \mathbf{h}_{i+1}}{\hat{\partial} \mathbf{h}_i} \in \mathbb{R}^{\left[N_1^{(h_{i+1})} \times \cdots \times N_{D_{f_{i+1_{\text{out}}}}}^{(h_{i+1})}\right] \times \left[K_1^{(h_i)} \times \cdots \times K_{D_{f_{i_{\text{out}}}}}^{(h_i)}\right]}. \tag{2.23}$$

The generalized Jacobian offers a convenient way to write the chain rule, but a look at its size in Eq. 2.23 shows that it is costly to compute and to store in memory. In a practical setting, however, most of the values of the generalized Jacobian are zero as not all outputs depend on all inputs. So a naive implementation of the formulas above will not be very efficient and there is a lot of optimization potential by skipping the computations where we can analytically show that the gradient is zero. Nevertheless the generalized Jacobian allows us to treat the computation in a principled way and, as we will see in the following, also allows us to recover the lost efficiency.

2.2.3 Computational graphs and automatic differentiation

A slightly different way to look at a neural network is in the form of a computational graph. Each node of this graph corresponds to an atomic operation which, in its most general form, computes output tensors from input tensors. Example operations include, e.g., element-wise addition and multiplication but also matrix multiplication or more complex operations like matrix decompositions. A computational graph describes the connections between individual operations and allows for automatic differentiation (AD) (see, e.g., [Bay+18] for a survey). AD can be implemented in two different ways:

[4]For readability we omit the parameters as it is directly comparable to the scalar case.

in forward mode and in reverse mode. Roughly speaking, the forward mode calculates the gradient by computing the chain rule from right to left (or inner to outer), while the reverse mode computes the chain rule from left to right (or outer to inner). Most of the available software frameworks implement the reverse mode which we will quickly summarize in the following. For a more detailed introduction (also to the forward mode) see, e.g., [Bay+18].

In the reverse mode, the gradient for all nodes is computed by reversing the order of the computation and, starting from a final node (e.g. the scalar loss for a neural network), propagating the gradient w.r.t. an input of each node to the output of all nodes connected to this input. Consider a node mapping the input tensors $\mathbf{I}_1, \ldots, \mathbf{I}_N$ to the output tensors $\mathbf{O}_1, \ldots, \mathbf{O}_M$ with some function f. Given the gradients of a scalar loss J w.r.t. the M outputs $\frac{\hat{\partial} J}{\hat{\partial} \mathbf{O}_m}$, the node calculates the gradient for the n-th input as

$$\frac{\hat{\partial} J}{\hat{\partial} \mathbf{I}_n} = \sum_m \frac{\hat{\partial} J}{\hat{\partial} \mathbf{O}_m} \frac{\hat{\partial} \mathbf{O}_m}{\hat{\partial} \mathbf{I}_n} \tag{2.24}$$

during the backpropagation phase.

The specific gradient for each input is then passed to all nodes connected to the input, i.e. those nodes who provided the input values during the forward propagation. In this view, all parameters of a model can be seen as nodes with a state (the current parameter values) and no input.

The computational graph simplifies the implementation of the gradient calculation significantly. Instead of implementing the gradient w.r.t. all parameters manually, it is sufficient to implement all operations as individual nodes which can calculate the gradient w.r.t. to their inputs according to Eq. 2.24. These nodes can then be connected in an arbitrary way to achieve the desired functionality. The gradients w.r.t. the parameters are obtained in a principled way following the chain rule and backpropagating them through the (reverse) computational graph.

The computational graph and AD do not only simplify the implementation of a complex network, they also help to solve the issue with the shape of the generalized Jacobian and the associated computational efficiency. To see this, note that each node calculates the gradient of J w.r.t. to its inputs given the gradient of J w.r.t. its outputs. One key insight is that we can exploit the fact that not all outputs depend on all inputs and avoid implementing the generalized Jacobian by finding an analytical solutions for Eq. 2.24. This can best be seen for some concrete examples.

One class of operations that can be readily simplified are element-wise operation. Consider a node mapping an input tensor

$$\mathbf{A} \in \mathbb{R}^{N_1 \times N_2 \times \ldots N_D} \tag{2.25}$$

to a tensor

$$\mathbf{B} \in \mathbb{R}^{N_1 \times N_2 \times \ldots N_D} \tag{2.26}$$

with the same dimension by applying a function $f : \mathbb{R} \to \mathbb{R}$ to each element of $\mathbf{A}$

$$(\mathbf{B})_{\mathbf{i}} = f((\mathbf{A})_{\mathbf{i}}). \tag{2.27}$$

Let $\mathbf{G} \in \mathbb{R}^{N_1 \times N_2 \times \dots N_D}$ be a tensor with

$$(\mathbf{G})_{\mathbf{i}} = \frac{\partial (\mathbf{B})_{\mathbf{i}}}{\partial (\mathbf{A})_{\mathbf{i}}}. \tag{2.28}$$

Note that each element of $\mathbf{G}$ is just a scalar partial derivative. Then, Eq. 2.24 can be computed as

$$\frac{\hat{\partial} J}{\hat{\partial} \mathbf{A}} = \frac{\hat{\partial} J}{\hat{\partial} \mathbf{B}} \circ \mathbf{G} \tag{2.29}$$

where $\circ$ denotes an element-wise multiplication (also known as the Hadamard product) and $\frac{\hat{\partial} J}{\hat{\partial} \mathbf{B}}$ is the gradient of the loss J w.r.t. the output of the node. Note that the dimension of $\mathbf{G}$ and $\frac{\hat{\partial} J}{\hat{\partial} \mathbf{B}}$ is only

$$\mathbb{R}^{N_1 \times N_2 \times \dots N_D} \tag{2.30}$$

whereas the generalized Jacobian $\frac{\hat{\partial} \mathbf{B}}{\hat{\partial} \mathbf{A}}$ has the shape

$$\mathbb{R}^{[N_1 \times N_2 \times \dots N_D] \times [N_1 \times N_2 \times \dots N_D]}. \tag{2.31}$$

Another example is the matrix product between a matrix $\mathbf{A} \in \mathbb{R}^{N \times M}$ and a vector $\mathbf{b} \in \mathbb{R}^{M}$ resulting in the vector $\mathbf{c} \in \mathbb{R}^{N}$ with $\mathbf{c} = \mathbf{A}\mathbf{b}$. This projection is a very common operation for neural networks where the elements of the projection matrix $\mathbf{A}$ are learnable parameters[5]. In the computational graph view, the node has two inputs ($\mathbf{A}$ and $\mathbf{b}$) and one output $\mathbf{C}$. The gradient of a scalar loss J w.r.t. $\mathbf{A}$ given the gradient $\frac{\partial J}{\partial \mathbf{C}}$ is given by [Böd+17]

$$\frac{\partial J}{\partial \mathbf{A}} = \left(\frac{\partial J}{\partial \mathbf{C}} \right)^{\mathsf{T}} \mathbf{b} \tag{2.32}$$

and w.r.t. $\mathbf{b}$ by

$$\frac{\partial J}{\partial \mathbf{b}} = \mathbf{A}^{\mathsf{T}} \frac{\partial J}{\partial \mathbf{C}}. \tag{2.33}$$

Comparing the dimensions to the ones of the generalized Jacobians, the operands in both equations are smaller and thus require less memory and computations. However, the savings are not as big as the ones achieved in the previous example since here, a single output depends on multiple inputs, resulting in a denser generalized Jacobian.

For the general case of an arbitrary tensor operation, there is no generic way to find an efficient expression for Eq. 2.24. In the case where the node performs an operation $\mathbf{C} = f(\mathbf{A}, \mathbf{B})$ with $\mathbf{A} \in \mathbb{R}^{N_1^{(a)} \times N_2^{(a)}}$, $\mathbf{B} \in \mathbb{R}^{N_1^{(b)} \times N_2^{(b)}}$, $\mathbf{C} \in \mathbb{R}^{N_1^{(c)} \times N_2^{(c)}}$, however, one can find such an expression in a principle way [Gil08]. Suppose $\mathbf{A}$ and $\mathbf{B}$ depend on a scalar

[5] In practice, this operation is implemented as an inner product of two matrices since an additional batch dimension is introduced. This enables a better parallelization of the computation when working with mini-batches but the principle remains the same.

x themselves, i.e. $\mathbf{C}(x) = f(\mathbf{A}(x), \mathbf{B}(x))$. Further let $J = g(\mathbf{C})$. Then we can express $\frac{\partial J}{\partial x}$ as

$$\frac{\partial J}{\partial x} = \sum_{n,m} \frac{\partial J}{\partial \mathbf{C}_{n,m}} \frac{\partial \mathbf{C}_{n,m}}{\partial x} = \mathrm{tr}\left\{ \left(\frac{\partial J}{\partial \mathbf{C}} \right)^\mathsf{T} \frac{\partial \mathbf{C}}{\partial x} \right\}. \tag{2.34}$$

Note that we use ∂ here to explicitly distinguish from a generalized Jacobian where, e.g., we have not defined a transpose. Here, both derivatives are matrices. The forward mode AD is expressed by

$$\frac{\partial \mathbf{C}}{\partial x} = \frac{\partial f}{\partial \mathbf{A}} \frac{\partial \mathbf{A}}{\partial x} + \frac{\partial f}{\partial \mathbf{B}} \frac{\partial \mathbf{B}}{\partial x}, \tag{2.35}$$

where $\frac{\partial f}{\partial \mathbf{A}}$ and $\frac{\partial f}{\partial \mathbf{B}}$ respectively describe the sensitivity of the function towards a small perturbation of the inputs and have the same shape as their inputs [Gil08].

Inserting Eq. 2.35 into Eq. 2.34 and exploiting $\mathrm{tr}\{\mathbf{Y} + \mathbf{Z}\} = \mathrm{tr}\{\mathbf{Y}\} + \mathrm{tr}\{\mathbf{Z}\}$ yields

$$\frac{\partial J}{\partial x} = \mathrm{tr}\left\{ \left(\frac{\partial J}{\partial \mathbf{C}} \right)^\mathsf{T} \frac{\partial \mathbf{C}}{\partial \mathbf{A}} \frac{\partial \mathbf{A}}{\partial x} \right\} + \mathrm{tr}\left\{ \left(\frac{\partial J}{\partial \mathbf{C}} \right)^\mathsf{T} \frac{\partial \mathbf{C}}{\partial \mathbf{B}} \frac{\partial \mathbf{B}}{\partial x} \right\}. \tag{2.36}$$

At the same time

$$\frac{\partial J}{\partial x} = \mathrm{tr}\left\{ \left(\frac{\partial J}{\partial \mathbf{A}} \right)^\mathsf{T} \frac{\partial \mathbf{A}}{\partial x} \right\} + \mathrm{tr}\left\{ \left(\frac{\partial J}{\partial \mathbf{B}} \right)^\mathsf{T} \frac{\partial \mathbf{B}}{\partial x} \right\} = \frac{\hat{\partial} J}{\hat{\partial} \mathbf{A}} \frac{\hat{\partial} \mathbf{A}}{\hat{\partial} x} + \frac{\hat{\partial} J}{\hat{\partial} \mathbf{B}} \frac{\hat{\partial} \mathbf{B}}{\hat{\partial} x}. \tag{2.37}$$

Comparing the coefficients and comparing to our definition of the generalized Jacobian shows that [Gil08]

$$\begin{aligned} \frac{\partial J}{\partial \mathbf{A}} &= \left(\frac{\partial \mathbf{C}}{\partial \mathbf{A}} \right)^\mathsf{T} \frac{\partial J}{\partial \mathbf{C}} = \frac{\hat{\partial} J}{\hat{\partial} \mathbf{A}} \\ \frac{\partial J}{\partial \mathbf{B}} &= \left(\frac{\partial \mathbf{C}}{\partial \mathbf{B}} \right)^\mathsf{T} \frac{\partial J}{\partial \mathbf{C}} = \frac{\hat{\partial} J}{\hat{\partial} \mathbf{B}}. \end{aligned} \tag{2.38}$$

For $\mathbf{C} = f(\mathbf{A}, \mathbf{B})$ an efficient expression for $\frac{\hat{\partial} J}{\hat{\partial} \mathbf{A}}$ and $\frac{\hat{\partial} J}{\hat{\partial} \mathbf{B}}$ is found by determining the sensitivity (i.e. forward mode AD) and then rearranging $\frac{\partial J}{\partial x}$ such that

$$\frac{\partial J}{\partial x} = \mathrm{tr}\left\{ (\dots)^\mathsf{T} \frac{\partial \mathbf{A}}{\partial x} \right\} + \mathrm{tr}\left\{ (\dots)^\mathsf{T} \frac{\partial \mathbf{B}}{\partial x} \right\}. \tag{2.39}$$

A big advantage of the described procedure is that it is also possible to find an expression for more complicated cases such as $\mathbf{C} = \mathbf{A}^{-1}\mathbf{B}$ by considering the more simpler implicit expression $\mathbf{AC} = \mathbf{B}$.

In this particular example we have

$$\begin{aligned} \frac{\partial \mathbf{B}}{\partial x} &= \frac{\partial \mathbf{A}}{\partial x}\mathbf{C} + \mathbf{A}\frac{\partial \mathbf{C}}{\partial x} \\ \Leftrightarrow \frac{\partial \mathbf{C}}{\partial x} &= \mathbf{A}^{-1}\frac{\partial \mathbf{B}}{\partial x} - \mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\mathbf{C}. \end{aligned} \tag{2.40}$$

Inserting in Eq. 2.34 yields

$$\begin{aligned}\frac{\partial J}{\partial x} &= \mathrm{tr}\left\{\left(\frac{\partial J}{\partial \mathbf{C}}\right)^{\mathsf{T}}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{B}}{\partial x} - \mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\mathbf{C}\right)\right\} \\ &= \mathrm{tr}\left\{\left(\mathbf{A}^{-\mathsf{T}}\frac{\partial J}{\partial \mathbf{C}}\right)^{\mathsf{T}}\frac{\partial \mathbf{B}}{\partial x}\right\} - \mathrm{tr}\left\{\mathbf{C}\left(\frac{\partial J}{\partial \mathbf{C}}\right)^{\mathsf{T}}\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial x}\right\} \\ &= \mathrm{tr}\left\{\left(\mathbf{A}^{-\mathsf{T}}\frac{\partial J}{\partial \mathbf{C}}\right)^{\mathsf{T}}\frac{\partial \mathbf{B}}{\partial x}\right\} - \mathrm{tr}\left\{\left(\mathbf{A}^{-\mathsf{T}}\frac{\partial J}{\partial \mathbf{C}}\mathbf{C}^{\mathsf{T}}\right)^{\mathsf{T}}\frac{\partial \mathbf{A}}{\partial x}\right\}.\end{aligned} \tag{2.41}$$

Comparing the coefficients results in the desired expression for the gradient w.r.t. $\mathbf{A}$ and $\mathbf{B}$ respectively

$$\begin{aligned}\frac{\delta J}{\delta \mathbf{A}} &= -\mathbf{A}^{-\mathsf{T}}\frac{\partial J}{\partial \mathbf{C}}\mathbf{C}^{\mathsf{T}} \\ \frac{\delta J}{\delta \mathbf{B}} &= \mathbf{A}^{-\mathsf{T}}\frac{\partial J}{\partial \mathbf{C}}.\end{aligned} \tag{2.42}$$

2.2.4 Wirtinger calculus

Later in this work, we will integrate signal processing algorithms into a neural network. Digital signal processing (DSP) usually uses a complex valued spectral representation of a signal, i.e. a representation where the signal is decomposed into its harmonics. In order to be able to efficiently train DSP related networks with gradient descent, the backpropagation algorithm must support intermediate complex valued tensors.

The Wirtinger calculus [Bra83] expresses the partial derivative w.r.t. to a complex valued scalar $z = x + jy$ as

$$\frac{\partial}{\partial z^*} = \frac{1}{2}\left(\frac{\partial}{\partial x} + j\frac{\partial}{\partial y}\right) \tag{2.43}$$

$$\frac{\partial}{\partial z} = \frac{1}{2}\left(\frac{\partial}{\partial x} - j\frac{\partial}{\partial y}\right) \tag{2.44}$$

and obeys the chain rule, i.e., given two functions $f : \mathbb{C} \to \mathbb{R}$ and $g : \mathbb{C} \to \mathbb{C}$ with $J = f(u + jv) = f(h)$ and $h = g(z) = g(x + jy)$ the partial derivative $\frac{\partial J}{\partial z^*}$ is given by

$$\frac{\partial J}{\partial z^*} = \left(\frac{\partial J}{\partial h^*}\right)^* \frac{\partial h}{\partial z^*} + \frac{\partial J}{\partial h^*}\left(\frac{\partial h}{\partial z}\right)^* \tag{2.45}$$

as can be shown (e.g. [Böd+17]) by using the real valued chain rule and Eq. 2.43.

Starting from this, the same methodology as for the real valued case can be developed ([DRH16; Böd+17]. Note that from the computational graph perspective only one partial derivative (i.e., Eq. 2.43 or Eq. 2.44) has to be calculated and propagated to the node connected to the inputs as one equation can be readily computed from the respective other one [DRH16]. Also, the same method can be applied to find an efficient expression for $\frac{\delta J}{\delta \mathbf{A}}$ and $\frac{\delta J}{\delta \mathbf{B}}$ when $\mathbf{C} = f(\mathbf{A}, \mathbf{B})$ and $\mathbf{A} \in \mathbb{C}^{N_1^{(a)} \times N_2^{(a)}}$, $\mathbf{B} \in \mathbb{C}^{N_1^{(b)} \times N_2^{(b)}}$,

$\mathbf{C} \in \mathbb{C}^{N_1^{(c)} \times N_2^{(c)}}$. For the complex valued case, Eq. 2.37 becomes

$$\frac{\partial J}{\partial z^*} = \mathrm{tr}\left\{ \left(\frac{\partial J}{\partial \mathbf{A}^*}\right)^{\mathsf{H}} \frac{\partial \mathbf{A}}{\partial z^*} + \left(\frac{\partial J}{\partial \mathbf{A}^*}\right)^{\mathsf{T}} \left(\frac{\partial \mathbf{A}}{\partial z}\right)^* \right\} + \mathrm{tr}\left\{ \left(\frac{\partial J}{\partial \mathbf{B}^*}\right)^{\mathsf{H}} \frac{\partial \mathbf{B}}{\partial z^*} + \left(\frac{\partial J}{\partial \mathbf{B}^*}\right)^{\mathsf{T}} \left(\frac{\partial \mathbf{B}}{\partial z}\right)^* \right\} \tag{2.46}$$

and Eq. 2.39

$$\frac{\partial J}{\partial z^*} = \mathrm{tr}\left\{ (\dots)^{\mathsf{H}} \frac{\partial \mathbf{A}}{\partial z^*} + (\dots)^{\mathsf{T}} \left(\frac{\partial \mathbf{A}}{\partial z}\right)^* \right\} + \mathrm{tr}\left\{ (\dots)^{\mathsf{H}} \frac{\partial \mathbf{B}}{\partial z^*} + (\dots)^{\mathsf{T}} \left(\frac{\partial \mathbf{B}}{\partial z}\right)^* \right\}. \tag{2.47}$$

Again, comparing the coefficients yields the desired expressions for $\frac{\delta J}{\delta \mathbf{A}}$ and $\frac{\delta J}{\delta \mathbf{B}}$.

2.2.5 Summary

This section briefly introduces NNs and important layers like the convolutional layer and the LSTM layer. Their parameters are optimized using stochastic gradient descent which requires to compute the gradient of a real valued loss function w.r.t. each parameter of the network. Exploiting the chain rule, this computation can be carried out efficiently with the backpropagation algorithm. Viewing the network as a computational graph in which each node defines a forward function and the gradient w.r.t. its inputs given the gradient of the loss w.r.t. its outputs allows for a modular implementation. Instead of implementing the gradient computation for a specific network, only the local gradient of the node needs to be implemented. These nodes can then be connected to build an arbitrary network structure with automatic differentiation. This mechanism still works if intermediate inputs and outputs are complex valued and allows the use of complex valued algorithms within a neural network architecture.

3 Automatic speech recognition

The ultimate goal of ASR is to find the most likely sequence of discrete linguistic tokens (usually words, the terms will be used interchangeably) for a given audio signal. The Maximum-a-posteriori (MAP) decision rule is expressed by the fundamental equation of ASR

$$\hat{\boldsymbol{\omega}}_{1:\hat{L}} = \underset{\boldsymbol{\omega}_{1:L}, L}{\operatorname{argmax}} \Pr(\boldsymbol{\omega}_{1:L} | \mathbf{z}_{1:N}). \tag{3.1}$$

Here, $\hat{\boldsymbol{\omega}}_{1:\hat{L}} = [\hat{\omega}_1, \hat{\omega}_2, \ldots, \hat{\omega}_{\hat{L}}]$ is the decoded sequence of tokens and the probability is conditioned on the sequence of input features $\mathbf{z}_{1:N} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]$. According to Eq. 3.1, $\hat{\boldsymbol{\omega}}_{1:\hat{L}}$ is the sequence of tokens among all possible token sequences which maximizes the posterior probability $\Pr(\boldsymbol{\omega}_{1:L} | \mathbf{z}_{1:N})$ given the observed feature sequence $\mathbf{z}_{1:N}$.

The difficulty of solving this maximization problem depends foremostly on the application. If the distance from the speaker to the sensor(s) is small, the number of possible interferences reduces significantly and the feature sequence is more predictive for the task at hand. For larger distances ($> 1\,\mathrm{m}$) this situation changes significantly. Interfering sources can now be captured with the same, or even greater power than the target source itself and realizations of the features are consequently less dependent on the target. Worse, different environments can lead to feature sequences very different to any comparable sequence available in the training data for the model as even big corpora cannot cover all possible interferences. This challenging scenario is termed far-field ASR and the ultimate goal is to build *robust* ASR systems, i.e. systems capable of dealing with severe environmental distortions, whether these are interfering sources, reverberation or a combination of both.

Another major factor for the difficulty of the task is the size of the vocabulary, i.e., the number of words the system knows about. With an increasing size, the classification becomes much more challenging as the number of possible sequences grows exponentially. If the size exceeds a few thousand words, one commonly refers to the task as large vocabulary ASR. This work concentrates on the problem of (robust) far-field large vocabulary ASR.

Today, the approaches to ASR can be roughly divided into two different ones: A statistical approach and the so called end-to-end approach which directly translates audio into text using a single model[1]. While the former benefits from neural networks

[1] The end-to-end approaches are technically still of statistical nature but there is no explicit generative modeling.

in terms of better performance, the latter was only made possible recently by the ability to train large networks. The statistical approach has a long history and is (still) used in most of the production systems for real world applications. We therefore also refer to it as the conventional approach. In the following, both approaches are reviewed with a focus on the statistical one as this one will be used throughout this work.

3.1 Statistical ASR

The statistical way to tackle the problem is to decompose the conditional probability in Eq 3.1 by applying Bayes' rule

$$\hat{\boldsymbol{\omega}}_{1:\hat{L}} = \underset{\boldsymbol{\omega}_{1:L}, L}{\operatorname{argmax}} \frac{\Pr(\boldsymbol{\omega}_{1:L}) p(\mathbf{z}_{1:N}|\boldsymbol{\omega}_{1:L})}{p(\mathbf{z}_{1:N})} \propto \Pr(\boldsymbol{\omega}_{1:L}) p(\mathbf{z}_{1:N}|\boldsymbol{\omega}_{1:L}). \tag{3.2}$$

The normalizing probability of the acoustic realization $p(\mathbf{z}_{1:N})$ can be neglected since it does not depend on the word sequence. Each factor is modeled by a distinct model[2].

Language model The first factor ($\Pr(\boldsymbol{\omega}_{1:L})$) can be identified as modeling the word sequence. Since the order of the words matters in a language and each word has an individual (context dependent) frequency, a certain probability can be assigned to any specific sequence of words. Thus, the corresponding model is referred to as the language model.

Acoustic model The second factor ($p(\mathbf{z}_{1:N}|\boldsymbol{\omega}_{1:L})$) is of a generative nature and describes the probability of an acoustic realization for a given word sequence. The model for this probability is consequently called the acoustic model.

Apart from these models, three other major components complete an ASR system.

Lexicon Instead of modeling $p(\mathbf{z}_{1:N}|\boldsymbol{\omega}_{1:L})$ directly, it is beneficial to introduce a mapping ϕ which maps a sequence of words to a sequence of so-called phonemes. A phoneme is the smallest acoustic unit that distinguishes one word from another for a particular language. Its relation to the acoustic realization and the, compared to the words, small size of the set of all phonemes, makes a phoneme sequence particularly suitable in this context. The mapping from a word to a phoneme sequence is called the lexicon. Note that this mapping is not unique; a word can be mapped to multiple phoneme sequences and vice versa.

Decoder The decoder combines the acoustic and the language model by solving Eq 3.2 and searching for the most likely transcription.

[2]Splitting the distribution not only makes the modeling easier but also has an important practical implication. Training the language model only requires text data while training an acoustic model requires labeled (i.e. transcribed) speech data. While the latter is time consuming and costly to acquire, the former is readily available for many domains.

Feature extraction Usually[3], the acoustic model does not model the raw waveform directly but rather models a sequence of features extracted from it. The feature extraction component extracts features suitable for the model from the raw waveform.

The following describes all of these components, except for the lexicon, in greater detail. Details for the lexicon are described whenever needed as it is the connecting element between the acoustic and language model.

3.1.1 Feature extraction

Extracting features from the raw audio signal is the task of the front-end. Ideally, these features should preserve all information necessary to distinguish the different acoustic units (i.e. phonemes) while being insensitive to acoustic variations introduced by the environment. Usually there is a trade-off between the compactness of the features and the information discarded while producing them. For a long time, the by far most commonly used features for ASR were the so-called Mel frequency cepstral coefficients (MFCCs). These features are partly motivated by the human auditory system and their extraction process has been standardized by the European Telecommunication Standards Institute (ETSI) [Pro03]. The following shortly explains how they are extracted.

After the analog-to-digital conversion (usually with a sampling rate of 8 kHz or 16 kHz), the signal is processed by a pre-emphasis block to compensate for an 6 dB/octave attenuation of the voiced speech [Owe93]. Analysis windows $w_{\mathrm{A}}(t)$ of length L_{window} and with an overlap of B samples then segments the signal $y(t)$ into a sequence of T overlapping frames to which the discrete Fourier transform (DFT) is applied.

$$y_{\mathrm{DFT}}(k,f) = \sum_{t=0}^{L_{\mathrm{window}}-1} w(t)y(t+kB)\exp\left\{-j\frac{2\pi}{F}tf\right\}, \quad f \in \{0,1,\ldots,F-1\} \tag{3.3}$$

The phase information of these F spectral coefficients is discarded[4] and only the magnitude (or power) of the current frame is considered. To these frequency coefficients, another set of N filters with $N < F$ is applied. Each of these overlapping bandpass filters has a triangular shape with a different center frequency and bandwidth. Motivated by the human perception, the frequency resolution is chosen in a non-linear way such that it results in a linear spacing on the Mel scale. This means that the bandwidth and the spacing between the center frequencies increase approximately logarithmically with frequency and also gives the array of filters the name Mel filter bank. Each filter Λ_n with the lower and upper cutoff frequencies F_n^{lower} and F_n^{upper} yields a Mel spectral coefficient (MSC)

$$y_{\mathrm{MSC}}(k,n) = \sum_{f=F_n^{\mathrm{lower}}}^{F_n^{\mathrm{upper}}} \Lambda_n(f)|y_{\mathrm{DFT}}(k,f)|. \tag{3.4}$$

[3]Exceptions will be discussed later in this section.

[4]Phase information has only a minor impact on human speech perception for the used fast Fourier transform (FFT) parameters [PA03]

Table 3.1: Typical parameters used for the feature extraction process depending on the acoustic model (AM).

AM	Frame shift	Frame length	FFT length	#LMSCs	#MFCCs
GMM	160	400	512	23	13
generic NN	160	400	512	40	40
CNN	160	400	512	80	–

To account for the also non-linear human perception of the intensity, the natural logarithm is applied afterwards and the resulting value is called log Mel spectral coefficient (LMSC)

$$y_{\mathrm{LMSC}}(k,n) = \ln y_{\mathrm{MSC}}(k,n). \tag{3.5}$$

Finally, a discrete cosine transformation (DCT) is applied to the stacked vector of LMSCs resulting in the MFCC feature vector

$$\mathbf{y}_{\mathrm{MFCC}}(k) = \mathrm{DCT}([y_{\mathrm{MSC}}(k,0), y_{\mathrm{MSC}}(k,1), \ldots, y_{\mathrm{MSC}}(k,N-1)]). \tag{3.6}$$

The DCT approximately decorrelates the coefficients which is a helpful property when modeling the features with gaussian mixture models (GMMs) as it allows to assume a diagonal covariance matrix. Most of the useful information is contained in the first coefficients and consequently the remaining ones are discarded to reduce the dimensionality of the final feature vector. While the DCT only removes the so-called intra-frame correlations, the vectors themselves are highly correlated (inter-frame correlation), mainly because of the overlapping windows during frame extraction. For models that are only exposed to a single frame (such as the GMM), the context provides helpful information. To incorporate this, Δ (delta) and $\Delta\Delta$ (delta-delta) features are concatenated to the vector of MFCCs resulting in dynamic features. The Δ and $\Delta\Delta$ features approximate the first and second order temporal derivative of the sequence of MFCCs feature vectors respectively.

With neural networks replacing the GMM to model the acoustic observations, the feature extraction pipeline also changes. While MFCCs are still heavily used (e.g., by most Kaldi recipes), their dimensionality increased (see Tbl. 3.1) as the benefits of keeping even weakly informative features outweighs the benefits of dimensionality reduction for a powerful discriminative model. But, depending on the architecture of the network, MFCCs might not be the right choice. For example, when using two dimensional CNNs, the decorrelation along the feature dimension becomes an undesired property. The LMSCs are more suitable for these architectures. And since the networks can learn to dismiss unimportant information and warp the frequency resolution by themselves, it is also possible to just use the magnitude spectrogram although this might require more training data. Also, the model is usually exposed to many frames

at once, eliminating the need for the Δ and $\Delta\Delta$ components. In fact, when using convolutional layers, a neural network can learn appropriate filters to produce these components if they are useful for the classification task.

It is also possible to dismiss the feature extraction altogether and use the raw audio signal. This has been successfully shown in, e.g., [Tüs+14] but also imposes a higher requirement on the amount of training data. Such architectures imitate most of the described extraction process (i.e. framing, filtering, logarithmic compression) but treat the filters as learnable parameters rather than manually determine them. Interestingly, the final filters after training show a lot of similarities to the fixed ones, especially the frequency resolution seems to match closely.

A discussion of suitable features for the multi-channel scenario considered in this work follows in Sec. 3.3.

Typical parameters for the extraction process are shown in Tbl. 3.1.

3.1.2 Acoustic model

The acoustic model estimates the probability $p(\mathbf{z}_{1:N}|\boldsymbol{\omega}_{1:L})$ of the observations given the label sequence. Note that the length of the two sequences differs, i.e. $N > L$. This results in an alignment or attribution problem since it is unknown which frames are associated with one specific label. To solve this, a hidden markov model (HMM) is used. This model introduces a sequence of latent states

$$p(\mathbf{z}_{1:N}|\boldsymbol{\omega}_{1:L}) = \sum_{s_{1:N} \in S} p(\mathbf{z}_{1:N}, s_{1:N}|\boldsymbol{\omega}_{1:L}). \tag{3.7}$$

Assuming each state sequence uniquely maps to a word sequence[5], describing the set of possible state sequences for the word sequence $\boldsymbol{\omega}_{1:L}$ with $S_{\boldsymbol{\omega}_{1:L}}$ and applying factorization yields

$$p(\mathbf{z}_{1:N}|\boldsymbol{\omega}_{1:L}) = \sum_{s_{1:N} \in S_{\boldsymbol{\omega}_{1:L}}} \prod_{n=1}^{N} p(\mathbf{z}_n|\mathbf{z}_{1:n-1}, s_{1:n}) \Pr(s_n|\mathbf{z}_{1:n-1}, s_{1:n-1}). \tag{3.8}$$

The two densities are still conditioned on all previous observations which makes them hard to model. Two assumptions further simplify the equation above. The first one is referred to as conditional independence assumption and assumes that the probability for an observation only depends on the current state

$$p(\mathbf{z}_n|\mathbf{z}_{1:n-1}, s_{1:n}) \approx p(\mathbf{z}_n|s_n). \tag{3.9}$$

The second one approximates the stochastic process of state emissions as a first order Markov process

$$\Pr(\mathbf{s}_n|\mathbf{z}_{1:n-1}, s_{1:n-1}) \approx \Pr(s_n|s_{n-1}). \tag{3.10}$$

[5] To ensure this, disambiguation symbols are introduced in practice.

The resulting approximation for the probability of an observation sequence given a word sequence is now related to the observation density and state transition probabilities by

$$p(\mathbf{z}_{1:N}|\boldsymbol{\omega}_{1:L}) \approx \sum_{s_{1:N} \in S_{\boldsymbol{\omega}_{1:L}}} \prod_{n=1}^{N} p(\mathbf{z}_n|s_n) \Pr(\mathbf{s}_n|s_{n-1}). \tag{3.11}$$

The states used in this model represent context-dependent phonemes also called senones. They describe a phoneme with its left and right context. The reasoning behind this choice is that an acoustic realization of a specific phoneme can be very different depending on the context it is used in. Since the number of possible context-dependent phonemes is usually too large, a clustering mechanism is used to combine contexts which result in a similar acoustic realization of the phoneme. In practice, depending on the size of the training data, a few thousand states are used.

GMM-HMM models

Despite many efforts to find alternatives, until the early 2010s the predominant way to model the observation density in Eq. 3.9 was to use a GMM resulting in a GMM-HMM model. These models can be trained efficiently with the expectation maximization (EM) and forward-backward algorithm [RJ86; MK07]. However, despite a wide variety of methods and great efforts to improve the performance, it eventually plateaued especially in challenging environmental conditions. Nevertheless the GMM-HMM models still play an important role as of today.

DNN-HMM models

First investigations of using neural networks for ASR started in the late 1980s and are reviewed in [Lip89]. The first works showed good performance on phoneme recognition tasks. Here, the alignment is known and the task is to infer a single phoneme class from a specific number of frames. Few works also demonstrated promising results for large vocabulary ASR tasks [RHR96]. However, GMM-HMM were easier to train and showed better performance. With more computational power, more data and the ability to train deeper networks [SLY11] this picture changed. The first works reporting a big performance gain for (applied) ASR using neural networks are [SLY11; Hin+12]. Different model architectures and training methods as well as even more computational power and data have led to significant improvements since then and no other type of (statistical) model has been shown to be competitive as of today. Some publications even report human like performance for specific tasks [Xio+17]. This work focuses on the so called hybrid models[6] that still make use of the HMM framework but instead of using a GMM to model the observation density found in Eq. 3.9, a neural network is

[6] There also exists the so-called TANDEM approach where a network is purely used as a feature extractor and the acoustic model is still a GMM-HMM. Sophisticated discriminatively trained TANDEM system achieve about the same performance as hybrid systems (see, e.g., [Tüs+17]) and some techniques developed for GMM-HMM systems can readily by applied. However, the majority of works including this thesis focuses on hybrid systems.

used. To this extent, Eq. 3.9 is rewritten using Bayes' rule to obtain

$$p(\mathbf{z}_n|s_n) = \frac{\Pr(s_n|\mathbf{z}_n)p(\mathbf{z}_n)}{\Pr(s_n)} \propto \frac{\Pr(s_n|\mathbf{z}_n)}{\Pr(s_n)}. \tag{3.12}$$

The state probability $p(s_n)$ is usually estimated on the training corpus from the occurrence frequencies of the states and $p(s_n|\mathbf{z}_n)$ is estimated with a neural network. To further improve the accuracy it is possible to include some frames as context $p(s_n|\mathbf{z}_{n-c_l:n+c_r})$ or, if there are no requirements on latency, even condition on the whole observation $p(s_n|\mathbf{z}_{1:N})$ by using, e.g., recurrent models. Since neural networks are able to handle highly correlated input data, this is a clear advantage over the GMMs where decorrelated data is beneficial. Additionally, a GMM-HMM system has an independent GMM per state which is usually not aware[7] of the models from other states whereas in a DNN-HMM system a single model is used for all states.

The reformulation in Eq. 3.12 transforms the problem into a classification task neural networks are particularly good at and also allows for a fast inference since the posterior probabilities can be calculated from the input independent of a hypothesis for the word sequence. Note that in this scenario a posterior is still estimated for each frame and its context independently, i.e. this does not lift the conditional independence assumption but might reduce the resulting approximation error by also considering neighboring frames (or even the whole sequence).

In order to train the network with a cross-entropy criterion in this setting, the state for each frame is required as supervision, i.e. the alignment must be known. This is commonly solved by using the hard alignment from a GMM-HMM system. A hard alignment can be extracted with a trained GMM-HMM system utilizing the Viterbi algorithm which yields the most likely state sequence[8]. The network can now be optimized to predict the most likely state for each frame, effectively minimizing the expected frame error rate.

One can identify this frame-wise training as the main drawback of this approach. First, the expected frame error rate is only a loosely related proxy for the actual target (minimizing the expected word error rate). And second, a GMM-HMM system is still required to obtain the alignment. Not only does this add additional complexity, it also introduces a dependency. The quality of the alignments is determined by the performance of the GMM-HMM system. And while neural networks are able to handle a certain amount of label noise [Kra+15] and even benefit from it as a form of regularization [Chi+17; Sze+15], empirical evidence shows that an erroneous alignment hurts the performance of the DNN-HMM system [Del+13; NW14]. Especially for low signal-to-noise ratio (SNR) or highly reverberant scenarios investigated in this work, the quality of the alignments can become an issue. If parallel data is available, one way to overcome this is to use the clean speech data to train the GMM-HMM system and to extract alignments which are then used for the distorted data of interest. Thus, when working with reverberation, care has to be taken that the reverberated data still aligns with the clean data as most room impulse responses (RIRs) introduce a group delay.

One way to overcome the problem of extracting the right frame labels that also addresses the loose relationship with the actual target is to switch from a frame-wise

[7] Training with a maximum mutual information (MMI) criterion addresses this issue

[8] A simple count of these is used to estimate the previously mentioned state probability $p(s_n)$

criterion to a sequence-wise one. CTC is one criterion that directly optimizes a variant of Eq. 3.11. Modifications include omitting the transition model as well as the state prior (i.e. assuming that both are fixed and uniformly distributed) and extending the frame context to the full sequence. The criterion also introduces an additional state called blank, changing the set of possible state sequence from $S_{\boldsymbol{\omega}_{1:L}}$ to $\tilde{S}_{\boldsymbol{\omega}_{1:L}}$. The blank state corresponds to no output and is ignored during decode. It is crucial for the convergence of the system and seems especially beneficial if recurrent neural networks are used as found by studies on the related task of handwriting recognition in [Blu+15]. The CTC likelihood is then expressed by

$$\mathcal{L}_{\text{CTC}} = \sum_{s_{1:N} \in \tilde{S}_{\boldsymbol{\omega}_{1:L}}} \prod_{n=1}^{N} p(s_n | \mathbf{z}_{1:N}). \tag{3.13}$$

With the help of the forward-backward algorithm, this criterion and its gradient can be computed efficiently during training. CTC thus eliminates the need for an alignment by considering all valid state sequences mapping to the target transcription. However, this assumes that all sequences are equally likely a priori and neglects the effects of the language model leading to a mismatch when both models are combined.

Sequence discriminative criteria such as MMI [Bah+86], minimum phone error (MPE) [Pov05] or state-level minimum Bayes risk (sMBR) [Kin09; Ves+13] account for the fact that not all sequences are equally likely and further close the gap between the training and evaluation criterion. One variant is the so called lattice-free maximum mutual information (LF-MMI) [Pov+16] which, as of today, is used by the popular open-source toolkit Kaldi [Pov+11] where it achieves the best performance on many different datasets. The MMI criterion for ASR relates the likelihood of the observation sequence given the state sequence $p(\mathbf{z}_{1:N}|s_{1:N})$ to the likelihood of the observation $\sum_{s' \in S} p(\mathbf{z}_{1:N}|s'_{1:N}) \Pr(s'_{1:N})$ which is obtained by marginalizing over all possible sequences. This way the model is encouraged to maximize the likelihood of the observation sequence conditioned on a valid state sequence such that it does not increase the likelihood of the observation sequence when conditioned on a different state sequence. Since the set of all possible sequences S is huge, the marginalization over all state sequences to compute the (unconditional) likelihood of the observation is computationally demanding and several simplifications are necessary to achieve practical speeds. LF-MMI for example uses a simplified transition model, a phone-level language model and a reduced frame-rate [Pov+16].

3.1.3 Language model

The language model estimates the probability $\Pr(\boldsymbol{\omega}_{1:L})$ of a word sequence. In its factorized form, this probability is expressed by

$$\Pr(\boldsymbol{\omega}_{1:L}) = \Pr(\omega_0) \prod_{i=1}^{L} \Pr(\omega_i | \boldsymbol{\omega}_{0:i-1}). \tag{3.14}$$

This can again be approximated by limiting the context length to the last $n-1$ words. Left padding the sequence with $n-1$ special sentence start labels then yields

$$\Pr(\boldsymbol{\omega}_{1:L}) \approx \prod_{i=0}^{L} \Pr(\omega_i | \boldsymbol{\omega}_{i-n:i-1}). \tag{3.15}$$

A language model based on this approximation is called an n-gram language model. The probabilities can be estimated by counting the occurrences in a large training dataset[9]. But even for a relatively small vocabulary size of 5000 words and a tri-gram language model there are $5000^3 = 125.000.000.000$ possible n-grams. Considering that the frequency of words approximately follows a power law [GOO53], even for a very large corpus many n-grams will never been seen and thus zero probability will be assigned to them if trained using the maximum-likelihood (ML) criterion. This issue can be mitigated by employing smoothing techniques and/or backoff methods [NEK94]. The former introduces pseudo counts for non-existing n-gram while the latter uses the probability of a reduced context multiplied with some back-off probability. During the combination with the acoustic model, the size of the search space is mainly determined by the size of the vocabulary and order of the language model. In practice, a good trade-off is found by using a tri-gram language model to narrow down the number of hypotheses (first pass) followed by a rescoring of these hypotheses with, e.g., a 5-gram language model.

A more powerful recurrent neural network language model (RNN-LM) [Mik+11] does not limit the size of the context and approximates the sequence probability as

$$\Pr(\boldsymbol{\omega}_{1:L}) \approx \prod_{i=0}^{L} \Pr(\omega_i | \mathbf{h}_{i-1}). \tag{3.16}$$

The state $\mathbf{h}_i$ of the recurrent network can, at least in theory, store all relevant information from all previous words. An analysis shows that the context spans roughly 200 words but the order is only important for the words nearby [Kha+18]. Even more context can be captured using a different architecture termed transformer [AlR+18; Dai+19] However, the statefulness of the recurrent model becomes an issue when combining the probabilities with the ones from the acoustic model since for each search path a state vector has to be stored now and due to the infinite context fewer states can be combined. Some works suggest that such a model can still be used within a single decoding pass, e.g., by using caches [HZD14; Bec+19]. But mostly the model is used to rescore n-best hypotheses during a second decoding pass.

3.1.4 Decoder

The decoder searches for the most probable word sequence combining the probabilities[10] of the acoustic and the language model, i.e. it solves the optimization problem from Eq. 3.2. This can be done in a frame synchronous or in a label synchronous way. The

[9] Contrary to transcribed speech data, a vast amount of plain text data is available for many domains (e.g., [Che+13]) or can be aggregated by, e.g., scraping the web.

[10] For numerical reasons the actual implementation uses log-probabilities also referred to as scores

hybrid approach considered in this work uses the former one but for other architectures the latter one is beneficial (see the following Sec. 3.2). Frame synchronous means that the decoder advances one step with each frame processed by the acoustic model. In order to find the most likely sequence, the decoder would theoretically need to evaluate the probabilities for all possible word sequences. This is computationally infeasible and the decoder uses a heuristic algorithm called beam search and dismisses paths with low probability after each step.

Typically, the decoder is efficiently implemented utilizing weighted finite state transducers (WFSTs) (see, e.g., [MPR02; MPR08]). To compute the probability of a hypothesis, the probabilities of the acoustic and language model are combined by adding them in the log domain. A weight factor, the so-called language model weight, is multiplied with the language model score to trade-off the influence between the acoustic and the language model.

After having described all components of a statistical ASR framework, we now turn to end-to-end ASR systems. These systems roughly consist of a single model directly transcribing the audio (features). In the following, the two major architectures, sequence-to-sequence models with attention and the RNN transducer (RNN-T) are briefly described.

3.2 End-to-end ASR

3.2.1 Sequence-to-sequence models

The initial work by Chorowski et al. [Cho+14b] in 2014 paved the way for all-neural sequence-to-sequence (Seq2Seq) models which directly transform audio (features) into a sequence of labels. They consist of an encoder network which transforms the audio features into some learned representation and a decoder network which is similar to a RNN-LM but conditioned on the representation of the encoder. For long sequences such as those occurring in ASR, it is not sufficient to just initialize the decoder with, e.g., the state of the encoder like the first models of this architecture did for different tasks [Cho+14a]. Instead, a mechanism introduced for machine translation tasks to address this issue [BCB14] called attention is used. This mechanism gives the decoder access to the whole representation for each decode step in form of a weighted sum where the weights are dynamically computed depending on a vector representation issued by the decoder. A follow-up work titled listen attend and spell (LAS) [Cha+16] changed the output labels from phonemes to characters effectively also learning the spelling (i.e. the lexicon).

One can identify three major advantages of the Seq2Seq models compared to the statistical approach. First, it allows to eliminate the conditional independence assumption which is crucial for the statistical HMM based system. Second, lifting the requirement for a lexicon greatly reduces the manual labor needed to build a system for a new language or domain. And third, it significantly simplifies the toolchain necessary to train the model. These advantages triggered a big research interest and there has been a significant progress in this area since the first work was published. Notably, recent systems now start to outperform the conventional ones on datasets with 1000 h

of speech data or more (e.g., [Chi+18; Iri+19; Par+19]). However, for smaller amounts of training data statistical approaches are still more effective than the end-to-end ones as we will demonstrate later. This can partly be attributed to one drawback caused by having a single model. As mentioned earlier, there is usually much more training data available for the language model than for the acoustic one. For an end-to-end approach however, the decoder is only trained using the transcribed audio data and despite some efforts to overcome this issue (e.g., [Kar+19]), it is still an open research problem. Better performance can be achieved by integrating an auxiliary language model during decode time [CJ17; Gül+15] or even during training [Sri+18]. But even then there is a remaining performance gap for smaller datasets compared to the statistical approach. The vanilla Seq2Seq also has another drawback, namely its latency caused by the attention that spans over the whole encoded input. Thus the system needs to consume the full audio signal to be transcribed while conventional systems can operate in a streaming and even frame-by-frame fashion. Models with a block-wise processing of the audio stream are subject to current research and have yet to reach the performance of their offline counterparts [Sai+18].

3.2.2 RNN transducer

A different end-to-end model which is closer related to the conventional systems is called RNN-T [Gra12]. Roughly speaking, it is a combination of the CTC idea combined with an RNN-LM and a loss function for joint training. The model interpolates the token probabilities from the transcription network (acoustic model) for each frame with the token probabilities for each token from the prediction network (RNN-LM). Here, token refers to either a grapheme or some other sub-word unit. By also considering the probabilities from the prediction network the model effectively relaxes the conditional independence assumption since the prediction network has access to the full hypothesis history up until the current label. Calculation of the likelihood for training can again be implemented efficiently using the forward-backward algorithm [Gra12]. The newly introduced dependency of the probability distribution for one frame on the history of hypothesized tokens complicates the decoding process which can either operate in a label or in a frame synchronous fashion.

Note that different from the language model in the statistical system the prediction network here does not describe word dependencies but rather works with the same tokens as the transcription network. This allows to again also implicitly learn the spelling (lexicon) but also might hurt the performance slightly as word level language models outperform those based on sub-units. Since the prediction network is nothing else then a RNN-LM, it can be pre-trained without any modifications allowing to exploit large text corpora. The biggest difference between RNN-T and Seq2Seq models is the way the alignment problem is solved. For Seq2Seq models this happens implicitly by learning the attention while for RNN-T there is an explicit mechanism of marginalization during the loss calculation and search phase. This explicit mechanism also incorporates a monotonic prior which seems suitable for the ASR task. It also allows RNN-T models to operate in a streaming fashion for low latency applications, even on resource constrained hardware such as mobile phones [He+19].

All systems described so far use features from a single microphone. In the following,

we discuss how the neural network can be extended upon to benefit from additional information provided by multiple microphones. These methods concentrate either on the feature extraction and/or the model architecture but can in principle be applied to both, statistical ASR as well as to end-to-end approaches.

3.3 Multi-channel ASR

Multiple approaches exist in order to exploit the additional information contained in a multi-channel input signal for NN-based ASR systems.

One straightforward option is to use the same feature extraction as described in Sec. 3.1.1 for each input channel individually and then either stack the resulting vectors to obtain a single multi-channel feature vector or use a CNN architecture with multiple input channels. This approach has been investigated in [SGR14] where it yields performance improvements over a single-channel system. It only uses magnitude information since the phase is dropped during the feature extraction (see Sec. 3.1.1). However, for a microphone array with no occlusion, the magnitudes of the individual signals are nearly identical, especially when the spacing between the microphones is small. Thus, the possible gains this approach can achieve are limited and no spatial information is exploited.

Finding a feature representation of the phase information that can be utilized effectively by a neural network to provide spatial information but that is also invariant to the array geometry and room characteristics remains an open challenge. Recently, interchannel phase difference (IPD) features showed improvements for a multi-channel speech separation task [WLH18]. They have also been used for acoustic modeling in the context of the CHiME 5 challenge and showed promising results [Kan+18]. However, for the speech separation task, only pairs of microphones are considered and the information of those pairs is aggregated in an embedding space. This allows for handling arbitrary microphone configurations. And although some techniques exist to combine information later in the process for the acoustic model (e.g., ROVER [Fis97]), the proposed method aggregates the information at the input level rendering it specific to the microphone configuration during training.

A different approach to exploit spatial information is to directly work on the raw waveform and leveraging temporal differences between the input signals. Previous works have already shown that neural networks with a suitable architecture can learn to extract useful information from the raw waveform in the single-channel case and even outperform similar models which rely on the manually designed feature extraction [Tüs+14; Sai+15c]. In [HWW15] Hoshen et al. report noticeable performance improvements when using multiple raw waveform signals on a larger scale voice search task. Further improvements in performance as well as reduction in computational demands were achieved in follow-up works [Sai+15a; Sai+16]. An analysis of the learned filters in [Sai+15a] shows some interesting properties of the system. First, the filters have a bandpass characteristic and spatial selectivity. Performance increases with a higher number of channels but only if the number of filters is also increased. Using a different array geometry during inference and training degrades the results. This can be avoided by sampling different microphone configurations during training but only 30 % of the learned filters remain spatially

selective afterwards. However, evaluation shows that this restores the performance gain achieved in the first place and now even for different microphone configurations. This suggests that multiple beamformers are learnt for different critical bands and the network is able to select one that supports classification depending on the observation. But while this is possible for two channels where the only variable is microphone spacing, a system with support for more channels will need to learn many more filters to achieve the same invariance. Also, the number of channel is determined at training time. If a channel is missing the system presumably cannot adapt and if additional channels become available, the system will not benefit.

3.4 Summary

This chapter reviewed the underlying methods of ASR systems. It introduced the statistical view and the split into an acoustic and language model. For both, a shorter context or independence assumption is key to obtain equations with tractable models. The conventional acoustic model is based on a HMM with GMMs to model the observation. Large improvements are achieved by replacing the GMMs with a DNN, leading to the so-called hybrid approach where the DNN classifies the states given the acoustic feature(s). Further improvements can be realized using a sequence training criterion. Besides the statistical approach, end-to-end models were discussed. These models make fewer assumptions, crucially they dismiss the independence assumption but need more data to reach comparable performance and are subject to active research. In this work we will use HMM-DNN models which are more suitable for smaller datasets.

Several options to exploit multiple microphones with neural network architectures were reviewed. Exploiting phase information is hard with neural networks. One major drawback of the current approaches is that they are not adaptive. Once trained, their filters are fixed and only work for certain microphone configurations and acoustic scenarios encountered during training. This prohibits, e.g., tracking a speaker movement during a longer utterance.

4 Speech signal processing

In the last chapter we already described the processing of the raw audio to obtain useful features for the ASR system. We also briefly discussed how to exploit multi-channel audio data recorded by an array of microphones. This discussion, however, was primarily focused on the (NN-based) acoustic model and its ability to directly handle multi-channel data.

Another approach is to perform signal processing on the audio data beforehand and, e.g., condense it to an enhanced single-channel signal from which the features for acoustic model are extracted afterwards. Speech signal enhancement aims to reduce the negative impact of interferences and reverberation on speech intelligibility for humans as well as for machines. It is based on a physical and/or statistical model of the involved signals and acoustic scenario. These models will be explained in Sec. 4.1. With this prerequisite, a physical (Sec. 4.2.1) and statistical method (Sec. 4.2.2) for acoustic beamforming is reviewed. We will focus on statistical beamforming, explain different optimization criteria to obtain the filter coefficient and relate them afterwards. All presented methods make use of signal statistics which need to be estimated (Sec. 4.3). One way these statistics can be obtained is by estimating a class affiliation for each time frequency bin (tf-bin). Spatial characteristics can be modeled by a statistical model for this task (Sec. 4.4). Finally, we will present methods specially designed to deal with reverberation. One of them is called weighted prediction error (WPE) and will be discussed in Sec. 4.5.2. But we can also use beamforming for this task as will be detailed in Sec. 4.5.1.

4.1 Signal model

4.1.1 General model

In general, we assume that the acoustic scene consists of Q direct sources as well as a diffuse signal which is a superposition of the signal of several diffuse sources. Direct sources are point sources which emit sound in a very limited region of space, while diffuse sources emit the sound in a very broad region of space. This acoustic scene is captured using M sensors which are usually part of one sensor array. The geometry of such an array can have a specific design to support the disentanglement of the signals (see, e.g., the beampattern in following Sec. 4.2.1) but is assumed to be unknown for most parts of this thesis. Because of the linear propagation of a sound wave [Kut16], an individual sensor m of such an array records the following superposition of time signals:

$$y_m(t) = \sum_{q=0}^{Q-1} \sum_{\tau=0}^{\infty} a_{m,q}(t,\tau) x_q(t-\tau) + d_m(t). \tag{4.1}$$

$x_q(t)$ is the signal emitted by the q-th direct source, $d_m(t)$ is the superposition of all diffuse sources as captured by the m-th microphone and $a_{m,q}(t,\tau)$ is the acoustic impulse response (AIR) from the q-th source to the m-th microphone at the current time t.

The AIR (or RIR) is the result of the summation of all propagation paths and can be decomposed into three (including the delay four) distinct parts as depicted in Fig. 4.1. First, there is the direct path from the source to the sensor. Because this is also the shortest path, it is characterized by the first peak in the AIR. This peak is followed by a number of smaller peaks, the so called early echos which are the result of waves reflected by nearby objects and the room boundaries. Finally, there is the long reverberation tail produced by signals which are reflected multiple times by various objects or walls. For this part, the individual peaks cannot be distinguished from each other anymore because of the high number of different paths with the same length. The power of the reverberation signal decreases exponentially as only a part of the signal energy is reflected each time the paths encounters a surface. The ratio of the absorbed power and the reflected power depends on the material and the angle of incidence. High ratios are usually encountered for, e.g., carpets (15 %). Harder materials reflect more signal power and the ratio is 7 % for a concrete wall and only 1 % for a tiled floor [Mar+14].

4.1.2 Simplified model

Without any simplifying assumptions, any enhancement of a (speech) signal $y_m(t)$ is difficult due to the time-varying and possibly infinitely long AIR. In this work, we therefore limit ourselves to a more specific acoustic scenario. Namely, we make the following main assumptions:

1. There is exactly one target source we want to recover and this source is a human speaker uttering a sentence we aim to transcribe. For most cases, we will not make any assumptions about the other sources except that these are non-speech sources. In a typical scenario, these sources will be noises typically encountered in a household setting (e.g., vacuum cleaner, television, appliances, ventilation / air conditioning, etc.) or an outdoor setting (e.g., cars, construction sites, diffuse babble, etc.).

2. The AIR is time invariant for a certain period of time. Depending on the concrete system, the length of this time span ranges from a few hundred milliseconds to a few seconds (e.g., the length of a sentence) and will be specified individually later. This assumption translates to a static scene where no source nor any object is moving. While this is still a strong assumption for some scenarios, in many cases it is reasonable given the short time period over which time invariance is assumed.

3. The length of the AIR is finite.

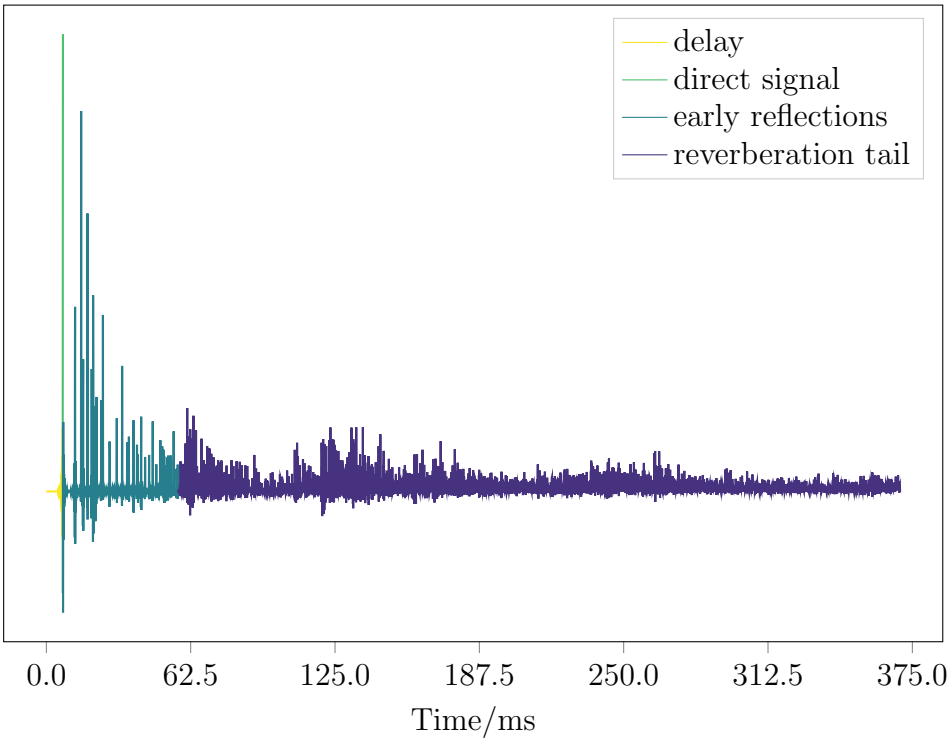

Figure 4.1: Example of a (simulated) RIR with colors to highlight the different parts of the signal. After a delay (yellow) caused by the distance between the source and the sensor, the direct signal (lighter green) arrives first. This also has the highest amplitude since it exhibits no reflections which absorb some of the power. Signals arriving within a range of 50 ms after the direct signal are the early reflections (darker green). Their individual peaks can still be distinguished. Afterwards, the reverberation tail (purple) is composed of numerous reflected signals of weak power with indistinguishable peaks.

Taking these assumptions into account, we can write Eq. 4.1 as

$$y_m(t) = \sum_{\tau=0}^{A_{m,s}} a_{m,s}(\tau)s(t-\tau) + \sum_{q=1}^{Q-1}\sum_{\tau=0}^{A_{m,q}} a_{m,q}(\tau)n_q(t-\tau) + d_m(t). \tag{4.2}$$

Here, we explicitly distinguished between the speech source $s(t)$ and the $Q-1$ noise sources $n_q(t)$. Note that, due to the second assumption formulated above, the AIRs do not depend on the time t anymore and the third assumptions leads to a finite length $A_{m,q}$. The signal resulting from the convolution of an AIR and a source is called the spatial image of the source in the following.

Although simplified, this model is still unsuitable for most signal processing methods. The length of the AIR usually ranges from hundreds to several thousand samples (taps). The computational requirements to estimate this AIR to, for example, compute an inverse filter, are too high for many applications due to the large matrices involved when calculating, e.g., second order statistics.

4.1.3 Spectral model

A common way to solve this issue is to work with a spectral representation of the signals. In this representation, the signal is decomposed into its individual frequency components. Throughout this thesis, all signals are considered in this domain if not explicitly noted otherwise.

Using the short-time Fourier transform (STFT) and k as the frame and f as the frequency index respectively, the spectral representation of the signal from Eq. 4.2 can be approximated as

$$y_m(k,f) = a_{m,s}(f)s(k,f) + \sum_{q=1}^{Q-1} a_{m,q}(f)n_q(k,f) + d_m(k,f). \tag{4.3}$$

To arrive at this equation, additional assumptions are necessary: The frame length of the STFT must be large enough to capture the longest AIR such that the differences between the circular convolution and a normal convolution can be neglected. Additionally, all adversarial effects (e.g., the leakage effect) must be sufficiently small.

The above approximation is also called the narrowband approximation [Gan+17] and is widely adopted by the speech enhancement community.

Decoupling the individual frequencies reduces the dimension of the problem significantly and allows complex algorithms to be computed efficiently. This, however, also comes at a cost: The gain and permutation ambiguity arise because many algorithms solve a problem locally for each frequency without access to global scaling and / or permutation [Gan+17]. Concrete realizations of these problems and possible solutions are presented later in this work.

To represent the complete captured signal, we now switch to vector notation, stack the signals of the individual microphones and arrive at

$$\boxed{\mathbf{y}(k,f) = \mathbf{a}_s(f)s(k,f) + \mathbf{A}_n(f)\mathbf{n}(k,f) + \mathbf{d}(k,f)}. \tag{4.4}$$

Here, $\mathbf{y}(k,f)$ is an M-dimensional vector

$$\mathbf{y}(k,f) = [y_0(k,f), y_1(k,f), \ldots, y_{M-1}(k,f)]^{\mathrm{T}},$$

$\mathbf{a}_s$ is an M-dimensional vector

$$\mathbf{a}_s(f) = [a_{0,x}(f), a_{1,x}(f), \ldots, a_{M-1,x}(f)]^{\mathrm{T}},$$

$\mathbf{A}_n$ is a $M \times Q-1$-dimensional matrix

$$\mathbf{A}_n = \begin{bmatrix} a_{0,1} & a_{0,2} & \ldots & a_{0,Q-1} \\ \vdots & \ddots & & \vdots \\ a_{M-1,1} & & \ldots & a_{M-1,Q-1} \end{bmatrix}$$

$\mathbf{n}(k,f)$ is a $Q-1$-dimensional vector

$$\mathbf{n}(k,f) = [n_1(k,f), n_1(k,f), \ldots, n_{Q-1}(k,f)]^{\mathrm{T}},$$

and $\mathbf{d}(k,f)$ is a M-dimensional vector

$$\mathbf{d}(k,f) = [d_0(k,f), d_1(k,f), \ldots, d_{M-1}(k,f)]^{\mathrm{T}}.$$

Equipped with a spectral representation of the audio signal, we now turn to acoustic beamforming. Acoustic beamforming can exploit spatial information to enhance the speech signal and also combines the individual channels resulting in an enhanced single-channel audio signal.

4.2 Acoustic beamforming

The goal of acoustic beamforming is to amplify the target speech signal $s(k,f)$ and suppress the interference signals $\mathbf{n}(k,f)$ and $\mathbf{d}(k,f)$ by exploiting spatial characteristics. Formally, we seek to find filter coefficients $\mathbf{w}(k,f)$, such that

$$\hat{s}(k,f) = \mathbf{w}^{\mathsf{H}}(k,f)\mathbf{y}(k,f). \tag{4.5}$$

Note that this is a linear filter operation. The filter itself is frequency dependent and, in general, also time dependent. With the second assumption from above, this can be relaxed and the filter is assumed to be static for the period of an utterance or a block of frames. Depending on the application, different constraints can be imposed on the filter. If the perceptual quality is important (e.g., for hearing aids), the distortions of the speech signal should be as small as possible and deteriorating effects on the intelligibility should be minimized. For an automatic transcription system on the other hand, the perceptual quality is not important as long as the features extracted from the signal are helpful for the classification. This leads to different optimization goals and trade-offs.

4.2.1 Physical model

If no reverberation is present[1] and the distance between the sources and sensors is large compared to the distances between sensors[2], the signals captured by the array differ only in a delay. Normalizing the AIR with an arbitrarily chosen reference channel (e.g., the 0-th sensor) leads to the relative transfer function (RTF). In this case, the RTF for source q is expressed by

$$\mathbf{a}_q(f) = \left[1, \exp(-j\Omega_f \Delta_{\tau_1,q}), \ldots, \exp\left(-j\Omega_f \Delta_{\tau_{M-1},q}\right)\right]^T \tag{4.6}$$

with the discrete angular frequency index $\Omega_f = 2\pi \frac{f}{L_{fft}}$ and the time difference of arrival (TDOA) for the q-th source $\Delta_{\tau_m,q}$.

The TDOA is given by the distance of the reference and the m-th sensor projected onto the directional vector of the impinging wavefront from angle θ_q divided by the speed of sound

$$\Delta_{\tau_{(m)},q} = \frac{\mathrm{dist}(m, m_{ref}) \sin(\theta_q)}{c}. \tag{4.7}$$

For the anechoic case specified before, the expression 4.6 is also called the steering vector. If we multiply the signal 4.4 with the hermitian of the steering vector for the target, we obtain

$$\mathbf{z}(k,f) = \mathbf{a}_x(f)^{\mathrm{H}} \mathbf{y}(k,f) = Mx(k,f) + \mathbf{a}_x(f)^{\mathrm{H}} \mathbf{A}_n(f) \mathbf{n}(k,f) + \mathbf{a}_x(f)^{\mathrm{H}} \mathbf{d}(k,f). \tag{4.8}$$

The target signal is amplified with a gain equal to the number of microphones. Other directional sources will be attenuated depending on the cosine similarity between the steering vector and their respective AIR. The diffuse noise is also reduced depending on its characteristics. If the noise is spatially white, i.e., is independent noise at each sensor, it will be more dampened compared to spatially correlated noise.

The operation 4.8 can be interpreted as acoustically "looking" or "steering the beam" into a certain direction in space. It is also called Delay-and-Sum beamformer since, in the time-domain, the resulting signal is the sum of all signals where each one has a specific delay to compensate for the TDOA. From the presence of the sine function in Eq. 4.7 it becomes clear that – depending on the array configuration – the steering vector is ambiguous. Different steering (or beamforming) vectors can lead to the same beampattern for a specific arrangement of the sensors. The plot of the sensitivity of an array steered towards a source is called beampattern. Exemplary ones for an array with two sensors with a spacing of 10 cm and 20 cm respectively are depicted in Fig 4.2. Note that the pattern depends on the frequency and the resolution increases with the frequency and microphone spacing at the cost of additional side lobes again caused by the sine function.

[1] Reverberation can be seen as introducing new mirrored sources with a specific delay from the early reflections and a diffuse source with the late reverberation.

[2] In free space, the signal is attenuated proportional to the inverse of the distance between the source and the sensor. If this distance is large, the attenuation is approximately the same for all sensors since the distances between the sensors is usually small.

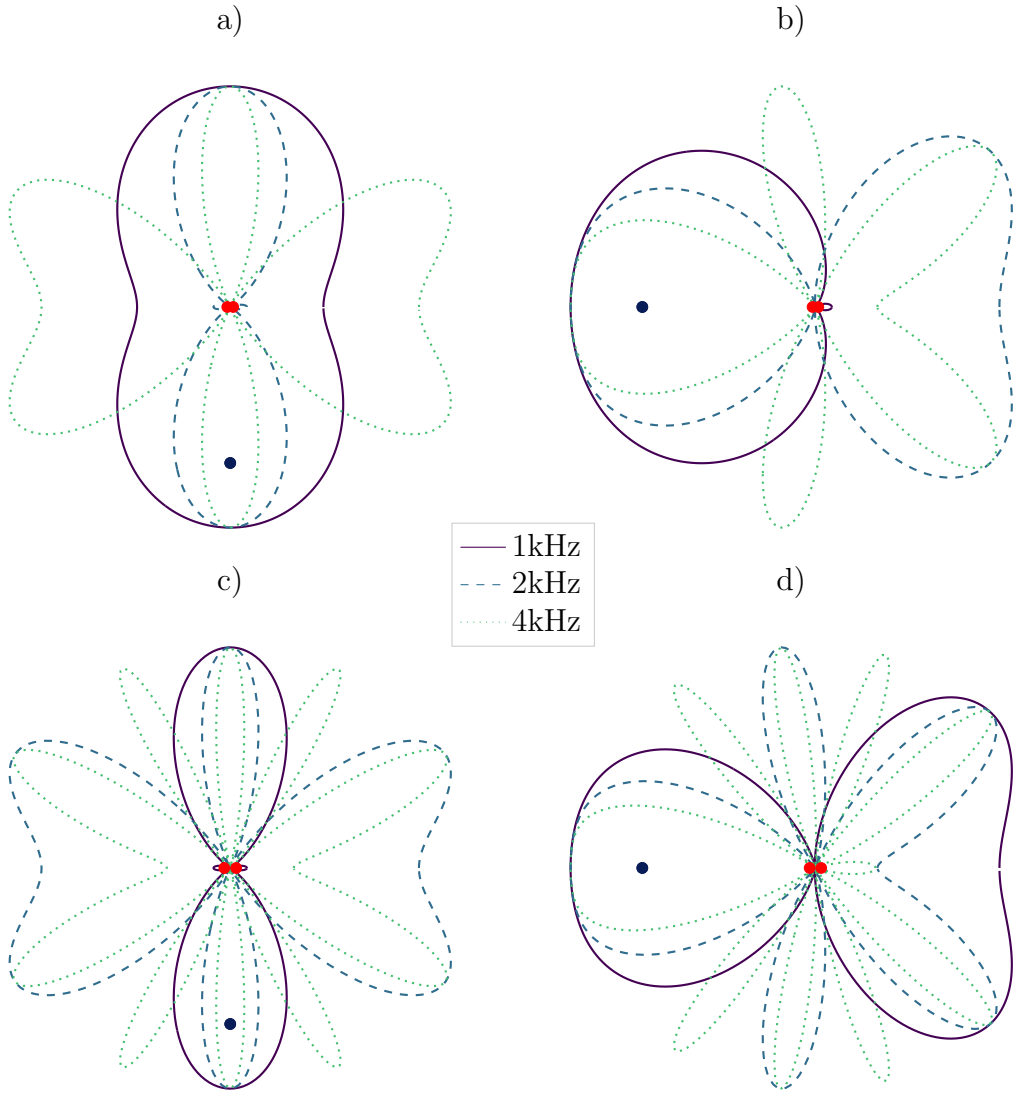

Figure 4.2: Example beampattern for frequencies of 1 kHz, 2 kHz and 4 kHz and for a microphone spacing **a)** of 10 cm and a source at an angle of $\frac{3}{2}\pi$ **b)** of 10 cm and a source at an angle of π **c)** of 20 cm and a source at an angle of $\frac{3}{2}\pi$ **d)** of 20 cm and a source at an angle of π The sensors are represented by red dots and assumed to be omnidirectional. The source is represented by a black dot. The distance between the source and the center of the array is always 3 m. The beampatterns are calculate such that the steering vector points towards the source.

4.2.2 Statistical model

The previous subsection gave a physical motivation of beamforming. In many situations, however, it is beneficial to approach the problem from a statistical point of view. One reason for this is that the mapping from a specific beampattern to a beamforming vector requires knowledge about the array configuration. That is, even if the desired beampattern is known, the steering vector is not. A second reason is the estimation of the desired beampattern itself. In the presence of reverberation it is not intuitively clear how a beampattern optimized for the goal formulated above looks like. For example, capturing early reflections instead of only the direct path can improve the perceptual quality as well as the recognition performance of ASR systems.

The following paragraph reviews three different criteria, each leading to a statistically optimal beamforming vector under the specific criterion. All of them make use of second-order spatial statistics of a multi-channel signal $\mathbf{z}(k, f)$:

$$\mathbf{\Phi}_{zz}(k, f) = \mathbb{E}\left[\mathbf{z}(k, f)\mathbf{z}^{\mathsf{H}}(k, f)\right] \tag{4.9}$$

The spatial covariance matrix (SCM) $\mathbf{\Phi}_{zz}(k, f)$ has the dimension $M \times M$ and captures how correlated the sensor signals are with each other. In general, the matrix is time-variant but can be approximated as time-invariant for many applications. The matrix has two properties which will be important:

1. It is hermitian, i.e. $\mathbf{\Phi}_{zz}(k, f) = \mathbf{\Phi}_{zz}{}^{\mathsf{H}}(k, f)$.
2. It is positive-semidefinite, i.e. $\mathbf{u}^{\mathsf{H}}\mathbf{\Phi}_{zz}(k, f)\mathbf{u} \geq 0$.

If we further assume that the target signal is uncorrelated to all noise sources, the SCM of the observation can be written as

$$\begin{aligned}
\mathbf{\Phi}_{yy}(k, f) =& \mathbb{E}\left[\mathbf{y}(k, f)\mathbf{y}^{\mathsf{H}}(k, f)\right] \\
=& \mathbb{E}\Big[\left(\mathbf{a}_s(f)s(k, f)\right)\left(\mathbf{a}_s(f)s(k, f)\right)^{\mathsf{H}} \\
&\quad + \left(\mathbf{A}_n(f)\mathbf{n}(k, f) + \mathbf{d}(k, f)\right)\left(\mathbf{A}_n(f)\mathbf{n}(k, f) + \mathbf{d}(k, f)\right)^{\mathsf{H}}\Big] \\
=& \mathbf{a}_s(f)\mathbf{a}_s{}^{\mathsf{H}}(f)\,\mathbb{E}\left[s(k, f)s^*(k, f)\right] + \mathbf{A}_n(f)\,\mathbb{E}\left[\mathbf{n}(k, f)\mathbf{n}^{\mathsf{H}}(k, f)\right]\mathbf{A}_n{}^{\mathsf{H}}(f) \\
&\quad + \mathbb{E}\left[\mathbf{d}(k, f)\mathbf{d}^{\mathsf{H}}(k, f)\right] \\
=& \mathbf{\Phi}_{ss}(f) + \tilde{\mathbf{\Phi}}_{nn}(f) + \mathbf{\Phi}_{dd}(f) \\
=& \mathbf{\Phi}_{ss}(f) + \mathbf{\Phi}_{nn}(f) \\
=& \mathbf{\Phi}_{yy}(f).
\end{aligned} \tag{4.10}$$

Note that, due to the assumptions about the AIRs formulated in Sec. 4.1.2, the SCM does not depend on the frame index k. Ways to estimate the matrix for a specific signal are discussed later in greater detail.

The following first reviews three important optimization criteria, their solutions and discusses their similarities and differences afterwards. To improve the readability, the time and frequency dependency will be omitted whenever it is not ambiguous.

Multi-channel Wiener filter

The multi-channel Wiener filter (MWF) aims to reduce the expected MSE between the beamformed signal and the target signal at a reference microphone[3]:

$$\mathbf{w}_{\mathrm{MWF}} = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathbb{E}\left[\left|\mathbf{w}^{\mathsf{H}}\mathbf{y} - s\right|^2\right] \tag{4.11}$$

With $\mathbf{y} = \mathbf{a}_s s + \mathbf{A}_n \mathbf{n} + \mathbf{d}$ and the introduction of a trade-off parameter μ, the problem can be reformulated as [Wan+18]

$$\mathbf{w}_{\mathrm{MWF}} = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathbb{E}\left[\left|(\mathbf{w}^{\mathsf{H}}\mathbf{a}_s - 1)s\right|^2\right] + \mu\, \mathbb{E}\left[\left|\mathbf{w}^{\mathsf{H}}(\mathbf{A}_n\mathbf{n} + \mathbf{d})\right|^2\right]. \tag{4.12}$$

The first expectation describes the speech distortion while the second expectation describes the residual noise power. The factor μ controls the trade-off between both terms.

Using the SCM for Eq. 4.11 and the one-hot vector $\mathbf{u}_{\mathrm{ref}} = [1, 0, \ldots, 0]$ which points to the reference microphone, the solution to this optimization problem is [SMW04]

$$\boxed{\mathbf{w}_{\mathrm{MWF}} = \mathbf{\Phi}_{ss}\left(\mathbf{\Phi}_{ss} + \mu\mathbf{\Phi}_{nn}\right)^{-1}\mathbf{u}_{\mathrm{ref}}}. \tag{4.13}$$

MPDR/MVDR beamformer

The minimum power distortionless response (MPDR) and minimum variance distortionless response (MVDR) beamformers share a similar criterion. While the MPDR aims to minimize the power of the beamformed signal, the MVDR aims to minimize the power of the residual noise. Both impose the constraint that the filter response is equal to one for a desired direction $\mathbf{h}$. Mathematically, this is expressed with the help of a Lagrange multiplier:

$$\mathbf{w}_{\mathrm{MPDR}} = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathbb{E}\left[\mathbf{w}^{\mathsf{H}}\mathbf{\Phi}_{yy}\mathbf{w}\right] \text{ subject to } \mathbf{w}^{\mathsf{H}}\mathbf{h} = 1, \tag{4.14}$$

$$\mathbf{w}_{\mathrm{MVDR}} = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathbb{E}\left[\mathbf{w}^{\mathsf{H}}\mathbf{\Phi}_{nn}\mathbf{w}\right] \text{ subject to } \mathbf{w}^{\mathsf{H}}\mathbf{h} = 1. \tag{4.15}$$

The solutions to these optimization problems are the MPDR and MVDR beamforming vector respectively

$$\boxed{\mathbf{w}_{\mathrm{MPDR}} = \frac{\mathbf{\Phi}_{yy}{}^{-1}\mathbf{h}}{\mathbf{h}^{\mathsf{H}}\mathbf{\Phi}_{yy}{}^{-1}\mathbf{h}}}, \tag{4.16}$$

$$\boxed{\mathbf{w}_{\mathrm{MVDR}} = \frac{\mathbf{\Phi}_{nn}{}^{-1}\mathbf{h}}{\mathbf{h}^{\mathsf{H}}\mathbf{\Phi}_{nn}{}^{-1}\mathbf{h}}}. \tag{4.17}$$

[3]Here we assume the RTF, i.e., $a_{\mathrm{ref}} = 1$

Although different covariance matrices are used, the resulting beamforming vectors are theoretically identical under the assumed signal model [BSH08, p. 275]. However, in practice the results may differ due to numerical issues and estimation errors. The MPDR is usually easier to estimate since the SCM of the observation is available whereas it is more difficult to estimate the statistics for the interferences only. However, the SCM of the observation is in general subject to a higher time variance compared to the SCM of the interferences which is why the MVDR is preferred in many applications.

GEV beamformer

The generalized eigenvalue (GEV) beamformer aims to maximize the SNR after the beamforming operation

$$\mathbf{w}_{\mathrm{GEV}} = \underset{\mathbf{w}}{\operatorname{argmax}} \frac{\mathbf{w}^{\mathsf{H}} \mathbf{\Phi}_{ss} \mathbf{w}}{\mathbf{w}^{\mathsf{H}} \mathbf{\Phi}_{nn} \mathbf{w}}. \tag{4.18}$$

The ratio in Eq. 4.18 is also known as the Rayleigh quotient and the solution to Eq 4.18 leads to the generalized eigenvalue problem

$$\mathbf{\Phi}_{ss} \mathbf{w} = \lambda \mathbf{\Phi}_{nn} \mathbf{w}. \tag{4.19}$$

With the help of the Cholesky factorization[4] $\mathbf{\Phi}_{nn} = \mathbf{L}\mathbf{L}^{\mathsf{H}}$ this can be reformulated as a regular eigenvalue problem[5]

$$\left(\mathbf{L}^{-1} \mathbf{\Phi}_{ss} \mathbf{L}^{-\mathsf{H}}\right) \left(\mathbf{L}^{\mathsf{H}} \mathbf{w}\right) = \lambda \left(\mathbf{L}^{\mathsf{H}} \mathbf{w}\right) \tag{4.20}$$

for the transformed vector $\tilde{\mathbf{w}} = \mathbf{L}^{\mathsf{H}} \mathbf{w}$ and matrix $\tilde{\mathbf{\Phi}} = \mathbf{L}^{-1} \mathbf{\Phi}_{ss} \mathbf{L}^{-\mathsf{H}}$. The vector maximizing the right hand side expression of Eq. 4.18 is the eigenvector corresponding to the largest eigenvalue, i.e.

$$\mathbf{e}, \mathbf{V} = \mathcal{P}\left(\tilde{\mathbf{\Phi}}\right) \tag{4.21}$$

$$\mathbf{w}_{\mathrm{GEV}} = \mathbf{L}^{-\mathsf{H}} \mathbf{v}_{\mathbf{e}_{\max}}, \tag{4.22}$$

where $\mathbf{e}$ and $\mathbf{V}$ are the eigenvalues and eigenvectors of $\tilde{\mathbf{\Phi}}$ respectively. The transformation with $\mathbf{L}^{-\mathsf{H}}$ is necessary to get back from the space of the regular eigenvalue problem to the one of the generalized eigenvalue problem.

[4] The factorization requires a positive-definite matrix. While covariance matrices are always positive-definite, the estimated *empirical* covariance matrices are not guaranteed to be positive-definite due to numerical issues. This can be mitigated by a proper conditioning of the matrices in the implementation.

[5] Another way is to multiply with the inverse of $\mathbf{\Phi}_{nn}$ but the resulting matrix $\mathbf{\Phi}_{nn}{}^{-1} \mathbf{\Phi}_{ss}$ is not hermitian anymore, which is why this approach is not recommend.

Rank-1 constraint

While the previous solutions exploit the assumption that the target and interference are uncorrelated, they do not take into account another assumption of the signal model. Namely the Rank-1 property of the SCM of the target. Considering this constraint reveals the connections between the different beamformer variants.

With $\mathbf{\Phi}_{ss} = \mathbf{a}_s \mathbf{a}_s^{\mathsf{H}} \phi_{ss}$, the MWF solution from Eq. 4.13 can be written as [Wan+18]

$$\mathbf{w}_{\mathrm{MWF_{R1}}} = \frac{\mathbf{\Phi}_{nn}{}^{-1} \mathbf{a}_s \mathbf{a}_s^{\mathsf{H}} \phi_{ss}}{\mu + \mathrm{tr}\left\{ \mathbf{\Phi}_{nn}{}^{-1} \mathbf{a}_s \mathbf{a}_s^{\mathsf{H}} \phi_{ss} \right\}} \mathbf{u}_{\mathrm{ref}}. \tag{4.23}$$

Setting $\mu = 0$ and using $\mathrm{tr}\left\{ \mathbf{\Phi}_{nn}^{-1} \mathbf{a}_s \mathbf{a}_s^{\mathsf{H}} \phi_{ss} \right\} = \mathbf{a}_s^{\mathsf{H}} \mathbf{\Phi}_{nn}^{-1} \mathbf{a}_s \phi_{ss}$ recovers the MVDR solution from Eq. 4.15 when the distortionless direction is chosen such that $\mathbf{h} = \mathbf{a}_s$

$$\mathbf{w}_{\mathrm{MWF_{R1}}}\Big|_{\mu=0} = \frac{\mathbf{\Phi}_{nn}^{-1} \mathbf{a}_s}{\mathbf{a}_s^{\mathsf{H}} \mathbf{\Phi}_{nn}^{-1} \mathbf{a}_s} = \mathbf{w}_{\mathrm{MVDR}}\Big|_{\mathbf{h}=\mathbf{a}_s}. \tag{4.24}$$

For the GEV-criterion, Eq. 4.20 becomes

$$\left(\mathbf{L}^{-1} \mathbf{a}_s \mathbf{a}_s^{\mathsf{H}} \mathbf{L}^{-\mathsf{H}} \right) \left(\mathbf{L}^{\mathsf{H}} \mathbf{w} \right) = \frac{\lambda}{\phi_{ss}} \left(\mathbf{L}^{\mathsf{H}} \mathbf{w} \right), \tag{4.25}$$

with the Rank-1 matrix $\left(\mathbf{L}^{-1} \mathbf{a}_s \mathbf{a}_s^{\mathsf{H}} \mathbf{L}^{-\mathsf{H}} \right) = \tilde{\mathbf{\Phi}} = \mathbf{u}\mathbf{v}^{\mathsf{H}}$ where $\mathbf{u}$ and $\mathbf{v}$ are the right and left eigenvectors respectively [Mon05]. In this case, the desired beamforming vector is

$$\mathbf{w}_{\mathrm{GEV_{R1}}} = \mathbf{L}^{-\mathsf{H}} \mathbf{L}^{-1} \mathbf{a}_s = \mathbf{\Phi}_{nn}^{-1} \mathbf{a}_s. \tag{4.26}$$

Comparing Eq. 4.24 with Eq. 4.26 reveals that – under the Rank-1 constraint – all three beamforming vectors point in the same direction in space but scale the signal differently.

In a real scenario, the spatial covariance matrix of the speech signal must be estimated and will never be exactly of Rank-1. To improve the estimate, it is possible to enforce the Rank-1 property. The simplest solution is to use the eigenvector with the largest eigenvalue as $\hat{\mathbf{a}}_{s_{\mathrm{EVD}}}$ and

$$\hat{\mathbf{\Phi}}_{ss_{\text{R1-EVD}}} = \hat{\mathbf{a}}_{s_{\mathrm{EVD}}} \hat{\mathbf{a}}_{s_{\mathrm{EVD}}}^{\mathsf{H}} \frac{\mathrm{tr}\{\mathbf{\Phi}_{ss}\}}{\mathrm{tr}\left\{ \hat{\mathbf{a}}_{s_{\mathrm{EVD}}} \hat{\mathbf{a}}_{s_{\mathrm{EVD}}}^{\mathsf{H}} \right\}} \tag{4.27}$$

as an approximation of the Rank-1 spatial covariance matrix [Wan+18].

A more robust estimation also includes the spatial covariance matrix of the noise during the approximation of $\hat{\mathbf{a}}_{x_{\mathrm{GEVD}}}$ [Wan+18] [MGC09]

$$\hat{\mathbf{a}}_{s_{\mathrm{GEVD}}} = \mathbf{\Phi}_{nn} \mathbf{w}_{\mathrm{GEV}} \tag{4.28}$$

$$\hat{\mathbf{\Phi}}_{ss_{\text{R1-GEVD}}} = \hat{\mathbf{a}}_{s_{\mathrm{GEVD}}} \hat{\mathbf{a}}_{s_{\mathrm{GEVD}}}^{\mathsf{H}} \frac{\mathrm{tr}\{\mathbf{\Phi}_{ss}\}}{\mathrm{tr}\left\{ \hat{\mathbf{a}}_{s_{\mathrm{GEVD}}} \hat{\mathbf{a}}_{s_{\mathrm{GEVD}}}^{\mathsf{H}} \right\}}. \tag{4.29}$$

Including the spatial covariance matrix of the noise is especially beneficial if the estimate of $\mathbf{\Phi}_{ss}$ also includes interference sources [MGC09]. With this approximation all three beamformers again point in the same direction and are equal up to a complex scalar factor.

Scaling

Recalling that the beamforming vectors are computed for each frequency independently shows that their scaling can have a big impact on the whole signal. For example, frequencies with high noise power can be almost muted by the GEV beamformer (due to the projection with $\mathbf{\Phi}_{nn}^{-1}$) leading to hearable distortions. As mentioned, this might be a desirable behavior when the task is speech recognition and the acoustic model handles such distortions (i.e. missing/suppressed frequency bands) better than certain noises. But for a perceptually motivated goal these distortions are almost always undesired. A way to mitigate them is to apply a post-filter. Note that this issue does not arise for the MVDR due to the distortionless constraint and the MWF because of the criterion which aims to recover the speech image at a reference microphone. In [WH07] the authors propose what they call a blind analytical normalization

$$\mathbf{w}_{\text{GEV}}^{\text{BAN}} = \frac{\sqrt{\mathbf{w}_{\text{GEV}}{}^{\mathsf{H}} \mathbf{\Phi}_{nn} \mathbf{\Phi}_{nn} \mathbf{w}_{\text{GEV}} / M}}{\mathbf{w}_{\text{GEV}}{}^{\mathsf{H}} \mathbf{\Phi}_{nn} \mathbf{w}_{\text{GEV}}} \mathbf{w}_{\text{GEV}}. \tag{4.30}$$

Again, the Rank-1 approximation reveals the connection to the other beamformer formulations. Under this constraint, the beamforming vector becomes

$$\mathbf{w}_{\text{GEV}_{\text{R1}}}^{\text{BAN}} = \frac{\sqrt{\mathbf{a}_s{}^{\mathsf{H}} \mathbf{a}_s / M}}{\mathbf{a}_s{}^{\mathsf{H}} \mathbf{\Phi}_{nn}{}^{-1} \mathbf{a}_s} \mathbf{\Phi}_{nn}{}^{-1} \mathbf{a}_s. \tag{4.31}$$

Comparing this with Eq. 4.24 shows that this corresponds to the distortionless MWF and MVDR except that it is scaled with the square-root of the mean power of the AIR vector instead of a reference value of this vector.

In this work we propose and use a different normalization. We divide the noise SCM by its trace which results in $\mathbf{L}'^{\mathsf{H}} = \frac{\mathbf{L}}{\operatorname{tr} \mathbf{\Phi}_{nn}}^{\mathsf{H}} = \sqrt{\sum_j \mathbf{e}_j{}^{\mathsf{H}} \mathbf{\Phi}_{nn} \mathbf{e}_j} \mathbf{L}^{\mathsf{H}}$ and thus the scaled GEV beamforming vector

$$\mathbf{w}_{\text{GEV}}^{\text{trace}} = \sqrt{\sum_j \mathbf{e}_j{}^{\mathsf{H}} \mathbf{\Phi}_{nn} \mathbf{e}_j} \mathbf{w}_{\text{GEV}}. \tag{4.32}$$

Using the Rank-1 approximation yields

$$\mathbf{w}_{\text{GEV}_{\text{R1}}}^{\text{trace}} = \sqrt{\sum_j \mathbf{e}_j{}^{\mathsf{H}} \mathbf{\Phi}_{nn} \mathbf{e}_j} \mathbf{\Phi}_{nn}{}^{-1} \mathbf{a}_s \tag{4.33}$$

4.3 Spatial covariance matrix estimation

All different beamformer formulations presented in Sec. 4.2.2 rely on the estimation of either the SCM of the target source and/or the SCM of the interferences. This information is not available in practice and has to be estimated from the observed signals.

A rather simple way to estimate the SCM for the interferences it to assume that the target is not active in the first seconds of a recording. The SCM can then simply be calculated with Eq. 4.9 using the observed signal. Depending on the beamformer, a direction of distortionless response or the SCM of the target speech must then be estimated. A coarse estimation can be achieved with a voice activity detection (VAD) extracting frames containing mostly the target speaker. However, working on the frame level inevitably introduces estimation errors. These can be avoided with a more fine grained consideration of individual tf-bins.

Defining continuous spectral masks $\in [0, 1]$ for the k-th frame and frequency bin f as $\gamma_{\text{target}}(k, f)$ and $\gamma_{\text{interferences}}(k, f)$ respectively, the SCMs can be approximated as [Sou+13]

$$\mathbf{\Phi}_{ss}(f) = \frac{1}{K} \sum_{k} \gamma_{\text{target}}(k, f) \mathbf{y}(k, f) \mathbf{y}(k, f)^{\mathsf{H}} \tag{4.34}$$

and

$$\mathbf{\Phi}_{nn}(f) = \frac{1}{K} \sum_{k} \gamma_{\text{interferences}}(k, f) \mathbf{y}(k, f) \mathbf{y}(k, f)^{\mathsf{H}}. \tag{4.35}$$

Estimating the SCMs with this weighted sum of outer products of the observed signals assumes that these signals are ergodic (i.e. averaging over time gives the same results as averaging over realizations). In other words, it is assumed that the speaker does not move and the scenario, i.e., the interferences, does not change during the aggregation of the statistics. Further assumptions are that the speech signal and interferences are sparse, uncorrelated and disjoint [Aok+01; YR04]. The later property implies that each tf-bin is dominated by one source and most of that source energy is captured by accumulating those tf-bins it dominates. Those tf-bins are indicated by $\gamma_{\text{target}}(k, f)$ and $\gamma_{\text{interferences}}(k, f)$ respectively which are also referred to as speech presence probability (SPP) and noise presence probability (NPP)[6].

4.4 Statistical mask estimation

Traditionally, the SPP and NPP have been estimated using statistical signal models [SAM07; Ito+14; IAN13; Yos+15; AN11; Ara+16]. Noticeable, the system presented in [Yos+15] showed the best performance in the third Computational Hearing in Multisource Environments (CHiME) challenge[7]. It uses a time-varying complex Gaussian mixture model (CGMM) [Ito+14] to estimate a spectral mask for the target speech

[6] During this work we will use the term mask and SPP/NPP synonymously

[7] http://spandh.dcs.shef.ac.uk/chime_challenge/chime2015/results.html

source to compute the SCMs for an MVDR beamformer. In [IAN16] Ito et al. introduce the closely related complex angular central Gaussian mixture model (cACGMM) model, show that the EM algorithm is equivalent to the one used for the time-variant CGMM and conclude that it can achieve the same (mask estimation) performance with less information as the (instantaneous) signal power is not used. The following explains this model in greater detail. It serves as the baseline for a statistical model for the evaluation and also plays an important role for the unsupervised training which will be presented later in Ch. 9.

Assuming sparseness of the target speech source as already mentioned in Sec. 4.3, the observations are modeled with a mixture model with Q classes. For the scenario considered in this work, we set $Q = 2$, i.e. one class for speech and one for the interferences. But in general, more classes can be represented, for example in blind source separation (BSS) scenarios with multiple speakers.

The distribution of the multi-channel observation can be modeled as a mixture of class conditional distributions $p(\mathbf{y}(k,f);\boldsymbol{\theta}_q)$ with class dependent parameters $\boldsymbol{\theta}_q$ for the q-th class

$$p(\mathbf{y}(k,f);\boldsymbol{\theta}) = \sum_q \pi_q(f) p(\mathbf{y}(k,f);\boldsymbol{\theta}_q). \tag{4.36}$$

The a-priori probability, that an observation belongs to mixture component q, is frequency dependent and expressed by $\pi_q(f)$. This generic formulation marginalizes over all classes and assumes that all observations are i.i.d. given the class affiliation.

As suggested above, there are multiple suitable conditional distributions $p(\mathbf{y}(k,f);\boldsymbol{\theta}_q)$ to choose from to obtain a concrete model. Successfully used examples include Gaussian distributions [MEJ07; Ara+09], complex Watson distribution [TH10; SAM11] and the mentioned time-varying complex Gaussian [Ito+14] and complex angular central Gaussian distribution [Ken97; IAN16].

The latter distribution with the class dependent parameter matrix $\mathbf{B}_q(f)$ is specified by

$$p(\tilde{\mathbf{y}}(k,f);\mathbf{B}_q(f)) = \frac{(M-1)!}{2\pi^M \det \mathbf{B}_q(f)} \frac{1}{(\tilde{\mathbf{y}}(k,f)^{\mathrm{H}} \mathbf{B}_q(f)^{-1} \tilde{\mathbf{y}}(k,f))^M}. \tag{4.37}$$

Instead of the observation itself, the distribution of the normalized observation $\tilde{\mathbf{y}}(k,f) = \mathbf{y}(k,f)/\|\mathbf{y}(k,f)\|$ is modeled. This way it does not account for the time-varying power of the observation but can still capture inter-channel level differences. Also, while inter-channel phase differences can be captured, the model is invariant to the absolute phase as can be seen by

$$(\tilde{\mathbf{y}}(k,f)\exp(\mathrm{j}\phi))^{\mathrm{H}} \mathbf{B}_q(f)^{-1} \tilde{\mathbf{y}}(k,f)\exp(\mathrm{j}\phi) = \tilde{\mathbf{y}}(k,f)^{\mathrm{H}} \mathbf{B}_q(f)^{-1} \tilde{\mathbf{y}}(k,f). \tag{4.38}$$

Note, that each tf-bin is treated independently. In other words, the model does not take into account any spectral or temporal correlations. As a consequence, this is a spatial only model, i.e. it cannot learn any concept of speech (spectro-temporal patterns) but rather models the spatial distribution of the signal and clusters it into Q classes. The model parameters $\boldsymbol{\theta}$ are estimated using the EM algorithm and the alternating updates

take the following form [IAN16]

$$\gamma_q(k,f) = \frac{\pi_q(f)\frac{1}{\det \mathbf{B}_q(f)}\frac{1}{(\tilde{\mathbf{y}}(k,f)^{\mathrm{H}}\mathbf{B}_q^{-1}(f)\tilde{\mathbf{y}}(k,f))^M}}{\sum_q \pi_q(f)\frac{1}{\det \mathbf{B}_q(f)}\frac{1}{(\tilde{\mathbf{y}}(k,f)^{\mathrm{H}}\mathbf{B}_q^{-1}(f)\tilde{\mathbf{y}}(k,f))^M}}, \tag{4.39}$$

$$\pi_q(f) = \frac{1}{K}\sum_k \gamma_q(k,f), \tag{4.40}$$

$$\mathbf{B}_q(f) = M\sum_k \gamma_q(k,f)\frac{\tilde{\mathbf{y}}(k,f)\tilde{\mathbf{y}}(k,f)^{\mathsf{H}}}{\tilde{\mathbf{y}}(k,f)\mathbf{B}_q^{-1}(f)\tilde{\mathbf{y}}(k,f)^{\mathsf{H}}} \Big/ \sum_t \gamma_q(k,f). \tag{4.41}$$

For the task at hand (mask estimation), we are interested in $\gamma_q(k,f)$ which is the posterior probability that the frequency bin of the k-th frame and frequency f is dominated by the source of class q given the observation. This can be plugged into Eq. 4.34 and Eq. 4.35 to obtain the estimates for the target and interferences SCM respectively.

Note that the update rules themselves contain an inner iteration due to the implicit definition of the covariance matrix $\mathbf{B}_q(f)$ in Eq. 4.41. The number of inner iterations and EM update steps can be chosen by optimization on sample data. In order to start with the iterations, the EM algorithm needs an initialization. There are two possible options here. If prior knowledge about the spatial characteristics of the scenario is available, this can be used to initialize the covariance matrices $\mathbf{B}_q(f)$ combined with informed or uninformative values for the mixture weights $\pi_q(f)$. This information is sufficient to obtain a first estimate of the affiliation posteriors $\gamma_q(k,f)$ and the iterative process can be started from there. However, for many scenarios such information is not available a priori. For these cases, one can start with Eq. 4.41, initially assume an identity matrix for the covariances $\mathbf{B}_q(f)$ and an initialization of the posteriors $\gamma_q(k,f)$. Again, prior knowledge can be exploited here for example by using an initialization depending on an estimated or known source activity (see e.g., [Boe+18a]). In more uninformed settings, the posteriors are initialized more or less randomly according to heuristics found to be working well for specific scenarios.

A big advantage of the statistical approach is that it is completely unsupervised[8]. The model parameters are estimated from the current observation and the spatial information contained in it. No knowledge about the characteristics of the sources is required, it is invariant against global scaling of the signals and no adaptation is needed for different domains. This flexibility, however, comes at a cost.

Since the model treats each frequency independently, a permutation problem arises (see Sec. 4.1.3). This frequency permutation problem [SAM07] describes the fact that the class indices are inconsistent over the frequency bins, i.e. for one frequency the target could be assigned to class index 0 while for other frequencies it is associated with index 1. The assignment is ultimately determined by the initialization of the EM algorithm described above. In the worst case this means that the initialization and therefore the assignment is random. An additional permutation solver step addresses this issue but cannot completely eliminate it [SAM07]. This work uses a method proposed in [SAM07]

[8]The number of classes has to be specified for the presented approach but is assumed to be known in the considered scenario.

which maximizes the correlation of the masks along neighboring frequencies to compute a permutation alignment.

The frequency permutation problem is not the only issue that arises due to the unsupervised nature of the statistical approach. Even if this permutation problem is solved and the class assignments are aligned perfectly across frequencies, there is still the problem of assigning one class to the target and one class to the noise. We refer to this as the global permutation problem and it needs to be solved before the posteriors can be used in a downstream task such as beamforming. Again, several heuristics exist to find a suitable alignment for a given scenario. In this work, we opt for a rather simple but yet effective one which exploits the sparsity of the target, namely the speech source. Since we assume there are only two classes and the posteriors for each time-frequency bin sum to one, the other class must broadly capture all other source of the acoustic scene. Therefore, the class with a lower value for the summation of the respective posterior over all tf-bins is associated with the target.

4.5 Dereverberation

The acoustic beamforming detailed in the previous section mainly aims at reducing the impact of undesired sources at different spatial locations. However, the signal can also be deteriorated without the presence of other sound sources due to delayed reflections of the target signal itself, i.e., reverberation. We now focus on this phenomenon.

We assume that there is only one source present (the speaker) and that the AIR is long and its power decays slowly. The delayed source signals then cause distortions which have a severe impact on the intelligibility of the speech signal and significantly impacts the performance of an ASR system, irrespective of the size of the training corpus [Li+17; Har15]. A common way to specify the reverberation is the T_{60} time (also called reverberation time (RT)). This measures the time it takes until the power of the AIR tail decays by 60 dB. Typical values range from 0.2 s – 2 s and are largely dependent on the room size, its floor and wall materials as well as objects positioned in the room absorbing the sound waves. Dereverberation aims to shorten this length, keeping the direct components and early reflections while suppressing the reverberation tail. The combination of the direct components and early reflections is called the speech image or also anechoic speech in the following.

Starting with the signal model from Eq. 4.2 and by splitting the AIR into two parts, the signal recorded by the m-th sensor can be decomposed into

$$y_m(t) = \sum_{\tau=0}^{L_{\text{early}}} a_{m,s}(\tau) s(t-\tau) + \sum_{\tau=L_{\text{early}+1}}^{A_{m,s}} a_{m,s}(\tau) s(t-\tau). \tag{4.42}$$

The length L_{early} should be chosen such that the direct signal and early reflections are captured[9]. Neglecting subband energy leakage and switching to vector notation, the

[9] A common choice, which is also adopted here, is 50 ms which corresponds to $L_{\text{early}} = 800$ samples for a 16 kHz signal

observed signal in the STFT-domain is denoted as [YN12]

$$\mathbf{y}(k,f) = \sum_{\tau=0}^{L_A} \mathbf{a}(\tau,f)s(k-\tau,f) \tag{4.43}$$

and can be decomposed into

$$\mathbf{y}(k,f) = \sum_{\tau=0}^{\Delta} \mathbf{a}(\tau,f)s(k-\tau,f) + \sum_{\tau=\Delta+1}^{L_A} \mathbf{a}(\tau,f)s(k-\tau,f). \tag{4.44}$$

Here, $L_A = A_{mx}/B$ is the length of the AIR filter in the STFT domain [Nak+10]. Δ distinguishes between the part regarded as the image and the one regarded as the reverberation tail[10].

There are many approaches tackling the task of dereverberation. They can be broadly categorized in two categories: Linear filter approaches and spectral subtraction methods [Li+15]. This work focuses on the first category which has shown to be effective for ASR [Kin+16].

4.5.1 Dereverberation with beamforming

Due to the multiple paths, the reverberation tail is received by the array as a spatially diffuse signal and some early reflections are received as spatial point sources originating from the point of reflection. Consequently, it is possible to use one of the spatial filters discussed in the previous section to enhance the signal. Doing so obviously violates some of the assumptions made earlier. First, depending on the choice of Δ, the target, i.e., the anechoic speech, and interfering signals, i.e., the late reverberations, are likely to be correlated since

$$\mathbb{E}\left[\sum_{\tau=0}^{\Delta} \mathbf{a}(\tau,f)s(k-\tau,f)\left(\sum_{\tau=\Delta+1}^{L_A} \mathbf{a}(\tau,f)s(k-\tau,f)\right)^{\mathrm{H}}\right] \neq 0. \tag{4.45}$$

Or, in other words, the cross-terms when calculating the SCM in Eq. 4.11 are not zero and the SCM of the observation cannot simply be expressed as the sum of the SCMs of the target and the interference.

Second, the AIR for the target signal is not a single vector anymore and the corresponding SCM $\mathbf{\Phi}_{ss}$ is not a Rank-1 matrix.

However, despite these violations, such an approach shows good performance in practice.

4.5.2 Weighted predictive error

A different approach, WPE, has shown good performance in the REVERB challenge [Kin+13] and also found its way into a commercial product [Li+17]. It operates

[10] In practice, depending on the STFT transformation parameters, a reasonable value for Δ ranges from 1 to 3.

on a single channel or in a multiple-input multiple-output fashion on multi-channel data.

WPE estimates the current reverberation tail using previous samples and then subtracts the estimate from the observation. This results in an optimal estimate of the anechoic speech in a maximum likelihood sense:

$$\hat{\mathbf{s}}(k,f) = \mathbf{y}(k,f) - \mathbf{G}(f)^{\mathrm{H}} \bar{\mathbf{y}}(k-\Delta, f). \tag{4.46}$$

The entries of the matrix $\mathbf{G}(f)^{\mathrm{H}} \in \mathbb{C}^{M \times ML_{\text{taps}}}$ are the filter coefficients and

$$\bar{\mathbf{y}}(k-\Delta, f) = \left[\mathbf{y}^{\mathrm{T}}(k-\Delta, f), \ldots, \mathbf{y}^{\mathrm{T}}(k-\Delta-L_{\text{taps}}+1, f)\right]^{\mathrm{T}} \in \mathbb{C}^{ML_{\text{taps}}} \tag{4.47}$$

is a stacked representation of the previous observations. Note that using $\Delta \geq 1$ avoids whitening of the speech source.

To obtain the filter coefficients, WPE maximizes the likelihood of the model under the assumption that the anechoic signal is a realization of a zero-mean circularly-symmetric complex Gaussian with an unknown time-varying variance $\lambda_{t,f}$. This value, which represents the power spectral density (PSD) of the speech image, needs to be estimated well enough in order to achieve good dereverberation performance. Because this signal is unknown in the first place, there is no closed form solution for the likelihood optimization and WPE is an iterative method which alternates between two steps:

Step 1)
$$\lambda(k,f) = \frac{1}{(\delta+1+\delta)\,M} \sum_{\tau=k-\delta}^{k+\delta} \sum_{m=0}^{M-1} |\hat{s}(\tau,f,m)|^2 \tag{4.48}$$

Step 2)
$$\mathbf{R}(f) = \sum_k \frac{\tilde{\mathbf{y}}(k-\Delta,f)\tilde{\mathbf{y}}(k-\Delta,f)^{\mathrm{H}}}{\lambda(k,f)} \in \mathbb{C}^{ML_{\text{taps}} \times ML_{\text{taps}}}, \tag{4.49}$$

$$\mathbf{p}(f,d) = \sum_k \frac{\tilde{\mathbf{y}}(k-\Delta,f)\mathbf{y}(k,f)^{\mathrm{H}}}{\lambda(k,f)} \in \mathbb{C}^{ML_{\text{taps}} \times M}, \tag{4.50}$$

$$\mathbf{G}(f) = \mathbf{R}(\mathbf{f})^{-1}\mathbf{p}(f) \in \mathbb{C}^{DL_{\text{taps}} \times M}. \tag{4.51}$$

The first step estimates the PSD given the current estimate of the anechoic signal using a heuristic smoothing scheme to improve the estimate [YN12]. The second step updates the filter coefficients given the estimate of the PSD.

4.6 Summary

This chapter introduced signal processing methods aiming to extract a (speech) source in the presence of interferences from a multi-channel mixture. Starting from a signal model for the mixture we especially focused on statistical beamforming methods. These methods estimate a linear spatial filter from the SCMs of the target and the interferences. Three different criteria to estimate these filters were presented, namely the MWF, the MVDR and GEV beamformer. We showed that, under a Rank-1 assumption of the target SCM, all of these criteria result in the same look direction but scale the frequencies differently. Afterwards, we discussed how the SCMs can be estimated from

the observation. A mask attributes individual tf-bins to one of the sources and the SCM is then estimated by a weighted average of the outer products. The masks can be obtained as the posterior for the class affiliations of a spatial mixture model. For this scenario, the cACGMM has been proven to be especially suitable and we extensively discussed this specific model. Finally, we focused on distortions caused by reverberation. These can also be mitigated with the help of a statistical beamformer when we divide the RIR into two parts and estimate the SCM for each. Another particular successful technique for dereverberation is WPE which predicts the current reverberation from past observations with a linear filter and then subtracts it from the current observation.

5 Datasets, setup and baselines

This chapter introduces two datasets that will be used throughout this thesis to evaluate and compare proposed systems. Namely, the CHiME (3) task and the Reverberant Voice Enhancement and Recognition Benchmark (REVERB) task. Both were created in the context of an official challenge to benchmark and advance ASR systems. While the difficulty of the CHiME corpus is mainly due to distorting noise sources, the REVERB data focus, as the name already suggests, on deterioration caused by reverberation. Combined, the two corpora cover the two main challenging environmental distortions for ASR discussed in this thesis. Using them allows for a meaningful evaluation and comparison of the systems which will be proposed in the next chapters. The evaluation setup and metrics will be the same for all of these systems throughout this thesis. They are detailed in Sec. 5.2. Finally, Sec. 5.3 summarizes results of other works on the two corpora which will serve as baselines for a comparison.

5.1 Datasets

5.1.1 CHiME

CHiME is a series of challenges with the goal of comparing and improving speech recognition in everyday environments. The first challenge dataset comprised artificial mixtures of speech commands and recorded noise typical of everyday listening conditions [Bar+13]. A larger vocabulary size and a more realistic mixing process was used in the second challenge to increase the level of difficulty [Vin+13]. The third and the fourth challenge share the same dataset which, compared to the second challenge, featured real recordings in noisy environments and a custom recording device [Bar+15]. While the previous challenges all used perfectly separated read speech sentences, the latest challenge, CHiME 5, deals with conversational speech recognition with recordings from a cocktail party scenario.

This work uses the dataset from the third and forth challenge to evaluate the presented systems.

The dataset consists of real and simulated recordings in four different environments, namely, in a cafe, a bus, a pedestrian area and on a street junction. For the real recordings, a tablet with six microphones is used. The microphones are located in the upper and lower frame with a spacing of 10 cm and aligned to the center. Five of them face forward towards the speaker in front of the tablet. The one in the center of the upper frame faces backwards. All channels are recorded sample synchronous. 12

Task	Name	Type	#utterances	#words	#hours
Training		simulated	7138	129410	15.1
		real	1600	26759	2.9
Development		simulated	1640	27118	2.9
		real	1640	21409	2.8
Evaluation	SIMU	simulated	1320	21411	2.3
	REAL	real	1320	21409	2.2

Table 5.1: Summary of the CHiME corpus.

speakers (6 male, 6 female) read prompts taken from the Wall Street Journal (WSJ) dataset (WSJ0 [Gar+93]) displayed by the tablet. Each speaker made roughly 100 recordings in each of the environments.

For the simulated data, speech recordings from the WSJ0 dataset are mixed with multi-channel noise recordings from the tablet device in one of the four environments. A time-varying filter is used to model the acoustic path between the speaker and the microphones before the filtered speech signal is added to the noise. To avoid larger mismatches with the real recordings, this filter is estimated from one of them.

Overall, there are 7138 simulated and 1600 real recordings available for training which are evenly distributed across the four environments. Further, two development and two evaluation sets are available.

One set with real recordings and one with simulated recordings. The development sets contain 410 real and simulated recordings for each environment, i.e. one set consists of 1640 recordings. The evaluation sets are a bit smaller with 330 recordings for each condition. Both sets have a closed vocabulary, i.e. all words appearing in these sets are known in advance and included in the lexicon and language model. In total, there are 5000 different words. Specific to this task are microphone failures during the recording of several utterances which might degrade overall system performance [Bar+17]. An overview of the whole dataset is given in Tbl. 5.1.

5.1.2 REVERB

The REVERB challenge [Kin+13; Kin+16] from 2013 had the goal to compare and advance different approaches to far-field ASR. More precisely, the main focus of the challenge was on the deterioration of the recorded speech signal due to reverberation. Additional non-stationary noise sources were not considered. Similar to the CHiME task described above, there are simulated as well as recorded data available.

For the simulated data, RIRs are recorded using a circular array equipped with eight microphones in six different scenarios. Three different room sizes (small, medium and large) are considered as well as two different speaker positions for each room. One 50 cm away from the array (near) and one 2 m away from the array (far). This results in RIRs with T_{60} times between 200 ms and 800 ms. Clean speech data from the British english WSJCAM0 [Rob+95] corpus is used to generate reverberated speech signals by convolving the speech signal with one of the recorded RIRs. Recorded broadband noise (mostly originating from air conditioning) is added afterwards such that the ratio

Task	Name	Type	distance	#utterances	#words	#hours
Training		simulated	mixed	7861	132778	15.5
Development		real	near	90	1463	0.17
		real	far	89	1603	0.17
Evaluation	`EVAL[+10dB]`	real	near	186	3131	0.35
		real	far	186	2962	0.32

Table 5.2: Summary of the REVERB corpus.

between the reverberant speech signal and the noise (reverberant-to-noise ratio (RNR)) is approximately 20 dB. The resulting data is split into a set for training, development and evaluation according to the WSJCAM0 corpus. The training set consists of 7861 utterances from 92 different speakers (53 male, 39 female).

The real recordings are only used for development and evaluation purposes. They are taken from the MC-WSJ-AV corpus [Lin+05] and recorded in a reverberant meeting room not used to record one of the RIRs for the simulated data. Two different speaker distances (100 cm: near and 200 cm: far) and multiple positions are covered. Exact T_{60} times are unknown but are estimated to be within 600 ms – 800 ms. Similar to the CHiME dataset, this is a closed vocabulary task with 5000 words. There is a significant mismatch between the simulated and the real recorded data as shown by the challenge results with the real recordings being more challenging [Kin+16].

A quantitative summary of the corpus is given by Tbl. 5.2.

5.2 Evaluation setup

The ultimate goal of our system is to transcribe speech data, even though only the signal processing part might have been replaced compared to the baseline system. This work therefore concentrates on a single metric for evaluation, namely the WER. It is calculated as the total Levenshtein [Lev66] distance divided by the number of words on an evaluation set and expressed as a percentage. In other words, it measures the average minimal number of insertions, substitutions and deletions needed to obtain the ground truth transcription per 100 words.

For evaluation, we use the same language modeling components as the baseline system provided by the specific task. I.e. neither the lexicon, nor the language model is modified and the only changes we make concern the estimation of the state posteriors (see Sec. 3.1.2). We also refrain from rescoring the hypotheses with a more sophisticated language model with the assumption[1] that this performance increase is orthogonal to the one we aim to achieve in this work and primarily because evaluation becomes computationally less demanding. For both datasets, the language model probabilities are estimated by a pre-computed closed vocabulary 3-gram language model which includes around 5000 different words and is provided by the respective dataset. The language

[1]Experience with the CHiME task has shown that the improvements obtained by rescoring are independent of the once obtained by a more sophisticated signal processing / acoustic model

Name	Reference	# channels	WER on REAL
CHiME 3 DNN baseline	[Bar+15]	6	33.4
CHiME 4 baseline	[Hor+15]	6	11.51
NTT CHiME 3 I	[Yos+15]	1	15.60
NTT CHiME 3 II	[Yos+15]	6	8.32
NTT CHiME 3 III	[Yos+15]	6	5.83
Kaldi setup I	[Che+18]	1	16.36
Kaldi setup II	[Che+18]	1	12.92
Kaldi setup III	[Che+18]	6	6.84
Kaldi setup IV	[Che+18]	6	4.01

Table 5.3: Reference results for the CHiME task on the real recordings evaluation set.

model score (see Sec. 3.1.4) is always optimized by a line search on the development set and then held fixed during the actual evaluation.

As described above, there is a simulated as well as a real evaluation set available for both datasets. For the CHiME task, both conditions are about equally challenging and we evaluate both sets. But for the REVERB task, the performance of a system on the simulated data can merely serve as a coarse approximation for the performance on the real dataset [Kin+16]. Additionally, the WERs achieved on the simulated data are already so low, that any enhancement can only hardly be distinguished from noise caused by different training runs and a likely non-optimal choice of hyper-parameters for the specific systems.

Ultimately, we are interested in the system performance in a real-world application and therefore only focus on the real evaluation set in this work for this task. A main difference between the evaluation and the training set can be the recording volume, i.e. the power of the observed signal. While for the CHiME task this is similar, there is a large difference for the REVERB task. As not all methods in the following are invariant to scale, an additional evaluation set is created for the REVERB dataset by increasing the power of the original one by 10 dB. Also for the REVERB task, there is only a minor difference between the far and near set and reporting them separately yields no new insights. We report the average WER across both.

Besides the multi-channel track with 6 and 8 microphones respectively, both tasks also feature a 2 channel and a single-channel track. We will use the single-channel track to evaluate performance of the acoustic model on unprocessed data and the multi-channel track to evaluate the overall system performance. This multi-channel system can also be applied to the two channel track with some performance loss as shown in, e.g., [HDH16b]. Since we do not develop a specialized system for the two-channel track, we opt to not evaluate all systems on this track since we do not expect to gain new insights from this evaluation. Different microphone configurations and their impact will be considered in Sec. 7.4.7.

Name	Reference	# channels	WER
Challenge baseline	[Kin+13]	8	47.23
NTT REVERB I	[Del+14b]	1	27.5
NTT REVERB II	[Del+14b]	1	21.7
NTT REVERB III	[Del+14b]	8	17.8
NTT REVERB IV	[Del+14b]	8	12.8
NTT REVERB V	[Del+14b]	8	9.0
Kaldi setup I	[Szu18]	1	19.78
Kaldi setup II	[Szu18]	8	10.25
NTT unified I	[NK19]	8	13.11
NTT unified II	[NK19]	8	9.27

Table 5.4: Reference results for the REVERB task averaged over the two real recordings evaluation set.

5.3 Baselines

In order to compare the performance of a proposed method, we specify different baselines for each of the tasks. The baselines are chosen such that they include the original challenge baseline, the winning system of the challenge as well as the current state-of-the-art system for both tasks. An overview of the CHiME baselines is given in Tbl 5.3 whereas those for the REVERB task can be found in Tbl. 5.4

5.4 Summary

The CHiME and the REVERB corpus will be used throughout this thesis to evaluate the models. While the former features everyday environmental distortions but little reverberation, the latter exclusively focuses on signal deterioration caused by reverberation. We also defined baselines for both tasks which reflect challenge as well as state-of-the-art results.

6 Contributions

The overall theme of this work is to develop a system for robust multi-channel ASR with improved recognition performance for far-field scenarios in everyday environments. Specifically, we set the following scientific goals:

- As few assumptions as possible should be made regarding the type of the distortions[1], environment and the microphone array.
- If possible, the system should make use of available data to learn to exploit spectro-temporal patterns and to avoid tedious manual tuning of parameters and erroneous model assumptions.
- The system should not only work in theory or controlled conditions but also be practical and deployable. A real-time factor < 1 should be possible with moderate computational resources and, ideally, the latency should be low such that it can be used in interactive voice response systems.
- If data is used for training, the supervision needed should be as small as possible to avoid costly labeling efforts.
- The interaction of the speech enhancement and the acoustic model and their joint optimization should be considered.

Pursuing these goals resulted in several publications which form the basis for this thesis. In this work, they are presented in a concise way, put into context and extended upon in many ways. All previously presented systems have been updated to include all insights gained during the course of this thesis work. They are trained from scratch and evaluated to obtain consistent results and to allow for a fair comparison. Additionally, new aspects and insights are discussed in this thesis that have not been published yet. The contributions of the publications and thus this work are highlighted in the following.

The main contribution of this work is the so-called neural network supported beamformer [Hey+15a; HDH17; HDH16a]. It combines classical statistical signal processing for beamforming with a neural network to estimate the necessary statistics from the high dimensional observation. The system is designed in a way that the advantages of both approaches are kept, the flexibility and optimality of the statistical beamforming

[1] The system should not be designed to distinguish between two competing speaker but should be able to handle concurrent speech as a diffuse noise source such as background babble.

and the classification capabilities of the data-driven neural network. Because of the first property, the system works with any microphone array and the second one ensures that statistics can be inferred even in challenging environments by exploiting temporal as well as spectral correlations for estimating the SPP and NPP. The system is described and evaluated in detail in Ch. 7. All other contributions are based on this system and they all have the goal to make the system more suitable for real-world applications.

The first improvement deals with latency. Although this is a factor of minor importance for many applications as long as computations are tractable, one important use case of far-field ASR are smart speakers with integrated virtual assistance where a low latency is crucial for a satisfactory user experience. The originally proposed neural network supported beamformer has a latency of at least a whole utterance as it aggregates the statistics over all available data. In order to reduce the latency, the network architecture as well as the beamformer itself has to be modified. Investigations on these modifications are published in [HHH18; Kit+16]. Additional to these prior works, further simplifications of the network architecture are explored and evaluated with a focus on scale invariance in Ch. 8.

The second improvement eliminates the need for parallel training data. With the introduction of a data-driven method comes the need for large amounts of data to train the system. In this specific case, parallel data is needed, i.e. data where the signal is separated into the target signal and the superposition of all interfering signals. But this kind of data is only available for simulated recordings with artificial mixtures of a clean target signal and interfering sources. Especially in the case of multi-channel signals, recording large amounts of this data is costly whereas real recordings become available just by using the system. Additionally, simulated data does not account for all effects, e.g., the Lombard effect [GHD10]. Thus, a domain mismatch between the training data and the actual real-world data is introduced. The publication [DHH19b] solves this problem by introducing a new loss function which maximizes the likelihood of a mixture model in a completely unsupervised way. This contribution is presented in Ch. 9.

Lastly, the signal processing part with the beamformer and mask estimator is integrated into the AM for a joint optimization. Not only does this mitigate the training data problem by just requiring a transcription for a real recording to train the mask estimator. It also promises to optimize the front-end model with an objective that is much more closely related to the ultimate goal of recognizing the correct word sequence than any signal level objective. To unify the two models it is necessary to backpropagate the gradients from the acoustic model through the complex valued beamforming operation. The required math was published in [Böd+17], and the whole system in [Hey+17]. All of this will be presented in Ch. 10. This thesis also includes unpublished further experiments and analysis of the system yielding new insights.

All contributions discussed above focus on the front-end part of the system. And although this is the central topic of this thesis, the back-end was not left out completely. For the CHiME 4 challenge, an acoustic model called wide residual bidirectional long short-term memory (BLSTM) network (WRBN) was presented which was among the top performing models in the challenge [HDH16b; Vin+]. This, along with an improved and simplified version of it will be presented in Ch. 7 and used throughout this thesis.

Collaboration with peers resulted in several other publications which are not all

focused on topics discussed in this work. A complete list of publications with (co-) authorship can be found appended to this thesis.

Besides publishing scientific articles, contributions were made to open-source projects during the work on this thesis. This includes an open-source variant of the presented neural network beamformer[2] and the WPE dereverberation algorithm [Dru+18a][3]. Both are now used within recipes of the Kaldi toolkit to achieve state-of-the art results on several datasets [Che+18]. Contributions were made to another open-source end-to-end speech recognition toolkit called ESPNet [Wat+18]. Additionally, the gradients for the operations needed for beamforming were implemented[4] and are now publicly available in the Tensorflow framework [Mar+15].

[2]https://github.com/fgnt/nn-gev
[3]https://github.com/fgnt/nara_wpe
[4]This work was done during an internship.

7 Robust multi-channel ASR with neural network supported beamforming

This chapter describes the proposed system for robust multi-channel ASR. Instead of finding a way to effectively handle multi-channel data with the acoustic model, this work adheres to a more conventional approach based on classical signal processing. A statistical beamformer condenses the multiple input channels into one channel which is then fed to a standard single-channel acoustic model. The system neither makes any assumptions about the microphone array configuration and interfering noise sources, nor about the RTF which encodes speaker position and room characteristics. All necessary signal statistics are estimated directly from the observed multi-channel signal. To achieve this, a neural network is integrated into the beamforming process. We call this model the neural mask estimator and the whole beamforming process neural network supported beamforming. Another neural network is specifically designed for the acoustic model. Combined, the system can leverage spatial information and is robust against interfering noise and reverberation. In the following, we describe this system in detail and extensively evaluate it on the two datasets discussed in Sec. 5.1.

7.1 Neural network mask estimation

In Sec. 4.3 we already described how SCMs necessary for the beamforming can be obtained using spectral masks which in turn can be estimated using statistical spatial models. But with some notable exceptions [TH12; TH13a; TH13b; Tra15], statistical models as described in Sec 4.4 make the conventional i.i.d. assumption and ignore the spectral and temporal dependencies. The mentioned works already show that incorporating these dependencies complicates parameter inference, can become computationally very expensive and requires other assumptions like first order Markov chains which might not be suitable to capture, e.g., the harmonics of a speech signal.

When dependencies become hard to model, data-driven methods and especially neural networks are known to excel. Indeed, many works consider using neural networks in the context of single-channel speech enhancement and mask estimation (see, e.g., [WC18] for an overview). For single-channel enhancement, the network either outputs the enhanced spectrogram or a gain which is multiplied with the signal to obtain the enhanced version.

For multi-channel enhancement we propose to output a mask for the target and one for the interferences to estimate the SCMs and then use beamforming to obtain the enhanced signal [Hey+15a; HDH17]. A schematic overview of this system is depicted

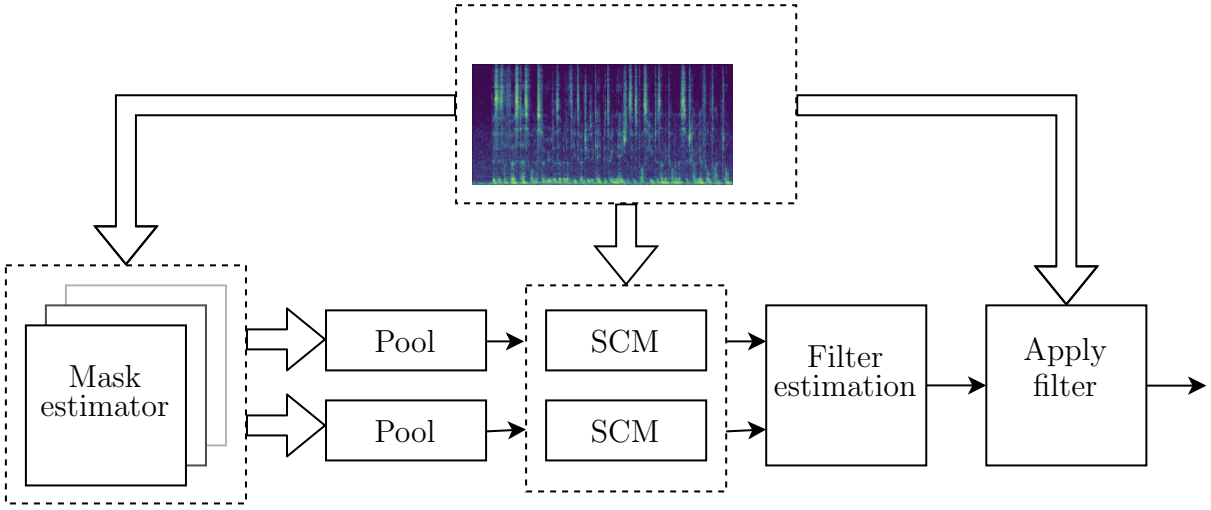

Figure 7.1: Schematic representation of the neural network supported beamformer. Depth illustrates multiple channels and copies of the network respectively. Multi-channel signals are indicated by bold arrows.

in Fig 7.1. While the task of the neural network is comparable to the single-channel enhancement with gain estimation, the overall system is very different. Using a gain (or directly approximating the spectral signal) can introduce arbitrary distortions like musical tones. Beamforming on the other hand is a linear filter operation based on signal statistics. And although it can introduce distortions due to an arbitrary scaling of the individual frequencies (see Sec. 4.2), the enhanced signal is a linear combination of all channels with different time shifts for each frequency and still represents a plausible signal. In other words, instead of having a neural network directly influencing the signal, we rely on classical signal processing and a statistical model which is only supported by a neural network in order to estimate necessary statistics. This combines the best of both worlds: the good classification performance of a data-driven neural network approach and the flexibility and interpretability of a statistical model.

To avoid making any assumptions about the microphone configuration, the system estimates the masks for each signal separately followed by a pooling operation to condense these masks into one for the target and one for the distortion. The pooling operates on each tf-bin individually across the available channels. While there are multiple operations (i.e. min/max, average, median, etc.) possible, we opt for the median. The reasoning behind this is that this operation still yields a reasonable mask even if some (and at most $(M-1)/2$) of the M sensors are defect. Such a malfunctioning is observed in the data of the CHiME challenge (see 5.1.1) for example.

Compared with the statistical approach from Sec 4.4, instead of relying on the spatial distribution of the sources, neural networks use spectro-temporal patterns of the individual sources to distinguish between them. In order to learn the characteristic patterns, a training step is necessary. This will be described in greater detail in Sec. 7.3.1. One advantage of this data-driven approach is that we do not need to make assumptions about the spectro-temporal patterns in the acoustic signal but instead infer those directly from the data itself. As another advantage, the class affiliation is already

Table 7.1: BLSTM network configuration for mask estimation

	Units	Type (Non-Linearity)	p_{dropout}
L1	2×1024	BLSTM	0.5
L2	1024	FF (ELU)	0.5
L3	1024	FF (ELU)	0.5
L4	2×257	FF (Sigmoid)	0.0

determined during the training and neither the frequency nor the global permutation problem arises. This, however, first comes at the cost of needing the right training data which in this case means that a separate signal for the two classes must be available. But in Sec. 9 and Sec. 10 this requirement will also be lifted.

We did not yet specify any particular architecture of the model and in this chapter we consider two different variants for the mask estimator. One is based on a BLSTM while the other one is a CNN. Both are detailed in the following.

BLSTM mask estimator

The considered BLSTM model for mask estimation is composed of a BLSTM layer (L1) with 1024 units for each direction. The output is concatenated and fed to a stack of two fully connected layers (L2 + L3) with 1024 units and an exponential linear unit (ELU) activation function. Masks for both classes are then estimated by two additional linear layers, this time with 257 units each. This naturally corresponds to the number of frequency bins.

We do not consider the masks to be mutually exclusive for this task, or, in other words, the masks are not forced to sum to one. This is motivated by the fact that for the estimation of the SCMs it is sufficient to only account for those tf-bins where one source certainly dominates. Tf-bins where both sources are about equally active or inactive at all are ignored. Consequently, instead of using a softmax for the output (which would mean mutually exclusive sources), we use a sigmoid here. The performance gain achieved with this choice will be analyzed later when discussing the unsupervised extension in Sec. 9.3.3.

To achieve a better generalization, we use dropout for the input of all layers except for the output layer. The dropout rate is fixed at $p_{\text{dropout}} = 0.5$ for every layer. Table 7.1 summarizes the network configuration. This architecture is a bigger variant of the network first proposed in [Hey+15a].

Using a BLSTM allows the network to take temporal context into account and therefore learn temporal patterns of speech and noise. The network also operates on the whole frame, making it possible to discover and exploit spectral patterns. As mentioned earlier, the network treats each channel separately but the parameters are shared among them.

The input to the network is the magnitude spectrogram of an observation. In general, the network is not invariant to a shift or the scaling of the data. However,

the output masks should be the same irrespective of the global gain which depends on the used sensors and the distance of the sources. To establish this invariance, the input is normalized using the mean and variance of the current utterance. In the spirit of batch normalization [IS15], we do not only normalize the input to the first layer but also the activations before the non-linearity for layers L2 and L3. Contrary to batch normalization where the statistics are calculated along the batch dimension, the statistics are calculated along the temporal dimension here. We therefore also refer to this kind of normalization as sequence normalization. A similar approach was presented in [Lau+16] where the normalization is calculated along the time and batch dimension. One advantage of our approach is that it can also be used during inference, allowing the network to adapt to the data and avoiding a (gain) mismatch between training and test. The drawback is that the whole utterance has to be available first which is prohibitive for low latency scenarios. Alternative approaches for these scenarios will be discussed later in this work in Ch. 8.

U-Net mask estimator

As an alternative to a recurrent structure, we also consider a convolutional structure. The U-Net architecture is named after its U-shaped schematic illustration and was proposed in [RPB15] for medical image segmentation. Since then it has been extensively (as of the time of writing, the paper has been cited 6925 times) and successfully used within the computer vision community for segmentation tasks (e.g. [Che+17; BKC17]) and beyond [Iso+17]. It is an all convolutional architecture, i.e. it is only composed of convolutional, pooling and stacking operations. As such, it is able to handle variable sized input without any modification. The main idea behind the architecture is to use clues from different scales of the signal to produce the final segmentation result. This is achieved by downscaling feature maps, applying a block of filters, upscaling it and then concatenate the result as additional feature maps to the original ones. The block of filters includes the same mechanism of reducing the resolution. Thus some filters are applied at the original resolution capturing local features, while others are applied at a much lower resolution and can capture more global features.

For the task at hand, the magnitude spectrogram can be interpreted as an image and its segmentation yields a class affinity for each tf-bin. Consequently, little changes are necessary to apply the architecture for spectral mask estimation. The biggest difference between the spectrogram and an image is that the temporal dimension of the spectrogram varies between the utterances while for images the corresponding dimension is assumed to be fixed. But since the architecture is all convolutional, no major changes are required. The only modification is to make sure that for each block the devision by the pooling factor yields an integer number. This is achieved by padding the spectograms with zeros at the end such that this requirement is fulfilled. At the final output of the network we append an additional 2-dimensional convolution with filter size 3×3, two output channels and a final (sigmoid) non-linearity to obtain the masks for the target and distortions.

The whole architecture of the network used here is shown in Fig. 7.2

Having described the two architectures of the neural mask estimator we now turn to the acoustic model.

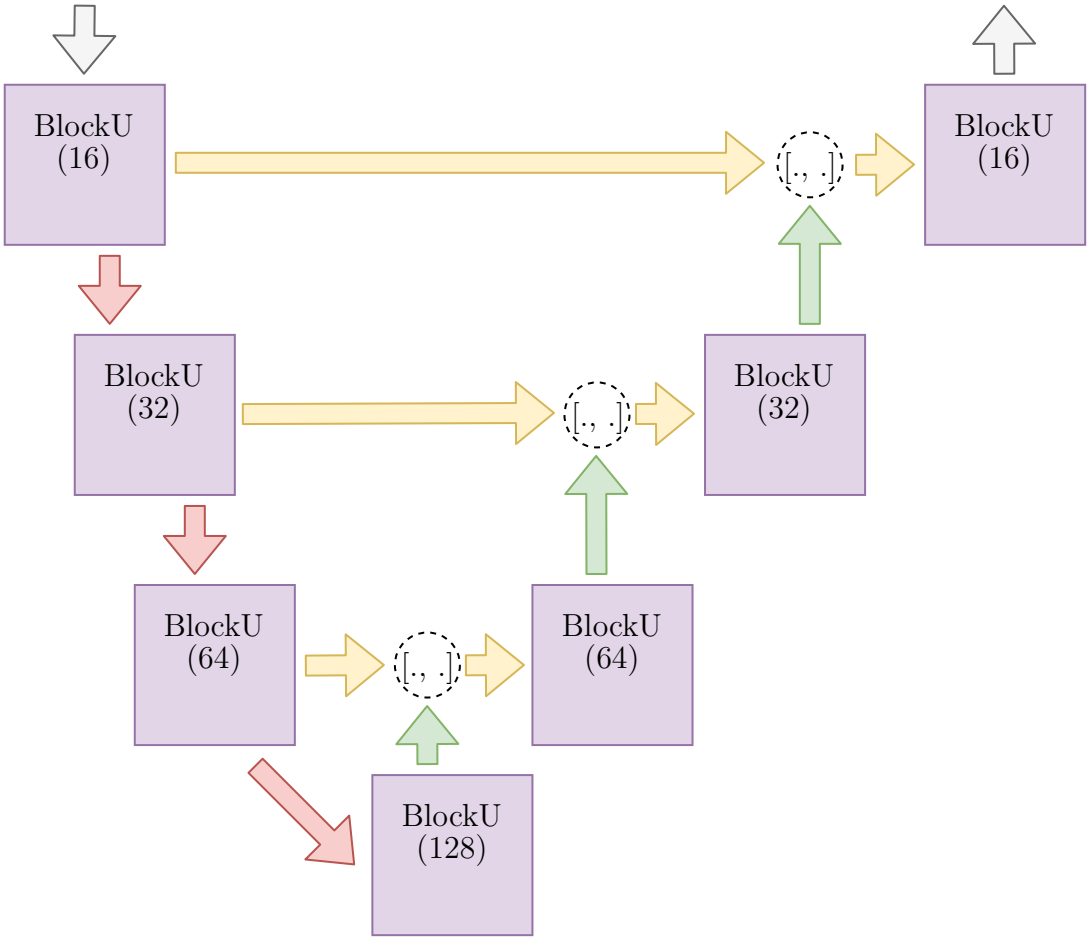

Figure 7.2: U-Net mask estimator. Red arrows indicate a max-pooling with size 2×2. The green arrows indicate a bi-linear upsampling by a factor of 2, the yellow arrows copy the feature map and the two gray arrows are the input and output respectively. The "[., .]" operator concatenates the two feature maps along the channel dimension. The details of the convolutional block "BlockU" are given in Fig. 7.3. Note that as we increase depths, the spatial resolution of the feature maps decreases while the number of channels increases. This allows the network to incorporate spectral as well as temporal context.

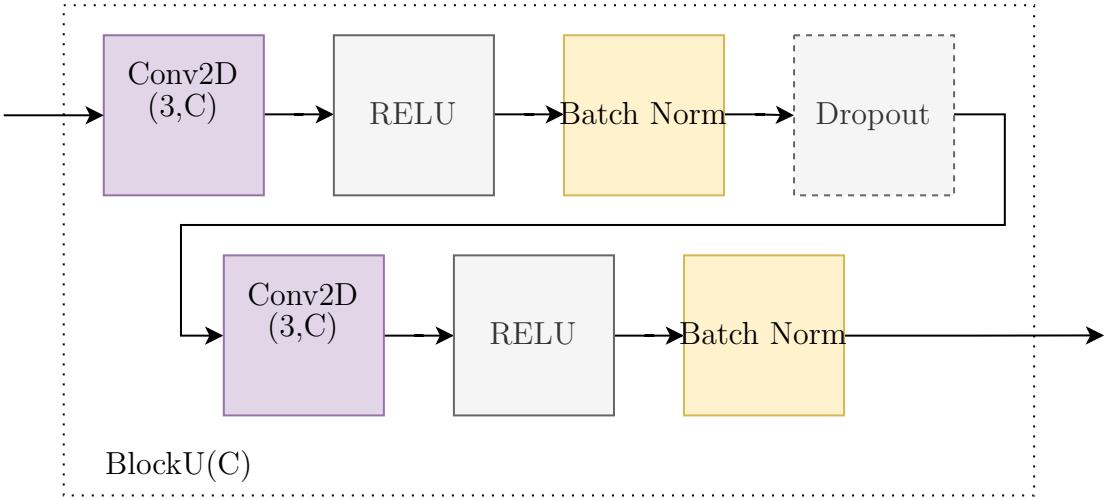

Figure 7.3: A single convolutional block as used by the U-Net shown in Fig. 7.2 above. The block is parameterized by the number of filter channels C for the 2-dimensional convolution. Dropout is only applied during training. The 2-dimensional convolution block Conv2D(F, C) is parameterized by the filter size $F \times F$ and the number of filters C.

7.2 Wide Residual BLSTM Network acoustic model

The network architecture used for the acoustic model in this work draws major inspiration from three other works. The first one is the Convolutional long short-term memory fully connected deep neural network (CLDNN) architecture [Sai+15b] which combines CNN, LSTM and feed-forward layers to profit from the characteristic advantages of each layer type. Second are the works by Sercu et al. [Ser+15; SG16]. In [Ser+15] the authors show that a slightly modified architecture from an image recognition task also works well for a speech recognition task. While previous works used large filter sizes for CNN layers, this work shows that filters of size 3×3 also work well when multiple layers are stacked. The follow-up work [SG16] describes full sequence training with these networks which leads to a much more efficient inference step when working on a whole utterance. We adopt this approach as it allows to use sequence normalization during test time and is suitable for combination with a BLSTM layer which anyway requires to process a longer sequence.

Additional inspiration is drawn from the findings about CNNs by the image community [ZK16; He+16a; He+16b] where residual connections improve convergence properties and *wide* CNN layers (i.e. a larger number of feature maps) show good performance even if the network is relatively shallow and compared to a network with $100+$ layers.

We name the resulting architecture WRBN and presented it first in [HDH16b]. Fig. 7.4 gives an overview of the complete architecture. The details for the building blocks are shown in Fig. 7.6 and Fig. 7.5.

The first part, a wide residual network (WRN) adopted for sequence processing,

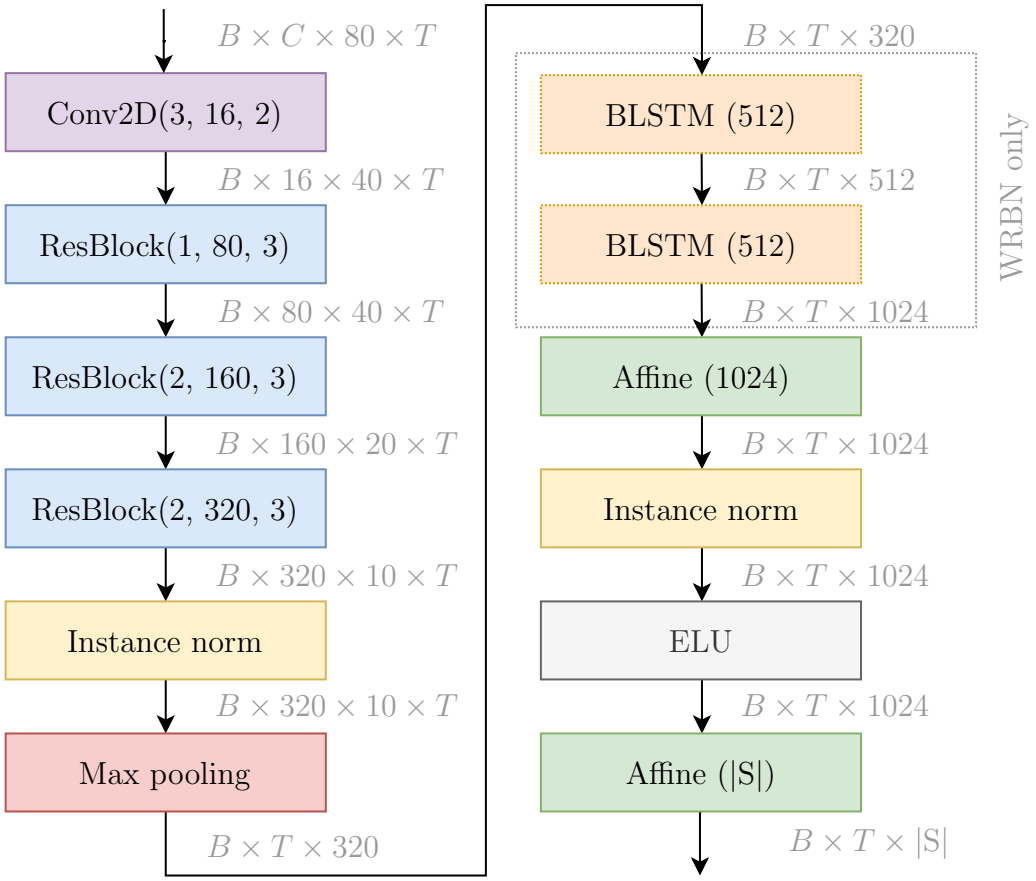

Figure 7.4: Overview of the back-end structure. The annotations in gray indicate the dimension of the tensors where B is the batch size and T is the number of frames of the largest utterance within the batch. "ResBlock"s are further explained in Fig. 7.5 and Fig. 7.6. The instance norm normalizes across the width and height of a feature channel which roughly corresponds to the time and frequency dimension. |S| is the number of unique states of the HMM.

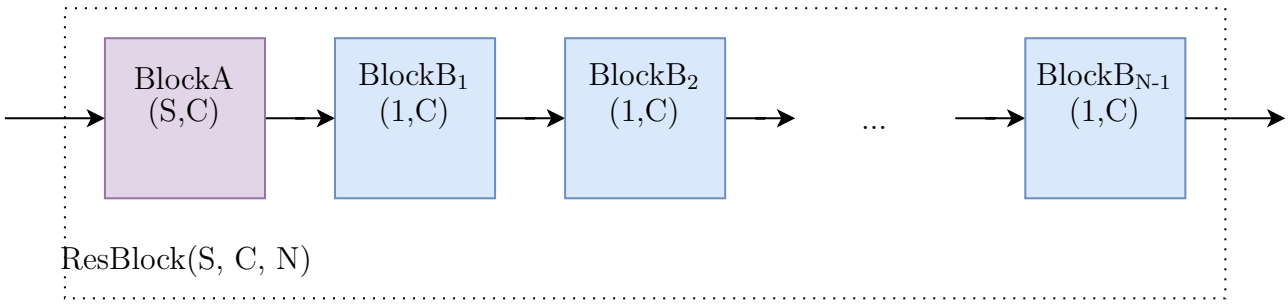

Figure 7.5: Detailed view of a ResBlock. A ResBlock is parameterized by its height striding S, the number of output channels C and the number of inner blocks N. Accordingly, BlockB is repeated N-1 times.

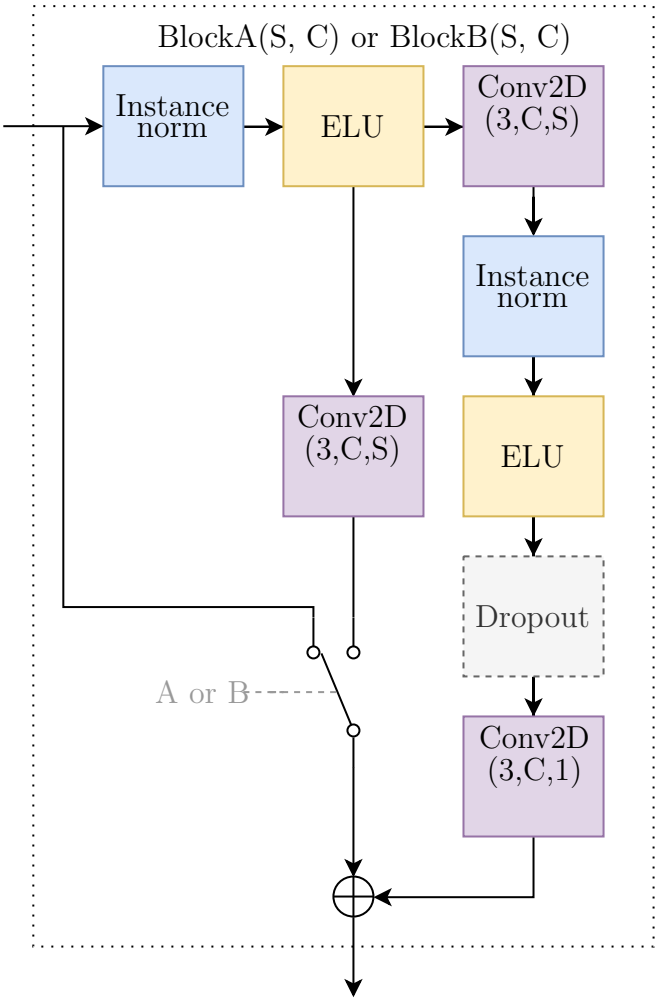

Figure 7.6: Detailed view of the building blocks BlockA and BlockB respectively. A convolution block Conv2D(F, C, S) is parameterized by the filter size $F \times F$, the number of filters C and the striding S in the height dimension. The striding for the time dimension, i.e. the width, is always 1. The input is padded with zeros such that the output has the same size.

consists of three residual building blocks. Each of these blocks consist itself of smaller building blocks, BlockA and BlockB. BlockA is always the first block in the chain of smaller building blocks. It can have a stride ≥ 1 to reduce the frequency resolution and it increases the number of channels. This prohibits a direct residual connection to the output of the block and an additional convolution operation with filter size 1×1 acts as the residual connection changing resolution and size accordingly. After BlockA a couple of BlockB blocks can follow. These are nearly identical but do not alter the resolution nor the number of feature maps, allowing for a direct residual connection. All blocks use an ELU non-linearity. Instead of batch normalization, we use an instance normalization which calculates the statistics for each feature map individually over height (frequency) and width (time). Reliable statistics can be estimated because we are working with full sequences. Similar to the mask estimator, this again allows to normalize within the network also during test time with the actual statistics of an utterance rather than accumulated ones from the training data. In [HDH16b] we show that this yields relative improvements of about 10 %.

The output of the WRN are 320 feature maps of size $10 \times K$. I.e. the frequency dimension has been reduced to 10 while there are still K frames. A max-pooling operation reduces the frequency dimension completely, yielding K features of size 320 which are provided to the two BLSTM layers. The first of these has 512 units for each direction and its final output is the sum of both directions. The second also has 512 units for each direction but this time the two outputs are stacked to K features of size 1024. These features then serve as an input to a feed-forward layer with an ELU non-linearity and a sequence normalization of its activations. A final linear transformation yields the logits for the senones (see Sec. 3.1.2).

7.3 Training

The system includes two models, the mask estimator and the acoustic model, which are trained separately. A joint training scheme will be discussed later in Ch. 10. Both models are trained in a supervised fashion. If not noted otherwise, a variant of multi-style training [LMP87] is used. This multi-style training directly uses the noisy observations instead of e.g. the clean speech image.

We use the same STFT parameters for all models. The window size is 400 samples which translates to 25 ms for the 16 kHz recordings. We shift the window by 160 samples (10 ms) and transform the signal with a FFT of size 512.

In order to avoid overfitting, a process on a separate workstation constantly evaluates the WER on the development set once the last evaluation finished and a new checkpoint is available. If the WER did not improve over the last 10 evaluations, the last best checkpoint is restored and the learning rate is multiplied by 0.5. Training is stopped after the learning rate has been decayed three times and the current best model is saved for final evaluation. To measure the WER for a mask estimator, it is combined with a GEV beamformer and a pre-trained acoustic model.

The training of the individual models including further optional regularizations like gradient clipping is described in more detail in the following.

7.3.1 Mask estimator

The training of the mask estimator requires parallel data. I.e. for a given observation knowledge of the predominant class (speech or interferences) for each tf-bin is required. This information is only available for simulated data where interferences and speech are mixed artificially such that the original signal is still available for each class. Different approaches exist to calculate a target from this signal. In [WNW14], Wang et al. compare an ideal binary mask (IBM) and an ideal ratio mask (IRM) in the context of speech separation. The IBM is one for all tf-bins where the SNR is greater than a pre-defined threshold while the IRM is the ratio between the instantaneous speech power and the instantaneous observation power for each tf-bin. While the IBM is of discrete nature and can only take a value out of $\{0, 1\}$, the value of the IRM is continuous and lies in the interval $[0, 1]$. Both targets are suitable for the task of SCM estimation. However, there is a notable difference between SCM estimation and source separation. The estimation of the SCM averages the contributions over time. Contrary to the direct application of the mask for speech separation, this estimation does not suffer from missing tf-bins as long as enough bins with a major contribution are available. In other words, while for speech separation both, precision and recall are equally important, in the context of SCM estimation a higher emphasis can be put on precision. Since we use the whole utterance to calculate the SCM and can thus deal with sparse masks, we favor IBMs over IRMs. In particular, the oracle masks for each channel are calculated as follows

$$\hat{\gamma}_{\text{target}}(k,f) = \begin{cases} 1, & \frac{||\mathbf{s}(k,f)||}{||\mathbf{n}(k,f)||} > 10^{\text{th}_s(f)}, \\ 0, & \text{else}, \end{cases} \tag{7.1}$$

and

$$\hat{\gamma}_{\text{noise}}(k,f) = \begin{cases} 1, & \frac{||\mathbf{n}(k,f)||}{||\mathbf{s}(k,f)||} > 10^{\text{th}_n(f)}, \\ 0, & \text{else}. \end{cases} \tag{7.2}$$

Note that the two thresholds th_s and th_n are frequency dependent and not identical. Both thresholds are hand-tuned on a few utterances of the development set and their values range from -5 to 10 depending on the frequency[1]. They are optimized for precision and sparsity and chosen such that a decision for speech/noise requires an instantaneous SNR which is high/low enough to ensure a low false acceptance rate. Note that these masks do not sum to one and are also not mutually exclusive. They can both be zero at the same time and also both be one at the same although the former case should happen much more frequently than the latter case.

The signals $\mathbf{s}(k,f)$ and $\mathbf{n}(k,f)$ describe the image of the target speech signal and the superposition of all interferences respectively. For a speaker vs. interfering noise sources scenario, these signals are clearly defined. But in the case of reverberation the distinction is not well defined. We follow the practice to attribute the first 50 ms after the first peak of the RTF to the speech image [Kin+13] and the remaining part to the distortion (see also Sec. 4.5). To generate the data, we accordingly split the RIRs for the target source into two parts to obtain a filter which, when convolved with the

[1]The concrete implementation with the default values can be found in the open source version here: https://github.com/fgnt/nn-gev/blob/master/fgnt/mask_estimation.py

speech signal, outputs the speech image and one for the late reverberation. The latter signal is then regarded as interferences and its energy is added to the noise energy before calculating the masks.

As discussed before, we interpret the task as a binary classification problem (dominant / not dominant) for each class and tf-bin independently and consequently use the binary cross-entropy (BCE) loss for each output. For a single example and class q, where q indicates either speech or noise class, this loss is defined as

$$\mathcal{L}_q = \frac{1}{K}\frac{1}{F}\sum_k \sum_f -(\hat{\gamma}_q(k,f)\log p_q(k,f) + (1-\hat{\gamma}_q(k,f))\log(1-p_q(k,f))). \quad (7.3)$$

This is the BCE averaged over all tf-bins with the oracle target $\hat{m}_q(k,f) \in \{0,1\}$ and the network prediction[2] $p_q(k,f) \in]0,1[$ for the class q at tf-bin (k,f). The total loss for a batch of size B is then calculate as $\frac{1}{B}\sum_b \mathcal{L}_s^{(b)} + \mathcal{L}_n^{(b)}$.

The resulting loss function is minimized using Adam [KB14] with an initial learning rate of $\alpha = 0.001$ and a limited gradient norm of 5 [PMB12]. To fully utilize the sequence normalization (see Sec. 7.1), one example always consists of a full utterance and for recurrent models full backpropagation through time [Wer90] is used.

As described before, the input to the network is only a single channel signal. To fully use the available data, all channels are used for training and randomly selected for each utterance for each iteration. One batch consists of 18 examples. In order to foster generalization for a broad range of SNRs, we modify the signal on-the-fly and alter its original SNR by randomly sampling a value between -5 dB and 3 dB.

7.3.2 Acoustic model

The alignments for the training of the acoustic model are obtained by training a GMM-HMM system according to the task baseline recipe provided by Kaldi[3][4]. This model is then used to force align the training data to get the target state for each frame. Note that contrary to the training of the mask estimator, the acoustic model can also be trained on real recordings where only the observation is available. Previous work [Yos+15] also has shown the benefits of including all channels in the training process and we follow this advice and sample the channel randomly for each example at each iteration.

With the obtained targets, the acoustic model is trained with a standard cross-entropy loss and LMSCs are used as input features. Although subsequent sequence training generally improves the performance, as with the language model, these improvements are assumed to be mostly orthogonal to the methods presented and evaluated in this work. We refrain from it in favor of a simpler and faster training process.

Like before, Adam is used as the optimization method but this time the initial learning rate is lower and set to $\alpha = 0.0001$ and the batch-size is 2. The gradient is likewise clipped when its norm exceeds 5.

Again, we always use a full utterance as an example to exploit the benefits of normalization. We also modify the SNR of the observation for simulated examples as

[2] In the implementation the value is clipped to avoid numerical issues.
[3] https://github.com/kaldi-asr/kaldi/tree/master/egs/chime4/s5_1ch
[4] https://github.com/kaldi-asr/kaldi/tree/master/egs/reverb/s5

Table 7.2: Baseline WERs on single-channel track for the acoustic model described in 7.2 for the CHiME and REVERB task in comparison to results published by others as described in 5.3.

	CHiME `SIMU`	CHiME `REAL`	REVERB
WRBN	17.42	16.05	15.52
Kaldi setup I ([Szu18])	-	-	19.78
NTT REVERB I ([Del+14b])	-	-	27.5
Kaldi setup I ([Che+18])	-	16.36	-
NTT CHiME 3 I ([Yos+15])	-	15.60	-

already described for the mask estimator but increase the range of the SNR change to $-7\,\text{dB} - 7\,\text{dB}$. The inclusion of higher SNR values aims at better performance when used in combination with the multi-channel front-end, which, as we will demonstrate soon, can achieve high SNR gains.

7.4 Evaluation

Results for the different systems are reported on the CHiME and REVERB task. First, we focus on the acoustic model with different configuration. Afterwards, the whole system is evaluated. The different approaches to mask estimation are compared as well as the different beamforming criteria. Finally, the influence of the number of channels is analyzed.

7.4.1 Acoustic model

The results for our baseline acoustic model as described in Sec. 7.2 for the CHiME and REVERB task are shown in Tbl. 7.2. Note that these are the results for the single-channel track of both tasks and without any processing of the observed signal. The results for the CHiME dataset are on par with the ones reported in Kaldi setup I ([Che+18]) which is, as of the time of writing, regarded as state-of-the art. They are also close to the ones of the winning CHiME 3 system [Yos+15] but without using a strong language model to rescore the hypotheses. Compared to the baseline system of the CHiME 3 challenge, the WER is reduced by nearly 50 % although the baseline system uses all channels and a beamformer to process the signal. Slightly worse results are achieved on the simulated set compared to the real set but this is a common phenomenon observed across different systems [Vin+].

On the REVERB task, the WER is improved by 20 % over the Kaldi setup I Baseline ([Szu18]). When compared to the winning solution of the challenge (NTT REVERB I [Del+14b]), the error rate is reduced by over 40 %.

Next, we analyze the impact of multi-style training exemplarily for the CHiME task. The results are reported in Tbl. 7.3. If only clean data is used during training, the

Table 7.3: Effect of multi-style training on the CHiME task.

training data	SIMU	REAL
clean	42.73	48.25
multi	17.42	16.05

Table 7.4: Performance of the acoustic model when modifying its architecture for the CHiME task.

Modification	Name	SIMU	REAL
None	WBRN	17.42	16.05
– norm		21.12	20.60
– CNN		22.46	22.61
– BLSTM	WRN	**16.03**	**15.23**
– BLSTM; – norm		20.00	21.10

performance severely degrades and the WER increases from 16.05 % to 48.25 % on the real evaluation set. The performance hit is slightly less for the simulated data but still large. This is expected as there is a huge mismatch between the evaluation set and the clean training data. In order to cope with distortions, the model needs to be trained on distorted data, the more similar the distortions, the better. This will become important again later in Ch 10 when discussing a joint optimization of the models.

To gain some insights on what drives the performance of the acoustic model, we remove important components of its architecture. Again results are reported on the CHiME task in Tbl. 7.4.

The first modification (– norm) removes all normalizations from the network and replaces them with a single normalization of the input features using the training data statistics. This modification has a significant impact on the systems performance and results in a WER increase of nearly 30 %.

For the second and third change, we remove either all CNN (– CNN) or all BLSTM

Table 7.5: Performance of the WRN acoustic model when trained with different seeds for the CHiME task.

Model	SIMU	REAL
WRN	16.03	**15.23**
WRN #2	16.01	15.47
WRN #3	**15.91**	15.36

Table 7.6: Performance of the WRN and WRBN acoustic model for the CHiME and REVERB task.

Model	EVAL	EVAL+10dB
WRBN	15.52	15.52
WRN	**15.02**	**15.02**

(– BLSTM) layers. While for the former the performance significantly degrades, the latter, which we refer to as WRN, surprisingly outperforms the baseline model even though it has much fewer parameters.

We also remove the normalization from the WRN (– BLSTM; – norm) which results in an even bigger performance drop than for the WRBN and again highlights the important role of the sequence normalization.

But the training process includes a lot of randomness. This starts with the random initialization of the network parameters and also includes things like the composition of examples to form a batch. Additionally, there are a lot of hyper-parameters governing the training process and although these are chosen from a vast experience, they are likely not the best ones possible for the specific model. On the other hand, the computational power required to train a single acoustic model is large, rendering it infeasible to run the training multiple times with different random seeds for all experiments and much less running a search to optimize the hyper-parameters for each system. But before drawing conclusions from a single run, we at the least want a rough estimate of the magnitude of the variance caused by the randomness of the training process. To this extent we train the same WRN model three times with different random seeds but the same hyper-parameters on the CHiME dataset. Results for this experiment are given in Tbl. 7.5. The range of the WER is fairly narrow. For the real set, absolute difference between the best and the worst run are 0.33 percent points. Again, this also cannot serve as a test for any statistical significance and should solely be regarded as an indication of the variance. So whenever we use the term significant in the following, this does not relate to statistical significance.

With this, we can conclude that there is a chance that the WRN and WRBN are actually on par but removing the normalization is most certainly harmful. This result also indicates that the broad receptive field of the CNN layers combined with the underlying HMM models the temporal dependencies sufficiently well such that no recurrent model is needed. Though it should be noted that the training corpus for both tasks is considered small[5] and the picture might change when more data is available. In the following, we will evaluate both, the WRBN and WRN for most of the upcoming experiments to test for any different behavior when the data is preprocessed.

Finally we also evaluate the WRN model on the REVERB task. The result is shown in comparison with the ones from the WRBN in Tbl. 7.6 for the official (`EVAL`) as well as for the amplified (`EVAL+10dB`) test set with real recordings. Again, the WER achieved by the WRN is slightly better compared to the WRBN (15.02 vs. 15.52) but

[5]Big corpora include thousands of hours of speech.

Table 7.7: Comparison between a BLSTM (Sec. 7.1) and a cACGMM (Sec. 4.4) mask estimator with a GEV beamformer for the CHiME task.

	WRBN		WRN	
Mask estimator	SIMU	REAL	SIMU	REAL
None	17.42	16.05	16.03	15.23
BLSTM	**6.93**	**7.79**	**6.78**	**7.34**
cACGMM	9.49	13.07	9.28	12.95

the difference is not large enough to draw any strong conclusions. Not surprisingly, the results are the same for both test sets. While this rather serves as a sanity check for the normalization rendering the acoustic model invariant against any scaling of the signal here, the gain difference will become more important in the experiments to follow.

7.4.2 cACGMM vs. neural network based mask estimator

Next, we evaluate and compare the performance of a neural network and cACGMM based mask estimator. Here, only the BLSTM mask estimator described in Sec. 7.1 is considered. Other estimators are assessed later in this work.

For the experiments, the respective mask estimator is used to obtain the weights for calculation of the SCMs (see Sec. 4.3). For each utterance, a single target and noise SCM is estimated for each frequency, i.e., we average the statistics over a whole signal. The SCMs are then used to estimate the filter coefficients of a GEV beamformer which are applied to the observation to obtain a single-channel STFT representation of the enhanced signal[6]. For the scaling we divide the interferences SCM by its trace as discussed in Sec. 4.2.2. LMSCs are then extracted from the enhance spectral representation and fed to the WRBN or WRN acoustic model trained in the previous experiments. Finally, the WER is evaluated based on the decoded transcription. Note that the acoustic model is not adapted in any way to the front-end processing method. The enhancement is simply *plugged* into the feature extraction process. A more detailed analysis on the integration will follow in Ch. 10.

The results for the CHiME task are shown in Tbl. 7.7 together with the results of the single-channel acoustic model for reference. Overall system performance improves for both estimators, showing the effectiveness of beamforming. However, the WER is considerably lower for the system with the neural network based mask estimator. These results are in line with our findings in [HDH16a] which also evaluates systems with a MVDR beamformer, i.e., this phenomenon is not specific to the GEV beamformer. With the BLSTM mask estimator, the WER on the real recordings reduces by a bit more than 50 % for both acoustic models. Using the cACGMM to estimate masks results in a reduction of about 18 % with the WRBN and 15 % with the WRN acoustic

[6]Note that we do transform the signal back to the time domain to extract the features. This avoids distortions which can be introduced by, e.g., a phase mismatch among the individually processed sub-bands and thus improves the overall system performance.

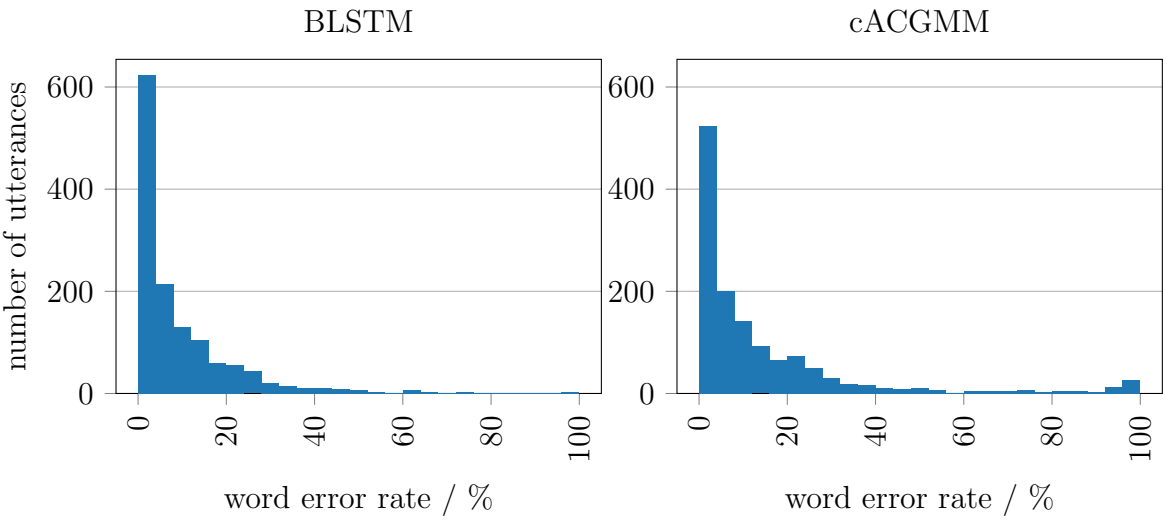

Figure 7.7: Histogram of the WERs for the systems with the BLSTM and cACGMM mask estimator. For the latter, there are several utterances with a WER >80 %, i.e. cases, where the system completely fails.

model. For the simulated data, the cACGMM performs slightly better, yet, the BLSTM system achieves lower WERs.

Further insights can be gained from the histograms in Fig 7.7. They show the distribution of the WERs for the real evaluation set for both systems. It is noticeable, that for the cACGMM system the distribution is not just slightly shifted towards higher WERs, but that there is a significant number of utterances where the system does not work at all. To be precise, 46 utterances have a WER >80 % while for the BLSTM system there are only 2 with this property. Analyzing these cases reveals two main causes.

For utterances with specifically low SNR, the frequency permutation problem is not solved correctly. An example for this is shown in Fig 7.8 which also shows the estimated mask from the BLSTM mask estimator for the same utterance. The other source of failure is the global permutation problem. Especially when one or more microphones are distorted, the system tends to focus on the artifacts introduced by this distortion. An example of the phenomenon is shown in Fig. 7.9.

For the only two utterances where the neural network based system has a WER >80 %, the acoustic model can be identified as the cause. The two utterances are very short (2 and 3 words respectively) and the masks for the two systems for one of these utterances is shown in Fig 7.10. Both masks look reasonable, yet the WER for the systems is 100 %. The same WER is obtained without any processing by the single-channel system.

When excluding all *hard* utterances, i.e., those with a WER >60 % for the cACGMM system, it achieves a WER on the remaining set of real recordings of 9.31 % while the neural network based system achieves a WER of 6.43 % on the same set. This demonstrates that, although the cACGMM clearly suffers from the permutation alignment problem, the neural network based system still outperforms the statistical one, even

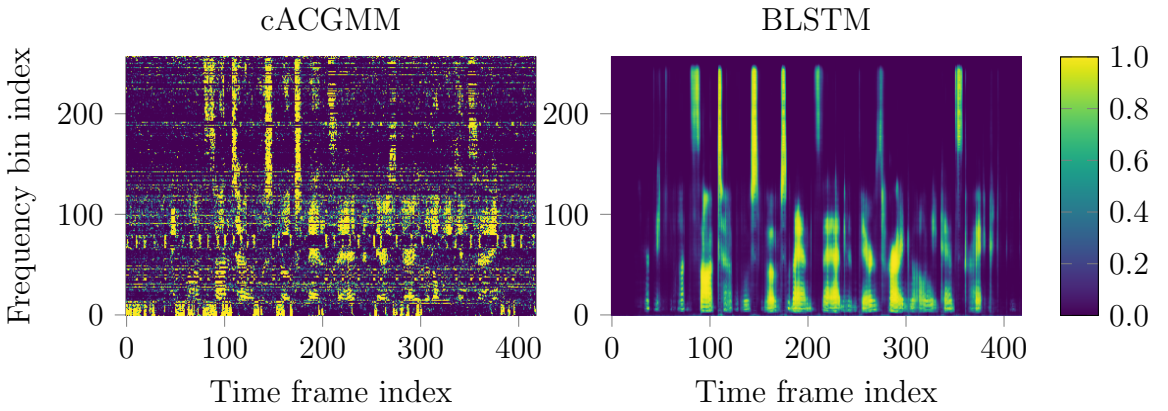

Figure 7.8: Masks for the utterance `m06_442c020g_bus`. Left: cACGMM mask estimator (100 % WER), right: BLSTM mask estimator (10 % WER). The frequency permutation alignment failed for the cACGMM model due to low SNR.

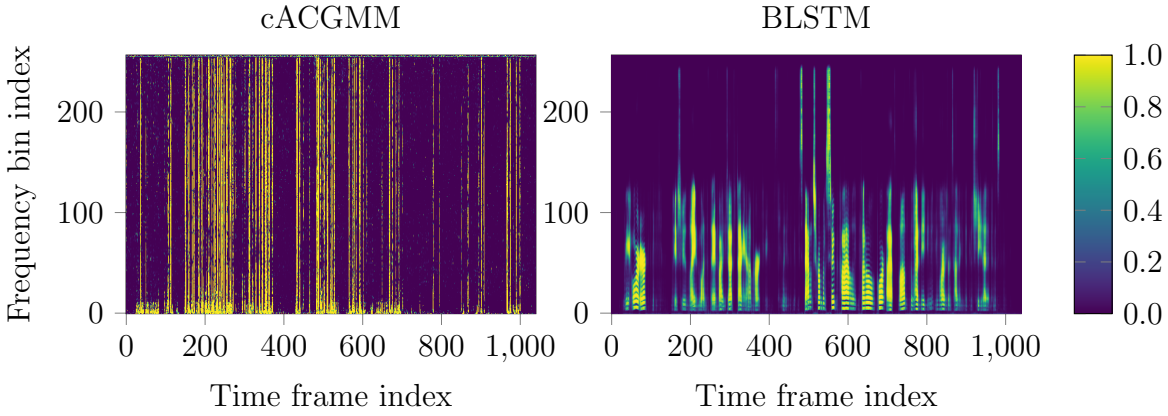

Figure 7.9: Masks for the utterance `m05_445c020g_bus`. Left: cACGMM mask estimator (100 % WER), right: BLSTM mask estimator (20 % WER). The global permutation alignment failed for the cACGMM model due to a microphone failure.

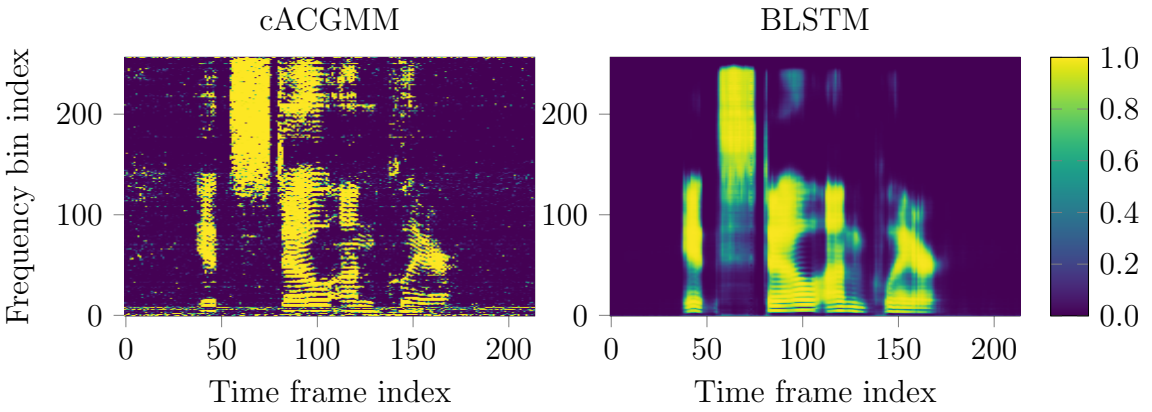

Figure 7.10: Masks for the utterance `m05_446c020p_str`. Left: cACGMM mask estimator (100 % WER), right: BLSTM mask estimator (100 % WER). Both models yield reasonable masks but all decoded words are wrong.

when factoring out most of the permutation issue.

Comparing the results with the baselines from Tbl. 5.3, we can see that the neural network based system yields a lower WER than the NTT CHiME3 II system, i.e. the winning contribution without the speaker adaptation. This confirms the results from the previous comparison as the NTT system also employs a mixture model with a time-varying complex distribution to obtain masks to estimate the SCMs. Their permutation alignment is more sophisticated than the one used for the previous experiments and their system can probably be regarded as the upper bound in terms of performance of a mixture model based system on this task.

Another interesting comparison is with the Kaldi setup III which uses a comparably strong acoustic model (although followed by a rescoring step with a RNN-LM) and the BeamformIt! [AWH07] beamformer to preprocess the multi-channel signal. This system achieves a WER of 6.84 % and therefore a slightly lower one than the BLSTM mask estimator system from this work. However, there is a significant gain from using a language model rescoring step as can be seen when also considering the Kaldi setup IV which uses our open source version of the BLSTM mask estimator and GEV beamformer and is conceptually identical to the system evaluated here. With a RNN-LM rescoring step, this setup yields a WER of 4.01 % which is significantly lower than the one achieved by the identical system but with the BeamformIt! beamformer. Indeed, analysis in our work for the CHiME 4 challenge [HDH16b] showed a similar performance improvement when replacing BeamformIt! with a neural network supported GEV beamformer.

While the previous analysis was concerned with speech signals deteriorated by interfering sources, we now turn to the REVERB dataset where the deterioration is mainly caused by the speech signal itself, namely by its reverberation. The two systems are set up in the same way as in the previous evaluations but this time the beamformer is used to remove the reverberant part of the signal (see Sec. 4.5.1). For the cACGMM model this does not change anything and for the neural network model this means that the targets during the training of the mask estimator are calculated differently (see Sec. 7.3.1).

We include a third system in this comparison. This system uses WPE (see Sec. 4.5.2)

Table 7.8: Comparison between a BLSTM (Sec. 7.1) and a cACGMM (Sec. 4.4) mask estimator with a GEV beamformer and WPE for the REVERB task.

			WRBN		WRN	
Mask estimator	Beamformer	WPE	`EVAL`	`EVAL+10dB`	`EVAL`	`EVAL+10dB`
None	–	✗	15.52	15.52	15.02	15.02
None	–	✓	13.77	13.55	13.13	12.96
BLSTM	GEV	✗	**8.71**	**7.86**	**8.43**	**8.09**
cACGMM	GEV	✗	13.23	12.66	13.07	12.54

to process and dereverberate the signal prior to feature extraction. In contrast to the beamforming based systems, WPE is a multiple inputs multiple outputs (MIMO) method. As the acoustic model only handles one channel, we select it according to the filelist for the single-channel task of the REVERB challenge after dereverberation. We use the same STFT parameters as in all previous experiments (a window size of 25 ms and a shift of 10 ms). The delay Δ and number of filter taps L_{taps} (see Sec. 4.5.2) is optimized to yield the best performance on the development set with a grid search where $\Delta \in \{1, 2, 3\}$ and $L_{\text{taps}} \in \{5, 10, 15\}$. Again, this method just extends the feature extraction process and the acoustic model is not adapted in any way.

The results are shown in Tbl. 7.8. We first note, that all systems show some sensitivity to the overall power of the signal. Simply increasing it by 10 dB improves the performance for all configurations where the signal is enhanced prior to feature extraction. For the two systems relying on beamforming the sensitivity is caused by the beamforming operation and not by the mask estimation. The latter is invariant to scale for both, the cACGMM and BLSTM since both models normalize the signal prior to processing. Again, the systems using the WRN acoustic model achieve lower WERs compared to the WRBN systems for all except one case. With the BLSTM mask estimator the WRBN acoustic model yields a WER of 7.86 % while with the WRN the WER is 8.09 %. However, as mentioned earlier, this is within a range where no conclusions can be drawn from a single experiment.

Overall, the system employing the neural network based mask estimator significantly outperforms the other approaches. It should be mentioned that the cACGMM was not designed for the task of dereverberation. Yet, it performs surprisingly well compared to WPE. It is also worth noting that all of the evaluated enhancement methods improve the performance over the corresponding single-channel baseline. This again highlights the benefits of exploiting multiple channels in a principled way. The WER of the best multi-channel system is nearly 50 % lower than the one of the best single-channel system.

Comparing the results with the baseline results from Tbl. 5.4 shows that the system with the BLSTM mask estimator achieves the lowest WER of all approaches, even though no language model (LM) rescoring is used. Compared to the challenge baseline result from 2014, the WER is reduced by over 80 %. It can even improve over the winning solution despite being a much simpler system with only a few components.

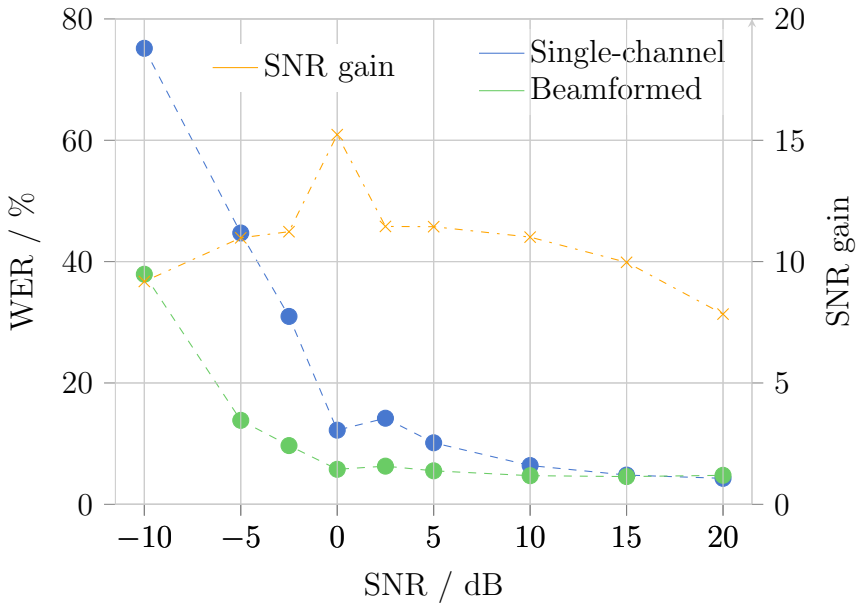

Figure 7.11: WERs (left scale) for different SNR conditions on the simulated development set from the CHiME task using only channel 5 ("Single-channel") or the output of a beamformer system ("Beamformer"). The orange line shows the SNR gain (right scale) achieved by the beamformer. See text for details.

7.4.3 Performance over SNR

The previous results show that the neural mask estimator performs well on the two datasets used for evaluation. For both, all utterances in the test sets have a fixed SNR. But an interesting property of an enhancement system to evaluate is its behavior for different SNR conditions. We already saw that significant gains are achieved in the low SNR regime (the average SNR for the CHiME test set is estimated to be around 4 dB [Bar+17]). However, for an enhancement system to be of practical use, the performance for high SNR conditions is equally important as the performance for the default scenario with no interferences should not degrade.

To evaluate the performance for different SNR conditions, we use the simulated data from the development set of the CHiME task and adjust the power of the target and interferences such that all utterances have the same SNR. We consider SNRs of −10 dB, −5 dB, −2.5 dB, 0 dB, 2.5 dB, 5 dB, 10 dB, 15 dB and 20 dB. Our baseline are the WERs achieved using only a single channel (the lower center microphone) with the WRN acoustic model. We compare this against a system using the BLSTM mask estimator in combination with the GEV beamformer to enhance the multi-channel data first and then the same WRN acoustic model to transcribe it. For this system, we also calculate the SNR gain by first estimating the beamforming filter, applying it to the spatial images of the target speech and interference signal and then calculating their ratio in dB.

Results are shown in Fig. 7.11 with the numbers given in Tbl 7.9. Up until an SNR of 2.5 dB, the WERs with the enhanced signal are decreased by more than 50 % relative.

Table 7.9: WERs for different SNR conditions on the simulated development set from the CHiME task using only channel 5 ("Single-channel") or the output of a beamformer system ("Beamformer").

	SNR in dB								
System	-10	-5	-2.5	0	2.5	5	10	15	20
Single-channel	75.16	44.72	30.97	12.22	14.19	10.14	6.39	4.82	**4.28**
Beamformed	**37.92**	**13.84**	**9.68**	**5.77**	**6.29**	**5.52**	**4.72**	**4.57**	4.78

Afterwards, the difference monotonically gets smaller such that for a SNR of 15 dB, the WER is only improved by 5 %. In the 20 dB scenario, the multi-channel system finally performs worse compared to the single-channel baseline.

The SNR gain achieved by the system ranges between 7.5 dB and 15 dB, with a visible peak at 0 dB. This might be explained by the training data, which, after augmentation, roughly has an average of 0 dB. Otherwise, it is surprisingly constant even for high input SNR values where the gain measured in WERs vanishes. This underlines that the SNR is only loosely related to the WER, supporting the choice of our evaluation criterion to be the WER.

In summary, the beamformer system shows good performance until a SNR > 15 dB is reached. In these higher SNR scenarios, processing the signal actually degrades the performance. A possible solution would be to include more data with high SNR values in the training process although this might degrade accuracy in low SNR scenarios. An alternative would be to coarsely estimate the SNR upfront and only process the multi-channel signal when the estimated SNR is below a certain threshold.

7.4.4 Comparison of different beamformers

So far, only the GEV has been considered for the beamforming operation. But the application of the mask estimator is not limited to this specific beamformer and Sec. 4.2.2 already presented different statistical beamformer criteria. These are evaluated using the BLSTM mask estimator to estimate the SCMs in the following. Note that the mask estimator is agnostic to the specific choice of the beamformer and the same model can be used to evaluate all of them. The same is true for the AM which we do not adapt to a specific front-end (this will be discussed in-depth in Sec. 10.2.4). For the MVDR, we make use of the Rank-1 approximation discussed in Sec 4.2.2 and set the direction of distortionless responses to $\mathbf{h} = \mathcal{P}(\mathbf{\Phi}_{ss})$. The reference channel for the MVDR and MWF is chosen to be the center microphone on the bottom frame.

WERs for the CHiME task are shown in Tbl. 7.10. All evaluated beamformers improve the performance and there is little differentiation among them. For the WRN acoustic model, the difference between the best and worst performing beamformer are merely 0.34 percentage points in the WER. A similar gap exists for the real recordings and the WRBN acoustic model. Only with the simulated recordings and the WRBN AM the difference is slightly higher (0.63 percent points). Given the fact that this is just a single configuration and implementation and the large amount of hyper-parameters, no

Table 7.10: Comparison of different beamformers with a BLSTM mask estimator for the CHIME task. Best results are marked in bold.

	WRBN		WRN	
Beamformer	SIMU	REAL	SIMU	REAL
GEV	6.93	**7.79**	6.78	7.34
MVDR	7.33	8.11	6.85	7.53
MWF	**6.70**	8.06	**6.51**	**7.21**
MWF with Rank-1 approx.	6.93	7.83	6.77	7.38

Table 7.11: Comparison of different beamformers with a BLSTM mask estimator for the REVERB task.

	WRBN		WRN	
	EVAL	EVAL+10dB	EVAL	EVAL+10dB
Beamformer				
GEV	8.71	**7.86**	**8.43**	**8.09**
MVDR	8.45	7.86	8.50	8.41
MWF	9.41	9.13	8.71	8.34
MWF with Rank-1 approx.	**8.41**	8.52	8.49	8.14

definite conclusions can be drawn from this result. The distortions possibly introduced by the GEV do not harm the system performance in this scenario. Also, contrary to the findings in [Wan+18], the Rank-1 approximation does not improve the results for our system.

As shown in Sec. 4.2.2, all vectors approximately (i.e., under the Rank-1 assumption) point into the same direction and the only difference is a frequency dependent scaling factor. Despite the sequence normalization this can have an influence since the feature extraction combines multiple frequencies. But the results suggests that this influence is rather small.

The results for the REVERB task as shown in Tbl. 7.11 speak roughly the same language. This time, the GEV gives the best results while the MWF lags behind especially in combination with the WRBN acoustic model. Again, the results can be improved by a simple gain factor for all beamformer types although the differences vary among them.

We conclude from these results that at least for the offline case there is no significant difference in performance between the evaluated beamforming methods. For the following experiments in this work, we will stick with the GEV beamformer as the one with an overall solid performance. However, we assume that any insights gained with

this beamformer apply to the other variants as well.

7.4.5 Combination with WPE

WPE can be formulated as a MIMO system and as such also has the property of preserving directional information contained in the signal [Del+14a]. Consequently, it can be combined with beamforming in a straightforward way by simply first using WPE to dereverberate the signal followed by a beamformer to condense it into a single-channel signal for ASR. While WPE and beamforming have been already evaluated separately for the REVERB task, we now evaluate a system that combines both methods. As before, the number of filter taps and the delay is optimized on the development set for the individual models. The system allows for many different ways to incorporate the dereverberated signal since there are three components which can use the signal independently. These are the mask estimation, the SCMs estimation and the beamforming operation itself. For example, one could estimate the masks and the SCMs on the original observation and then use the resulting beamforming vector to filter the dereverberated signal. Not all possible combinations are intuitively reasonable and in the following we only investigate to use the WPE dereverberated signal for:

1. The mask estimation
2. The mask and SCMs estimation
3. The mask, SCMs estimation and the beamforming operation
4. The SCMs estimation and the beamforming operation
5. The beamforming operation

All results of this evaluation are shown in Tbl. 7.12. We first note that even though WPE on its own was not able to improve the results as much as the beamformer alone, a front-end combining the two methods achieves the best performance. Adding WPE as an additional enhancement step before the mask estimation, SCM estimation and beamforming operation improves the WER by 10 % – 18 %.

The biggest gains are achieved for the matched condition, i.e. when the dereverberated signal is used for all components. Using it only for the mask estimation improves the WERs by about one percent point absolute. Also using the dereverberated signal for the SCM estimation deteriorates this result slightly when the estimated filter is then applied to the unprocessed observation. This is to be expected as the resulting filter will not try remove reverberation from the signal which was already removed by WPE. Yet, the performance of this setup is still reasonable, indicating that there is still enough reverberation left in the WPE signal to estimate spatial fiters to remove it. When the unprocessed signal is used for mask estimation and the dereverberated signal for SCM estimation and beamforming, the performance is on-par with the case where the dereverberated signal is used for all components. This result underlines how well the mask estimator can cope also with reverberation.

Table 7.12: Performance for a system with WPE, a BLSTM mask estimator and a GEV beamformer for the REVERB task. Reference results for WPE and GEV beamformer only are shown in gray. A checkmark in columns 2 – 4 indicates if the dereverberated signal has been used for the respective component while a cross denotes the use of the unmodified observation.

		WPE		WRBN		WRN	
Front-End	ME	SCM	BF	`EVAL`	`EVAL+10dB`	`EVAL`	`EVAL+10dB`
GEV BF	✗	✗	✗	8.71	7.86	8.43	8.09
WPE	✗	✗	✗	13.77	13.55	13.13	12.96
GEV BF	✓	✓	✓	**7.22**	**7.02**	7.23	7.31
GEV BF	✓	✗	✗	7.76	7.49	7.92	7.95
GEV BF	✓	✓	✗	7.93	7.96	8.04	7.96
GEV BF	✗	✗	✓	7.37	7.38	7.45	7.49
GEV BF	✗	✓	✓	7.36	7.14	**7.08**	**7.09**

7.4.6 Comparison between BLSTM and U-Net

In the previous experiments only the BLSTM based mask estimator was considered. In the following, we now compare the performance with one based on the U-Net architecture, i.e. a fully convolutional one. For the U-Net, we investigate two different variants. One, which operates on each sensor individually and one, which estimates a single mask for all available sensors. The latter configuration leaves the pooling of the masks to the network. It might be able to learn to select appropriate channels resulting in a greater robustness against broken sensors. Also, inter-channel relationships could be exploited and further improve the estimation. Note however, that the input for the network is still the magnitude spectrogram, i.e. it is not possible to use phase information. It also sacrifices the array independence property and the network is specific to the microphone configuration which was used during training. But depending on the application, this might be an acceptable trade-off as long as it yields improved results.

All systems are evaluated using the GEV beamformer.

Results for the CHiME task are shown in Tbl. 7.13. The U-Net operating on a single channel achieves WERs which are around 4 % better than those achieved by the BLSTM mask estimator. Given the already low WERs, one can conclude that both mask estimators work equally well. However, one advantage of the U-Net is the possibility to process all frames in parallel while for the BLSTM the frames need to be processed sequentially. With an appropriate hardware setup, this results in a significant improvement in execution time. Similar to the findings for the acoustic model, this again indicates that the receptive field of the CNN captures enough temporal information and recurrence might not be needed.

When all channels are used as an input, the performance on the simulated data

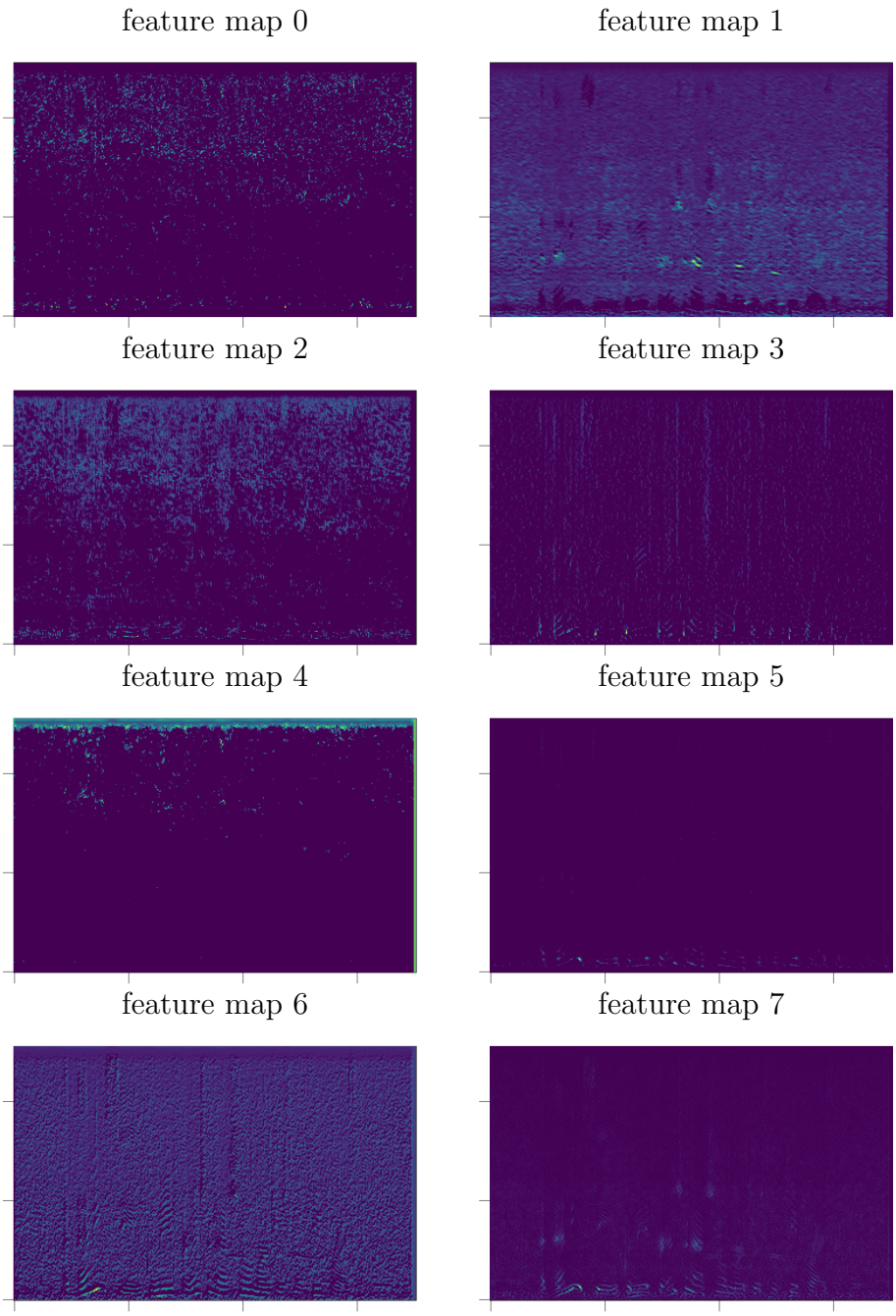

Figure 7.12: First eight feature maps calculated from the input for a U-Net operating on all channels for a real recording.

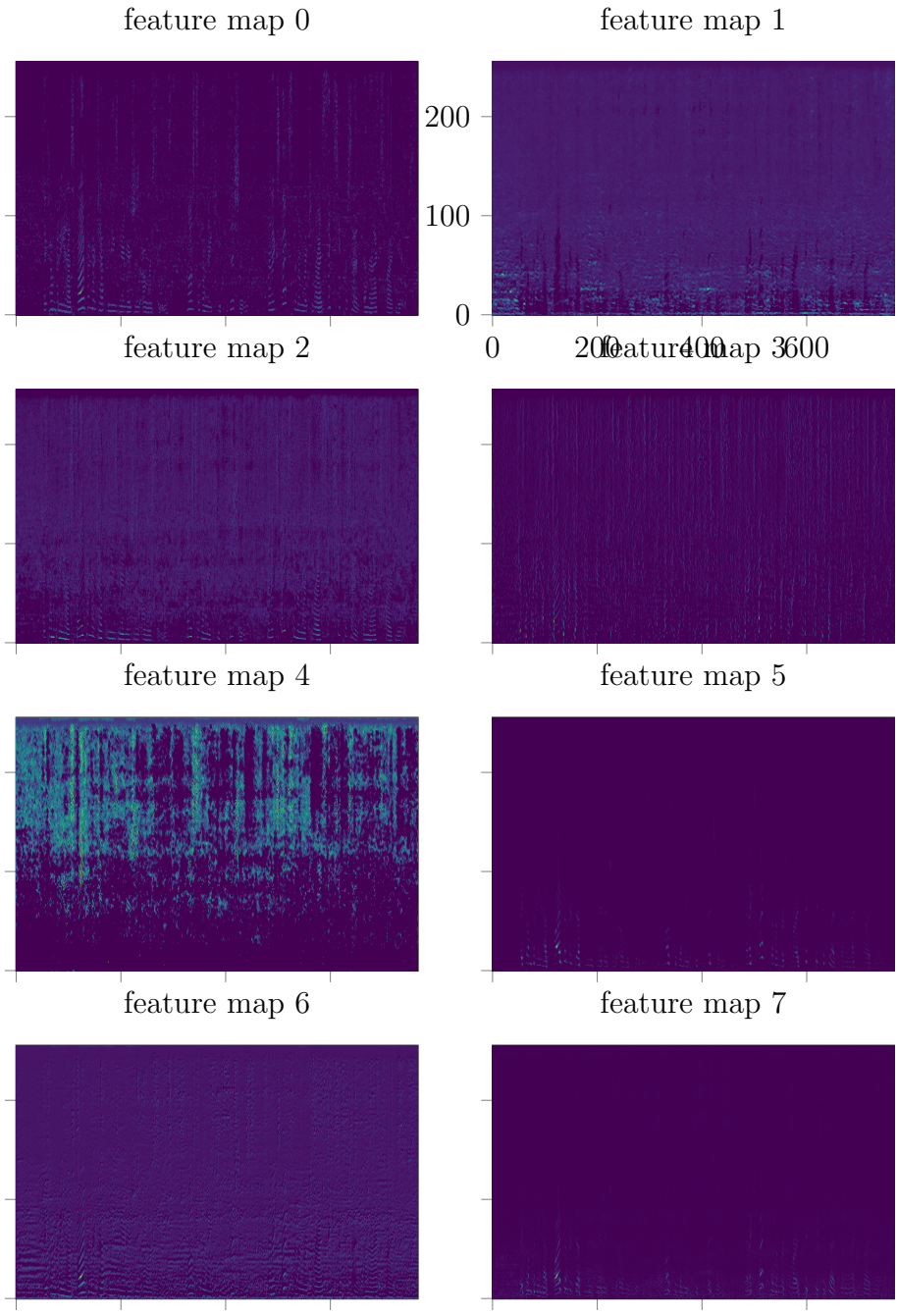

Figure 7.13: First eight feature maps calculated from the input for a U-Net operating on separate channels for a simulated recording and the lower center microphone channel.

Table 7.13: Performance comparison between the BLSTM and U-Net mask estimator on the CHiME task. *U-Net single-channel* treats each channel independently while *U-Net all channels* works on all channels simultaneously. See text for details.

	WRBN		WRN	
Mask estimator	SIMU	REAL	SIMU	REAL
BLSTM	6.93	7.79	6.78	7.34
U-Net single-channel	6.63	**7.54**	6.69	**7.11**
U-Net all channels	**6.58**	38.78	**6.24**	38.82

improves slightly. But for the real recordings, the system breaks and the WERs are higher than the ones achieved without any beamforming. For a further investigation of this issue, we look at the first feature maps which are calculated from the input. The first eight of them are shown in Fig. 7.12 for an exemplary real recording (which does not suffer from any microphone failures) while Fig. 7.13 shows them for a simulated recording. We can see that for example the first filter is already able to extract speech patterns from simulated recordings but only produces artifacts for the real recordings. Similarly, the forth filter seems to focus on background noise but again only works for the simulated recordings. These observations can be made for many of the filters and are also not specific to this example but apply generally when looking at other utterances. Since this feature map is the first one and directly computed from the input, this suggests that there is already a mismatch in the microphone configuration for the CHiME task between simulated and real data[7]. And while the system has learned to account for inter-channel differences on the simulated training data, this cannot be transferred to real recordings. Even worse, the system becomes very sensitive to a slight mismatch and fails once applied to real data. This highlights the benefits of processing each channel individually and pooling the estimated masks later. Not only can such a system cope with arbitrary array geometry mismatches, but also yields very similar performance compared to a system that uses the information of all channels.

Although there is a significant gain mismatch between the training and evaluation data for the REVERB task, the U-Net using all channels achieves similar performance compared to the single-channel version as can be seen in Tbl. 7.14. Note that for this task the array geometry does not change and no microphones fail. Overall, the WERs are even marginally better, but given the test set size the difference is not significant and does not justify giving up the array geometry independence.

Both CNN based models are more sensitive to a gain mismatch compared to BLSTM mask estimator. The reason for this is probably the use of batch normalization instead of sequence normalization, rendering the network sensitive to the scaling of the signal. As a consequence, for this task, the BLSTM performs better than the U-Net(s) but the difference can be attributed to the different normalization rather than the different

[7]Recall that the mask estimator is trained only on simulated data.

Table 7.14: Performance comparison between the BLSTM and U-Net mask estimator on the REVERB task. *U-Net single-channel* treats each channel independently while *U-Net all channels* works on all channels simultaneously. See text for details.

	WBRN		WRN	
Mask estimator	`EVAL`	`EVAL+10dB`	`EVAL`	`EVAL+10dB`
BLSTM	**8.71**	**7.86**	**8.43**	**8.09**
U-Net single-channel	9.48	8.33	10.38	8.36
U-Net all channels	9.32	8.28	9.75	8.46

architecture.

In summary, there is no big difference between the mask estimators in terms of performance for both tasks. Using all channels as input does not significantly improve the results in case of matching array geometry and is too sensitive against possible mismatches. As long as no phase differences can be exploited, using a single-channel mask estimator is the better option.

7.4.7 Array independence

The system was designed with the goal of being independent of the array geometry, including the number of microphones. But the performance will inevitably depend on the number of microphones as more microphones allow to create a sharper beampattern with better attenuation of interfering sources (see Sec. 4.2.1). In the previous experiments we always used all available channels in the same order. The following experiments now examine the differences in performance for different numbers of microphones. For all experiments we use the BLSTM mask estimator, the GEV beamformer and the WRN acoustic model. As the previous findings suggest, the results are likely also valid for other configurations although the absolute numbers may differ slightly.

For the CHiME task, we conduct two experiments.

The first one gradually decreases the number of microphones from six to two. Note that the same components (mask estimator, beamformer, acoustic model) are used for all configurations without any specific training or adaptation. The only difference is the dimension of the SCMs and the beamforming vector. For the six microphones, we permute the order to show that the system is indeed invariant to a permutation of the microphones. The other configurations are chosen in a way that results in a symmetric configuration with a varying amount of microphone spacings but are arbitrary otherwise.

In the second experiment we test for the influence of the individual microphones by leaving out a single microphone.

Results for the different microphone configurations are shown in Tbl. 7.15. As expected, permuting the channels does not change the WER as can be seen in the first row of the results table where the order of the microphones in inverted. There is also a visible trend towards higher WERs when less microphones are considered. Again, this

Table 7.15: WERs on the CHiME task for different microphone configurations.

	SIMU	REAL
Microphone #		
6+5+4+3+2+1	6.78	7.34
1+3+4+5+6	7.32	7.67
1+3+4+6	8.08	8.75
2+4+6	7.90	10.03
5+2	8.48	12.57

Table 7.16: WERs on the CHiME task when leaving out one single microphone.

	SIMU	REAL
Microphone #		
2+3+4+5+6	6.84	7.80
1+3+4+5+6	7.32	7.67
1+2+4+5+6	6.82	7.62
1+2+3+5+6	7.32	7.97
1+2+3+4+6	7.20	8.14
1+2+3+4+5	6.98	7.92

is an expected behavior as elaborated before. What stands out is the difference between the simulated and the real recordings. While reducing the number of microphones from six to two leads to relative WER increase of 25 % for the simulated data, the WER for the real data increases by over 70 %. A major jump in WER can be observed when reducing the number of microphones from four to three for the real recordings while for the simulated recordings the WER even decreases slightly. This might be attributed to the specific choice of the microphones which includes the backwards facing one (microphone #2) for the three and two microphones configuration. Additionally, the robustness to a microphone failure also decreases with fewer microphones considered.

The influence of individual microphones can be inferred from Tbl. 7.16. All configurations perform worse compared to using all channels, i.e. the system is able to exploit additional information from any of the microphones. Interestingly this is even true for the microphone facing backwards. The microphone in the middle of the bottom frame contributes the most, not including it leads to the worst WER (on real recordings). This is in agreement with the findings of the authors of the dataset who identified this channel as the most reliable one and used it for the baseline system [Bar+15]. But overall, the WER only varies in a narrow range and no single microphone has a huge influence on the performance.

Table 7.17: WERs on the REVERB task for different microphone configurations.

	EVAL	EVAL+10dB
Microphone #		
7+6+5+4+3+2+1+0	8.43	8.36
7+5+3+1	9.37	9.14
0+4	11.83	11.30

For the REVERB task we only consider three different configurations. One with a flipped ordering and all channels, one with four channels and one with two channels. The microphones are again arbitrarily chosen but such that the minimum distance between any two microphones is maximized for the circular array.

The results are displayed in Tbl. 7.17 Reducing the number of microphones from eight to four increases the WER by a little more than 10 % while a reduction to only two microphones leads to a degradation by 40 %. Again, this was expected although the magnitude of the decline was unknown. The impact of reducing the number of microphones is smaller compared to the CHiME task. One explanation for this might be that the array for the REVERB task is circular and symmetric, i.e. the channels are interchangeable. For the very same reason we refrain from evaluating the influence of individual channels for this task.

Overall, the results show that the system profits from multiple microphones although the achievable gains saturate as the number of microphone increases. However, at least in this study the performance never degrades by adding a microphone. Combined with the geometry independence property of the system this is a desirable feature since performance can be improved by simply switching to a larger array without any further modifications required. In Sec. 3.3 we discussed several alternatives to exploit multiple channels for ASR and found that the ones using IPD features and those working on the raw waveform cannot easily be scaled to using more microphones. This is a distinctive advantage of the presented approach made possible by the classical statistical signal processing ultimately combining the channels.

7.5 Related work

It should be mentioned that, although we were the first to publicly describe neural network supported beamforming in our CHiME 3 submission [Hey+15a], we were not the only ones working on this idea. Notably, the works by Erdogan et al. pursue a similar idea and were published shortly afterwards [Erd+16]. A slightly different approach is taken by [PZP17] where the authors use features based on the eigenvectors as an input for a neural network which then estimates the masks for the SCM estimation.

Others have built upon this system an extended it in different ways. Examples include the works by Žmolíková et al. [Kat+17] where an additional speaker embedding vector allows to extract a specific speaker from a mixture. The embedding vector guides

the mask estimator to only mark those tf-bins where a certain speaker is active. This speaker is then extracted from a mixture via beamforming. In [Liu+18] the system is specifically tailored towards a setting where only two microphones are available. The IPD features already discussed in Sec. 3.3 are used to provide the mask estimator with spatial information. With some other additional improvements, the authors were able to achieve significant performance gains for their dataset. Nakatani et al. combine the neural network based mask estimator with the cACGMM and obtain better performance on the real recordings for the CHiME task [Nak+17]. An unsupervised extension building upon this will be presented in Ch. 9.

A similar concept is also used for multi-channel source separation. Here, the task is to separate multiple (speech) sources from a mixture. This problem can be tackled by increasing the number of outputs of the mask estimator to match the (maximum) number of sources. In this setting, the global permutation problem arises even for the network as it is not defined which output will correspond to which source. During training, this problem can be avoided with permutation invariant training [Yu+17]. If the number of sources is unknown upfront and the system should be able to handle an arbitrary number of sources, techniques like deep clustering [Her+16] or deep attractor networks [CLM17] can be used to estimate the masks for each source. These can also be integrated with a spatial mixture model for improved performance [DH19].

Overall, the use of neural networks for mask based beamforming has been widely adopted by the community[8] and is regarded as the current state-of-art. For example, nearly all submissions for the CHiME 4 challenge use a neural network based beamformer as a front-end processing technique [Vin+16]. Kaldi recipes using it as a front-end technique outperform all previous baselines, sometimes by a large margin [Che+18]. Works by the author of this thesis [HBS18] and by Boeddeker et al. [Boe+18b] show that the technique also works on large scale data and in more practical settings for smart home devices. In fact, with slight modifications and additional components, the neural network based beamforming also found its way into commercial products such as the Apple HomePod[9]. Some of the necessary modifications which reduce the latency of the system will be presented and discussed in the next chapter.

7.6 Summary

The biggest advantages of using a neural network to estimate masks for beamforming lie in its ability to exploit spectro-temporal correlation and the avoidance of any permutation problem. While the parameters of a spatial model have to be estimated on the signal we aim to enhance, the parameters of the neural network are inferred from a big training corpus and generalize well so they can be readily used for new signals. And since the output classes are determined and specified during the training, no additional steps are necessary to determine which class is which – a problem that can be hard to solve when only considering the spatial distribution and no spectral patterns. This

[8] On March 30th 2020 the relevant publication ([Hey+15a], [HDH16a], [HDH16b] and [Hey+17]) had a total of 379 citation according to Google Scholar.

[9] https://machinelearning.apple.com/2018/12/03/optimizing-siri-on-homepod-in-far-field-settings.html

makes neural networks especially suitable for the task at hand. For the CHiME as well as for the REVERB task, performance improvements of about 50 % can be achieved compared to a single-channel system which is considerably higher than those achieved by a cACGMM based system. The gains are stable over a wide SNR range and can be achieved with a BLSTM based mask estimator as well as one based on CNNs. For all systems, the choice of the beamformer criterion plays a minor role and has little influence on the final performance. The system is agnostic to the microphone configuration and can profit from adding more sensors without any modification necessary.

8 Reducing latencies

In the previous chapter we presented and evaluated a neural network supported beamformer and the WRBN / WRN acoustic model. We could show that combined, state-of-the art results are achieved for the CHiME and REVERB task. However, all considered configurations of the system operated in batch-mode and required access to the whole utterance, i.e. the latency is at least the length of the utterance plus the computational time required by the system.

Especially for applications with immediate interaction with the user, such a high latency is not practical and has a severe impact on the user experience. This chapter builds upon the developed system with the goal of reducing the latency to only a few frames. In particular, with a frame shift of 10 ms we aim for an inherent latency $<$ 100 ms. With inherent, we mean the latency of the system without taking into account the computational time required for the models.

The system can be broadly divided into three components; The mask estimator, the beamformer and the ASR back-end. In the following we will analyze the dominant source of latency for each of those components and propose ways to reduce the latency. We mainly focus on the front-end (i.e. the mask estimator and the beamformer) and only consider the back-end for completeness.

8.1 Mask estimator

In principle, the mask estimator could operate in a frame-online fashion, i.e. process each frame as soon as it becomes available and estimate the class affiliations for each frequency bin. But one single frame does not always contain all necessary information to solve this task and, intuitively, adding temporal context helps to resolve many arising ambiguities. It is, for example, very hard to distinguish fricatives from noise from a single frame.

Another problem is related to the signal power. Neural networks, as a non-linear model, are sensitive to the scaling of the input signal. I.e. if the power of the signal is doubled (e.g. by using sensors with a different dynamic range or simply by placing the sensors closer to the sources), the output can be very different although the acoustic scene did not change. To circumvent this issue, the input to the model is usually normalized. When the whole utterance is available upfront, this can be done on an utterance level by calculating the (frequency-wise) mean and variance to normalize the signal. However, in the low-latency setting, the required statistics are not available. One option is to calculate a global mean and variance on, for example, the training set

Table 8.1: Feedforward network configuration for mask estimation

	Units	Type	Non-Linearity	p_{dropout}
L1	2048	FF	RELU	0.0
L2	1024	FF	RELU	0.0
L3	512	FF	RELU	0.0
L4	2×257	FF	Sigmoid	0.0

and use these values to normalize the signal. While this strategy works well when the power levels for this set are comparable to the ones encountered in practice, performance deteriorates severely once this assumption does not hold anymore. One example for this is the REVERB task where the test set deviates significantly from the training set. This also conflicts with the goal of being independent of the microphone configuration as using microphones with different gain characteristics will also lead to the same issue with this approach.

We explore three different approaches to cope with the two problems.

The first one is a LSTM based network with different normalizations of the input. The second and the third one are based on simple feedforward networks operating on a window of frames.

8.1.1 LSTM

The LSTM mask estimator basically resembles the BLSTM mask estimator (see Tbl. 7.1) but without the time-reverse LSTM layer and normalizations. To compensate for the sequence normalization, we experiment with three different input normalizations. Two which update the signal statistics with each new frame and one which calculates the statistics on the training set. As already discussed, the latter is sensitive to a gain mismatch but serves as a baseline. The former two calculate the running average (variant 1) and the running variance (variant 2). Statistics are updated with each new frame and the current frame is centered (and scaled) with the most recent estimate of the mean (and variance). Note that only the second variant is actually invariant to a constant gain but the variance may be too susceptible to estimation errors so a mean-only normalization is provided as a reference.

Both, the normalization and the model work in a frame-online fashion, i.e. as soon as a new frame becomes available, the mask can be calculated immediately without any additional latency (except for the computation time which we ignore here). Nevertheless, a large context in form of all past frames can be utilized for the current estimation.

8.1.2 Instance norm feedforward network

Instead of storing the context in an aggregated vector and working on a single frame only, the instance norm feedforward (IN-FF) network uses a left and right context of two frames. A single mean and variance value is calculated for the resulting feature

vector with $5 * 257 = 1285$ elements. The mean and variance value is then used to normalize the vector, rendering the estimator invariant to the scaling of the signal and also contributing the instance norm [UVL16] part in the name. The network, whose configuration is summarized in Tbl. 8.1, simply consists of four feedforward layers. Masks are estimated only for the center frame and thus the step size of the context window is one. Using only two past and future frames is a trade-off between the context information available, the latency and the size of the feature vector. With the shift size of 10 ms, the future frames introduce a latency of only 20 ms. Contrary to the LSTM network, we do not use any dropout during training as it hindered learning during initial experiments.

8.1.3 Scale invariant feedforward network

The gain ambiguity of the IN-FF network is achieved by normalizing each feature vector. But this might cause (numerical) problems for those parts of the signal where no source is active at all. As an alternative, we here propose an architecture we term scale invariant feedforward (SI-FF) network. This network does not require any feature normalization and is, as the name suggests, invariant to a scaling of the input signal. To derive the network, we start by identifying which parts of the network are not scale invariant and replace them with appropriate operations.

The affine transformation $\mathbf{W}\mathbf{x} + \mathbf{b}$ becomes scale invariant once we only keep the linear mapping and avoid the translation. For the non-linearities we have the ReLU, i.e. $\max(\mathbf{x}, \mathbf{0})$ which is already scale invariant and the sigmoid, which is not invariant. However, if we slightly reframe the problem and consider[1] IRMs instead of IBMs, we can replace this non-linearity by the ratio of the output for one class and the sum over all classes. This ratio is scale invariant. With these minimal changes, the network is now completely invariant to any scaling of the input signal as we have

$$\begin{aligned} \mathbf{M}_s &= \frac{\tilde{\mathbf{M}}_s}{\tilde{\mathbf{M}}_s + \tilde{\mathbf{M}}_n}, \\ \mathbf{M}_n &= \frac{\tilde{\mathbf{M}}_n}{\tilde{\mathbf{M}}_s + \tilde{\mathbf{M}}_n} \end{aligned} \tag{8.1}$$

$$\tilde{\mathbf{M}}_i = \left(\max\left(\mathbf{W}_{i\mathrm{L4}}\left(\max\left(\mathbf{W}_{\mathrm{L3}}\left(\max\left(\mathbf{W}_{\mathrm{L2}}\left(\max\left(\mathbf{W}_{\mathrm{L1}}\mathbf{y}_{\mathrm{context}}, \mathbf{0}\right)\right), \mathbf{0}\right)\right), \mathbf{0}\right)\right), \mathbf{0}\right)\right), \tag{8.2}$$

and

$$\alpha\tilde{\mathbf{M}}_i = \left(\max\left(\mathbf{W}_{i\mathrm{L4}}\left(\max\left(\mathbf{W}_{\mathrm{L3}}\left(\max\left(\mathbf{W}_{\mathrm{L2}}\left(\max\left(\mathbf{W}_{\mathrm{L1}}\alpha\mathbf{y}_{\mathrm{context}}, \mathbf{0}\right)\right), \mathbf{0}\right)\right), \mathbf{0}\right)\right), \mathbf{0}\right)\right). \tag{8.3}$$

8.2 Beamformer

Once the beamforming vector is estimated, applying it to the signal does not introduce any additional latency. Also, for any criterion presented in Sec. 4.2.2, the optimal filter

[1] See Sec. 9.3.3 for a discussion and implications on the performance.

can be readily calculated once the SCMs are known. The estimation of the latter is the main issue when it comes to low latency applications as the statistics are aggregated over time.

A possible solution, which we will use in this work, is a block-wise processing with an exponential smoothing of the estimated SCMs. The SCM for the block b and signal x is then calculated as

$$\mathbf{\Phi}_{xx}(b, f) = \alpha \mathbf{\Phi}_{xx}(b-1, f) + (1-\alpha) \sum_{k=(b-1)L_b}^{bL_b} M_x(k, f)\mathbf{x}(k, f)\mathbf{x}^{\mathrm{H}}(k, f), \tag{8.4}$$

with a block length of L_b and a smoothing factor α. The block length determines the latency of the system and, in combination with α, the rate at which the system can adapt to a changing scenario (e.g. if the speaker moves).

If prior information about the target position and/or noise is available, this can be integrated by choosing the initial estimate $\mathbf{\Phi}_{ss}(0, f)$ accordingly. Information for the target can be obtained for example from a wake word interaction.

However, here we assume that no information is available and initialize the SCM for the target with zeros and the one for the interferences with an identity matrix scaled by 0.001.

The exponential smoothing biases the estimates towards the prior for the first blocks[2]. For larger values of α and/or block length L_b this can become an issue. We therefore divide the SCMs by $(1-\alpha^b)$ to compensate for the bias in the beginning.

Another problem arises at the beginning of an utterance when the target has not been active yet. As we consider each frequency separately, this problem becomes even more pronounced. One way to mitigate this is to wait until the target becomes active by choosing a threshold on the sum of its mask as we proposed in [Kit+16]. The threshold allows to trade-off latency against estimation errors but also is another hyper-parameter to tune and the threshold works on a global scale, i.e. individual frequencies can still be affected. This can be solved by smoothing across frequencies as presented in [HHH18].

Here, however, we propose a different solution which exploits a property of the GEV beamformer. Namely, the GEV allows to replace the SCM of the target with the SCM of the observation under the assumption that the target signal and interferences are uncorrelated (also see Sec. 4.2.2). This can be easily seen by looking at the criterion itself:

$$\begin{aligned}
\mathbf{w}_{\mathrm{GEV}} &= \underset{\mathbf{w}}{\operatorname{argmax}} \frac{\mathbf{w}^{\mathrm{H}}\mathbf{\Phi}_{ss}\mathbf{w}}{\mathbf{w}^{\mathrm{H}}\mathbf{\Phi}_{nn}\mathbf{w}} && (8.5)\\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \frac{\mathbf{w}^{\mathrm{H}}\mathbf{\Phi}_{ss}\mathbf{w}}{\mathbf{w}^{\mathrm{H}}\mathbf{\Phi}_{nn}\mathbf{w}} + 1 && (8.6)\\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \frac{\mathbf{w}^{\mathrm{H}}\mathbf{\Phi}_{ss}\mathbf{w}}{\mathbf{w}^{\mathrm{H}}\mathbf{\Phi}_{nn}\mathbf{w}} + \frac{\mathbf{w}^{\mathrm{H}}\mathbf{\Phi}_{nn}\mathbf{w}}{\mathbf{w}^{\mathrm{H}}\mathbf{\Phi}_{nn}\mathbf{w}} && (8.7)\\
&= \underset{\mathbf{w}}{\operatorname{argmax}} \frac{\mathbf{w}^{\mathrm{H}}(\mathbf{\Phi}_{ss} + \mathbf{\Phi}_{nn})\mathbf{w}}{\mathbf{w}^{\mathrm{H}}\mathbf{\Phi}_{nn}\mathbf{w}}. && (8.8)
\end{aligned}$$

[2] The same problem appears when an optimizer estimates the first two moments of the gradients and is solved in the same way by ADAM [KB14]

Contrary to the MPDR beamformer (where the SCM of the observation replaced the SCM of the interferences) the SCM of the observation is unlikely to have a much higher time variance than the SCM of the target (speech) signal. The SCM of the observation can be estimated at any point in time, also at the beginning when the target is not yet active. However, this will inevitably introduce an error but might capture some statistics for the target for those tf-bins misclassified by the mask estimator.

8.3 ASR back-end

As mentioned earlier, we do not focus on the back-end here and so the only measure we take to reduce the latency of the acoustic model is to remove the normalizations within the layers and replace the sequence normalization of the input with one that uses fixed values inferred from the training data (see Sec 7.4.1).

8.4 Evaluation

We evaluate the influence of each individual component described in the previous three sections by replacing the offline components with the proposed low latency ones one after another. Beginning with the mask estimator and followed by the beamformer and acoustic model, we successively transform the baseline system into one that works in an online fashion. For all evaluations we use the (modified) GEV beamformer and (modified) WRN acoustic model. All models are trained as described in Sec.7.3.2 and Sec. 7.3.1 respectively.

8.4.1 Online mask estimation

First, we compare the different approaches to reduce the latency of the mask estimator presented in Sec. 8.1. To assess the influence of the mask estimator only, we evaluate the models with a GEV beamformer and WRN acoustic model both operating in batch-mode, i.e. we use the same models as in the previous chapter.

Results for the CHiME task are shown in Tbl. 8.2. The WERs for all low latency variants are higher compared to the baseline BLSTM mask estimator. However the differences are small when compared to the IN-FF and SI-FF and can be mostly explained by the reduced context and lower model complexity as the results for the FF network with sequence normalization shows. This holds for both, simulated and real recordings. Not using any future context has no noticeable effect as can be seen from the results for the LSTM with sequence normalization. Replacing the normalization with one that relies on the training statistics also barely has any influence for the simulated data but performance degrades for real recordings by 20 % relative. Removing offsets with a running mean works better and leads to WERs roughly on par with the FF networks. But additional estimation of the variance to scale the signal fails, resulting in very high WERs. This is mostly due to an unstable training process which does not converge[3].

[3]We tried to alter the learning rate but did not find a setting where the training would converge.

Table 8.2: Comparison of different low-latency mask estimators for the CHiME task. Results for the BLSTM mask estimator are given as reference. All systems use the GEV beamformer and the WRN acoustic model operating in batch-mode. Training for the system with the running variance did not converge (see text).

Mask estimator	Input normalization	SIMU	REAL
BLSTM	Sequence	6.78	7.34
FF	Sequence	6.86	7.59
IN-FF	Instance	6.81	7.87
SI-FF	–	6.81	7.63
LSTM	Sequence	6.83	7.37
LSTM	Training statistics	6.87	8.95
LSTM	Running mean	6.95	7.93
LSTM	Running mean & variance	(33.96)	(37.30)

Table 8.3: Comparison of different low-latency mask estimators for the REVERB task. Results for the BLSTM mask estimator are given as reference. All systems use the GEV beamformer and the WRN acoustic model operating in batch-mode.

Mask estimator	Input normalization	EVAL	EVAL+10dB
BLSTM	Sequence	8.43	8.09
FF	Sequence	8.61	8.03
IN-FF	Instance	9.63	7.98
SI-FF	–	10.95	8.54
LSTM	Sequence	8.52	7.96
LSTM	Training statistics	11.01	8.31
LSTM	Running mean	12.11	8.16
LSTM	Running mean & variance	11.04	8.14

The WERs for the REVERB task are given in Tbl. 8.3. Again, switching the network architecture to a LSTM or FF network while keeping the same normalization does not have any significant effect on the results. But the other proposed systems behave differently here. While the IN-FF again shows a slight drop in performance compared to the baseline, this gap further increases for the SI-FF. We speculate that this might be caused by the type of distortion which includes the same speech pattern but with a lower energy making it hard to distinguish for the scale invariant network and also harder to classify when the normalization only relies on five frames.

The gap when using the training statistics widens, especially for the unamplified data. The system relying on the running mean normalization also only achieves the baseline results for the test set with the gain. This time, a running estimation of the variance does work (i.e., the training converged), but cannot close the gap for the `EVAL` set. These observations can be explained by the big mismatch between the training and (unamplified) test set for this task. While a similar trend was already visible for the real recordings of the CHiME task which features only a slight mismatch, the big mismatch ed here highlights the gain sensitivity of the system. As discussed before, using pre-computed values or only removing the mean inevitably leads to a performance loss.

Overall, it is possible to design a low-latency variant of the mask estimator with only little impact on the performance of the overall system. The biggest issue is a gain mismatch between training and test.

8.4.2 Online front-end

We now turn to the estimation of the SCMs and evaluate the block online variant as described in Sec. 8.2. For the block-size we choose a value of 5, resulting in an additional latency of 50 ms and a size of 1 which corresponds to frame-online processing. The smoothing factor α is set to 0.999 in all experiments motivated by the fact that the scenes are mostly static for both datasets. Masks are estimated with either one of the low-latency mask estimators SI-FF and IN-FF respectively, or the BLSTM mask estimator to evaluate the impact of the low-latency SCM estimation. We do not include the LSTM mask estimator in this experiment as the feedforward networks worked on-par or better in the previous experiment. The beamforming filters are calculated using the GEV criterion in the two variants also described in Sec. 8.2. One variant uses the target speech and noise SCMs while the other one uses the SCMs of the observation and the noise. For all combinations we use the WRN acoustic model to obtain the WERs.

Results for the CHiME task are shown in Tbl. 8.4. Switching from an offline to an online estimation of the SCMs imposes a performance penalty of about 1 percentage point or 15 % relative when no other components are modified. With an online mask estimator, the gap increases to more than 20 % for the real recordings. Using $\mathbf{\Phi}_{yy}$ instead of $\mathbf{\Phi}_{ss}$ yields on-par (with the BLSTM mask estimator (ME)) or better WERs. Especially for the online estimation and with a weaker mask estimator, this improves the results. This supports the discussion of the issue above. Interestingly, the frame-online variant performs slightly better on average compared to the block-online variant although the differences are small. A possible reason for this could be the choice of

Table 8.4: Comparison of different block-size for low-latency SCM estimation combined with different mask estimators for the CHiME task. All systems use the WRN acoustic model. A block-size of ∞ corresponds to the offline variant which serves as a baseline.

		SIMU			REAL		
Block-size	SCMs	BLSTM	SI-FF	IN-FF	BLSTM	SI-FF	IN-FF
∞	$\mathbf{\Phi}_{ss}, \mathbf{\Phi}_{nn}$	6.66	6.81	6.81	7.29	7.63	7.87
∞	$\mathbf{\Phi}_{yy}, \mathbf{\Phi}_{nn}$	6.66	6.81	6.75	7.29	7.62	7.71
5	$\mathbf{\Phi}_{ss}, \mathbf{\Phi}_{nn}$	7.57	8.38	8.04	8.68	9.68	9.68
5	$\mathbf{\Phi}_{yy}, \mathbf{\Phi}_{nn}$	7.55	8.25	7.91	8.49	9.56	9.52
1	$\mathbf{\Phi}_{ss}, \mathbf{\Phi}_{nn}$	7.38	7.76	7.87	8.46	9.79	9.58
1	$\mathbf{\Phi}_{yy}, \mathbf{\Phi}_{nn}$	7.34	7.73	7.79	8.11	9.64	9.40

the smoothing factor which is maybe too high for the block-online variant which only performs $\frac{1}{5}$ of the updates compared to the frame-online variant.

For the REVERB task, where the results are shown in Tbl. 8.5, the benefit of using $\mathbf{\Phi}_{yy}$ is even bigger and this time also apparent for the BLSTM. Here, the WER increases by 22 % instead of 32 % when switching to an online SCMs estimation on the official test set. Scaling the signal reduces the impact to 8 % and 12 % respectively, again highlighting the scale sensitivity. Results are especially worse for the SI-FF on the official test set. For the scaled set, the relative decrease in performance is within the same range as with the BLSTM. Again the frame-online variant yields slightly better results compared to the block-online variant which might be caused by the same reason hypothesized for the CHiME task.

Overall, replacing the SCM estimation with a low-latency version degrades the performance by roughly 15 %. This effect is in addition to the impact of replacing the mask estimator with an online variant but both seem to be orthogonal. The proposed modification of the GEV beamformer (using $\mathbf{\Phi}_{yy}$ instead of $\mathbf{\Phi}_{ss}$) shows its desired effect and helps to slightly improve the results in a low-latency setting.

8.4.3 Online system

Finally, we also evaluate the low-latency front-ends with a block-online version of the acoustic model (see Sec. 8.3).

Results for the CHiME task are reported in Tbl 8.6. Compared to the results with the offline AM (see Tbl. 8.4), the performance again degrades by roughly an additional 20 %. But all tendencies identified in the experiments above still hold true for the block-online AM. Recalling the results without any enhancement (see Tbl. 7.4), the drop in performance is also expected and within a reasonable range. Yet, given the same block-online AM, replacing the single-channel front-end with a multi-channel front-end with the low-latency components leads to a relative improvement of nearly

Table 8.5: Comparison of different block-size for low-latency SCM estimation combined with different mask estimators for the REVERB task. All systems use the WRN acoustic model. A block-size of ∞ corresponds to the offline variant which serves as a baseline.

		EVAL			EVAL+10dB		
Block-size	SCMs	BLSTM	SI-FF	IN-FF	BLSTM	SI-FF	IN-FF
∞	$\mathbf{\Phi}_{ss}, \mathbf{\Phi}_{nn}$	8.43	10.95	9.63	8.09	8.54	7.98
∞	$\mathbf{\Phi}_{yy}, \mathbf{\Phi}_{nn}$	8.60	10.75	12.69	8.09	8.49	8.79
5	$\mathbf{\Phi}_{ss}, \mathbf{\Phi}_{nn}$	11.26	18.44	11.48	9.16	10.13	9.02
5	$\mathbf{\Phi}_{yy}, \mathbf{\Phi}_{nn}$	10.64	17.22	11.21	8.76	9.39	8.91
1	$\mathbf{\Phi}_{ss}, \mathbf{\Phi}_{nn}$	11.06	14.31	12.05	9.02	9.30	8.66
1	$\mathbf{\Phi}_{yy}, \mathbf{\Phi}_{nn}$	10.54	14.28	11.40	8.82	9.08	8.79

Table 8.6: Comparison of different block-size for low-latency SCM estimation combined with different low-latency mask estimators for the CHiME task. All systems use the WRN acoustic model without normalization, i.e., a block-online variant.

		SIMU		REAL	
Block-size	SCMs	SI-FF	IN-FF	SI-FF	IN-FF
5	$\mathbf{\Phi}_{ss}, \mathbf{\Phi}_{nn}$	8.87	8.45	11.90	11.72
5	$\mathbf{\Phi}_{yy}, \mathbf{\Phi}_{nn}$	8.77	8.18	11.87	11.36
1	$\mathbf{\Phi}_{ss}, \mathbf{\Phi}_{nn}$	8.45	8.14	11.63	11.18
1	$\mathbf{\Phi}_{yy}, \mathbf{\Phi}_{nn}$	8.45	8.07	11.93	11.22

Table 8.7: Comparison of different block-size for low-latency SCM estimation combined with different low-latency mask estimators for the REVERB task. All systems use the WRN acoustic model without normalization, i.e., a block-online variant.

		EVAL		EVAL+10dB	
Block-size	SCMs	SI-FF	IN-FF	SI-FF	IN-FF
5	$\mathbf{\Phi}_{ss}, \mathbf{\Phi}_{nn}$	32.37	26.09	25.42	20.66
5	$\mathbf{\Phi}_{yy}, \mathbf{\Phi}_{nn}$	32.20	26.41	24.94	20.61
1	$\mathbf{\Phi}_{ss}, \mathbf{\Phi}_{nn}$	30.04	26.23	24.85	20.68
1	$\mathbf{\Phi}_{yy}, \mathbf{\Phi}_{nn}$	30.13	26.16	24.65	20.64

50 % for the real recordings and even more for the simulated ones.

Results for the REVERB task are worse as can be seen from Tbl. 8.7. The WERs are more than twice as high with the block-online model compared to the previous experiment with the offline model, especially for the amplified test set. We hypothesize that this is again mainly due to a scaling issue but leave this issue to future work as our focus here lies on the front-end part.

8.5 Discussion

Both, the mask estimator as well as the beamformer can be modified to work with a very low latency (2 frames or 20 ms respectively). For the mask estimator, there is nearly no performance penalty when switching to the low-latency variant when the distortions are caused by interfering source. Reverberation impacts performance slightly. The biggest challenge seems to result from gain mismatches. Compared to the statistical model, the ability to work seamlessly in a low-latency setting is a distinct advantage of the neural mask estimator. Although work exist on block-online processing with spatial mixture models (e.g., [Hig+16] and [Hig+18]), they require a good prior for the spatial scene and also require to solve the permutation alignment problem. The neural network mask estimator on the other hand profits from its ability to recognize tempo-spectral patterns learned from the training data. In some sense this can also be regarded as a prior (information) but, compared to spatial priors, it generalizes better. As we will see in the next chapter, it is also possible to combine both, the data-driven and the statistical model approach, and consequently exploit both prior information. Especially for voice assistants which are activated by a wake word interaction, this interaction can also be used to extract spectral and spatial priors for the target speaker. Works in this direction include, e.g., [Mar+19].

The results in this chapter should also be interpreted with care. According to the numbers, using low-latency components instead of offline ones roughly leads to a performance loss of 10 % – 15 % for each replaced component. But the datasets used throughout this thesis heavily favor the offline system due to their mostly static nature.

For a scenario with a moving speaker or with a quickly changing noise environment, the impact is likely less severe and the ability of the low-latency SCM estimation to adapt to a changing signal statistic might actually outweigh the loss caused by an inaccurate estimation of the SCMs.

8.6 Summary

In this chapter we presented a low-latency variant of all three major components of the robust ASR system from the previous chapter: the mask estimator, the beamformer and the acoustic model. For the mask estimator, we focused on the problem of a gain mismatch and discussed a scale invariant variant. We explained a frame-online and block-online SCM estimation and finally a block-online acoustic model. Evaluation showed the performance impact of each individual component which turned out to be mostly negative for the datasets used in this work. Leaving aside issues due to a gain mismatch, the online mask estimator achieves comparable performance to its offline variant. But the SCM estimation with a sliding window increases the WER noticeably. However, in the succeeding discussion we hypothesized that the impact will be less severe for more dynamic scenarios.

9 Unsupervised neural mask estimator training

While in the previous chapters we were able to show that a system with a neural network based mask estimator achieves excellent performance and works in low-latency scenarios, there is one major drawback when compared to a spatial mixture model based mask estimator: The neural network requires parallel training data and cannot be trained on real recordings. In this chapter, we lift this requirement and devise a method to train the neural network using observed data only, not even requiring a teacher model. To achieve this, we combine the cACGMM and the neural network. More concretely, we use the spatial mixture model to calculate the likelihood of the observation given class affiliations provided by the neural mask estimator. We then optimize the likelihood with gradient descent, i.e. by differentiating the likelihood function to calculate the gradients w.r.t. the parameters of the neural network and update them accordingly.

A different way to utilize the cACGMM is to use it as a teacher model in a student-teacher training scheme [HVD15]. This also avoids the need for parallel data and will be explored as an alternative direction. The following describes both methods in detail and evaluates them by comparing their results to the ones obtained with the supervised system.

9.1 cACGMM likelihood loss

With Eq. 4.36 and Eq. 4.37, the log-likelihood [Bis+06, Eq. 9.28] of the normalized observation $\tilde{\mathbf{y}}(k,f) = \mathbf{y}(k,f)/\|\mathbf{y}(k,f)\|$ under the cACGMM is

$$\begin{aligned}\mathcal{L} &= \sum_{k,f} \ln \sum_{q} \pi_q(f) \frac{(M-1)!}{2\pi^M \det \mathbf{B}_q(f)} \frac{1}{\left(\tilde{\mathbf{y}}(k,f)^{\mathrm{H}} \mathbf{B}_q(f)^{-1} \tilde{\mathbf{y}}(k,f)\right)^M} \\ &= \sum_{k,f} \ln \sum_{q} \pi_q(f)\, \mathrm{cACG}(\tilde{\mathbf{y}}(k,f); \mathbf{B}_q(f)).\end{aligned} \tag{9.1}$$

It is parameterized by the mixture weights $\pi_q(f)$ and the SCMs $\mathbf{B}_q(f)$ for the Q classes. Like before, we only consider two classes here (i.e. $Q = 2$), namely the target (speaker) and the superposition of all interferences. In Sec. 4.4 the model parameters were estimated with the EM algorithm (compare Eq. 4.39 – Eq. 4.41) but ultimately our interest was to obtain $\gamma_q(k,f)$ – the posterior probability that the frequency bin of the

k-th frame and frequency f is dominated by the source of class q. Once this is known, all other parameters can be inferred with Eq. 4.40 and Eq. 4.41 respectively.

In Ch. 7 we already demonstrated that a neural network can successfully estimate a value very similar to the posterior from the observation. Note that previously we did not exactly estimate the posterior since the two estimated masks did not necessarily sum to one for each tf-bin. But we can easily enforce this property by using a softmax instead of a sigmoid for the outputs. As a consequence, the model has to spread the probability mass among the two classes for each tf-bin while it could previously opt to ignore certain bins. In Sec. 9.3 we will examine the influence of this change on the system performance.

But for now this allows us to obtain an estimate for $\gamma_q(k, f)$ which we can plug into Eq. 4.40 and Eq. 4.41 to compute[1] the parameters for the log-likelihood. To make the dependency on the parameters of the neural network more explicit, we denote the cACGMM parameters in a vector as $\pi_q^{(\boldsymbol{\theta}_{\text{NNET}})}(f)$ and $\mathbf{B}_q^{(\boldsymbol{\theta}_{\text{NNET}})}(f)$ and obtain the following log-likelihood:

$$\mathcal{L} = \sum_{k,f} \ln \sum_q \pi_q^{(\boldsymbol{\theta}_{\text{NNET}})}(f)\, \text{cACG}(\tilde{\mathbf{y}}(k, f); \mathbf{B}_q^{(\boldsymbol{\theta}_{\text{NNET}})}(f)). \tag{9.2}$$

Now, we can optimize the log-likelihood by calculating the gradient w.r.t. the network parameter using backpropagation. Note that this is the derivative of a single real valued scalar w.r.t. real valued parameters but involves complex valued operations. We use the Wirtinger calculus reviewed in Sec. 2.2.4 and AD offered by a computation graph (see Sec. 2.2.3) implemented in the Tensorflow [Mar+15] framework to compute the gradients. Detailed derivations for the necessary complex valued derivatives can be found in our technical report [Böd+17].

When explaining the cACGMM mask estimator in Sec. 4.4, we also discussed the two permutation problems that arise due to the invariance of the model, namely the local and the global permutation problem. Both are also present when using the likelihood to train a neural network based mask estimator and need to be considered in this context. Here, they correspond to a random permutation of a pair of two rows of the weight matrix and bias of the output layer. Since the likelihood and thus the gradient is independent of the order within a pair (they depend on the output value, not the index) the parameters of the output layer will be updated with the same gradient, irrespective of the ordering. The gradient for a single unit's activation of the last layer is the weighted sum of the gradients for the output layer. Reordering the summands does not change the total gradient and, as a result, the permutation invariance does not influence the network training.

Nonetheless, the downstream task requires a solution for the permutation problem since the class correspondence needs to be known. But crucially, this is much easier and reliable than for the spatial mixture model alone because the permutation only needs to be resolved once. The weight matrix and bias vector of the output layer can then be

[1] As mentioned in Sec. 4.4, the SCM is defined implicitly and can be solved by iterating [IAN16]. When using the likelihood to train a neural network, this can quickly become computational too demanding. We therefore opt to initialize the matrix with an identity matrix followed by a single update according to Eq. 4.41. The evaluation will show that this is sufficient to train the network properly.

reordered accordingly. Also, several utterances can be taken into account for solving the problem and the final solution can then be obtained by, e.g., a majority vote. In practice, we solve the permutation problem in the beginning of the training[2] by using the same method as for the cACGMM mask estimator in the first 100 iterations. This is already sufficient to fix the class correspondence for the model outputs.

Due to the i.i.d. assumption, the spatial mixture model has no concept of speech, i.e., it does not use any spectral cues and only performs a spatial clustering. This is also true for the likelihood and can be the cause of degenerated solutions. It is, for example, possible to select only a few tf-bins which can be modeled almost perfectly with a specific SCM and allocate the rest to the other class. Although the second component will have a low likelihood, the overall likelihood can be reasonably high due to the contribution of the first component. Another scenario would be that a few tf-bins which are hard to explain under the model are compounded into a single component while the rest is left to the other one which explains them reasonably well. Because of the resulting high mixture weight of the second component, the likelihood can again improve over the course of the training but ultimately reach a sub-optimal value. We therefore also experiment with different optimization goals: One where we replace the mixture weights with the constant $\frac{1}{Q} = \frac{1}{2}$ for all classes and one where we replace the log-likelihood with the auxiliary function [Bis+06, Eq. 9.30]. The motivation for the latter is the direct influence of the neural network output on the loss function which might improve the feedback, i.e. the gradient. These variants are formulated as follows:

$$\mathcal{L}^{(\text{equal})} = \sum_{k,f} \ln \sum_{q} \frac{1}{Q} \operatorname{cACG}(\tilde{\mathbf{y}}(k,f); \mathbf{B}_q^{(\theta_{\text{NNET}})}(f)), \tag{9.3}$$

$$\mathcal{L}^{(\text{auxiliary})} = \sum_{q,k,f} \gamma_q(k,f) \ln \pi_q^{(\theta_{\text{NNET}})}(f) \operatorname{cACG}(\tilde{\mathbf{y}}(k,f); \mathbf{B}_q^{(\theta_{\text{NNET}})}(f)). \tag{9.4}$$

The mixture weights in Eq. 9.2 and the posteriors in Eq. 9.4 can also be calculated in different ways. We can perform the E-step of the EM algorithm (Eq. 4.39) and use the obtained posteriors in Eq. 9.4 to calculate the mixture weights in Eq. 9.2. Or we can skip the E-step and use the outputs of the neural network $\gamma_q^{(\theta_{\text{NNET}})}(k,f)$ directly. The argument for the former would be that the posteriors better match the model under which we want to calculate the likelihood while an argument for the latter is again the more direct gradient. In the limit, i.e. if the network estimates the posteriors perfectly (that is according to the model) both variants yield the same result.

At inference time, two ways to determine a class affiliation mask are now possible. The first one dismisses the whole spatial model and just uses the outputs of the neural network. Once trained, this system is essentially the same as the ones discussed in the chapters before with the only difference being that the estimate is now a proper probability, i.e. the class affiliation sum to one for all tf-bins. The second option is to use the neural network outputs as an initial guess for the posteriors and then do some EM iterations to increase the likelihood of the current observation under the cACGMM. One important benefit of this method is that it includes spatial (the cACGMM) as well as tempo-spectral cues (the neural network). Ambiguities present in the spectral signal

[2] We already solve it during the training for an easier monitoring of the process, not because it is required for the model to work.

might be resolved spatially. Please note that this does not reintroduce the permutation problem from which the cACGMM suffered before due to the strong prior from the posteriors. One drawback might be that the resulting estimate of the posterior is ultimately limited by the assumptions introduced with the cACGMM.

9.2 cACGMM teacher

Another possibility to exploit the cACGMM for training the neural mask estimator and lifting the requirement for parallel data is a student-teacher training scheme. In this scheme, the cACGMM is used to estimate masks with are in turn used as oracle targets for the neural network. At first it might seem that such a training would limit the neural network to the performance of the (teacher) cACGMM. However, two effects allow the student to actually supersede its teacher. First, the feature both systems can access are very different. While the cACGMM uses spatial information, the neural network uses spectral cues. It thus cannot simply replicate its teacher but has to find different representations to infer the targets. The second one is of statistical nature and concerns the permutation alignment. When this fails for single utterances, the network will not immediately adapt to this. So if it works for the larger amount of observations, the errors will average out and the network will learn a correct alignment.

9.3 Evaluation

We now evaluate the different presented optimization criteria and compare the likelihood-based approach to the student-teacher approach. Finally we analyze the influence of the output non-linearity.

All experiments are conducted using the GEV beamformer and the WRN acoustic model which is the same as in Sec 7.4.1, or, in other words, a model trained on noisy observations only without any adaptation. For the mask estimator, we use the (modified)[3] BLSTM mask estimator model as presented in Sec. 7.1. Note that we do not require any oracle target masks anymore and can therefore extend the training set for the CHiME task to also include the real recordings. As mentioned, it is also possible to use the network output to initialize the cACGMM during inference. Here, we found that in practice a single EM iteration is sufficient as the likelihood does not increase with additional iterations. Consequently we fix the number of iterations to one.

9.3.1 Optimization criterion

First, we asses the performance after training with different variants of the log-likelihood loss. We only conduct this experiment for the CHiME task and will use the best configuration in the following. The optimization criteria considered are the log-likelihood (Eq. 9.2), the likelihood the under assumption of equal mixture weights (Eq. 9.3) and the auxiliary function (Eq. 9.4). Where applicable, we either calculate the posterior according to the E-step (Eq. 4.39) or use the posterior predicted by the network directly.

[3] The non-linearity of the output units is – depending on the use-case – either the sigmoid or the softmax function.

Table 9.1: WERs for the CHiME task and different loss functions for the likelihood training. The additional EM step determines if a single EM step is used at inference time.

		SIMU		REAL	
		Additional EM step			
Loss function	γ	no	yes	no	yes
Eq. 9.2	$\gamma_q^{(\theta_{\text{NNET}})}$	8.09	7.39	8.83	8.25
	Eq. 4.39	7.86	7.29	8.73	8.00
Eq. 9.3	$1/Q$	**7.43**	**7.09**	**8.36**	**7.88**
Eq. 9.4	$\gamma_q^{(\theta_{\text{NNET}})}$	7.71	7.26	8.83	8.18
	Eq. 4.39	7.94	7.26	8.82	8.15

Table 9.2: WERs for the CHiME task with an estimator trained on oracle targets (typeset in gray) and using the likelihood loss with equal class weights (Eq. 9.3). The additional EM step determines if a single EM step is used at inference time or if the output of the network is used directly.

		SIMU		REAL	
		Additional EM step			
Training target	Eq.	no	yes	no	yes
BCE (oracle targets)	(7.3)	6.93	–	7.46	–
Likelihood	(9.3)	7.43	7.09	8.36	7.88

Results are reported in Tbl. 9.1. For all configurations, the additional EM-step on the test utterance which includes the spatial information to estimate the class affiliation posteriors improves the WERs compared to using the output of the network directly. For the loss functions Eq. 9.2 and Eq. 9.4, performing an E-step to obtain an updated estimate of the posterior during training (Eq. 4.39) slightly improves the results. Overall, the best performance is achieved by the loss function which assumes an equal a-priori probability for each mixture component (Eq. 9.3) but the differences compared to the other loss functions are small. Nonetheless, we will use this loss function in the following experiments.

9.3.2 Comparison with oracle target training

Next, we compare the results of the likelihood training with the ones obtained by a system trained with oracle targets from Ch. 7. WERs for the CHiME and REVERB

Table 9.3: WERs for the REVERB task with an estimator trained on oracle targets (typeset in gray) and using the likelihood loss with equal class weights (Eq. 9.3). The additional EM step determines if a single EM step is used at inference time or if the output of the network is used directly.

		EVAL		EVAL+10dB	
		Additional EM step			
Training target	Eq.	no	yes	no	yes
BCE (oracle targets)	(7.3)	8.71	–	7.86	–
Likelihood	(9.3)	8.63	7.90	7.97	7.36

task are shown in Tbl. 9.2 and Tbl. 9.3 respectively. For the CHiME task, the system trained with the likelihood criterion and without an additional EM step yields a WER which is about 10 % higher than the one from the baseline system trained on oracle targets. Adding the EM step during inference and thus using the cACGMM posterior, reduces the gap and the performance of the system trained with the likelihood criterion is only slightly worse. As the likelihood criterion includes the cACGMM during training, this is not surprising.

Whereas for the REVERB challenge, training with the likelihood criterion leads to an improved system performance with the additional EM step and an on-par performance without it. Note that for the CHiME challenge, the system with oracle targets is trained on simulated data only while the unsupervised training also includes the real recordings. In theory, this should be an advantage of the unsupervisedly trained system. However, the results show that this does not translate in a better performance on real recordings during inference.

9.3.3 Softmax vs. Sigmoid

Apart from the training procedure, one major difference between the baseline system and the one trained with the likelihood loss is the non-linearity at the output of the network. While the baseline system views the affiliations independently for each class and is not forced to assign a tf-bin to either of the classes, the proposed unsupervised system estimates a proper posterior over two classes. To assess how that influences the results, we train a system with oracle targets but with a softmax at the output rather than a sigmoid. Consequently, we use the cross-entropy criterion instead of the independent binary cross-entropy. The oracle targets are calculated by determining the dominant class (in terms of instantaneous power) for each tf-bin.

We report the WERs with and without an additional EM step during inference in Tbl 9.4 for the CHiME task and Tbl. 9.5 for the REVERB task respectively. From the results we can see that the CHiME task profits slightly when not all tf-bins need to be considered. For the real recordings this improves the WER by about 5 % relative. Again we can see a different picture for the REVERB task where estimating a proper posterior improves the WER by 10 % relative for the mismatch case with no gain factor.

Table 9.4: WERs for the CHiME task with an estimator with a sigmoid non-linearity at the output and one with a softmax. Both estimators are trained with oracle masks. The additional EM step determines if a single EM step is used at inference time or if the output of the network is used directly.

	SIMU		REAL	
	Additional EM step			
Non-linearity	no	yes	no	yes
Sigmoid	**6.93**	**6.84**	**7.46**	**7.71**
Softmax	7.05	**6.84**	7.97	**7.71**

Table 9.5: WERs for the REVERB task with an estimator with a sigmoid non-linearity at the output and one with a softmax. Both estimators are trained with oracle masks. The additional EM step determines, if a single EM step is used at inference time or if the output of the network is used directly.

	EVAL		EVAL+10dB	
	Additional EM step			
Non-linearity	no	yes	no	yes
Sigmoid	8.71	7.96	7.86	7.02
Softmax	**7.74**	**7.52**	**7.85**	**6.87**

For the set with a gain, there is no difference between the systems. The additional EM step reduces the differences between the two systems and both achieve the same WERs for the CHiME task and comparable WERs for the REVERB task. This is to be expected as the masks from both systems are very similar and are likely to converge to the same posterior when used as an initial guess for the cACGMM.

These results suggest that there is no clear advantage for either case. While in the noisy scenario (CHiME) using the softmax degrades the performance slightly, it likewise improves in reverberant environments (REVERB).

9.3.4 Comparison with teacher-student training

Sec. 9.2 describes an alternative way to avoid the need for parallel data with the help of a cACGMM, namely using it as a teacher system. We train two systems, one with a sigmoid output and one with a softmax output and compare them against a system trained with the likelihood criterion. Note that the student-teacher scheme also allows to train on the real recordings which we therefore include during the training.

A comparison can be found in Tbl. 9.6 for the CHiME task and Tbl. 9.7 for the REVERB task. These results show that the student-teacher system achieves better

Table 9.6: WERs for the CHiME task with an estimator with a sigmoid non-linearity at the output and one with a softmax. Both estimators are trained using a cACGMM teacher and compared against a system trained with the likelihood criterion. The additional EM step determines, if a single EM step is used at inference time or if the output of the network is used directly.

		SIMU		REAL	
		Additional EM step			
Training scheme	Non-linearity	no	yes	no	yes
Student-Teacher	Sigmoid	7.08	7.13	7.79	7.86
Student-Teacher	Softmax	7.08	7.07	7.95	7.86
Likelihood	Softmax	7.43	7.09	8.36	7.88

Table 9.7: WERs for the REVERB task with an estimator with a sigmoid non-linearity at the output and one with a softmax. Both estimators are trained using a cACGMM teacher and compared against a system trained with the likelihood criterion. The additional EM step determines, if a single EM step is used at inference time or if the output of the network is used directly.

		EVAL		EVAL+10dB	
		Additional EM step			
Training scheme	Non-linearity	no	yes	no	yes
Student-Teacher	Sigmoid	9.38	7.71	8.41	8.39
Student-Teacher	Softmax	9.45	7.75	8.37	8.37
Likelihood	Softmax	8.63	7.90	7.97	7.36

WERs when using the outputs of the neural network directly but once the EM step is added the WERs are on-par. Again, this is not surprising as the likelihood criterion does not optimize for the output of the neural network explicitly but for the whole system including the cACGMM. The student-teacher training on the other hand, although using a cACGMM as the teacher, directly optimizes the output of the network during training.

In summary, both unsupervised methods show competitive performance compared to the supervised training. Comparing both methods directly, there is a trade-off between training and inference. The student-teacher system requires a reasonably well performing teacher. For new tasks this can translate into tedious manual tuning of the cACGMM system and the EM algorithm, especially when it comes to initialization and solving the permutation problems. On the other hand, the system trained with the likelihood criterion requires an additional EM step during inference to achieve

comparable performance. But this can be an obstacle for example in low-latency settings considered in the previous chapter. One way to circumvent this problem would be to train a system with the likelihood criterion first and then use this system as a teacher to train a network which does not require the EM step. However, we will leave this to future work.

Application of the two presented methods are not limited to this particular use-case but can also be extended to blind source separation of multiple sources in general. A student-teacher system for this scenario has already been presented in e.g. [DHH19a]. It should be noted though, that the teacher in this work uses a guided source separation [Boe+18a], requiring a speaker diarization as annotation and is thus not completely unsupervised. Using the likelihood approach might allow to drop this requirement and is subject to ongoing research.

9.4 Summary

The likelihood of the observation under a spatial mixture model initialized by a neural network can be used as a training criterion for the latter. The presented approach thus allows for an unsupervised training of a neural mask estimator with almost no drop in performance on the evaluated datasets. It also enables training on real recordings where no targets can be computed, extending the applicability of the system. During inference, spatial cues can be integrated which might help to further improve the performance on other datasets similar to [Nak+17] and [DH19]. We also explored using a spatial mixture model as a teacher which yields on-par results compared to training the model with oracle targets. Both methods share the same benefit of unsupervised training but with different trade-offs. The likelihood based approach requires an additional EM step during inference while the student-teacher approach requires a (manually) tuned teacher model.

10 Joint optimization

So far we have treated the mask estimator and the acoustic model separately. This corresponds to the classical view of a front-end responsible for signal processing and feature extraction and a back-end to infer a word sequence from the features. Both systems are optimized and tuned for their respective task and are unaware of each other. A loose coupling can be achieved in a straightforward way by using enhanced features during the training of the acoustic model, but there usually is no feedback from the back-end to the front-end. Notable exceptions include e.g. [SRS04], where the criterion for the beamformer is to enhance the signal in a way that the extracted features maximize the likelihood of generating the correct transcription (under a GMM-HMM acoustic model).

A different way of combining the front-end with the back-end is to directly use the waveform as the input modality for the acoustic model as mentioned in Sec 3.3. For these systems, no distinction can be made anymore between the two parts as they are merged into a single neural network with the already discussed advantages and drawbacks.

In general, a tighter coupling of both components seems to be a desirable goal. Especially optimizing the front-end with a criterion much more closely related to the final goal of a low word error rate rather than signal level criteria promises better results. Another benefit would be the possibility to train on transcribed data only. Similar to the unsupervised training presented in the previous chapter, this eliminates the need for parallel data and allows to train the system on real recordings.

The proposed system with its neural network based mask estimator has the best prerequisites for a tight integration with the acoustic model. All parameters of the combined system are parameters of a neural network and trained with gradient descent. The gradients can be efficiently computed with backpropagation for both models. Connecting them only requires to calculate the partial derivative of the feature vectors w.r.t. to the estimated mask. Crucially, we need to backpropagate the gradients through the complex valued beamforming operation. Once the derivative is known, the beamformer essentially just becomes another layer (without parameters) of the neural network and, similar to the systems working on the raw waveform, the distinction between the front-end and back-end vanishes. However, compared to direct waveform approach discussed above, the signal processing part of this model is still interpretable using classical signal processing. It also promises to be more flexible regarding the array configuration and acoustic scenario in general. Yet, it is trained jointly with an ASR related goal.

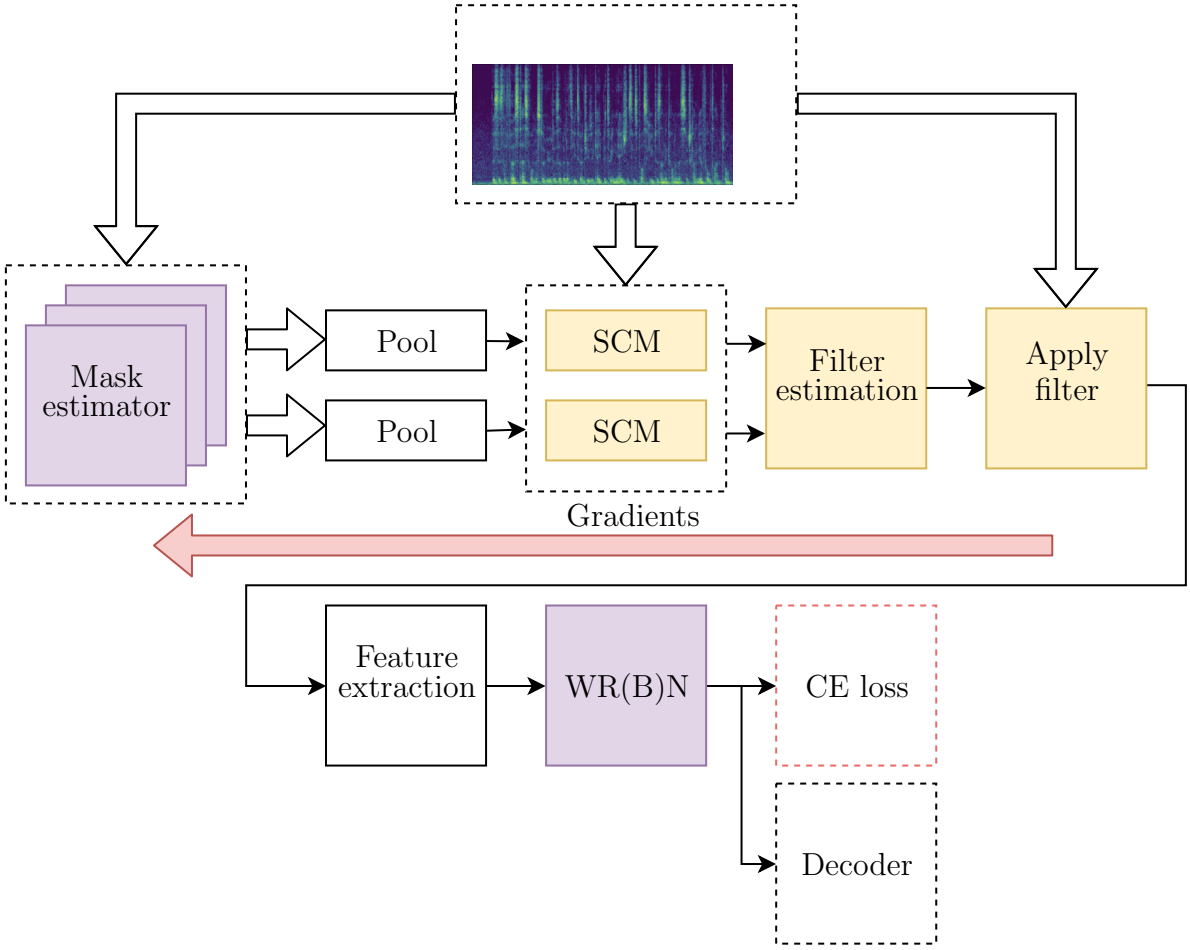

Figure 10.1: Schematic representation of the joint system with a neural network supported beamformer and the acoustic model. Depth illustrates multiple channels and copies of the network respectively. Multi-channel signals are indicated by bold arrows. Violet blocks have trainable variables while yellow indicates that the block uses complex values. The gradient flow is sketched by the red arrow.

In the following, we detail the necessary steps to connect the two components followed by an extensive evaluation of the joint system. We especially focus on different training schemes and also analyze the importance of the data the acoustic model is ultimately trained with.

10.1 Backpropagating gradients

Since the gradients are propagated from the loss to the parameters, the order of the nodes flips (reverse computational graph, see Sec. 2.2.3) and in the following we will adhere to this order when describing relations. As sketched in the overview of the system in Fig. 10.1, the gradient has to be backpropagated through four major blocks.

1. Feature extraction

2. Acoustic beamformer
3. SCM estimation
4. Pooling

Note that the loss is real valued and so is the gradient of the LMSC features. The same holds for the application of the logarithm and the Mel-filterbank which can be formulated as matrix-matrix product. Afterwards, the gradient becomes complex valued, representing the one of the spectogram. To cope with the complex valued gradients we again turn to the Wirtinger calculus as described in Sec. 2.2.4.

The spectogram is the outcome of the beamforming operation, i.e. the frequency-wise weighted sum of the spectograms from all channels. We are interested in the gradient for the weights $\mathbf{w}_{\mathrm{BF}}$ of the individual sums which are straightforward to compute. Now, depending on the beamformer criterion, the math gets more complicated when calculating the partial derivative of the filter coefficients w.r.t. the SCMs. Formally, given the gradient $\frac{\hat{\partial} J}{\hat{\partial} \mathbf{w}_{\mathrm{BF}^*}}$ of the (conjugated) beamforming vector w.r.t. to the loss J, we seek to calculate

$$\frac{\hat{\partial} J}{\hat{\partial} \mathbf{\Phi}^*} = \frac{\hat{\partial} J}{\hat{\partial} \mathbf{w}_{\mathrm{BF}^*}} \frac{\hat{\partial} \mathbf{w}_{\mathrm{BF}}}{\hat{\partial} \mathbf{\Phi}^*}. \tag{10.1}$$

Here, we focus on the GEV beamformer which, with the generalized eigenvalue problem, is probably the most challenging one among the statistical beamformers presented in Sec 4.2.2. In this case, we have

$$\mathbf{w}_{\mathrm{GEV}} = \mathbf{L}^{-\mathsf{H}} \mathbf{v}_{\mathbf{e}_{\max}}, \tag{10.2}$$

$$\mathbf{\Phi} = \left(\mathbf{L}^{-1} \mathbf{\Phi}_{ss} \mathbf{L}^{-\mathrm{H}} \right), \tag{10.3}$$

and

$$\mathbf{\Phi}_{nn} = \mathbf{L}\mathbf{L}^{\mathrm{H}}. \tag{10.4}$$

To backpropagate the gradient to the SCMs, we need the generalized Jacobians $\frac{\hat{\partial} \mathbf{w}_{\mathrm{GEV}}}{\hat{\partial} \mathbf{\Phi}_{ss}}$ and $\frac{\hat{\partial} \mathbf{w}_{\mathrm{GEV}}}{\hat{\partial} \mathbf{\Phi}_{nn}}$. These involve the gradients of the Cholesky factorization (see A.1), the matrix inverse and the eigenvalue problem (see A.2). All gradients have been implemented in the Tensorflow [Mar+15] framework and additional derivations can also be found in our technical report [Böd+17]. The crucial gradient is the one of the eigenvalue problem $\mathbf{E}, \mathbf{V} = \mathcal{P}(\mathbf{\Phi})$ which is given by

$$\frac{\partial J}{\partial \mathbf{\Phi}^*} = \mathbf{V}^{-\mathsf{H}} \left(\frac{\partial J}{\partial \mathbf{E}} + \mathbf{F}^* \circ \mathbf{V}^{\mathsf{H}} \frac{\partial J}{\partial \mathbf{V}} \right) \mathbf{V}^{\mathsf{H}}. \tag{10.5}$$

with $F_{ij} = (e_j - e_i)^{-1}$ and $F_{ii} = 0$, i.e. the element in the i-th row and j-th column of the matrix $\mathbf{F}$ is the reciprocal of the difference between the j-th and i-th eigenvalue[1].

[1] This property can lead to exploding gradients during training if the first and second eigenvalue become almost equal. In practice, we continuously monitor the difference between the two and skip utterances for which the values are too close. However, we did not encounter such a singularity during our experiments.

Table 10.1: Initial results for a jointly trained system compared to the same system where the mask estimator and the acoustic model is trained independently.

ME training	AM training	SIMU	REAL
independent	independent	6.78	7.34
jointly	jointly	6.95	9.12

Note that for $\frac{\partial J}{\partial \mathbf{V}}$ all rows except for one are equal to zero as only the eigenvector with the highest eigenvalue receives a gradient. Also, the eigenvectors are scaled to have unit norm during the forward path by the algorithm solving the eigenvalue problem. We make this scaling explicit by calculating

$$\mathbf{w}_{\text{GEV}} = \mathbf{L}^{-\mathsf{H}} \frac{\mathbf{v}_{\mathbf{e}_{\max}}}{||\mathbf{v}_{\mathbf{e}_{\max}}||}, \tag{10.6}$$

in the implementation.

Now that we have obtained the gradient for the SCMs, calculation of the gradient for the outputs of the mask estimator is again straightforward as the calculation of the SCM in Eq. 4.9 shows. However, note that we limit the masks to be real valued and consequently drop the imaginary part of the gradient (see [Böd+17]). For the pooling we use the average instead of the median during training to avoid sparse and, for some points, not well defined gradients. From the outputs of the network we can then calculate the gradients for the parameters of the mask estimator using the usual calculus for backpropagation for neural networks.

10.2 Evaluation

10.2.1 Initial experiment

As an initial experiment, we evaluate the performance of a system where the mask estimator is jointly trained with the acoustic model from scratch using the CE loss only on the CHiME task. We use the BLSTM mask estimator, the GEV beamformer and the WRN acoustic model for this experiment. Except for the joint optimization, the system is trained as described in Sec. 7.3 and Sec. 7.3.2. Note that, contrary to the baseline system, the mask estimator is also trained on the 1600 real recordings of the training set.

From the results and the comparison to the independently trained system in Tbl 10.1 we can see that for the simulated evaluation data, the jointly trained system performs almost on-par. For the real recordings however, there is a noticeable gap of nearly 2 percent points. This is especially surprising since the mask estimator of the jointly trained system was already exposed to real recordings during training.

10.2.2 Research questions

Several questions arise from this result

a Can the performance loss be attributed to only one of the models or do both perform worse?

b Previously the acoustic model has been trained using all available channels, increasing the size of the training data by a factor of six (for CHiME) although the effective factor might be less as the samples are still highly correlated. When training the model jointly, all channels are used and condensed into one by the beamformer. The acoustic model is thus trained with a single fold of the training corpus only. How does this affect the final system performance?

c When training jointly and assuming that the mask estimator is trained reasonably well, the features for the acoustic model have a higher SNR as the training progresses. Does this hurt the robustness of the acoustic model?

d Although the influence of the mask estimator on the beamformed signal is restricted by the linear filtering of the beamformer, it could potentially lead to filters tailored towards the specific acoustic model it is optimized with. Does the mask estimator overfit to the acoustic model it is optimized with?

e The acoustic model itself is already robust against interferences to a certain extent when trained with multi-style data. As a consequence, particular interferences only have a minor impact on the estimation of the state posteriors as the model has learned to ignore those variations in the features. But this also means that the magnitude of the gradient will be small and the acoustic model will not strongly encourage the front-end to remove these interferences. Can a better mask estimator be trained by using an acoustic model which has been trained on clean data only?

f If training both models from scratch leads to a performance drop, we can pre-train some components before jointly optimizing them. Ultimately, what is the best training strategy for the joint system?

In the following we seek to answer those questions by conducting various experiments on the CHiME and REVERB task.

10.2.3 Impact of the training data

First, we investigate the impact of the training data on the performance of the acoustic model. This is directly related to the questions (b) and (c).

We train the WRN acoustic model on five different data sources:

1. On all available channels with noisy data
2. On 1., additionally augmented with beamformed data
3. On beamformed data only
4. On noisy data from the fifth microphone only
5. On clean data only

Table 10.2: Impact of the training data used to train the acoustic model on the performance on beamformed and single-channel (channel 5) data for the CHiME task.

	SIMU		REAL	
Training data	Beamformed	Channel #5	Beamformed	Channel #5
All channels	6.78	16.03	7.34	15.23
All channels + Beamformed	6.40	18.22	7.66	17.28
Beamformed	6.89	55.89	9.64	30.97
Channel #5	6.94	17.48	7.62	17.60
Clean	11.69	42.73	16.31	48.25

This varies the total size of the training corpus as well as the average SNR during the training. We then evaluate all five models on the beamformed test sets (using a BLSTM mask estimator and the GEV beamformer) and on the (unprocessed) signal as recorded by by the lower center microphone (microphone #5).

The results are summarized in Tbl. 10.2. When the model is trained on beamformed data only, its performance significantly degrades compared to the baseline model trained on all channels. But this cannot be attributed to the smaller size of the training data. Using only data from the fifth microphone results in the same size of the training corpus, however, this model achieves noticeable lower WERs for all but the beamformed simulated test set. The differences are especially visible when the model is evaluated on the single-channel data. At the same time, the model trained on all channels is only marginally better than the one trained on a single channel only. This suggests that in this specific case, the size of the training corpus is not the decisive factor for the performance of the acoustic model. Further evidence is found by looking at the performance of the model trained on all channels plus the beamformed data. This model performs nearly on-par with the model trained on single-channel data and worse than the baseline model. In other words, adding additional training data even hurts the performance in this case. Overall, the results are a clear indication that the acoustic model trained on beamformed data is not as robust against variations as the one trained on all channels. Another observation is that the beamforming seemingly works better for the simulated data, i.e. the features have less variations due to interferences after the beamforming. This is supported by the performance of the models trained on data with higher SNRs (beamformed and clean).

10.2.4 Model analysis

To gain insights into specific properties of the model and find answers for the questions formulated above, we evaluate various variants of training and combinations of the two model components. One of these variants has already been explored exhaustively in the previous chapters: training and evaluating both models independently. On the

other side of the spectrum, both models can be trained jointly from scratch using the acoustic model cross-entropy loss only. In between exist a multitude of strategies using pre-trained models and possibly finetuning and/or mixing them.

The different variants we evaluate are explained in the following.

I The ME and the AM are trained independently with their own loss.

II Both models are optimized jointly from scratch using the cross-entropy loss between the predicted senone probabilities and the alignment, effectively turning the whole system into a single model.

III Uses both models from I as initialization. The AM is then fine-tuned using (pre-computed) beamformed data (related questions: a, b, c, f)

IV Uses both models from I as initialization. The ME is then fine-tuned using the CE loss while the parameters of the AM are fixed during this fine-tuning stage (related questions: a, d, f)

V Uses both models from I as initialization. Both models are then fine-tuned jointly using the CE loss (related questions: a, c, d, f).

VI Uses the AM from II and combines it with the ME from I (related questions: a, b, c, f).

VII Uses the AM from I and combines it with the ME from II (related questions: a, d, f).

VIII Uses the ME from I to calculate beamformed features. The AM is then trained (from scratch) using pre-computed data. (related questions: a, b, c, f).

IX Uses the AM from I. The ME is then trained (from scratch) using the CE loss while the parameters of the pre-trained AM are held fixed (related questions: a, e, f).

X The AM is trained independently on clean data. The ME is then trained (from scratch) using the CE loss while the parameters of the pre-trained AM are held fixed. For evaluation, the AM is swapped for one trained independently with multi-style data. (related questions: a, e, f).

XI Similar to (II) except that for each example, the AM receives an unprocessed single-channel observation with a 50 % chance or the output of the beamformer otherwise (related questions: f)

For all variants, we follow the same training setup as described in Sec. 7.3.2 and Sec. 7.3.1. When fine-tuning a model, we start with $\frac{1}{10}$-th of the initial learning rate for the respective model. As described in Sec. 7.3, we continuously monitor the WER to stop the training before the model starts to overfit.

The WERs on the CHiME task for the different variants are shown in Tbl. 10.3. Fine-tuning the mask estimator or the whole model (variant (IV) and (V) respectively) yields no improvements over the pre-initialized model. Also, fine-tuning the acoustic

Table 10.3: WERs for different training variants and combinations on the CHiME task. Please refer to Listing 10.2.4 for an overview of the variants and the text for further explanations. For the variants (IV) and (V) no best model was exported during training, i.e. the WER during cross-validation did no improve over the pre-initialized model. We therefore report the numbers of (I).

Variant	SIMU	REAL
I	6.78	7.34
II	6.95	9.12
III	6.67	7.41
IV	(6.78)	(7.34)
V	(6.78)	(7.34)
VI	6.59	8.94
VII	7.43	7.64
VIII	6.89	9.64
IX	7.03	8.63
X	6.73	7.18
XI	6.90	8.50

model with data from the beamformer results in small improvements on the simulated data but also slightly deteriorates the performance for the real recordings. This shows that the beamformer with the mask estimator trained with the BCE loss is already a very good match for the acoustic model (trained with noisy data). A finding, that is surprising since the mask estimator was optimized with a very loosely related criterion at best and the beamformer maximizes the signal level SNR which is only weakly correlated to the WER.

The results for the variants (VI), (VII) and (VIII) provide further evidence for the hypothesis that the worse result of the jointly trained model can be attributed solely to a weaker acoustic model. When the jointly trained acoustic model is combined with a separately trained mask estimator, there is still a significant performance degradation for the real recordings (compare with (II) and (I)). However, if we use the mask estimator from the jointly trained model and combine it with a separately trained acoustic model, the performance on the real recordings are much closer to those of the separately trained system (I) and the performance on simulated data degrades. The variant (VIII) shows a result we already saw in the last section. Namely, the WERs increase noticeably when the acoustic model is trained (from scratch) on beamformed data. In summary, these results support the claim that the acoustic model is the problem when the system is optimized jointly. Although a small portion of the performance gap can be attributed to the mask estimator, the largest part is caused by the acoustic model.

The reason for the performance loss of the mask estimator can be inferred from the variants (IX) and (X). When the mask estimator is trained with the CE loss and an acoustic model trained on noisy data, the resulting WERs are higher compared to the separately optimized model with the BCE loss ((IX) vs. (I)). However, when we use an acoustic model trained on clean data and then combine the trained mask estimator with the acoustic model from (I), we achieve on-par WERs compared to the baseline system (I). This shows that the gradients from the acoustic model trained with noisy data are indeed not the best choice to train the mask estimator as they are presumably small for interferences that the acoustic model already learnt to ignore.

Sampling the features for the acoustic model as in (XI) partly mitigates the training data issue arising in (II) but still suffers from the finding about the gradient discussed above.

10.2.5 Performance on REVERB

Next, we evaluate the different variants on the REVERB task. The results are shown in Tbl 10.4. Variants (III) – (V) were left out this time as the fine-tuning again did not show improvements over the pre-initialized model. For this task, we get a different picture. The WERs for the jointly trained system (II) are actually slightly better than the baseline with separately trained components (I). When the jointly trained acoustic model is combined with a pre-trained mask estimator there is no clear performance hit as observed for the CHiME task ((VI) vs. (I)). The same holds true when the acoustic model is trained on dereverberated data (VIII). But the jointly trained mask estimator is able to improve the performance when combined with a separate acoustic model ((VII) vs. (I) + (II)). Similar results are achieved when training the mask estimator with an acoustic model trained on reverberant data (IX) and on clean data (X). Finally,

Table 10.4: WERs for different training variants and combinations on the REVERB task. Please refer to Listing 10.2.4 for an overview of the variants and the text for further explanations.

Variant	`EVAL`	`EVAL+10dB`
I	8.43	8.09
II	8.04	7.91
VI	8.49	8.23
VII	7.62	7.55
VIII	8.57	8.12
IX	7.71	7.63
X	7.65	7.57
XI	7.44	7.47

when the system is trained jointly and the input to the acoustic model is sampled from the unprocessed and processed training data, the lowest WERs are reached. But overall, with a relative gap of 10 % between the worst and best WER, the differences between the systems are rather small.

These results are in contrast to the ones on the CHiME task. First, the performance of the acoustic model is not affected in the same way as seen before and a model trained on dereverberated data performs almost on-par with the one trained on unprocessed data. And second, the results for a mask estimator trained with the CE objective are better compared to the ones achieved by a mask estimator trained with the BCE loss.

A possible explanation for the first difference is that the training data of the REVERB corpus is only simulated and for most utterances the distortion due to reverberation is small. In fact, as mentioned in Sec. 5.2, we refrained from evaluating the system on the simulated test sets for exactly the same reason. The distortions do not noticeably degrade the performance of the acoustic model. Training on an enhanced set might thus make little difference. We hypothesize that this is a specific observation for this task and cannot be attributed to the fact that the distortions are caused by reverberation instead of noise sources. For a task with real recordings (like in the test set), we expect to see the same results as for the CHiME task.

The second difference might actually reveal a benefit of the joint optimization. While for the CHiME task we can clearly distinguish between the target speech and the interferences, this is not possible for reverberation. As described in Sec. 4.5, the signal can be decomposed into three parts and we aim to attenuate the late reverberation which is the third part. However, it is unclear at which point the line should be drawn, i.e. which part of the signal is still useful and when does it become harmful for the acoustic model. But in order to calculate the targets for the training of the mask estimator with the BCE loss, we need to specify this point. And although we made an

informed choice, it was likely not the optimal one, especially for the specific acoustic model. On the other hand, when the mask estimator is trained with the gradients from the acoustic model, this choice becomes part of the optimization problem and the system intrinsically learns where to make the distinction to achieve an optimal state classification accuracy.

Fig 10.2 shows the estimated interferences masks for an example utterance from the training set to visualize the difference between the jointly and separately trained model. Additionally, the spectogram of one channel of the observed signal and the target for the separately trained model are depicted. The separately trained model learned to reproduce the target very well, only some details are missing and overall it looks like a slightly blurred version of the target. A very different mask is produced by the jointly trained system. At first it looks very much like the output of a VAD system on a frame level. However, some of the "stripes" are longer than others and seemingly do capture some of the reverberation visible in the observed signal. The mask does not look very intuitive but from the results we know that this model works best. Also, it is not a phenomenon of this specific example, masks for other utterances look very similar and show the same pattern.

10.2.6 Answers

Equipped with the insights from the evaluations above, we can now answer the questions raised in Sec 10.2.2.

a Yes. Most of the performance loss (for the CHiME task) can be attributed to the acoustic model which is less robust as it is exposed to less diverse, i.e. noisy, data during training.

b For the CHiME task, training the model on a single fold of the training data (i.e. a single channel or beamformed data) does only slightly degrade the performance compared to using all channels and is not the reason for the worse performance of the jointly optimized system. The same holds for the REVERB task although there is no performance loss at all.

c Yes. When the acoustic model is trained on enhanced features, the final performance suffers (see also (a)). This holds true for the CHiME data. For the REVERB task, we cannot observe this behavior but hypothesize that this is caused by the training data already showing little distortions.

d No. A mask estimator trained jointly with an acoustic model can be used with any acoustic model.

e Yes. Using an acoustic model trained on clean data to train the mask estimator yields a better performing model. Again, this is based on the CHiME task while for the REVERB task is not conclusive in this point due to the training data (see (c)).

f The acoustic model should not be optimized jointly with the mask estimator. Instead, the acoustic model can be trained on noisy data and then used to train the mask estimator. If clean data is available, performance can be improved by using an

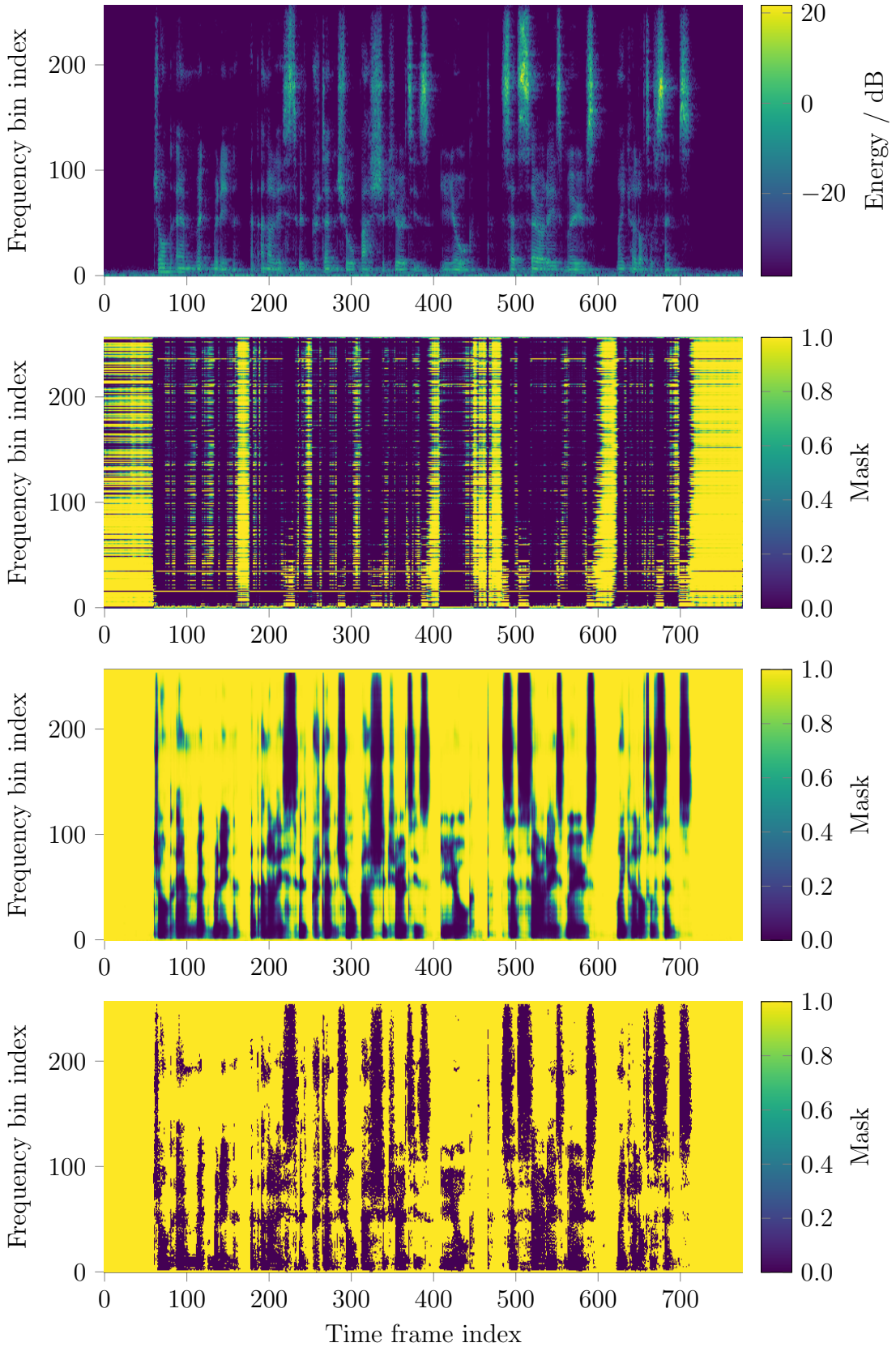

Figure 10.2: Comparison of masks estimated by a BLSTM mask estimator trained with the BCE and one trained with the CE loss. The first picture shows the spectogram for one channel of the observed signal. Below that is the interference mask estimated by the jointly trained mask estimator. The two pictures on the bottom show the mask estimated by the separately trained model and the target it was trained with.

acoustic model trained on clean data to train the mask estimator. Surprisingly, training the models separately is a very strong baseline worth considering if parallel data is available and good training targets for the mask estimator can be extracted.

10.3 Summary

Merging the beamformer front-end and the acoustic back-end into one model by propagating the gradients through the beamforming filter allows to train the system jointly using data only annotated with their transcription. Compared to an approach directly operating on the waveform, this keeps the flexibility of the statistical signal processing with regard to the microphone configuration combined with its interpretability. But care has to be taken when jointly training the system. As the front-end learns to enhance the features, the acoustic model becomes less robust, resulting in a performance degradation during inference. An appropriate training schedule is to train the acoustic model first on multi-style data and then use it to train the front-end or, alternatively, sample from enhanced and unprocessed features and jointly train both models. But this might only be true for datasets where a performant acoustic model can be trained on the available data. For hard tasks, like for example the CHiME 5 challenge, where the recordings are so distorted that it is very hard to train an acoustic model in the first place, a different strategy likely leads to better results. At this point, however, we leave an appropriate evaluation for future work. Another interesting direction to explore is how the architecture of the acoustic model influences the front-end training. Parallels to the influence of the discriminator in a generative adversarial network framework [Goo+14] might exist, although the front-end (generator) here is much more restricted due to the linear beamforming filter operation.

11 Summary

The role of ASR fundamentally changed in the last few years. While the first systems were only designed for close-talk dictation tasks or pre-defined commands with a limited vocabulary, the systems nowadays have become a commodity and serve as a complete user interface, even in far-field scenarios. However, these scenarios are especially challenging and the systems are still far away from human level perception in those cases. One key question is how to use multi-channel data in a way that allows the system to exploit spatial information to reduce the number of transcription mistakes.

NNs have been established as the model of choice for many tasks, including speech recognition. Particularly using a NN as the acoustic model drove down error rates in more challenging scenarios to a level which was unreachable before and allows many new practical applications. The model parameters are optimized using stochastic gradient descent on a vast amount of training data. In Ch. 2 we shortly reviewed the basic building blocks of a NN and focused on the gradient computation necessary for the update rule. We showed how by viewing the network as a computational graph and exploiting the chain rule we can derive the gradient computation in a principled way, also if intermediate values are complex valued. This enabled us later to integrate beamforming into the neural network.

Statistical beamforming is at the center of this thesis and was thoroughly discussed in Ch 4. Starting from a general signal model for the multi-channel mixture we obtained a spectral model with a couple of assumptions. With this model, we introduced the MWF, the MVDR and GEV beamformer and showed that, under a Rank-1 assumption of the target SCM, all of these criteria result in the same look direction but scale the frequencies differently. All of these beamformers rely on the estimation of the SCM which can be estimated with the help of a mask from the observation. We showed of such masks can be estimated with a statistical model, namely the cACGMM. Finally, we focused on reverberation and how it can be mitigated with the help of either a beamformer or WPE.

Equipped with the necessary background knowledge, we then presented the main contribution of this thesis: a neural network support beamformer. In a comprehensive evaluation in Ch. 7, we showed how this can achieve improvements of about 50 % compared to a single-channel system for the CHiME and REVERB corpora which were presented in Ch. 5. We also demonstrated advantages of this approach compared to the statistical cACGMM approach. The biggest advantages are the ability to exploit spectro-temporal correlation and the avoidance of any permutation problem. We also showed that the gains are stable over a wide SNR range and two different types of neural

networks, namely a LSTM and CNN based one. As formulated in one of the goals, the system is indeed agnostic to the microphone configuration and can scale to an arbitrary number of microphones without modification. After having presented the approach, we also discussed it in the context of other works which happened simultaneously or built upon the neural network supported beamformer.

The rest of the thesis was dedicated to extensions of the presented system. Motivated by a smart home scenario with a seamless user interaction, a low-latency variant of the mask estimator and a frame-online and block-online SCM estimation was introduced in Ch. 8. We also presented an alternative formulation of the GEV beamformer more suitable for online-processing. Evaluation showed that the performance loss due to a low latency mask estimator is minimal and the major factor is the SCM estimation. We discussed how this can be mitigated by, e.g., using statistics from wake word utterances and pointed to further and future work.

The limitation of requiring parallel data for training was lifted in two different ways in the next chapters. First, in Ch. 9 we used the likelihood of an observation under a spatial mixture model as a loss criterion. This connected the previously discussed cACGMM approach and the neural network based beamformer. We showed, that the resulting system can be trained without any supervision on observed multi-channel data and achieve comparable results to the system trained in a supervised fashion. As an additional benefit, spatial cues can be integrated during inference. We also considered a student-teacher approach which shows the same performance and concluded by discussing the different trade-offs and giving an outlook on further use-cases of the presented method.

Finally, in Ch. 10 we considered a joint training approach which combines the mask estimator and the acoustic model into a single model with the beamformer as a specialized signal processing layer. After initial results revealed an unexpected performance gap to the separately trained baseline, we devised different training and evaluation strategies to analyze the model behavior. We found that as the front-end learns to enhance the features, the acoustic model becomes less robust, resulting in a performance degradation during inference. Two ways to mitigate this effect without requiring additional (parallel) data or information were presented. Either one first trains the acoustic model on multi-style data and then use it to train the front-end, or, more in the spirit of joint training, the acoustic model is randomly presented either an enhanced or an unprocessed feature. Both methods were able to match the performance of the baseline but do not require parallel data.

Overall, this thesis shows that (classical) signal processing can provide several benefits in combination with powerful neural networks.

Appendix

A.1 Gradient for the Cholesky factorization

In the following we will derive the gradient for the Cholesky factorization of a positive definite hermitian matrix as published in [Böd+17]. The factorization is defined as

$$\mathbf{C} = \mathbf{L} = \text{cholesky}\left(\mathbf{A}\right) \tag{A.1}$$

such that

$$\mathbf{A} = \mathbf{L}\mathbf{L}^{\mathsf{H}}, \tag{A.2}$$

where $\mathbf{L}$ is a lower left triangular matrix ($\mathbf{L} \in \blacktriangleleft$). In the following we will exploit some properties of $\mathbf{L}$ and triangular matrices:

1. $\mathbf{L} \in \blacktriangleleft \Rightarrow \mathbf{L}^{-1} \in \blacktriangleleft$
2. $\mathbf{A} \in \blacktriangleleft, \mathbf{B} \in \blacktriangleleft \Rightarrow \mathbf{A}\mathbf{B} \in \blacktriangleleft$
3. $\mathbf{L} \in \blacktriangleleft \Rightarrow \left[\mathbf{L}^{-1}\right]_{ii} = {L_{ii}}^{-1}$
4. $\text{diag}\left(\mathbf{L}\right) \in \mathbb{R}^{+}$
5. $\mathbf{L} \in \blacktriangleleft, L_{i,i} \in \mathbb{R}, \mathbf{C} = \mathbf{C}^{\mathsf{H}}, \mathbf{C} = \mathbf{L} + \mathbf{L}^{\mathsf{H}} \Rightarrow \left(\blacktriangle - \frac{1}{2}\mathbf{I}\right) \circ \mathbf{C} = \mathbf{L}$ where $\blacktriangle$ is a lower triangular matrix filled with ones and $\circ$ is the element-wise multiplication (Hadamard product).

With these properties we start with the definition in Eq. A.2 and the forward sensitivity (see, e.g. [Gil08])

$$\frac{\partial \mathbf{A}}{\partial z^*} = \frac{\partial \mathbf{L}}{\partial z^*}\mathbf{L}^{\mathsf{H}} + \mathbf{L}\frac{\partial \mathbf{L}^{\mathsf{H}}}{\partial z^*} \tag{A.3}$$

which we reformulate as

$$\mathbf{L}^{-1}\frac{\partial \mathbf{A}}{\partial z^*} = \mathbf{L}^{-1}\frac{\partial \mathbf{L}}{\partial z^*}\mathbf{L}^{\mathsf{H}} + \frac{\partial \mathbf{L}^{\mathsf{H}}}{\partial z^*} \tag{A.4}$$

$$\Leftrightarrow \mathbf{L}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{L}^{-\mathsf{H}} = \mathbf{L}^{-1}\frac{\partial \mathbf{L}}{\partial z^*} + \frac{\partial \mathbf{L}^{\mathsf{H}}}{\partial z^*}\mathbf{L}^{-\mathsf{H}} \tag{A.5}$$

$$\Leftrightarrow \mathbf{L}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{L}^{-\mathsf{H}} = \mathbf{L}^{-1}\frac{\partial \mathbf{L}}{\partial z^*} + \left(\mathbf{L}^{-1}\frac{\partial \mathbf{L}}{\partial z^*}\right)^{\mathsf{H}}. \tag{A.6}$$

From 1. and 2. we can see that $\mathbf{L}^{-1}\frac{\partial \mathbf{L}}{\partial z^*} \in \blacktriangle$. Further, from 3. and 4., it follows that $[\mathbf{L}^{-1}]_{ii} \in \mathbb{R}^+$. The diagonal elements of $\mathbf{A}$ are all real valued ($\mathbf{A} = \mathbf{A}^{\mathsf{H}} \Rightarrow [\mathbf{A}]_{ii} = ([\mathbf{A}]_{ii})^{\mathsf{H}}$) and thus $\mathrm{Im}\left(\left[\frac{\partial \mathbf{A}}{\partial z^*}\right]_{ii}\right) = 0$ and $\mathrm{Im}\left(\left[\frac{\partial \mathbf{L}}{\partial z^*}\right]_{ii}\right) = 0$. Consequently, $\mathrm{Im}\left(\left[\mathbf{L}^{-1}\frac{\partial \mathbf{L}}{\partial z^*}\right]_{ii}\right) = 0$. With this, we can apply 5. to Eq. A.6 to obtain

$$\left(\blacktriangle - \frac{1}{2}\mathbf{I}\right) \circ \mathbf{L}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{L}^{-\mathsf{H}} = \mathbf{L}^{-1}\frac{\partial \mathbf{L}}{\partial z^*} \tag{A.7}$$

and

$$\frac{\partial \mathbf{L}}{\partial z^*} = \mathbf{L}\left(\left(\blacktriangle - \frac{1}{2}\mathbf{I}\right) \circ \mathbf{L}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{L}^{-\mathsf{H}}\right). \tag{A.8}$$

Following the instructions given in Sec. 2.2.3 and Sec. 2.2.4 we insert Eq. A.8 in Eq. 2.46 resulting in

$$\frac{\partial J}{\partial z^*} = \mathrm{tr}\left\{\left(\frac{\partial J}{\partial \mathbf{L}^*}\right)^{\mathsf{H}}\mathbf{L}\left(\left(\blacktriangle - \frac{1}{2}\mathbf{I}\right) \circ \mathbf{L}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{L}^{-\mathsf{H}}\right) + \ldots\right\}. \tag{A.9}$$

Using $\mathrm{tr}\{\mathbf{X}\,(\mathbf{Y} \circ \mathbf{Z})\} = \mathrm{tr}\{\left(\mathbf{X} \circ \mathbf{Y}^{\mathsf{T}}\right)\mathbf{Z}\}$ the above equation can be rewritten as

$$\begin{aligned}
\frac{\partial J}{\partial z^*} &= \mathrm{tr}\left\{\left(\left(\left(\frac{\partial J}{\partial \mathbf{L}^*}\right)^{\mathsf{H}}\mathbf{L}\right) \circ \left(\blacktriangle - \frac{1}{2}\mathbf{I}\right)^{\mathsf{T}}\right)\mathbf{L}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{L}^{-\mathsf{H}}\right\} + \ldots \\
\frac{\partial J}{\partial z^*} &= \mathrm{tr}\left\{\mathbf{L}^{-\mathsf{H}}\left(\left(\left(\frac{\partial J}{\partial \mathbf{L}^*}\right)^{\mathsf{H}}\mathbf{L}\right) \circ \left(\blacktriangle - \frac{1}{2}\mathbf{I}\right)^{\mathsf{T}}\right)\mathbf{L}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\right\} + \ldots \\
\frac{\partial J}{\partial z^*} &= \mathrm{tr}\left\{\left(\mathbf{L}^{-\mathsf{H}}\left(\left(\mathbf{L}^{\mathsf{H}}\frac{\partial J}{\partial \mathbf{L}^*}\right) \circ \left(\blacktriangle - \frac{1}{2}\mathbf{I}\right)\right)\mathbf{L}^{-1}\right)^{\mathsf{H}}\frac{\partial \mathbf{A}}{\partial z^*}\right\} + \ldots
\end{aligned}$$

Comparing this result to Eq. 2.47 finally yields the desired gradient

$$\frac{\partial J}{\partial \mathbf{A}^*} = \mathbf{L}^{-\mathsf{H}}\left(\left(\mathbf{L}^{\mathsf{H}}\frac{\partial J}{\partial \mathbf{L}}\right) \circ \left(\blacktriangle - \frac{1}{2}\mathbf{I}\right)\right)\mathbf{L}^{-1}. \tag{A.10}$$

A.2 Gradient for the complex valued eigenvalue decomposition

In the following we will derive the gradient for the eigenvalue decomposition of a complex valued hermitian matrix. Again, this has been published in [Böd+17] and inspired by the works by Giles on the real valued problem [Gil08]. The implicit definition of the eigenvalue decomposition of a quadratic matrix A is given by

$$\mathbf{AV} = \mathbf{VE}. \tag{A.11}$$

As described in Sec. 4.2.2, for the specific application of beamforming, $\mathbf{A}$ corresponds to the (projected) SCM $\mathbf{\Phi}$ and we seek to find the eigenvector $\mathbf{v}$ which corresponds to the highest eigenvalue $e_{\max}$. So in Eq. A.11 above $\mathbf{V}$ is a quadratic matrix with all

eigenvectors and $\mathbf{E}$ is a diagonal matrix with the eigenvalues. The forward sensitivity is then expressed as

$$\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{V} + \mathbf{A}\frac{\partial \mathbf{V}}{\partial z^*} = \frac{\partial \mathbf{V}}{\partial z^*}\mathbf{E} + \mathbf{V}\frac{\partial \mathbf{E}}{\partial z^*}. \tag{A.12}$$

Multiplying Eq. A.12 with $\mathbf{V}^{-1}$ from the left-hand side yields

$$\mathbf{V}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{V} + \mathbf{V}^{-1}\mathbf{A}\frac{\partial \mathbf{V}}{\partial z^*} = \mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial z^*}\mathbf{E} + \frac{\partial \mathbf{E}}{\partial z^*} \tag{A.13}$$

and rearranging the terms

$$\mathbf{V}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{V} - \frac{\partial \mathbf{E}}{\partial z^*} = \mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial z^*}\mathbf{E} - \mathbf{V}^{-1}\mathbf{A}\frac{\partial \mathbf{V}}{\partial z^*}. \tag{A.14}$$

Using $\mathbf{A} = \mathbf{V}\mathbf{E}\mathbf{V}^{-1}$ results in

$$\mathbf{V}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{V} - \frac{\partial \mathbf{E}}{\partial z^*} = \mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial z^*}\mathbf{E} - \mathbf{E}\mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial z^*}. \tag{A.15}$$

For a diagonal matrix $\mathbf{\Lambda}$, the following equation holds [Gil08; Böd+17]:

$$\mathbf{C}\mathbf{\Lambda} - \mathbf{\Lambda}\mathbf{C} = \mathbf{D} \circ \mathbf{C} \tag{A.16}$$

with $[\mathbf{D}]_{ij} = \lambda_j - \lambda_i$ and "$\circ$" denoting the Hadamard product. We apply this rule to Eq. A.15 to obtain

$$\mathbf{V}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{V} - \frac{\partial \mathbf{E}}{\partial z^*} = \mathbf{D} \circ \left(\mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial z^*}\right). \tag{A.17}$$

From this equation, the forward sensitivity for the eigenvalues can be calculated by realizing that $\frac{\partial \mathbf{E}}{\partial z^*}$ is a diagonal matrix, applying the Hadamard product with the identity matrix $\mathbf{I}$ to both sides and exploiting that $\operatorname{diag}(\mathbf{D}) = \mathbf{0}$ and thus $\mathbf{I} \circ \mathbf{D} = \mathbf{0}$. This results in

$$\frac{\partial \mathbf{E}}{\partial z^*} = \mathbf{I} \circ \left(\mathbf{V}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{V}\right). \tag{A.18}$$

To obtain the forward sensitivity for the eigenvectors, we introduce with $\mathbf{F}$ the Hadamard inverse of $\mathbf{D}$ whose elements are defined as $[\mathbf{F}]_{ij} = \frac{1}{\lambda_j - \lambda_i}$ and $[\mathbf{F}]_{ii} = 0$, i.e., $\mathbf{F} \circ \mathbf{D} = \mathbf{1} - \mathbf{I}$. Applying the Hadamard product with $\mathbf{F}$ to Eq. A.17 yields

$$\mathbf{F} \circ \left(\mathbf{V}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{V}\right) - \mathbf{F} \circ \frac{\partial \mathbf{E}}{\partial z^*} = (\mathbf{1} - \mathbf{I}) \circ \left(\mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial z^*}\right). \tag{A.19}$$

Noting that $\mathbf{F} \circ \frac{\partial \mathbf{E}}{\partial z^*}$ is zero since the diagonal elements of $\mathbf{F}$ are zero and that any change of input does not influence the magnitude of the eigenvectors resulting in $\operatorname{diag}\left(\mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial z^*}\right) = \mathbf{0}$ (see [Gil08]), the equation can be simplified to

$$\mathbf{F} \circ \left(\mathbf{V}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{V}\right) = \mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial z^*}. \tag{A.20}$$

Rearranging the terms finally results in the desired formulation

$$\frac{\partial \mathbf{V}}{\partial z^*} = \mathbf{V}\left(\mathbf{F} \circ \left(\mathbf{V}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{V}\right)\right). \tag{A.21}$$

Again, we follow the instructions given in Sec. 2.2.3 but since we have two outputs now, we reformulate Eq. 2.46 using the chain rule resulting in

$$\frac{\partial J}{\partial z^*} = \mathrm{tr}\left\{\left(\frac{\partial J}{\partial \mathbf{E}^*}\right)^{\mathsf{H}}\frac{\partial \mathbf{E}}{\partial z^*} + \left(\frac{\partial J}{\partial \mathbf{V}}\right)^{\mathsf{H}}\frac{\partial \mathbf{V}}{\partial z^*}\right\} + \ldots. \tag{A.22}$$

Inserting the forward mode sensitivities derived above yields

$$\frac{\partial J}{\partial z^*} = \mathrm{tr}\left\{\left(\frac{\partial J}{\partial \mathbf{E}^*}\right)^{\mathsf{H}}\mathbf{I} \circ \left(\mathbf{V}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{V}\right) + \left(\frac{\partial J}{\partial \mathbf{V}}\right)^{\mathsf{H}}\mathbf{V}\left(\mathbf{F} \circ \left(\mathbf{V}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\mathbf{V}\right)\right)\right\} + \ldots. \tag{A.23}$$

which, with the help of the following rules

$$\mathrm{tr}\{\mathbf{XY}\} = \mathrm{tr}\{\mathbf{YX}\} \tag{A.24}$$

$$\mathrm{tr}\{\mathbf{X}\,(\mathbf{Y} \circ \mathbf{Z})\} = \mathrm{tr}\big\{\big(\mathbf{X} \circ \mathbf{Y}^{\mathsf{T}}\big)\,\mathbf{Z}\big\}, \tag{A.25}$$

can be reformulated to obtain

$$\frac{\partial J}{\partial z^*} = \mathrm{tr}\left\{\mathbf{V}\left(\left(\frac{\partial J}{\partial \mathbf{V}}\right)^{\mathsf{H}}(\mathbf{V} \circ \mathbf{F}^{\mathsf{T}}) + \left(\frac{\partial J}{\partial \mathbf{E}^*}\right)^{\mathsf{H}}\right)\mathbf{V}^{-1}\frac{\partial \mathbf{A}}{\partial z^*}\right\} + \ldots. \tag{A.26}$$

A comparison of the coefficients with Eq. 2.47 finally yields the desired gradient

$$\frac{\partial J}{\partial \mathbf{A}^*} = \mathbf{V}^{-\mathsf{H}}\left(\frac{\partial J}{\partial \mathbf{E}} + \mathbf{F}^* \circ \mathbf{V}^{\mathsf{H}}\frac{\partial J}{\partial \mathbf{V}}\right)\mathbf{V}^{\mathsf{H}}. \tag{A.27}$$

A.3 Reproducibility

The main model described in Ch. 7 for the CHiME data is released on GitHub[1]. This is also part of some Kaldi recipes[2]. The code for WPE dereverberation has been made available also on Github[3]. Otherwise, all code required to reproduce the experiments presented in this thesis is available to the communications engineering group at Paderborn University and can be found in the git repository.

[1] https://github.com/fgnt/nn-gev
[2] https://github.com/kaldi-asr/kaldi/tree/master/egs/chime4/
[3] https://github.com/fgnt/nara_wpe

Symbols and notation

We use a lower-case bold character for one-dimensional vectors. A matrix or other higher dimensional tensors use a bold upper-case character. Sequences are denoted by a subscript and their start and end index, e.g., $\square_{1:N}$. For distributions over a set of discrete values we use $\Pr(\cdot)$ and over continuous values $p(\cdot)$.

Most of the symbols in this thesis have a local context and are introduced on a need-by basis whenever they appear first. But some are used throughout the thesis and listed in Lst. 1. For time-domain signals we use t to index the sample while for spectral signals we use k as the frame and f as the frequency index. Sources are indexed by q and microphones by m. Whenever there is no ambiguity we omit the indices for brevity.

$y_m(t)$	Signal as captured by the m-th sensor
$a_{m,q}(t, \tau)$	AIR from source q to sensor m at time t
$x_q(t)$	Signal emitted by the q-th source
$d_m(t)$	Diffuse signal as captured by the m-th sensor
$s(t)$	Target (speech) signal
$n_q(t)$	Signal of the q-th interfering source with $q \in \{1, \dots, Q-1\}$
$y_m(k, f)$	Signal as captured by the m-th sensor
$a_{mq}(f)$	AIR from source q to sensor m for frequency f at frame k
$x_q(k, f)$	Signal emitted by the q-th source
$d_m(k, f)$	Diffuse signal as captured by the m-th sensor
$s(k, f)$	Target (speech) signal
$\mathbf{\Phi}_{qq}(k, f)$	Spatial covariance matrix of multi-channel signal $\mathbf{s}_q(k, f)$
$\mathbf{e}$	Vector with eigenvalues of an eigenvalue decomposition
$\mathbf{V}$	Matrix of all eigenvectors of an eigenvalue decomposition
$\mathbf{w}$	Beamforming vector
J	Loss function or value

List of Figures

List of Tables

Acronyms

AD automatic differentiation.

AIR acoustic impulse response.

AM acoustic model.

ASR automatic speech recognition.

BCE binary cross-entropy.

BLSTM bidirectional long short-term memory.

BSS blind source separation.

cACGMM complex angular central Gaussian mixture model.

CE cross-entropy.

CGMM complex Gaussian mixture model.

CHiME Computational Hearing in Multisource Environments.

CLDNN Convolutional long short-term memory fully connected deep neural network.

CNN convolutional neural network.

CTC connectionist temporal classification.

DCT discrete cosine transformation.

DFT discrete Fourier transform.

DNN deep neural network.

DSP digital signal processing.

ELU exponential linear unit.

EM expectation maximization.

ETSI European Telecommunication Standards Institute.

FF feedforward.

FFT fast Fourier transform.

GEV generalized eigenvalue.

GMM gaussian mixture model.

GPU graphics processing unit.

HMM hidden markov model.

IBM ideal binary mask.

IN-FF instance norm feedforward.

IPD interchannel phase difference.

IRM ideal ratio mask.

LAS listen attend and spell.

LF-MMI lattice-free maximum mutual information.

LM language model.

LMSC log Mel spectral coefficient.

LSTM long short-term memory.

MAP Maximum-a-posteriori.

ME mask estimator.

MFCC Mel frequency cepstral coefficient.

MIMO multiple inputs multiple outputs.

ML maximum-likelihood.

MMI maximum mutual information.

MPDR minimum power distortionless response.

MPE minimum phone error.

MSC Mel spectral coefficient.

MSE mean squared error.

MVDR minimum variance distortionless response.

MWF multi-channel Wiener filter.

NN neural network.

NPP noise presence probability.

PSD power spectral density.

ReLU rectified linear unit.

REVERB Reverberant Voice Enhancement and Recognition Benchmark.

RIR room impulse response.

RNN recurrent neural network.

RNN-LM recurrent neural network language model.

RNN-T RNN transducer.

RNR reverberant-to-noise ratio.

RT reverberation time.

RTF relative transfer function.

SCM spatial covariance matrix.

Seq2Seq sequence-to-sequence.

SGD stochastic gradient descent.

SI-FF scale invariant feedforward.

sMBR state-level minimum Bayes risk.

SNR signal-to-noise ratio.

SPP speech presence probability.

STFT short-time Fourier transform.

TDOA time difference of arrival.

tf-bin time frequency bin.

VAD voice activity detection.

WER word error rate.

WFST weighted finite state transducer.

WPE weighted prediction error.

WRBN wide residual BLSTM network.

WRN wide residual network.

WSJ Wall Street Journal.

Bibliography

[AlR+18] R. Al-Rfou, D. Choe, N. Constant, M. Guo, and L. Jones. *Character-Level Language Modeling with Deeper Self-Attention*. 2018. eprint: ArXiv: 1808.04444.

[AN11] S. Araki and T. Nakatani. "Hybrid approach for multichannel source separation combining time-frequency mask with multi-channel Wiener filter". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2011.

[Aok+01] M. Aoki, M. Okamoto, S. Aoki, H. Matsui, T. Sakurai, and Y. Kaneda. "Sound source segregation based on estimating incident angle of each frequency component of input signals acquired by multiple microphones". In: *Acoustical Science and Technology* 22.2 (2001), pp. 149–157.

[Ara+09] S. Araki, Tomohiro Nakatani, H. Sawada, and S. Makino. "Blind sparse source separation for unknown number of sources using Gaussian mixture model fitting with Dirichlet prior". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2009.

[Ara+16] S. Araki, M. Okada, T. Higuchi, A. Ogawa, and T. Nakatani. "Spatial correlation model based observation vector clustering and MVDR beamforming for meeting recognition". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016.

[AWH07] X. Anguera, C. Wooters, and J. Hernando. "Acoustic beamforming for speaker diarization of meetings". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.7 (2007), pp. 2011–2021.

[Bah+86] L. Bahl, P. Brown, P. de Souza, and R. Mercer. "Maximum mutual information estimation of hidden Markov model parameters for speech recognition". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1986.

[Bar+13] J. Barker, E. Vincent, N. Ma, H. Christensen, and P. Green. "The PASCAL CHiME speech separation and recognition challenge". In: *Computer Speech & Language* 27.3 (2013), pp. 621–633.

[Bar+15] J. Barker, R. Marxer, E. Vincent, and S. Watanabe. "The third 'CHiME'speech separation and recognition challenge: Dataset, task and baselines". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 2015.

[Bar+17] J. Barker, R. Marxer, E. Vincent, and S. Watanabe. "The third 'CHiME'speech separation and recognition challenge: Analysis and outcomes". In: *Computer Speech & Language* 46 (2017), pp. 605–626.

[Bay+18] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind. "Automatic Differentiation in Machine Learning: a Survey". In: *Journal of Machine Learning Research (JMLR)* 18.153 (2018), pp. 1–43.

[BCB14] D. Bahdanau, K. Cho, and Y. Bengio. *Neural machine translation by jointly learning to align and translate*. 2014. eprint: ArXiv:1409.0473.

[Bec+19] E. Beck, W. Zhou, R. Schlüter, and H. Ney. *LSTM Language Models for LVCSR in First-Pass Decoding and Lattice-Rescoring*. 2019. eprint: arXiv:1907.01030.

[Bis+06] C. M. Bishop et al. *Pattern recognition and machine learning*. Vol. 1. springer New York, 2006.

[BKC17] V. Badrinarayanan, A. Kendall, and R. Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[Blu+15] T. Bluche, H. Ney, J. Louradour, and C. Kermorvant. "Framewise and CTC training of Neural Networks for handwriting recognition". In: *13th International Conference on Document Analysis and Recognition (ICDAR)*. 2015.

[Boe+18b] C. Boeddeker, H. Erdogan, T. Yoshioka, and R. Haeb-Umbach. "Exploring practical aspects of neural mask-based beamforming for far-field speech recognition". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018.

[Bra83] D. H. Brandwood. "A complex gradient operator and its application in adaptive array theory". In: *IEE Proceedings F (Communications, Radar and Signal Processing)*. Vol. 130. 1983, pp. 11–16.

[BSH08] J. Benesty, M. M. Sondhi, and Y. A. Huang. *Springer handbook of speech processing*. Springer handbooks. Springer, 2008.

[Cha+16] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016.

[Che+13] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, and P. Koehn. *One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling*. 2013. eprint: ArXiv:1312.3005.

[Che+17] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE transactions on pattern analysis and machine intelligence* 40.4 (2017), pp. 834–848.

[Che+18] S.-J. Chen, A. S. Subramanian, H. Xu, and S. Watanabe. "Building State-of-the-art Distant Speech Recognition Using the CHiME-4 Challenge with a Setup of Speech Enhancement Baseline". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (2018).

[Chi+17] C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, K. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani. *State-of-the-art Speech Recognition With Sequence-to-Sequence Models*. 2017. eprint: ArXiv:1712.01769.

[Chi+18] C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani. "State-of-the-Art Speech Recognition with Sequence-to-Sequence Models". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018.

[Cho+14a] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio. *On the Properties of Neural Machine Translation: Encoder-Decoder Approaches*. 2014. eprint: ArXiv:1409.1259.

[Cho+14b] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio. *End-to-end continuous speech recognition using attention-based recurrent NN: First results*. 2014. eprint: ArXiv:1412.1602.

[CJ17] J. Chorowski and N. Jaitly. "Towards Better Decoding and Language Model Integration in Sequence to Sequence Models". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (2017).

[CLM17] Z. Chen, Y. Luo, and N. Mesgarani. "Deep attractor network for single-microphone speaker separation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[Com+90] D. V. Compernolle, W. Ma, F. Xie, and M. V. Diest. "Speech recognition in noisy environments with the aid of microphone arrays". In: *Speech Communication* 9.5 (1990), pp. 433–442.

[Dai+19] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. *Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context*. 2019. eprint: ArXiv:1901.02860.

[Del+13] M. Delcroix, Y. Kubo, T. Nakatani, and A. Nakamura. "Is speech enhancement pre-processing still relevant when using deep neural networks for acoustic modeling?" In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2013.

[Del+14a] M. Delcroix, T. Yoshioka, A. Ogawa, Y. Kubo, M. Fujimoto, N. Ito, K. Kinoshita, M. Espi, S. Araki, T. Hori, and T. Nakatani. "Defeating reverberation: Advanced dereverberation and recognition techniques for hands-free speech recognition". In: *2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. 2014.

[Del+14b] M. Delcroix, T. Yoshioka, A. Ogawa, Y. Kubo, M. Fujimoto, N. Ito, K. Kinoshita, M. Espi, T. Hori, T. Nakatani, et al. "Linear prediction-based dereverberation with advanced speech enhancement and recognition technologies for the REVERB challenge". In: *Reverb workshop*. 2014.

[DH19] L. Drude and R. Haeb-Umbach. "Integration of Neural Networks and Probabilistic Spatial Models for Acoustic Blind Source Separation". In: *IEEE Journal of Selected Topics in Signal Processing* 13.4 (Aug. 2019), pp. 815–826.

[DHH19a] L. Drude, D. Hasenklever, and R. Haeb-Umbach. "Unsupervised training of a deep clustering model for multichannel blind source separation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019.

[DRH16] L. Drude, B. Raj, and R. Haeb-Umbach. "On the Appropriateness of Complex-Valued Neural Networks for Speech Enhancement". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2016.

[Erd+16] H. Erdogan, J. R. Hershey, S. Watanabe, M. I. Mandel, and J. Le Roux. "Improved mvdr beamforming using single-channel mask prediction networks." In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2016.

[Fis97] J. G. Fiscus. "A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER)". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 1997.

[Gan+17] S. Gannot, E. Vincent, S. Markovich-Golan, and A. Ozerov. "A Consolidated Perspective on Multimicrophone Speech Enhancement and Source Separation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.4 (2017), pp. 692–730.

[Gar+93] J. Garofolo, D. Graff, D. Paul, and D. Pallett. "CSR-I (WSJ0) Complete LDC93S6A". In: *Philadelphia: Linguistic Data Consortium* (1993).

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[GHD10] M. Garnier, N. Henrich, and D. Dubois. "Influence of sound immersion and communicative interaction on the Lombard effect". In: *Journal of Speech, Language, and Hearing Research* 53.3 (2010), pp. 588–608.

[Gil08] M. Giles. *An extended collection of matrix derivative results for forward and reverse mode automatic differentiation*. 2008.

[Goo+14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[GOO53] I. J. GOOD. “THE POPULATION FREQUENCIES OF SPECIES AND THE ESTIMATION OF POPULATION PARAMETERS”. In: *Biometrika* 40 (Dec. 1953), pp. 237–264.

[Gra12] A. Graves. *Sequence Transduction with Recurrent Neural Networks*. 2012. eprint: ArXiv:1211.3711.

[Gül+15] Ç. Gülçehre, O. Firat, K. Xu, K. Cho, L. Barrault, H. Lin, F. Bougares, H. Schwenk, and Y. Bengio. *On Using Monolingual Corpora in Neural Machine Translation*. 2015. eprint: ArXiv:1503.03535.

[Har15] M. Harper. “The Automatic Speech recognition In Reverberant Environments (ASpIRE) challenge”. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 2015.

[He+16a] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *IEEE Computer Vision and Pattern Recognition (CVPR)*. 2016.

[He+16b] K. He, X. Zhang, S. Ren, and J. Sun. “Identity mappings in deep residual networks”. In: *European conference on computer vision*. 2016.

[He+19] Y. He, T. N. Sainath, R. Prabhavalkar, I. McGraw, R. Alvarez, D. Zhao, D. Rybach, A. Kannan, Y. Wu, R. Pang, Q. Liang, D. Bhatia, Y. Shangguan, B. Li, G. Pundak, K. C. Sim, T. Bagby, S. Chang, K. Rao, and A. Gruenstein. “Streaming End-to-end Speech Recognition for Mobile Devices”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019.

[Her+16] J. R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe. “Deep clustering: Discriminative embeddings for segmentation and separation”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016.

[Hig+16] T. Higuchi, N. Ito, T. Yoshioka, and T. Nakatani. “Robust MVDR beamforming using time-frequency masks for online/offline ASR in noise”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016.

[Hig+18] T. Higuchi, K. Kinoshita, N. Ito, S. Karita, and T. Nakatani. “Frame-by-frame closed-form update for mask-based adaptive MVDR beamforming”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018.

[Hin+12] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. “Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups”. In: *IEEE Signal Processing Magazine* 29.6 (Nov. 2012), pp. 82–97.

[Hoc91] S. Hochreiter. *Untersuchungen zu dynamischen neuronalen Netzen*. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München. Advisor: J. Schmidhuber. 1991.

[Hor+15] T. Hori, Z. Chen, H. Erdogan, J. R. Hershey, J. Le Roux, V. Mitra, and S. Watanabe. "The MERL/SRI system for the 3RD CHiME challenge using beamforming, robust feature extraction, and advanced speech recognition". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 2015.

[HS97] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[HVD15] G. Hinton, O. Vinyals, and J. Dean. *Distilling the Knowledge in a Neural Network*. 2015. eprint: ArXiv:1503.02531.

[HWW15] Y. Hoshen, R. J. Weiss, and K. W. Wilson. "Speech acoustic modeling from raw multichannel waveforms". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015.

[HZD14] Z. Huang, G. Zweig, and B. Dumoulin. "Cache based recurrent neural network language model inference for first pass speech recognition". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014.

[IAN13] N. Ito, S. Araki, and T. Nakatani. "Permutation-free convolutive blind source separation via full-band clustering based on frequency-independent source presence priors". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2013.

[IAN16] N. Ito, S. Araki, and T. Nakatani. "Complex angular central Gaussian mixture model for directional statistics in mask-based microphone array signal processing". In: *European Signal Processing Conference (EUSIPCO)*. 2016.

[Iri+19] K. Irie, R. Prabhavalkar, A. Kannan, A. Bruguier, D. Rybach, and P. Nguyen. *Model Unit Exploration for Sequence-to-Sequence Speech Recognition*. 2019. eprint: ArXiv:1902.01955.

[IS15] S. Ioffe and C. Szegedy. *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. 2015. eprint: arXiv:1502.03167.

[Iso+17] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[Ito+14] N. Ito, S. Araki, T. Yoshioka, and T. Nakatani. "Relaxed disjointness based clustering for joint blind source separation and dereverberation". In: *International Workshop on Acoustic Signal Enhancement (IWAENC)*. 2014.

[Kan+18] N. Kanda, R. Ikeshita, S. Horiguchi, Y. Fujita, K. Nagamatsu, X. Wang, V. Manohar, N. E.-Y. Soplin, M. Maciejewski, S.-J. Chen, et al. "The Hitachi/JHU CHiME-5 system: Advances in speech recognition for everyday home environments using multiple microphone arrays". In: *Proc. CHiME 2018 Workshop on Speech Processing in Everyday Environments*. 2018, pp. 6–10.

[Kar+19] S. Karita, S. Watanabe, T. Iwata, M. Delcroix, A. Ogawa, and T. Nakatani. "Semi-supervised End-to-end Speech Recognition Using Text-to-speech and Autoencoders". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019.

[Kat+17] Kateřina Žmolíková, M. Delcroix, K. Kinoshita, T. Higuchi, A. Ogawa, and T. Nakatani. "Speaker-Aware Neural Network Based Beamformer for Speaker Extraction in Speech Mixtures". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2017.

[KB14] D. P. Kingma and J. Ba. *Adam: A method for stochastic optimization*. 2014. eprint: arXiv:1412.6980.

[Ken97] J. T. Kent. "Data analysis for shapes and images". In: *Journal of statistical planning and inference* 57.2 (1997), pp. 181–193.

[Kha+18] U. Khandelwal, H. He, P. Qi, and D. Jurafsky. "Sharp Nearby, Fuzzy Far Away: How Neural Language Models Use Context". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 2018, pp. 284–294.

[Kin+13] K. Kinoshita, M. Delcroix, T. Yoshioka, T. Nakatani, A. Sehr, W. Kellermann, and R. Maas. "The REVERB challenge: A common evaluation framework for dereverberation and recognition of reverberant speech". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. 2013.

[Kin+16] K. Kinoshita, M. Delcroix, S. Gannot, E. A. P. Habets, R. Haeb-Umbach, W. Kellermann, V. Leutnant, R. Maas, T. Nakatani, B. Raj, A. Sehr, and T. Yoshioka. "A summary of the REVERB challenge: state-of-the-art and remaining challenges in reverberant speech processing research". In: *EURASIP Journal on Advances in Signal Processing* 1 (Jan. 2016), p. 7.

[Kin09] B. Kingsbury. "Lattice-based optimization of sequence classification criteria for neural-network acoustic modeling". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2009.

[Kra+15] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and F. Li. *The Unreasonable Effectiveness of Noisy Data for Fine-Grained Recognition*. 2015. eprint: ArXiv:1511.06789.

[KSH12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012.

[Kut16] H. Kuttruff. *Room acoustics*. CRC Press, 2016.

[Lau+16] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio. "Batch normalized recurrent neural networks". In: 2016.

[Lev66] V. I. Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Vol. 10. 8. 1966, pp. 707–710.

[Li+15] J. Li, L. Deng, R. Haeb-Umbach, and Y. Gong. *Robust Automatic Speech Recognition – a Bridge to Practical Applications*. Elsevier, 2015.

[Li+17] B. Li, T. Sainath, A. Narayanan, J. Caroselli, M. Bacchiani, A. Misra, I. Shafran, H. Sak, G. Pundak, K. Chin, et al. "Acoustic modeling for Google home". In: 2017.

[Lin+05] M. Lincoln, I. McCowan, J. Vepa, and H. K. Maganti. "The multi-channel Wall Street Journal audio visual corpus (MC-WSJ-AV): specification and initial experiments". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 2005.

[Lip89] R. Lippmann. "Review of Neural Networks for Speech Recognition". In: *Neural Computation* 1.1 (1989), pp. 1–38.

[Liu+18] Y. Liu, A. Ganguly, K. Kamath, and T. Kristjansson. "Neural network based time-frequency masking and steering vector estimation for two-channel mvdr beamforming". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018.

[LMP87] R. Lippmann, E. Martin, and D. Paul. "Multi-style training for robust isolated-word speech recognition". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1987.

[Mar+14] D. Marković, K. Kowalczyk, F. Antonacci, C. Hofmann, A. Sarti, and W. Kellermann. "Estimation of Acoustic Reflection Coefficients Through Pseudospectrum Matching". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.1 (2014), pp. 125–137.

[Mar+15] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[Mar+19] J. M. Martın-Donas, J. Heitkaemper, R. Haeb-Umbach, A. M. Gomez, and A. M. Peinado. "Multi-channel block-online source extraction based on utterance adaptation". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2019.

[MEJ07] M. I. Mandel, D. P. Ellis, and T. Jebara. "An EM algorithm for localizing multiple sound sources in reverberant environments". In: *Advances in neural information processing systems*. 2007.

[MGC09] S. Markovich, S. Gannot, and I. Cohen. "Multichannel Eigenspace Beamforming in a Reverberant Noisy Environment With Multiple Interfering Speech Signals". In: *IEEE Transactions on Audio, Speech, and Language Processing* 17.6 (2009), pp. 1071–1086.

[Mik+11] T. Mikolov, S. Kombrink, A. Deoras, L. Burget, and J. H. Cernocky. "RNNLM - Recurrent Neural Network Language Modeling Toolkit". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 2011.

[MK07] G. McLachlan and T. Krishnan. *The EM algorithm and extensions*. John Wiley & Sons, 2007.

[Mon05] S. Monica Osnaga. "On rank one matrices and invariant subspaces". In: *Balkan Journal of Geometry and its Applications (BJGA)* 10 (Jan. 2005).

[MPR02] M. Mohri, F. Pereira, and M. Riley. "Weighted finite-state transducers in speech recognition". In: *Computer Speech & Language* 16.1 (2002), pp. 69–88.

[MPR08] M. Mohri, F. Pereira, and M. Riley. "Speech recognition with weighted finite-state transducers". In: *Springer Handbook of Speech Processing*. Springer, 2008, pp. 559–584.

[Nak+10] T. Nakatani, T. Yoshioka, K. Kinoshita, M. Miyoshi, and B.-H. Juang. "Speech dereverberation based on variance-normalized delayed linear prediction". In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.7 (2010), pp. 1717–1731.

[Nak+17] T. Nakatani, N. Ito, T. Higuchi, S. Araki, and K. Kinoshita. "Integrating DNN-Based and Spatial Clustering-Based Mask Estimation for Robust MVDR Beamforming". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[NEK94] H. Ney, U. Essen, and R. Kneser. "On structuring probabilistic dependences in stochastic language modelling". In: *Computer Speech & Language* 8.1 (1994), pp. 1–38.

[NH10] V. Nair and G. E. Hinton. "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *International Conference on Machine Learning*. 2010.

[NK19] T. Nakatani and K. Kinoshita. "A Unified Convolutional Beamformer for Simultaneous Denoising and Dereverberation". In: *IEEE Signal Processing Letters* 26.6 (June 2019), pp. 903–907. DOI: 10.1109/LSP.2019.2911179.

[NW14] A. Narayanan and D. Wang. "Joint noise adaptive training for robust automatic speech recognition". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pp. 2504–2508.

[Owe93] F. J. Owens. *Signal processing of speech*. Macmillan International Higher Education, 1993, p. 36.

[PA03] K. K. Paliwal and L. Alsteris. “Usefulness of phase spectrum in human speech perception”. In: *Eighth European Conference on Speech Communication and Technology*. 2003.

[Par+19] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le. *SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition*. 2019. eprint: ArXiv:1904.08779.

[PMB12] R. Pascanu, T. Mikolov, and Y. Bengio. *Understanding the exploding gradient problem*. 2012. eprint: ArXiv:1211.5063.

[Pov+11] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. “The Kaldi Speech Recognition Toolkit”. In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 2011.

[Pov+16] D. Povey, V. Peddinti, D. Galvez, P. Ghahremani, V. Manohar, X. Na, Y. Wang, and S. Khudanpur. “Purely sequence-trained neural networks for ASR based on lattice-free MMI.” In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2016.

[Pov05] D. Povey. “Discriminative training for large vocabulary speech recognition”. PhD thesis. University of Cambridge, 2005.

[Pro03] S. Processing. “Transmission and Quality Aspects (STQ); Distributed speech recognition; Extended advanced front-end feature extraction algorithm; Compression algorithms; Back-end speech reconstruction algorithm”. In: *ETSI ES* 202.212 (2003).

[PZP17] L. Pfeifenberger, M. Zöhrer, and F. Pernkopf. “DNN-based speech mask estimation for eigenvector beamforming”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[RHR96] T. Robinson, M. Hochberg, and S. Renals. “The Use of Recurrent Neural Networks in Continuous Speech Recognition”. In: *Automatic Speech and Speaker Recognition: Advanced Topics*. Ed. by C.-H. Lee, F. K. Soong, and K. K. Paliwal. Boston, MA: Springer, 1996, pp. 233–258.

[RJ86] L. R. Rabiner and B. Juang. “A tutorial on hidden Markov models”. In: *IEEE ASSP Magazine* 3.1 (1986), pp. 4–16.

[Rob+95] T. Robinson, J. Fransen, D. Pye, J. Foote, and S. Renals. “WSJCAM0: a British English speech corpus for large vocabulary continuous speech recognition”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1995.

[Ros58] F. Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[RPB15] O. Ronneberger, P.Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. Springer, 2015, pp. 234–241.

[Rus+15] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[Sai+15a] T. N. Sainath, R. J. Weiss, K. W. Wilson, A. Narayanan, M. Bacchiani, and Andrew. "Speaker location and microphone spacing invariant acoustic modeling from raw multichannel waveforms". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 2015.

[Sai+15b] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. "Convolutional, long short-term memory, fully connected deep neural networks". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015.

[Sai+15c] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals. "Learning the speech front-end with raw waveform CLDNNs". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2015.

[Sai+16] T. Sainath, R. Weiss, K. Wilson, A. Narayanan, and M. Bacchiani. "Factored Spatial and Spectral Multichannel Raw Waveform CLDNNs". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016.

[Sai+18] T. N. Sainath, C. Chiu, R. Prabhavalkar, A. Kannan, Y. Wu, P. Nguyen, and Z. Chen. "Improving the Performance of Online Neural Transducer Models". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018.

[SAM07] H. Sawada, S. Araki, and S. Makino. "Measuring dependence of bin-wise separated signals for permutation alignment in frequency-domain BSS". In: *IEEE International Symposium on Circuits and Systems (ISCAS)*. 2007.

[SAM11] H. Sawada, S. Araki, and S. Makino. "Underdetermined Convolutive Blind Source Separation via Frequency Bin-Wise Clustering and Permutation Alignment". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.3 (Mar. 2011), pp. 516–527.

[Sch15] J. Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural Networks* 61 (2015), pp. 85–117.

[Ser+15] T. Sercu, C. Puhrsch, B. Kingsbury, and Y. LeCun. *Very Deep Multilingual Convolutional Neural Networks for LVCSR*. 2015. eprint: ArXiv:1509.08967.

[SG16] T. Sercu and V. Goel. "Advances in Very Deep Convolutional Neural Networks for LVCSR". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (2016).

[SGR14] P. Swietojanski, A. Ghoshal, and S. Renals. "Convolutional neural networks for distant speech recognition". In: *IEEE Signal Processing Letters* 21.9 (2014), pp. 1120–1124.

[SLY11] F. Seide, G. Li, and D. Yu. "Conversational speech transcription using context-dependent deep neural networks". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2011.

[SMW04] A. Spriet, M. Moonen, and J. Wouters. "Spatially pre-processed speech distortion weighted multi-channel Wiener filtering for noise reduction". In: *Signal Processing* 84.12 (2004), pp. 2367–2387.

[Sou+13] M. Souden, S. Araki, K. Kinoshita, T. Nakatani, and H. Sawada. "A Multichannel MMSE-Based Framework for Speech Source Separation and Noise Reduction". In: *IEEE Transactions on Audio, Speech, and Language Processing* 21.9 (2013), pp. 1913–1928.

[Sri+18] A. Sriram, H. Jun, S. Satheesh, and A. Coates. "Cold Fusion: Training Seq2Seq Models Together with Language Models". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)* (2018).

[SRS04] M. L. Seltzer, B. Raj, and R. M. Stern. "Likelihood-Maximizing Beamforming for Robust Hands-Free Speech Recognition". In: *IEEE Transactions on Speech and Audio Processing* 12 (2004). ISSN: 1063-6676. DOI: 10.1109/TSA.2004.832988.

[Sze+15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. *Rethinking the Inception Architecture for Computer Vision*. 2015. eprint: ArXiv: 1512.00567.

[Szu18] Szu-JuiChen. *Pull request to update REVERB recipe*. https://github.com/kaldi-asr/kaldi/pull/2753. [Online]. 2018.

[TH10] D. H. Tran Vu and R. Haeb-Umbach. "Blind speech separation employing directional statistics in an expectation maximization framework". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2010.

[TH12] D. H. Tran Vu and R. Haeb-Umbach. "Exploiting temporal correlations in joint multichannel speech separation and noise suppression using hidden Markov models". In: *International Workshop on Acoustic Signal Enhancement (IWAENC)*. 2012.

[TH13a] D. H. Tran Vu and R. Haeb-Umbach. "Blind speech separation exploiting temporal and spectral correlations using 2D-HMMs". In: *European Signal Processing Conference (EUSIPCO)*. 2013.

[TH13b] D. H. Tran Vu and R. Haeb-Umbach. "Using the turbo principle for exploiting temporal and spectral correlations in speech presence probability estimation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2013.

[Tra15] D. H. Tran Vu. "Integrated Multi-Channel Blind Speech Separation and Noise Reduction Using 2D Hidden Markov Models". PhD thesis. University of Paderborn, 2015.

[Tüs+14] Z. Tüske, P. Golik, R. Schlüter, and H. Ney. "Acoustic modeling with deep neural networks using raw time signal for LVCSR". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2014.

[Tüs+17] Z. Tüske, W. Michel, R. Schlüter, and H. Ney. "Parallel Neural Network Features for Improved Tandem Acoustic Modeling." In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2017.

[UVL16] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. *Instance Normalization: The Missing Ingredient for Fast Stylization*. 2016. eprint: ArXiv:1607.08022.

[Ves+13] K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey. "Sequence-discriminative training of deep neural networks." In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2013.

[Vin+] E. Vincent, S. Watanabe, J. Barker, and R. Marxer. *CHiME4 result overview*. http://spandh.dcs.shef.ac.uk/chime_challenge/chime2016/results.html. Accessed: 2019-01-31.

[Vin+13] E. Vincent, J. Barker, S. Watanabe, J. Le Roux, F. Nesta, and M. Matassoni. "The second 'CHiME'speech separation and recognition challenge: Datasets, tasks and baselines". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 126–130.

[Vin+16] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer. "An analysis of environment, microphone and data simulation mismatches in robust speech recognition". In: *Computer Speech & Language* (2016).

[Wan+18] Z. Wang, E. Vincent, R. Serizel, and Y. Yan. "Rank-1 constrained Multichannel Wiener Filter for speech recognition in noisy environments". In: *Computer Speech & Language* 49 (2018), pp. 37–51.

[WB+19] G.-Y. Wei, D. Brooks, et al. *Benchmarking TPU, GPU, and CPU Platforms for Deep Learning*. 2019. eprint: ArXiv:1907.10701.

[WC18] D. Wang and J. Chen. "Supervised speech separation based on deep learning: An overview". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.10 (2018), pp. 1702–1726.

[Wer90] P. J. Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[WH07] E. Warsitz and R. Haeb-Umbach. "Blind acoustic beamforming based on generalized eigenvalue decomposition". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.5 (2007), pp. 1529–1539.

[WLH18] Z.-Q. Wang, J. Le Roux, and J. R. Hershey. "Multi-channel deep clustering: Discriminative spectral and spatial embeddings for speaker-independent speech separation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, pp. 1–5.

[WNW14] Y. Wang, A. Narayanan, and D. Wang. "On Training Targets for Supervised Speech Separation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.12 (Dec. 2014), pp. 1849–1858.

[Xio+17] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke. *The Microsoft 2017 Conversational Speech Recognition System*. 2017. eprint: ArXiv:1708.06073.

[YN12] T. Yoshioka and T. Nakatani. "Generalization of multi-channel linear prediction methods for blind MIMO impulse response shortening". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.10 (2012), pp. 2707–2720.

[Yos+15] T. Yoshioka, N. Ito, M. Delcroix, A. Ogawa, K. Kinoshita, M. Fujimoto, C. Yu, W. J. Fabian, M. Espi, T. Higuchi, S. Araki, and T. Nakatani. "The NTT CHiME-3 system: Advances in speech enhancement and recognition for mobile multi-microphone devices". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 2015.

[YR04] O. Yilmaz and S. Rickard. "Blind separation of speech mixtures via time-frequency masking". In: *IEEE Transactions on Signal Processing* 52.7 (2004), pp. 1830–1847.

[Yu+17] D. Yu, M. Kolbæk, Z.-H. Tan, and J. 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.

[ZK16] S. Zagoruyko and N. Komodakis. *Wide Residual Networks*. 2016. eprint: ArXiv:1605.07146v4.

Own publications

[Böd+17] C. Böddeker, P. Hanebrink, L. Drude, J. Heymann, and R. Haeb-Umbach. *On the Computation of Complex-valued Gradients with Application to Statistically Optimum Beamforming*. 2017. eprint: ArXiv:1701.00392.

[Boe+17] C. Boeddeker, P. Hanebrink, L. Drude, J. Heymann, and R. Haeb-Umbach. "Optimizing neural-network supported acoustic beamforming by algorithmic differentiation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[Boe+18a] C. Boeddecker, J. Heitkaemper, J. Schmalenstroeer, L. Drude, J. Heymann, and R. Haeb-Umbach. "Front-end processing for the CHiME-5 dinner party scenario". In: *International Workshop on Speech Processing in Everyday Environments (CHiME'18)*. 2018.

[Chi+16] A. Chinaev, J. Heymann, L. Drude, and R. Haeb-Umbach. "Noise-presence-probability-based noise PSD estimation by using DNNs". In: *Speech Communication; 12. ITG Symposium*. 2016.

[DHH19b] L. Drude, J. Heymann, and R. Haeb-Umbach. "Unsupervised training of neural mask-based beamforming". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2019.

[Dru+18a] L. Drude, J. Heymann, C. Boeddeker, and R. Haeb-Umbach. "NARA-WPE: A Python package for weighted prediction error dereverberation in Numpy and Tensorflow for online and offline processing". In: *Speech Communication; 13. ITG-Symposium*. 2018.

[Dru+18b] L. Drude, C. Boeddeker, J. Heymann, R. Haeb-Umbach, K. Kinoshita, M. Delcroix, and T. Nakatani. "Integrating Neural Network Based Beamforming and Weighted Prediction Error Dereverberation." In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2018.

[Ebb+17] J. Ebbers, J. Heymann, L. Drude, T. Glarner, R. Haeb-Umbach, and B. Raj. "Hidden Markov Model Variational Autoencoder for Acoustic Unit Discovery." In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2017.

[HBS18] J. Heymann, M. Bacchiani, and T. N. Sainath. "Performance of mask based statistical beamforming in a smart home scenario". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018.

[HDH16a] J. Heymann, L. Drude, and R. Haeb-Umbach. "Neural network based spectral mask estimation for acoustic beamforming". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2016.

[HDH16b] J. Heymann, L. Drude, and R. Haeb-Umbach. "Wide residual BLSTM network with discriminative speaker adaptation for robust speech recognition". In: *International Workshop on Speech Processing in Everyday Environments (CHiME'16)*. 2016.

[HDH17] J. Heymann, L. Drude, and R. Haeb-Umbach. "A Generic Neural Acoustic Beamforming Architecture for Robust Multi-Channel Speech Processing". In: *Computer Speech & Language* (2017).

[Hey+13] J. Heymann, O. Walter, R. Haeb-Umbach, and B. Raj. "Unsupervised word segmentation from noisy input". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 2013.

[Hey+14] J. Heymann, O. Walter, R. Haeb-Umbach, and B. Raj. "Iterative Bayesian word segmentation for unsupervised vocabulary discovery from phoneme lattices". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014.

[Hey+15a] J. Heymann, L. Drude, A. Chinaev, and R. Haeb-Umbach. "BLSTM supported GEV beamformer front-end for the 3rd CHiME challenge". In: *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding (ASRU)*. 2015.

[Hey+15b] J. Heymann, R. Haeb-Umbach, P. Golik, and R. Schlüter. "Unsupervised adaptation of a denoising autoencoder by bayesian feature enhancement for reverberant asr under mismatch conditions". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015.

[Hey+17] J. Heymann, L. Drude, C. Boeddeker, P. Hanebrink, and R. Haeb-Umbach. "BEAMNET: End-to-end training of a beamformer-supported multi-channel ASR system". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2017.

[Hey+18] J. Heymann, L. Drude, R. Haeb-Umbach, K. Kinoshita, and T. Nakatani. "Frame-online DNN-WPE dereverberation". In: *International Workshop on Acoustic Signal Enhancement (IWAENC)*. 2018.

[Hey+19] J. Heymann, L. Drude, R. Haeb-Umbach, K. Kinoshita, and T. Nakatani. "Joint optimization of neural network-based wpe dereverberation and acoustic model for robust online asr". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019.

[HHH18] J. Heitkaemper, J. Heymann, and R. Haeb-Umbach. "Smoothing along Frequency in Online Neural Network Supported Acoustic Beamforming". In: *Speech Communication; 13. ITG-Symposium*. 2018.

[HSL19] J. Heymann, K. C. Sim, and B. Li. "Improving CTC Using Stimulated Learning for Sequence Modeling". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019.

[Kit+16] M. Kitza, A. Zeyer, R. Schlueter, J. Heymann, and R. Haeb-Umbach. "Robust Online Multi-Channel Speech Recognition". In: *Speech Communication; 12. ITG Symposium*. 2016.

[Kit+18] M. Kitza, W. Michel, C. Boeddeker, J. Heitkaemper, T. Menne, R. Schlüter, H. Ney, J. Schmalenstroeer, L. Drude, J. Heymann, et al. "The RWTH/UPB system combination for the CHiME 2018 workshop". In: *International Workshop on Speech Processing in Everyday Environments (CHiME'18)*. 2018.

[Men+16] T. Menne, J. Heymann, A. Alexandridis, K. Irie, A. Zeyer, M. Kitza, P. Golik, I. Kulikov, L. Drude, R. Schlüter, et al. "The RWTH/UPB/-FORTH system combination for the 4th CHiME challenge evaluation". In: *International Workshop on Speech Processing in Everyday Environments (CHiME'16)*. 2016.

[Sch+17] J. Schmalenstroeer, J. Heymann, L. Drude, C. Boeddecker, and R. Haeb-Umbach. "Multi-stage coherence drift based sampling rate synchronization for acoustic beamforming". In: *2017 IEEE 19th International Workshop on Multimedia Signal Processing (MMSP)*. 2017.

[Wat+18] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai. "ESPnet: End-to-End Speech Processing Toolkit". In: *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. 2018.