KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Joint Speaker Segregation and Recognition

Jeroen Zegers

Supervisor:
Prof. dr. ir. H. Van hamme

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

July 2020

Joint Speaker Segregation and Recognition

Jeroen ZEGERS

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Electrical Engineering

July 2020

Preface

As children we have all been taught to listen politely to others before speaking ourselves. Unfortunately, sometimes this lesson is forgotten, causing simultaneous or overlapping speech. Overlapping speech also occurs naturally when many people are present in the same environment, sometimes referred to as *the cocktail party problem.* While normal-hearing individuals can deal with this to some extent, the same cannot be said for hearing-impaired persons or hearing-aid users. Machines also struggle with automatic speech processing in such scenarios. This is unfortunate, as speech is a user friendly way of communication between human and computer. This PhD thesis addresses this problem by applying automatic speech separation. More specifically, the link between speaker characterization and speaker separation will be studied. *Who said what?*

I would not be able to present this manuscript without the help of many people around me. The most important one being Hugo Van hamme, my supervisor. Probably being one of the university's most approachable professors, we had many formal and informal meetings. These discussions, where there were no limits but our imagination, are the moments I cherish most during my time at the university.

Tinne Tuytelaars and Alexander Bertrand, as members of my supervisory committee, have provided useful feedback. I am also glad to have had the opportunity to co-author with both of them on one or more publications. I appreciate the time invested in reading my manuscript by them and the other members of the jury: Sarah Verhulst, Patrick Wambacq and Erik Marchi. I thank the chair Hendrik Van Brussel.

I acknowledge the SB grant received by FWO, which allowed me to work on this thesis without distractions. I also acknowledge KU Leuven to allow me to write and present this manuscript.

I have many great memories of technical, but mostly non-technical discussions, moments, activities and coffee breaks with my colleagues of the past and present. I would like to thank Deepak, Joris, Reza, Alec, Vincent, Lyan, Louis, Yinan, Wim, Jalil, Bernd, Jinzi, Jakob, Lies, Pu, Quentin and Bagher.

Without the unconditional support of my mother and father, I can't imagine ever to have started my engineering education, let alone writing a PhD manuscript. I would like to thank them from the bottom of my hearth for all the wonderful opportunities they have given me. I would also like to thank my sister Sofie as she has been a great example for me. Finally, I would like to thank my partner Gwen for showing me everything that is important in life outside of the university.

Abstract

Many speech technology applications expect speech input from a single speaker and usually fail when multiple speakers are active, especially when speech overlaps. However, in many situations there are multiple people within reach of the recording device and therefore there is a high chance of multiple active speakers. In Speech Source Separation (SSS), the different speech sources are separated to obtain an audio signal for each speaker. As the sources are separated, we would also like to know the identity or the characteristics of the speaker by Speaker Recognition (SR) so we can re-identify the speaker later on. If both SSS and SR are being applied, we can track a single speaker throughout, for example, a recording of a business meeting with overlapping speech. SSS and SR are usually being treated as separate problems. However, when (blindly) separating a speech mixture, characterization of the sources is inherently necessary. Moreover, when recognizing speakers in overlapping speech, every speaker is associated with part of the audio fragment and thus source separation is inherently active. The main research question of the PhD thesis is whether a joint approach to SSS and SR makes them constructively help each other to achieve greater performance. A *sequential* approach where first SSS is done, followed by SR, is expected to be less efficient as each step is optimized independently and neglects the other step.

A first attempt in building such a joint model is done using Nonnegative Matrix Factorization (NMF). NMF is an often used method in SSS that looks for spectral patterns in an audio example. An existing multi-channel NMF model for SSS is adapted such that the same spectral patterns, that were extracted for SSS, can be used for SR. This model is compared with two sequential baselines in terms of SR performance in overlapping speech. One sequential baseline uses NMF, while the other uses i-vectors, a state-of-the-art method for SR in non-overlapping speech. The joint model is shown to outperform both.

In recent years, Deep Neural Networks (DNNs) have gained considerable attention in the SSS field. A DNN is shared in a joint model for SSS and

SR. It is studied how best to learn such a model for multiple tasks. The joint model outperforms a sequential and a parallel baseline in SR performance. Furthermore, a first adjustment to the joint model is made such that the SR task better takes into account the uncertainty of SSS. A second adjustment allows the model to handle both overlapping and non-overlapping speech, a useful feature if for some speakers enrollment can be done in a controlled, single-speaker environment.

One of the major drawbacks of DNNs in general is that due to their nested and non-linear structure, it is difficult to understand what makes them arrive at their prediction and therefore DNNs lack explainability. Two novel methods are developed that are generally applicable to study the memory of Recurrent Neural Network (RNN)s. These methods allow to restrict the duration that information is kept in the memory of RNNs. This duration is also referred to as memory span in this text. It is then measured how the RNN performs on a specific task for different arbitrarily chosen memory spans. These methods are applied to the task of SSS to determine to which extent SR and other factors play a role in a successful separation. By modifying the memory span per layer in a deep RNN, we discover hierarchical structures in the memory of an RNN. Furthermore, for a bidirectional RNN, a distinction is made between the importance of the forward network and the backward network.

Finally, the use of SSS for a neuro-steered hearing aid is studied. In summary, brain activity of a hearing-aid user is measured in a multi-speaker scenario, where the user tries to focus on a single target speaker. The measured brain signals are used to estimate the speech envelope of the attended target speaker using an Auditory Attention Decoder (AAD). The use of SSS here is twofold. First, the envelopes of the estimated speech signals are compared with the decoded speech envelope of the AAD module to estimate the attended speaker. Secondly, once the target speaker is estimated, this speaker's separated speech signal is sent to the hearing aid. A DNN for SSS is compared with a traditional linear SSS method. The DNN manages to outperform the traditional method in the most challenging cases.

Beknopte samenvatting

De meeste toepassingen van spraaktechnologieën verwachten een spraakinvoer waarbij slechts één spreker actief is. Deze toepassingen falen dan meestal wanneer meerdere sprekers actief zijn, in het bijzonder als de spraak overlapt. Echter, in vele situaties bevinden er zich meerdere personen in de buurt van een opnameapparaat en dus is er een grote kans dat meerdere sprekers actief zijn. Door spraaksignaalscheiding (*Speech Source Separation* of SSS) worden de verschillende spraakbronnen van elkaar gescheiden zodat voor elke spreker een individueel audiosignaal wordt bekomen. Terwijl we de sprekers scheiden, zouden we graag ook de sprekeridentiteit of de sprekerkarakteristieken leren, ook wel sprekerherkenning (*Speaker Recognition* of SR) genoemd. Dit laat ons toe om later eenzelfde spreker te herkennen. Wanneer SSS en SR worden toegepast, kunnen we een enkele spreker traceren doorheen bijvoorbeeld een opname van een meeting met overlappende spraak. Meestal worden SSS en SR beschouwd als gescheiden problemen. Echter, wanneer de sprekers van elkaar worden gescheiden op een blinde manier, is karakterisatie van de bronnen inherent nodig. Verder geldt ook dat wanneer men sprekers tracht te herkennen in overlappende spraak, men elke spreker moet verbinden aan een deel van het audiofragment en dus is spraakscheiding inherent nodig. De hoofdvraag van deze doctoraatsthesis is of een gemeenschappelijke aanpak van SSS en SR ervoor zorgt dat ze elkaar constructief helpen om een betere performantie te halen. Een *sequentiële* aanpak waarbij eerst SSS wordt toegepast, gevolgd door SR, is waarschijnlijk minder efficiënt aangezien elke stap onafhankelijk van de andere wordt geoptimaliseerd.

Een eerste poging om een dergelijk gemeenschappelijk model te bouwen, gebeurt door middel van *Nonnegative Matrix Factorization* (NMF). NMF is een vaak gebruikte methode voor SSS die op zoek gaat naar spectrale patronen in een audiovoorbeeld. Een bestaand meerkanaals NMF model voor SSS wordt aangepast zodat dezelfde spectrale patronen, die waren ontdekt voor SSS, kunnen worden gebruikt voor SR. Dit model wordt vergeleken met twee sequentiële basismodellen in termen van accuraatheid in SR. Het ene basismodel gebruikt

NMF, terwijl het andere i-vectors gebruikt, een *state-of-the-art* methode voor SR in niet-overlappende spraak. Het gemeenschappelijke model blijkt beter te presteren dan beide basismodellen in termen van SR performantie.

De laatste jaren hebben diepe neurale netwerken (*Deep Neural Networks* of DNNs) veel aandacht gewonnen in het domein van SSS. Een gemeenschappelijk DNN wordt gebruikt voor SSS en SR. Er wordt onderzocht hoe een dergelijk model het best kan leren van beide taken. Het gemeenschappelijk model blijkt beter te presteren dan een sequentieel en een parallel basismodel in termen van SR performantie. Bovendien zorgt een eerste aanpassing aan het gemeenschappelijk model ervoor dat de SR taak beter de onzekerheid van het SSS deel in beschouwing neemt. Een tweede aanpassing laat het model toe om zowel overlappende als niet-overlappende spraak te verwerken. Dit kan handig zijn indien voor sommige sprekers de sprekerkarakteristieken op voorhand kunnen worden geleerd in een gecontroleerde omgeving waarbij enkel die spreker actief is.

Een van de grote nadelen van DNNs in het algemeen is dat, door hun geneste structuur, het moeilijk is om te begrijpen hoe DNNs tot hun uiteindelijke beslissing komen. Twee nieuwe methodes worden ontwikkeld die algemeen toepasbaar zijn om de geheugenwerking van een recurrent neuraal netwerk (*Recurrent Neural Network* of RNN) te onderzoeken. Deze methodes laten toe om een beperking te plaatsen op de tijd dat informatie in het geheugen van het RNN wordt gehouden. Er kan dan worden gemeten hoe het RNN presteert op een bepaalde taak, afhankelijk van verschillende toegestane geheugentijden. Deze methodes worden toegepast op de SSS taak om te onderzoeken in welke mate SR en andere factoren een rol spelen in een succesvolle sprekerscheiding. Door de geheugenrestricties aan te passen per laag in een diep RNN, worden hiërarchische structuren in het geheugen van het RNN ontdekt. Verder ontdekken we dat voor een bi-directionieel RNN, een onderscheid kan worden gemaakt tussen het belang van de voorwaartse en de achterwaartse richting.

Ten slotte wordt het gebruik van SSS in een neurologisch gestuurd gehoorapparaat bestudeerd. In een dergelijk gehoorapparaat wordt de hersenactiviteit van een gebruiker van een gehoorapparaat gemeten. Deze gebruiker probeert te focussen op een enkele spreker in overlappende spraak. De gemeten hersensignalen worden gebruikt om de spraakenveloppe van de gefocuste spreker te reconstrueren aan de hand van een auditieve aandachtsdecoder (*Auditory Attention Decoder* of AAD). SSS wordt hier voor twee redenen gebruikt. Ten eerste, de enveloppes van de gescheiden spraaksignalen kunnen worden vergeleken met de gedecodeerde spraakenveloppe van de AAD om te schatten naar welke spreker de gebruiker aan het luisteren is. Ten tweede, eens de relevante spreker is bepaald, kan het gescheiden spraaksignaal van deze spreker worden doorgestuurd naar het gehoorapparaat. Een DNN voor SSS zal worden

vergeleken met een traditionele, lineaire SSS methode. Het DNN zal beter blijken te presteren dan de traditionele methode in de meest uitdagende scenarios.

Abbreviations

AAD Auditory Attention Decoder. iv, vi, xxv, 36, 109–115, 118, 122, 124–127, 129–133, 137

ASR Automatic Speech Recognition. 33, 36, 49, 78, 90

BPTT Backpropagation Through Time. 24

BSS Blind Source Separation. xxv, 109–114, 125, 127–133

BSSS Blind Speech Source Separation. xxi, xxiii, xxvii, 1, 2, 5, 6, 8–10, 12, 17, 26, 27, 30, 34–37, 39, 40, 47, 49, 50, 52–55, 57, 58, 60–62, 64, 66–69, 71, 73, 75–78, 82, 88–90, 93, 95, 97, 100, 103, 108–110, 117, 135–138

CASA Computational Auditory Scene Analysis. 5

CGNM Corpus Gesproken Nederlands Mix. 120, 123

CNN Convolutional Neural Network. 138

DANet Deep Attractor Networks. xxii, xxviii, 27, 30–32, 58, 67, 97, 100–103, 105–107

DC Deep Clustering. xxii, xxviii, 5, 27, 28, 30, 32, 51, 52, 54, 97–103, 106, 107, 112, 117

DL Deep Learning. 6, 17, 21, 26, 32, 35–37, 49, 97, 108, 110, 136–138

DNN Deep Neural Network. iii, iv, vi, xxv, xxvi, xxviii, 2, 6, 7, 17, 26, 32–35, 47, 49, 52, 54, 55, 57, 60, 73, 75, 98, 103, 109–115, 117, 118, 120–133, 135, 137

DOA Direction of Arrival. 12, 42, 111

ECoG Electro-Corticography. 111, 112

EEG Electroencephalography. 36, 109–115, 119, 121, 122, 124, 129, 132

EER Equal Error Rate. xxi–xxiii, xxvii, 7, 8, 66–72

fMLLR feature-space Maximum Likelihood Linear Regression. 49, 50

GCC-PHAT Generalized Cross Correlation with Phase Transform. 42, 43

GEVD Generalized Eigenvalue Decomposition. 15, 117

GMM Gaussian Mixture Model. xxi, 15, 16

HMM Hidden Markov Model. 5

ICA Independent Component Analysis. 5, 6

IPD Interchannel Phase Difference. 118

IS Itakura-Saito. 10, 12, 139

KL Kullback-Leibler. 10

LCMV Linearly Constrained Minimum Variance. 111

LDA Linear Discriminant Analysis. 15, 51, 53

LMMSE Linear minimum MSE. 116

LS LibriSpeech. xxiv, 93, 94

LSTM Long Short-term Memory. xxi, xxiii, 24–26, 35, 52, 69, 75–79, 81–83, 85, 88, 90, 92, 95, 99, 103, 121, 136, 138, 147

M-NICA Multiplicative Nonnegative ICA. xxv, xxvi, 109, 111–113, 115–118, 121, 122, 125–131, 137

MFCC Mel-Frequency Cepstral Coefficients. 44, 53

MSE Mean Square Error. 19, 114, 116

MWF Multi-channel Wiener Filter. xxv, xxvi, 111–118, 121, 122, 125–132, 137

NMF Nonnegative Matrix Factorization. iii, v, xxii, 2, 5, 7, 10–14, 34, 35, 39, 43–47, 57, 135

PESQ Perceptual evaluation of speech quality. xxvi, 6, 112, 127–129

RBM RadioBoeken Mix. xxviii, 119, 120, 123–125, 127

RIR Room Impulse Response. 3, 42

RNN Recurrent Neural Network. iv, vi, xxi–xxiii, 21–26, 28, 30, 31, 35, 52, 55, 59, 63, 75–79, 81, 83, 85, 88, 90, 92, 95, 96, 121, 136, 138, 147

SAR Signal-to-Artefact Ratio. xxvii, 46, 47

SDR Signal-to-Distortion Ratio. xxii, xxiii, xxvii, xxviii, 6, 46, 47, 53–55, 68–72, 100–102, 107, 123

SE Speech Enhancement. 3, 11, 26

SGD Stochastic Gradient Descent. 20, 21, 60–62

SID Speaker Identification. 6, 7

SINR Signal-to-Interference-plus-Noise Ratio. xxv, 112, 127–129

SIR Signal-to-Interference Ratio. xxvii, 46, 47

SNR Signal-to-Noise Ratio. xxviii, 107, 111, 119–121

SR Speaker Recognition. iii–vi, xxi, xxvii, 1, 2, 6–10, 14, 17, 32, 34–36, 39, 41–47, 50, 55–58, 60, 62, 64, 66–69, 71–73, 75, 135–138

SRE Speaker Recognition Evaluation. 66, 73

SS Source Separation. 1–3, 10, 11, 26, 34, 115, 129

SSS Speech Source Separation. iii–v, xxi, xxvii, 1–5, 10–12, 14, 26, 27, 34, 39, 40, 42–47

STFT Short-time Fourier Transform. xiv, 3–5, 43, 44, 52, 81, 88, 117, 121

STOI Short-Term Objective Intelligibility. 6

SV Speaker Verification. xxi, 6–8, 66

TDOA Time Difference of Arrival. 12

UBM Universal Background Model. xxi, 15, 16, 44, 53

uPIT utterance Permutation Invariant Training. xxii, xxviii, 27, 29, 30, 32, 97–99

VAD Voice Activity Detection. xxv, xxvi, 53, 92, 115, 116, 118, 125–129, 133

WER Word Error Rate. 78

WSJ Wall Street Journal. xxiv, 52, 53, 66, 72, 73, 93, 94, 106

List of Symbols

$\mathbf{a}$ Attractors for DANet. xxii, 30, 32, 58, 59, 62, 64

D The number of dimensions in the embedding vectors. 27, 29, 52, 66

F The number of frequencies or features. 10, 12, 27, 52, 104, 121

f The frequency or feature index. 12, 115, 117

I The number of enrollment speakers or *known* speakers. xxi, 6–9, 11, 12, 14, 40–43, 66, 72, 100

i enrollment speaker index. 11, 12, 40

J The number of microphones. 4, 5, 12, 13, 42, 116, 140

j Microphone index. 116

$\mathbf{K}$ The number of instances. As dictionary elements in NMF or memory instances in a reset RNN. xxiii, 10, 12, 41, 43–45, 83–87

$\mathbf{k}$ The instance index. xxiii, 10, 12, 39, 83–87

$\mathbf{M}$ Also noted as $m(t, f)$. The desired time-frequency mask to extract a source in a mixture. 5, 6

$\mathbf{Q}$ The activations in NMF. 10–12, 14, 15, 43, 139, 140, 142

S The number of speakers or sources in a mixture. xxi, xxii, xxvii, 3–5, 8, 9, 12, 13, 27, 29, 31, 39–43, 52, 58, 59, 63–65, 67–69, 98, 99, 103, 105, 113–115, 117, 118

s Speaker or source index in a mixture. 3–6, 12, 26, 27, 29, 40, 105, 115, 116

$\mathbf{T}$ The dictionary in NMF. 10–12, 14, 40, 41, 43, 139, 140, 142

T The number of time frames or the sequence length. xxi, xxiii, 10, 12, 14, 21, 23, 25, 27, 83, 85, 104

t The time frame index. 10, 12, 21, 22, 115

$\mathbf{V}$ The embedding matrix for DC or DANet. 27–29, 32, 53

$\mathbf{v}$ An embedding vector for DC or DANet. 27–29, 32, 35, 58, 62, 103, 105, 117

$\mathbf{X}$ Multiple model input vectors. 6, 10–12, 21, 40, 60, 139, 141

$\mathbf{x}$ A general model input vector. 10, 17–19, 21, 22, 24, 25, 84, 85

$\mathbf{Y}$ Multiple desired model output vectors. 6, 60

$\mathbf{y}$ A general desired model output vector. 17, 19, 22

$\mathcal{Y}$ Also noted as $y(t, f)$. The STFT of a time-domain signal $y[n]$. 4–6, 11, 27, 117, 118

$\mathbf{Z}$ Target matrix for DC. 27, 28

$\circ$ The Hadamard product. 5

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automatic speech processing is a major research area since it allows a user friendly way of communication between human and computer. When multiple sound sources are active, the processing of the speech signal(s) from a recording is more difficult compared to a so-called *clean* speech signal where only one sound source is present. For example, a speech signal can become corrupted in a crowded bar or even at home, which hinders applications such a voice assistants, automatic transcriptions and hearing aids. Many of these applications expect speech input from a single source and usually fail when multiple sources are active, especially when they overlap in time. In Source Separation (SS) the original source signals are estimated from such a multi-source audio recording, also called a mixture. This is particularly difficult when all sources are speech sources, which will be referred to as Speech Source Separation (SSS). Often the speakers are unknown and SSS is performed blindly, called Blind SSS (BSSS). As the speech sources are separated, it is often beneficial to look for speaker characteristics such that the speaker can be re-identified later on, using Speaker Recognition (SR). If both BSSS and SR are being applied we can track a single speaker throughout, for example, a recording of a business meeting.

BSSS and SR are usually being treated as separate problems. Specialists in SR mostly consider single speaker audio recordings. In case of overlapping speech they rely on BSSS specialists to convert the multi-speaker audio recording into multiple single-speaker tracks, or they simple ignore this type of data. A BSSS specialist, on the other hand, will consider his task done after successfully separating all speech sources in an audio recording. However, when blindly separating a speech mixture, characterization of the sources seems inherently necessary for a consistent mapping of speech fragments to the correct source.

Moreover, when recognizing speakers in overlapping speech, every speaker is associated with part of the audio fragment and thus BSSS seems inherently active. The two problems seem to be linked. The main research question of this doctoral thesis is whether a joint approach to BSSS and SR makes them constructively help each other to achieve greater performance. A sequential approach [113] where first BSSS is done, followed by SR, seems less efficient as each step is optimized independently and neglects the other step.

Chapters 2 and 4 of the thesis will look for such joint models and indeed find that they outperform sequential approaches. In Chapter 2 this is done using Nonnegative Matrix Factorization (NMF), while in Chapter 4 Deep Neural Networks (DNNs) are used. Given that they reach related conclusions, it is expected that the dependencies between BSSS and SR are independent of the model choice and rather intrinsic to the tasks themselves. Chapters 3 and 5 will not explicitly search for a joint model, but will rather show that indeed speaker characteristic information is inherently required when performing SS. Finally, Chapters 6 and 7 will consider more applied and practical aspects of BSSS.

The remainder of this chapter introduces some basic concepts used in this thesis text. In Section 1.1 the tasks of BSSS and SS will be described, as well as the joint BSSS-SR problem. Sections 1.2, 1.3 and 1.4 will describe the NMF method, i-vectors and DNNs, respectively. NMF and DNN can be used for both BSSS and SR and are therefore suited for building a joint model. The i-vector method is used for SR but has no direct application in BSSS. Finally, in Section 1.5 a short introduction to each chapter will be given.

1.1 Problem statement

The tasks of (B)SSS and SR will be briefly explained in Sections 1.1.1 and 1.1.2, respectively. In Section 1.1.3 the joint problem will then be defined.

1.1.1 Speech Source Separation

The cocktail party problem, where multiple sound sources, usually speakers, are simultaneously active, has been studied in the speech community for decades [20]. The problem is especially challenging when few assumptions are made. For a general solution we want to be speaker independent, text independent, using a single channel and so on. Source Separation (SS) refers to the task of retrieving the original sound signal of multiple sound source objects that have been recorded in a mixture. More specifically, this task focuses on the case when the sound sources are (partially) active at the same time.

SS can be divided into two domains: speech vs. noise separation and speech vs. speech separation. In the former, typically only speech is of interest and therefore it is also called Speech Enhancement (SE). In this text, the latter is referred to as Speech Source Separation (SSS) to stress that multiple speech sources are present. An illustration of (B)SSS is given in Figure 1.1. A mixture, or recording, of S speech sources in its simplest form can be defined as

$$y[n] = \sum_{s=1}^{S} x_s[n] \tag{1.1}$$

where $y[n]$ is the microphone signal, $x_s[n]$ is the audio signal of source s as received by the microphone and n is the sample index of the microphone. The formulation of (1.1) will be used in the remainder of the text when *mixture* is mentioned, unless stated otherwise. We give two additional formulations of the *mixture* as they will be used in some chapters. In noisy speech mixtures, also non-speech signals are present and these will be bundled in the noise term $n[n]$.

$$y[n] = \sum_{s=1}^{S} x_s[n] + n[n] \tag{1.2}$$

In Chapters 6 and 7 noisy mixtures will be considered. Separating a noisy mixture into its source components, can be seen as a combination of the SE and SSS tasks. In (1.1) and (1.2) $x_s[n]$ was defined as the audio signal of source s as received by the microphone. However $x_s[n]$ can also be defined as the audio signal measured at the source. (1.2) is then changed to

$$y[n] = \sum_{s=1}^{S} h_s[n] \circledast x_s[n] + n[n] \tag{1.3}$$

where $h_s[n]$ is the Room Impulse Response (RIR) of source s to the microphone and $\circledast$ is the convolution operation.

The above formulations were done in the time-domain. However, it is often more useful to consider speech or audio signals in the time-frequency domain. The Short-time Fourier Transform (STFT) is used to transform a time-domain signal into the time-frequency domain. In the time-frequency domain (1.1)–(1.3) become

$$\underline{y}(t,f) = \sum_{s=1}^{S} \underline{x}_s(t,f) \tag{1.4}$$

$$\underline{y}(t,f) = \sum_{s=1}^{S} \underline{x}_s(t,f) + \underline{n}(t,f) \tag{1.5}$$

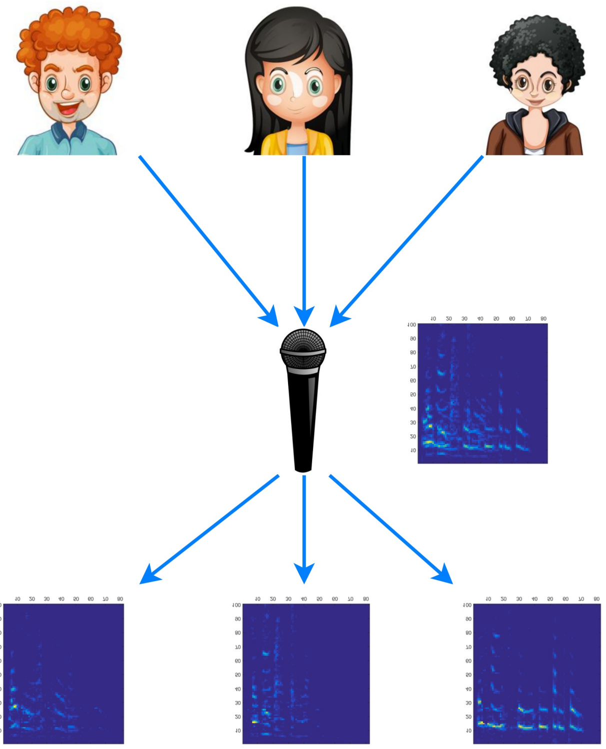


Figure 1.1: An illustration of (B)SSS for $S = 3$

$$\boldsymbol{y}(t,f) = \sum_{s=1}^{S} \boldsymbol{\hbar}_s(f)\boldsymbol{x}_s(t,f) + \boldsymbol{n}(t,f), \tag{1.6}$$

respectively, with $\boldsymbol{x}_s(t,f)$, $\boldsymbol{y}(t,f)$, $\boldsymbol{n}(t,f)$ and $\boldsymbol{\hbar}_s(f)$ the STFT of $x_s[n]$, $y[n]$, $n[n]$ and $h_s[n]$, respectively. Here, $\boldsymbol{\hbar}_s(f)$ is considered time-invariant and therefore independent of t. For ease of notation, (1.4) can also be written in matrix notation as

$$\boldsymbol{\mathcal{Y}} = \sum_{s=1}^{S} \boldsymbol{\mathcal{X}}_s \tag{1.7}$$

A recording device needs not be limited to one microphone, but in general has J microphones. For this multi-channel set-up (1.6) would be extended to

$$\boldsymbol{y}(t,f) = \sum_{s=1}^{S} \boldsymbol{\hbar}_s(f)\boldsymbol{x}_s(t,f) + \boldsymbol{n}(t,f) \tag{1.8}$$

with $\boldsymbol{y}(t,f) = [y_1(t,f), y_2(t,f), \ldots, y_J(t,f)]^{\mathrm{T}}$ and similar for $\boldsymbol{h}_s(f)$ and $\boldsymbol{n}(t,f)$.

The goal of (B)SSS is to find an estimate $\hat{\boldsymbol{\mathcal{X}}}_s$ for every source signal $\boldsymbol{\mathcal{X}}_s$ from the mixture $\boldsymbol{\mathcal{Y}}$. The estimated signal $\hat{x}_s[n]$ in the time domain can then be found using the Inverse Short-time Fourier Transform (ISTFT). Most BSSS approaches do not try to find the estimate $\hat{\boldsymbol{\mathcal{X}}}_s$ directly, but rather estimate a mask $\hat{\mathbf{M}}_s$ and then find the estimate $\hat{\boldsymbol{\mathcal{X}}}_s$ as

$$\hat{\boldsymbol{\mathcal{X}}}_s = \hat{\mathbf{M}}_s \circ \boldsymbol{\mathcal{Y}} \tag{1.9}$$

or

$$\hat{x}_s(t,f) = \hat{m}_s(t,f) y(t,f). \tag{1.10}$$

with $\circ$ the Hadamard product. Typically, the masks are constrained to $0 \leqslant \hat{m}_s(t,f) \leqslant 1$ and $\sum_{s=1}^{S} \hat{m}_s(t,f) = 1$. The latter constraint is usually not applied in the case of an additive noise source $\boldsymbol{n}(t,f)$. Furthermore, it is common that the mask $\hat{\mathbf{M}}_s$ is real-valued such that the magnitude is estimated as $|\hat{\boldsymbol{\mathcal{X}}}|_s = \hat{\mathbf{M}}_s \circ |\boldsymbol{\mathcal{Y}}|$ and the phase or argument is taken from $\boldsymbol{\mathcal{Y}}$ ($\angle \hat{\boldsymbol{\mathcal{X}}}_s = \angle \boldsymbol{\mathcal{Y}}$). When $\hat{x}_s(t,f) \approx y(t,f)$, the phase will thus be reasonably approximated. Since speech spectra are sparse [110], it is actually rare that multiple sources have high energy in a given time-frequency bin and usually a bin is dominated by a single source. For this dominant source, the phase of $y(t,f)$ is thus a reasonable approximation for the phase of $x_s(t,f)$. For the other sources it is not, but since the optimal mask estimates $m_{s' \neq s}(t,f)$ for those non-dominant sources should be close to zero, the estimate of the phase is of less importance. In fact, some BSSS approaches (like Deep Clustering (DC) in Section 1.4.2) estimate binary masks where for each time-frequency bin the mask for one source is set to 1 and all others are set to 0. However, mask approaches will always suffer, to some extent, from these phase prediction errors. Recently, some BSSS approaches that work directly in the time domain (using (1.1)) have shown to exceed the upper bound performance of an ideal mask [158, 159].

For the multi-channel case (1.10) is generalized to

$$\hat{x}_s(t,f) = \hat{\mathbf{m}}_s^{\mathrm{H}}(t,f) \boldsymbol{y}(t,f) \tag{1.11}$$

with $.^{\mathrm{H}}$ the Hermitian transpose. In the multi-channel case, often $\hat{\mathbf{m}}$ is in the complex domain to facilitate beamforming.

Two methods for BSSS will be discussed in this thesis: Nonnegative Matrix Factorization (NMF) in Section 1.2.1 and Deep Neural Networks (DNNs) in Section 1.4.2. Other methods like Computational Auditory Scene Analysis (CASA) [180, 156], Wiener filtering [114], spectral subtraction [19], factorial Hidden Markov Models (fHMMs) [178, 79, 125] and Independent Component

Analysis (ICA) [98] exist but their use for BSSS has been limited since DL methods have become state-of-the-art. One of the major advantages of a DNN is that it manages well to perform single-channel BSSS, where as others have to impose restrictions on the speakers or vocabulary or have to resort to multi-channel approaches.

The general input-output notation used in these models is $\mathbf{X}$ and $\hat{\mathbf{Y}}$. To avoid confusion, the reader is reminded that for BSSS, the mixture spectrogram $\boldsymbol{\mathcal{Y}}$ is given and speech source spectrograms $\boldsymbol{\mathcal{X}}_s$ are to be estimated. For instance, often used inputs to BSSS models are the power spectrogram $\mathbf{X} = |\boldsymbol{\mathcal{Y}}|.^2$, with $.^2$ the element-wise square, or the log spectrogram $\mathbf{X} = \log(|\boldsymbol{\mathcal{Y}}|)$, while the output is typically a collection of masks $\hat{\mathbf{Y}} = [\hat{\mathbf{M}}_1, \hat{\mathbf{M}}_2, \ldots, \hat{\mathbf{M}}_s]$ to be used with (1.9) to estimate $\boldsymbol{\mathcal{X}}_s$.

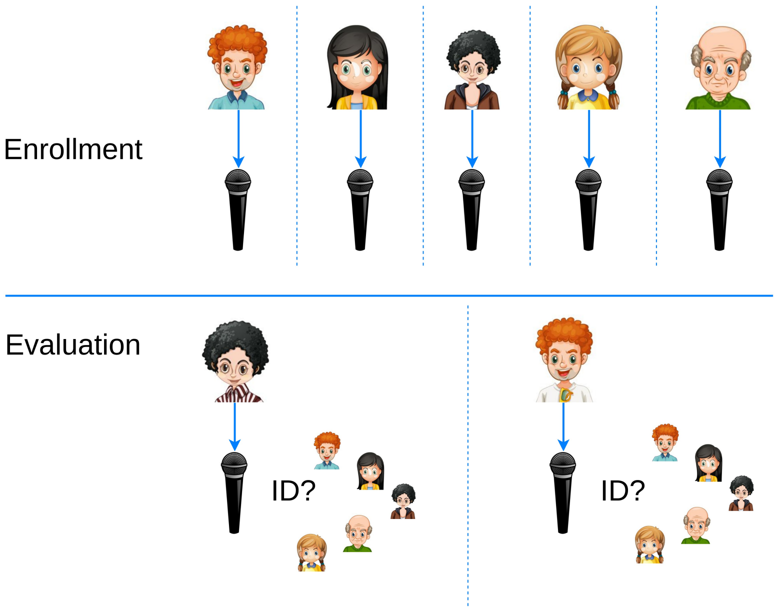
The performance of a BSSS algorithm depends on how well $x_s[n]$ approximates $\hat{x}_s[n]$. The most straightforward and most used evaluation metric is the Signal-to-Distortion Ratio (SDR) [102]. The `bss_eval` toolbox [176] is used to determine the SDR in all BSSS experiments in this text. Other evaluation metrics exists that focus more on speech intelligibility and perceptual evaluation, such as PESQ [144] and STOI [168, 100]. However, it is important to note that these are still determined by algorithms and do not contain human assessors.

1.1.2 Speaker Recognition

In Speaker Recognition (SR), one wants to recognize a speaker from an utterance. There are two stages in an SR problem: the *enrollment* stage and the *test* stage, where the latter is also called the *evaluation* or *inference* stage. In the enrollment phase, a set of I known speakers is built where enrollment data is required for each speaker. In the test phase, a previously unused audio fragment has to be matched to one of the I speakers[1]. An illustration of SR is given in Figure 1.2.

Two types of SR can be distinguished: Speaker Identification (SID) and Speaker Verification (SV). In SID the identity of the speaker in a test utterance is requested, given a set of I enrolled speakers. SID can thus be seen as a classification problem with I classes. In SV one wants to determine whether the utterance belongs to a specific speaker or not. Most SV models output a score, when asked if a test utterance belongs to a specific speaker. If this score exceeds a certain threshold, the trial is classified as positive. Naturally, these two SR tasks are linked and often methods developed for SID can be used for Speaker Verification (SV) (and vice versa).

[1]In some cases, the test fragment can belong to none of the I known speakers and the system is asked to provide an out-of-set trigger. However, this is outside the scope of the thesis.

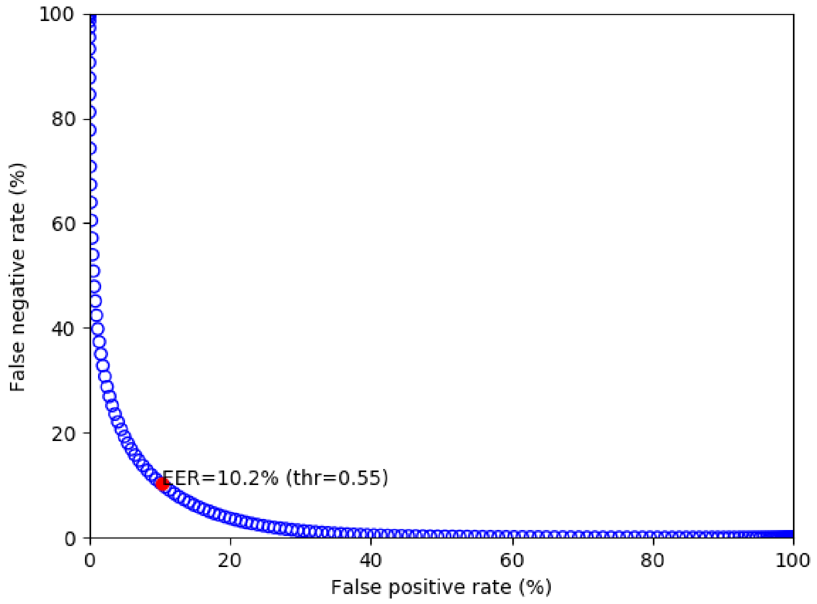


.

Figure 1.2: An illustration of SR for $I = 5$

For many years, the i-vector approach has been the most used approach in the SR field (Section 1.3). Only recently DNN approaches have been outperforming i-vectors (Section 1.4.3). Finally, there have been some attempts to use NMF for SR, although these have been rather limited (Section 1.2.2).

Since SID is a classification problem, the classification accuracy (or speaker recognition accuracy) can be used for system evaluation. For SV, two types of errors can be made: false positives and false negatives, also known as type I and type II errors. By tuning the score threshold for a positive decision, different false positive and false negative rates can be achieved. When a higher threshold is chosen, the false positive rate will decrease and the false negative rate will increase. When determining the false positive and false negative rates for different thresholds, a result as in Figure 1.3 can be achieved. When evaluating a SV system, it can be cumbersome to consider the false negative and false positive rates for many different thresholds. The Equal Error Rate (EER) metric tries to summarize this in a single number. It is defined as the false positive rate for the threshold were the false positive rate equals the false negative rate (see Figure 1.3).



.

Figure 1.3: Illustration of the false positive rate and false negative rate for an SV system. Each blue circle indicates a different threshold. The EER metric can be used to summarize the figure.

1.1.3 Joint BSSS and SR

A BSSS model should be capable to separate a multi-speaker mixture into its source components. On the other hand, an SR model should be able to recognize a previously enrolled speaker. A joint model should be able to do both. It should thus be able to separate the speech signals of S_{enr} previously unknown speakers and learn the speakers. When a new mixture is presented, the speech signals should again be separated and for each of the S_{test} speech signal estimates, a speaker identity should be estimated corresponding to one of the I enrolled speakers. An illustration of joint BSSS and SR is given in Figure 1.4.

It is noted that the terms Blind Speech Source Separation and Speaker Recognition might seem contradictory when used in the same context as *blind* implies there is no prior information on the speakers to separate, while SR

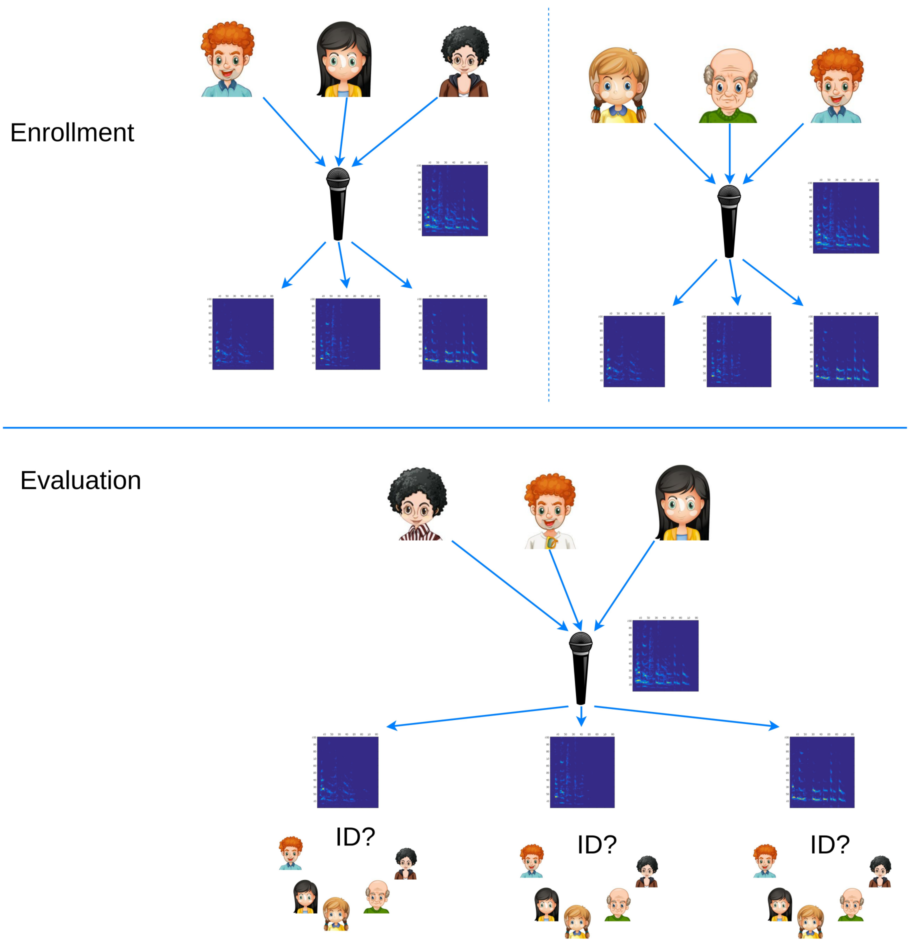


Figure 1.4: An illustration of joint BSSS and SR. $S_{\text{enr}} = S_{\text{test}} = 3$ and $I = 5$.

requires enrollment data for each speaker in the set. To clarify, SSS is only strictly blind when there is overlapping speech during the enrollment phase and it is technically not considered blind during the speaker recognition evaluation phase.

1.2 Nonnegative Matrix Factorization

Non-negative Matrix Factorization (NMF) is a frequently used factorization method to identify patterns in data. It was originally developed to recognize parts-based representation in images [103] and has since been used in various fields such as bioinformatics [63], noise-robust automatic speech recognition [66] and age and gender estimation [10]. It has been used for SR tasks and obtained comparable results to state-of-the-art i-vector based approaches, for small speaker sets [89, 57]. Moreover, NMF has been used in many scenarios of SS and music transcription [35, 179]. Multi-channel extensions of NMF have been developed with applications to BSSS [151, 131]. These approaches combine spatial cues from phase differences between microphones and the segmentation strength from NMF without any prior knowledge of the sources.

In NMF a non-negative matrix $\mathbf{X} \in \mathbb{R}_+^{F\times T}$ is approximated by a factorization using a non-negative dictionary matrix $\mathbf{T} \in \mathbb{R}_+^{F\times K}$ and a non-negative activation matrix $\mathbf{Q} \in \mathbb{R}_+^{K\times T}$, such that

$$\mathbf{X} \approx \hat{\mathbf{X}} \triangleq \mathbf{TQ} \tag{1.12}$$

or

$$\mathbf{x}_t \approx \hat{\mathbf{x}}_t \triangleq \sum_{k=1}^{K} q_{kt}\mathbf{t}_k. \tag{1.13}$$

For example, in speech processing $\mathbf{X} = |\boldsymbol{\mathcal{X}}|.^2$ can be a speech power spectrogram. $\mathbf{X}$ is a matrix with F frequency bins (or features) and T time frames. NMF tries to capture the most frequent patterns of the speech in K F-dimensional basis vectors that form a dictionary $\mathbf{T}$ for the speech. The matrix $\mathbf{Q}$ contains the coefficients of the linear combination and thus indicates how the k^{th} basis vector is activated in the t^{th} time frame. Usually $K < \min(F, T)$ such that NMF is a rank reduction operation. A discrepancy measure is chosen between the original $\mathbf{X}$ and the reconstruction $\hat{\mathbf{X}}$ and can be minimized by finding optimal dictionaries and activations. The Euclidean distance, the Kullback-Leibler (KL) divergence and the Itakura-Saito (IS) divergence are well known measures. In this chapter the IS divergence will be used since it has a multi-channel extension

(see Section 1.2.1)

$$d_{IS}(x_{ft}, \hat{x}_{ft}) = \frac{x_{ft}}{\hat{x}_{ft}} - log\left(\frac{x_{ft}}{\hat{x}_{ft}}\right) - 1. \tag{1.14}$$

To minimize this divergence, multiplicative iterative update formulas with convergence guarantees have been derived [126]

$$t_{fk} \leftarrow t_{fk}\sqrt{\frac{\sum_t \frac{x_{ft}}{\hat{x}_{ft}}\frac{q_{kt}}{\hat{x}_{ft}}}{\sum_t \frac{q_{kt}}{\hat{x}_{ft}}}} \tag{1.15}$$

$$q_{kt} \leftarrow q_{kt}\sqrt{\frac{\sum_f \frac{x_{ft}}{\hat{x}_{ft}}\frac{t_{fk}}{\hat{x}_{ft}}}{\sum_f \frac{t_{fk}}{\hat{x}_{ft}}}}, \tag{1.16}$$

where the sub-indices refer to the corresponding element in the matrix. To avoid scaling ambiguities the columns of $\mathbf{T}$ are to be normalized: $t_{fk} \leftarrow t_{fk}/\sum_{f'} t_{f'k}$[2].

1.2.1 NMF for SSS

NMF has shown good performance in supervised SS problems, both for SE and SSS.

NMF for supervised SSS

In non-blind SSS the speakers that will be present in the mixture are known beforehand. For every speaker i some training data $\mathbf{X}_i^{\text{train}}$ is present and a speaker dependent dictionary $\mathbf{T}_i^{\text{train}}$ can be obtained by iteratively applying (1.15) and (1.16)[3]. The learnt speaker dictionaries will be used during evaluation and hence $\mathbf{T}_i^{\text{train}} = \mathbf{T}_i^{\text{test}} = \mathbf{T}_i$. All dictionaries are then collected in a library $\mathbf{T}_{\text{tot}} = [\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_I]$. During evaluation, a reconstruction $\hat{\mathbf{X}}_s^{\text{test}}$ has to be estimated for every speaker, from a mixture input $\mathbf{X}^{\text{test}}$ (e.g. $\mathbf{X}^{\text{test}} = |\mathcal{Y}^{\text{test}}|.^2$). This can be done by iteratively estimating $\mathbf{Q}_{\text{tot}}^{\text{test}}$ using (1.16) and keeping the dictionary $\mathbf{T}_{\text{tot}}$ fixed. The library activations can then be decomposed to find the speaker dependent dictionary activations $\mathbf{Q}_i^{\text{test}}$. $\hat{\mathbf{X}}$ can then be found using

[2]It has been suggested that this normalization should be included while deriving the update formulas, rather than applying it afterwards [101]. However this is outside the scope of this text.

[3]The activations $\mathbf{Q}_i^{\text{train}}$ will also be obtained but will no longer be of interest.

(1.12) as

$$\hat{x}_{i,ft} = \sum_{k \in \kappa_i} t^{\text{tot}}_{fk} q^{\text{tot,test}}_{kt} = \sum_{k=1}^{K_i} t_{i,kt} q^{\text{test}}_{i,kt} \tag{1.17}$$

where κ_i indicates the dictionary elements of speaker i in the library $\mathbf{T}_{\text{tot}}$ and K_i is the number of dictionary elements in dictionary $\mathbf{T}_i$. The source signal estimates $\hat{x}_i(t, f)$ can be found using Wiener filtering [179] via the mask based approach of (1.10) and

$$\hat{m}_i(t, f) = \frac{\hat{x}_i(t, f)}{\sum_{i'=1}^{I} \hat{x}_{i'}(t, f)}. \tag{1.18}$$

Not all I known speakers need to be active in the test mixture ($S \leqslant I$). In that case ideally $\hat{x}_{i,ft} \approx 0$ for all speakers i not active in the mixture. In fact, in Chapter 2 a novel method will be described that allows to determine the identity of each active speaker s in the test mixture.

The problem with NMF for BSSS

For blind or unsupervised SSS, no library $\mathbf{T}_{\text{tot}}$ can be built prior to separation. Instead $\mathbf{T}^{\text{test}}_{\text{tot}}$ and $\mathbf{Q}^{\text{test}}_{\text{tot}}$ have to be estimated directly from $\mathbf{X}^{\text{test}}$ using (1.15)-(1.16). However, it is unclear how to determine to which speaker s each basis vector k belongs. Therefore, (1.17) cannot be applied.

Multi-channel NMF for BSSS

Usually one resorts to multi-channel NMF techniques in the BSSS case, where Time Difference of Arrival (TDOA) is a cue for Direction of Arrival (DOA) which can be used to assist the source separation.

We again consider (1.8) but leave out the noise term (although this can also be included in the NMF formulation [9]). Sawada et al. proposed a multi-channel IS divergence [151]

$$\begin{aligned} D_{\text{IS}}(\mathbf{X}, \{\mathbf{T}, \mathbf{Q}, \mathbf{H}, \mathbf{C}\}) &= \sum_{f=1}^{F} \sum_{t=1}^{T} d_{\text{IS}}(\mathbf{X}_{ft}, \hat{\mathbf{X}}_{ft}) \\ d_{\text{IS}}(\mathbf{A}, \mathbf{B}) &= \text{tr}(\mathbf{A}\mathbf{B}^{-1}) - \text{logdet}(\mathbf{A}\mathbf{B}^{-1}) + J \end{aligned} \tag{1.19}$$

where tr() is the trace of a matrix, logdet() is the natural logarithm of the determinant of a matrix, $\mathbf{X}_{ft} = \boldsymbol{y}_{ft}\boldsymbol{y}^{\text{H}}_{ft}$ with $\boldsymbol{y}_{ft}$ as a J-dimensional vector as in equation (1.8). $\hat{\mathbf{X}}_{ft}$ is set to be

$$\hat{\mathbf{X}}_{ft} = \sum_{k=1}^{K_{\mathrm{mix}}} \left(\sum_{s=1}^{S} \mathbf{H}_{fs} c_{sk} \right) t_{fk} q_{kt} = \sum_{k=1}^{K_{\mathrm{mix}}} \sum_{s=1}^{S} c_{sk} t_{fk} q_{kt} \mathbf{H}_{fs}. \tag{1.20}$$

The same interpretations are given to t_{fk} and q_{kt} as in single-channel NMF. c_{sk} is a latent speaker-indicator under the constraints $0 \leqslant c_{sk} \leqslant 1$ and $\sum_s c_{sk} = 1$. It can be interpreted that the k^{th} basis vector belongs to speaker s if c_{sk} is close to 1. K_{mix} is the total number of basis vectors, to be partitioned over the S active speakers. $\mathbf{H}_{fs}$ is a $J \times J$ Hermitian positive semi-definite matrix with on its diagonals the estimated power gain of the s^{th} speaker to each microphone, at the f^{th} frequency bin. $\mathbf{H}_{fs}$ is assumed independent of the time t. The off-diagonal elements include the estimated phase differences between microphones and thus contain spatial information of the speaker. In fact, $\mathbf{H}_{fs}$ can be interpreted by using the rank-1 convolutive model [59] as follows

$$\mathbf{H}_{fs} = \hat{\boldsymbol{\hbar}}_{fs} \hat{\boldsymbol{\hbar}}_{fs}^{H} \tag{1.21}$$

with $\hat{\boldsymbol{\hbar}}_{fs}$ an estimate for $\boldsymbol{\hbar}_{fs}$ as in (1.8).

Multiplicative update formulas have been found in [151, eq. (42)-(47)] that minimize the divergence in (1.19)[4]:

$$t_{fk} \leftarrow t_{fk} \sqrt{\frac{\sum_t \sum_s c_{sk} q_{kt} \mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{X}_{ft} \hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs} \right)}{\sum_t \sum_s c_{sk} q_{kt} \mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs} \right)}} \tag{1.22}$$

$$q_{kt} \leftarrow q_{kt} \sqrt{\frac{\sum_f \sum_s c_{sk} t_{fk} \mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{X}_{ft} \hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs} \right)}{\sum_f \sum_s c_{sk} t_{fk} \mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs} \right)}} \tag{1.23}$$

$$c_{sk} \leftarrow c_{sk} \sqrt{\frac{\sum_f \sum_t t_{fk} q_{kt} \mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{X}_{ft} \hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs} \right)}{\sum_f \sum_t t_{fk} q_{kt} \mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs} \right)}} \tag{1.24}$$

To update $\mathbf{H}_{fs}$ an algebraic Riccati equation is solved

$$\mathbf{H}_{fs} \mathbf{A} \mathbf{H}_{fs} = \mathbf{B} \tag{1.25}$$

$$\mathbf{A} = \sum_t \sum_k c_{sk} t_{fk} q_{kt} \hat{\mathbf{X}}_{ft}^{-1} \tag{1.26}$$

[4]A modified algorithm is derived in Appendix A.1

$$\mathbf{B} = \mathbf{H}'_{fs} \left(\sum_t \sum_k c_{sk} t_{fk} q_{kt} \hat{\mathbf{X}}_{ft}^{-1} \mathbf{X}_{ft} \hat{\mathbf{X}}_{ft}^{-1} \right) \mathbf{H}'_{fs} \tag{1.27}$$

where $\mathbf{H}^{'}_{fs}$ is the $\mathbf{H}_{fs}$ of the previous update. To avoid scale ambiguity these normalizations should follow: $\mathbf{H}_{fs} \leftarrow \mathbf{H}_{fs}/\mathrm{tr}(\mathbf{H}_{fs})$, $t_{fk} \leftarrow t_{fk}/\sum_{f'} t_{f'k}$ and $c_{sk} \leftarrow c_{sk}/\sum_{s'} c_{s'k}$.

The separated signals are then, similar to (1.18), obtained through Wiener filtering where the mask is obtained as

$$\hat{\mathbf{m}}_{s,ft} = \left(\sum_{k=1}^{K_{\text{mix}}} c_{sk} t_{fk} q_{kt} \right) \mathbf{H}_{fs} \hat{\mathbf{X}}_{ft}^{-1}, \tag{1.28}$$

which can be used with (1.11) to obtain the source signal estimates $\hat{x}_s(t, f)$.

1.2.2 NMF for Speaker Recognition

When using NMF for SR, every speaker i requires training or enrollment data $\mathbf{X}_i^{\text{train}}$, similarly as in non-blind SSS. $\mathbf{X}_i^{\text{train}}$ is then factorized using (1.15) and (1.16). The obtained dictionaries $\mathbf{T}_i$, for each of the I enrolled speakers, are assumed to be speaker dependent and are collected in the library $\mathbf{T}_{\text{tot}} = [\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_I]$.

During testing the identity of the speaker, in a previously unseen utterance $\mathbf{X}^{\text{test}}$, has to be found. NMF is applied to this utterance with a fixed library $\mathbf{T}_{\text{tot}}$ and the activations $\mathbf{Q}_{\text{tot}}^{\text{test}}$ are found iteratively using (1.16). The activation matrix quantitatively indicates the activation of each basis vector for each enrolled speaker in each time frame. The combined activity of all basis vectors in a speaker's dictionary is a measure of the activity of the enrolled speaker in the test segment. A simple way of estimating the speaker identity is to determine the enrolled speaker for which the sum of the activations, over all its basis vectors and over all the time frames, is maximal. This way of classification can be seen as a per frame speaker activity estimation where the final estimation is a weighted average over all frames, giving more weight to frames with higher activation or more energy

$$\hat{ID} = \arg\max_i \sum_{k \in \mathcal{K}^i} \sum_t^T q_{kt}^{\text{tot}} = \arg\max_i \sum_{k=1}^{K_i} \sum_t^T q_{i,kt}^{\text{tot}}. \tag{1.29}$$

This approach has been applied in [57] and will also be used in Chapter 2. It is possible to include Group Sparsity (GS-NMF) constraints on the activations

$\mathbf{Q}_{\text{tot}}^{\text{test}}$ to enforce solutions where it is unlikely that basis vectors from different speakers are active at the same time frame [104, 87, 25]. However, this is outside the scope of this thesis.

1.3 i-vectors for SR

A speaker representation that is often used for speaker identification tasks is the i-vector [51, 68]. To obtain such an i-vector, first a Universal Background Model (UBM), built using a Gaussian Mixture Model (GMM), is trained on development data. For each utterance, the UBM is adapted, using the data samples of the utterance, as shown in Figure 1.5. A supervector $\mathbf{s}$ is derived for each utterance, using the UBM, by stacking the speaker-adapted Gaussian mean vectors. $\mathbf{s}$ is then represented by an i-vector $\mathbf{w}$ and its projection based on the total variability space,

$$\mathbf{s} \approx \mathbf{m} + \mathbf{T}\mathbf{w}, \tag{1.30}$$

where $\mathbf{m}$ is the UBM mean supervector, $\mathbf{w}$ is the total variability factor or i-vector and $\mathbf{T}$ is a low-rank matrix spanning a subspace with important variability in the mean supervector space and is trained on development data [51, 68].

Linear Discriminant Analysis (LDA) can be used to further reduce the dimension of the i-vector. LDA is a rank reduction operation that tries to minimize intra-class variance and maximizes inter-class variance. Here, each class is represented by a single speaker. LDA tries to find the orthogonal directions $\mathbf{d}$ that optimize the following ratio:

$$J(\mathbf{d}) = \frac{\mathbf{d}^T\mathbf{S}_b\mathbf{d}}{\mathbf{d}^T\mathbf{S}_w\mathbf{d}} \tag{1.31}$$

with $\mathbf{S}_b$ the intra-speaker variance and $\mathbf{S}_w$ the inter-speaker variance. The following Generalized Eigenvalue Decomposition (GEVD) can then be considered

$$\mathbf{S}_b\mathbf{v} = \lambda\mathbf{S}_w\mathbf{v}, \tag{1.32}$$

with $\mathbf{v}$ the eigenvector and λ the eigenvalue. The eigenvectors with the highest eigenvalues are then stored in the projection matrix $\mathbf{A}$ and the LDA i-vectors $\mathbf{w}^*$ are then obtained as follows

$$\mathbf{w}^* = \mathbf{A}^T\mathbf{w}. \tag{1.33}$$

Thus, a vector with inter-speaker discriminating dimensions is found.

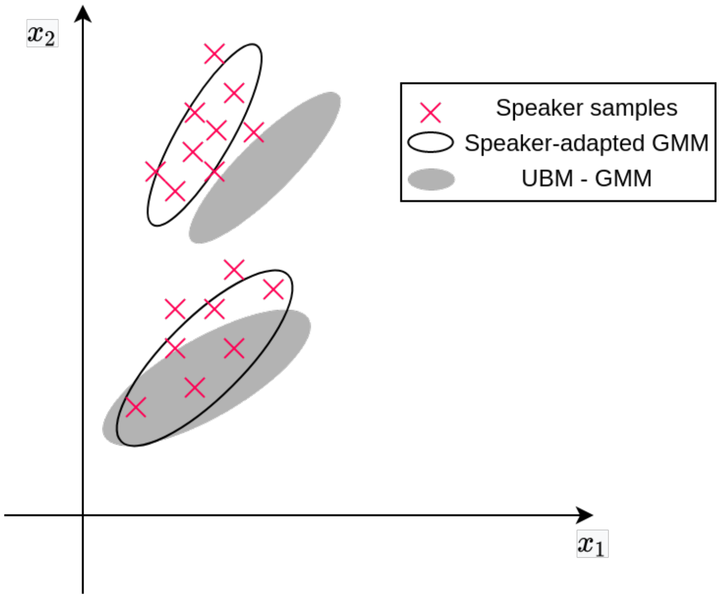


Figure 1.5: Illustration a speaker-adapted GMM. In gray a UBM-GMM with two mixture components is shown. This is UBM is determined on data samples from development data. The speaker samples from an utterance, marked with red crosses, allow for a speaker-adapted GMM, which are shown as black ellipses. Based on [32].

1.4 Deep neural networks

Deep Neural Networks (DNNs) are used in a wide variety of speech and non-speech related tasks and are often considered state-of-the-art in these domains [80]. This is also the case for BSSS and SR, although for the latter i-vector approaches are also still common (see Section 1.3). The training of such a DNN is referred to as Deep Learning (DL). The concepts of Deep Neural Network and Deep Learning are often used interchangeably. Section 1.4.1 gives a general description of DNNs and Sections 1.4.2 and 1.4.3 describe how DNNs can be used for BSSS and SR, respectively.

1.4.1 Background

The most fundamental building block of a DNN was originally described by McCulloch and Pitts in 1943 where it was used as a mathematical model of a biological neuron [115]. It was later on extended by Rosenblatt to what is now called a single-layer perceptron [145]. In its simplest form the perceptron's input-output behaviour can be described as

$$\boldsymbol{\xi}(\mathbf{x}, \Theta) = \mathbf{W}\mathbf{x} + \mathbf{b} \tag{1.34}$$

$$\hat{\mathbf{y}} = f(\boldsymbol{\xi}(\mathbf{x}, \Theta)) \tag{1.35}$$

where $\mathbf{x}$ and $\hat{\mathbf{y}}$ are the input and output of the model, $\Theta = \{\mathbf{W}, \mathbf{b}\}$ are the model's parameters called *weights* and *biases*, respectively, $\boldsymbol{\xi}$ is the activation and $f()$ is a non-linear activation function. Often used non-linearities (which are differentiable) are:

- The sigmoid σ function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{1.36}$$

- The hyperbolic tangent *tanh* function

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{1.37}$$

- The rectified linear unit *ReLU* function

$$\mathrm{ReLU}(x) = max(0, x) \tag{1.38}$$

- The softmax function (which can only be applied to a vector)

$$\mathrm{softmax}(\mathbf{x}) = \frac{e^{\mathbf{x}}}{\sum_i e^{x_i}} \tag{1.39}$$

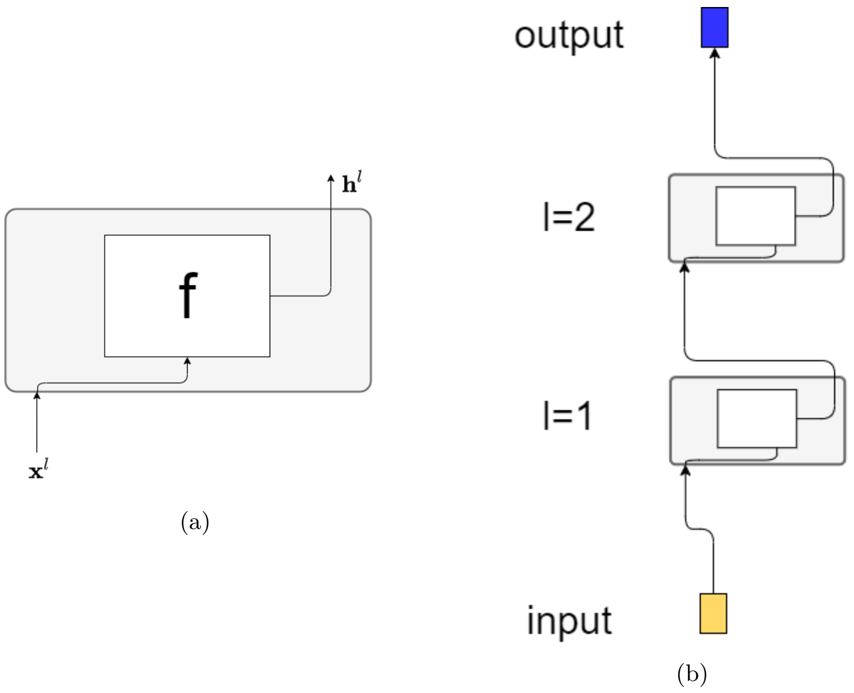


Figure 1.6: a) A block diagram illustration of a single-layer perceptron. b) A block diagram illustration of a 2-layer feedfoward neural network or multi-layer perceptron.

> with x_i the i^{th} element of $\mathbf{x}$. Note that from the above non-linearities, only the softmax function does not have an element-wise behaviour, meaning that its i^{th} output is not only dependent on the i^{th} input.

The output of a perceptron can be used as input to another perceptron to create a two layer perceptron. In fact, it has been shown that such a depth-2 perceptron is a universal approximator. However, it could take an exponential amount of units to achieve a desired accuracy, making it a very wide network [111]. Typically, multiple layers are stacked to create a multi-layer perceptron or a feedforward neural network.

The feedforward neural network or multi-layer perceptron is a stacking of multiple single layer perceptrons, where the input of layer l is set to be the output of layer $l-1$ below. Such a network with L layers can be described as

$$\mathbf{h}^l = f^l(\boldsymbol{\xi}^l) = f^l(\mathbf{W}^l\mathbf{x}^l + \mathbf{b}^l) \quad \text{for} \quad l = 1 \dots L \tag{1.40}$$

$$\mathbf{x}^l = \mathbf{h}^{l-1} \quad \text{for} \quad l = 2 \dots L \tag{1.41}$$

with $\mathbf{x}^l$ the input of layer l, $\mathbf{h}^l$ the output or hidden units of layer l, $\mathbf{W}^l$ the weights connecting the hidden units $\mathbf{h}^{l-1}$ with $\mathbf{h}^l$, $\mathbf{b}^l$ the bias for layer l, $\boldsymbol{\xi}^l$ the activations of layer l and $f^l()$ the non-linearity of layer l (although this is typically chosen to be the same for all layers, hence $f^l() = f()$). A block diagram illustration of the feedforward neural network can be seen in Figure 1.6. Finally, we set

$$\mathbf{x}^1 = \mathbf{x} \tag{1.42}$$

$$\hat{\mathbf{y}} = \mathbf{h}^L \tag{1.43}$$

as the input and output of the model.

The goal (for supervised learning) is to find the model parameters ($\mathbf{W}^l$ and $\mathbf{b}^l$) such that the model has a desired input-output behaviour. For this, N labeled training examples ($\mathbf{x}_n^{\text{train}}$, $\mathbf{y}_n^{\text{train}}$ for $n = 1 \dots N$) are needed as well as a differentiable loss function $\mathcal{L}_n^{\text{train}}(\mathbf{y}_n^{\text{train}}, \hat{\mathbf{y}}_n^{\text{train}}(\mathbf{x}_n^{\text{train}}, \Theta))$ that scores how well the output $\hat{\mathbf{y}}_n^{\text{train}}$ matches the targets $\mathbf{y}_n^{\text{train}}$ for a given input $\mathbf{x}_n^{\text{train}}$ and set of parameters Θ. The loss function over the complete data set is then

$$\mathcal{L}_\Theta^{\text{train}} = \frac{1}{N}\sum_n^N \mathcal{L}_n^{\text{train}}(\mathbf{y}_n^{\text{train}}, \hat{\mathbf{y}}_n^{\text{train}}(\mathbf{x}_n^{\text{train}}, \Theta)). \tag{1.44}$$

In the remainder of this section we will omit the superscript *train* for ease of notation. An example of such a loss function is the Mean Square Error (MSE) loss:

$$\mathcal{L}_\Theta = \frac{1}{N}\sum_n^N \|\hat{\mathbf{y}}_n(\mathbf{x}_n, \Theta) - \mathbf{y}_n\|^2. \tag{1.45}$$

Since (1.40) and the loss function $\mathcal{L}_\Theta$ were chosen to be differentiable, the gradient $\partial\mathcal{L}_\Theta/\partial\theta$ for each parameter θ in the model with respect to the loss function $\mathcal{L}_\Theta$ can be found and thus standard optimization techniques can be used. To determine this gradient $\partial\mathcal{L}_\Theta/\partial\theta$, the backpropagation algorithm is used [75]. The backpropagation starts by determining the gradient of the output with respect to the loss function and then works its way down to the input. The gradient $\partial\mathcal{L}_\Theta/\partial\hat{\mathbf{y}} = \partial\mathcal{L}_\Theta/\partial\mathbf{h}^L$ of the output of the network with respect to the loss function is usually straightforward. For example, using the MSE loss of (1.45) would give

$$\frac{\partial\mathcal{L}_\Theta}{\partial\hat{\mathbf{y}}} = \frac{2}{N}\sum_n^N (\hat{\mathbf{y}}_n(\mathbf{x}_n, \Theta) - \mathbf{y}_n). \tag{1.46}$$

The gradient of the loss function $\mathcal{L}_\Theta$ with respect to the activation of the last layer $\boldsymbol{\xi}^L$ is then

$$\frac{\partial\mathcal{L}_\Theta}{\partial\boldsymbol{\xi}^L} = \frac{\partial\mathcal{L}_\Theta}{\partial\hat{\mathbf{y}}}\frac{\partial\hat{\mathbf{y}}}{\partial\boldsymbol{\xi}^L} = \frac{\partial\mathcal{L}_\Theta}{\partial\hat{\mathbf{y}}}\text{diag}\left[f'(\boldsymbol{\xi}^L)\right], \tag{1.47}$$

with f' the derivative of the activation function, diag $[\mathbf{a}]$ a diagonal matrix with $\mathbf{a}$ as diagonal entries. In the last step (1.40) was used and an element-wise non-linearity, such as (1.36)-(1.38), was assumed. The gradient of the loss function with respect to the activations of the penultimate layer $L-1$ can be found as

$$\frac{\partial \mathcal{L}_\Theta}{\partial \boldsymbol{\xi}^{L-1}} = \frac{\partial \mathcal{L}_\Theta}{\partial \boldsymbol{\xi}^{L}} \frac{\partial \boldsymbol{\xi}^{L}}{\partial \boldsymbol{\xi}^{L-1}} = \frac{\partial \mathcal{L}_\Theta}{\partial \boldsymbol{\xi}^{L}} \frac{\partial \boldsymbol{\xi}^{L}}{\partial \mathbf{h}^{L-1}} \frac{\partial \mathbf{h}^{L-1}}{\partial \boldsymbol{\xi}^{L-1}} = \frac{\partial \mathcal{L}_\Theta}{\partial \boldsymbol{\xi}^{L}} \mathbf{W}^L \mathrm{diag}\left[f'(\boldsymbol{\xi}^{L-1})\right]. \tag{1.48}$$

where (1.40) and (1.41) were used in the last step. When applying this iteratively, the gradient of the loss function with respect to the activation of layer l can be found as

$$\frac{\partial \mathcal{L}_\Theta}{\partial \boldsymbol{\xi}^{l}} = \frac{\partial \mathcal{L}_\Theta}{\partial \boldsymbol{\xi}^{L}} \prod_{l'=l}^{L-1} \left(\frac{\partial \boldsymbol{\xi}^{l'+1}}{\partial \boldsymbol{\xi}^{l'}} \right) = \frac{\partial \mathcal{L}_\Theta}{\partial \boldsymbol{\xi}^{L}} \prod_{l'=l}^{L-1} \left(\mathbf{W}^{l'+1} \mathrm{diag}\left[f'(\boldsymbol{\xi}^{l'})\right]\right). \tag{1.49}$$

The gradients of the weights and biases of the layer l with respect to the loss function can then be found as

$$\frac{\partial \mathcal{L}_\Theta}{\mathbf{W}^{l}} = \frac{\partial \mathcal{L}_\Theta}{\partial \boldsymbol{\xi}^{l}} \frac{\partial \boldsymbol{\xi}^{l}}{\partial \mathbf{W}^{l}} = \frac{\partial \mathcal{L}_\Theta}{\partial \boldsymbol{\xi}^{l}} \mathbf{h}^{l-1} \tag{1.50}$$

$$\frac{\partial \mathcal{L}_\Theta}{\partial \mathbf{b}^{l}} = \frac{\partial \mathcal{L}_\Theta}{\partial \boldsymbol{\xi}^{l}} \frac{\partial \boldsymbol{\xi}^{l}}{\partial \mathbf{b}^{l}} = \frac{\partial \mathcal{L}_\Theta}{\partial \boldsymbol{\xi}^{l}}, \tag{1.51}$$

where (1.40) was used in the last step of (1.50) and (1.51).

Now that the gradients of the loss function with respect to all parameters have been found, optimization algorithms can be used to search for the optimal parameters. The most straightforward optimization algorithm is the iterative gradient descent where the following iterative update formula is used for a parameter θ until some stopping or convergence criterion is met.

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}_\Theta}{\partial \theta} \tag{1.52}$$

where the step size η is arbitrarily chosen and θ is randomly initialized at the start. However, for very large data sets determining $\partial \mathcal{L}_\Theta / \partial \theta$ can become computationally infeasible. Stochastic Gradient Descent (SGD) splits the training set into B mini-batches with N_b examples each. Every optimization iteration calculates gradients with respect to the loss of a single mini batch $\mathcal{L}_b$

$$\theta \leftarrow \theta - \eta \frac{\partial \mathcal{L}_{b,\Theta}}{\partial \theta} \tag{1.53}$$

$$b \leftarrow b + 1. \tag{1.54}$$

When b exceeds B, it is reset to 1. Since $\mathcal{L}_b$ is only an approximation for $\mathcal{L}$ (for a well balanced mini-batch), so is (1.53) for (1.52). Therefore, the number of steps before convergence is typically larger for SGD, but each step is much cheaper to evaluate. SGD has some additional advantages such as better generalization to unseen examples and avoiding local minima [70]. In this text mini-batching will always be used, but for ease of notation $\mathcal{L}_{b,\Theta}$ is written as $\mathcal{L}_\Theta$ in the remainder of the text.

There are many optimization algorithms that address some of the problems of SGD [146]. One of these is *Adam* [93], which will be used in all DL experiments in this text. *Adam*, short for *adaptive momentum estimates*, estimates first and second order momentum terms of the gradient, to counter the oscillation behaviour of SGD [141]. The *Adam* algorithm is defined as follows

$$m_{\theta,1} \leftarrow \beta_1 m_{\theta,1} + (1-\beta_1)\frac{\partial \mathcal{L}_\Theta}{\partial \theta} \tag{1.55}$$

$$m_{\theta,2} \leftarrow \beta_2 m_{\theta,2} + (1-\beta_2)\left(\frac{\partial \mathcal{L}_\Theta}{\partial \theta}\right)^2 \tag{1.56}$$

$$\hat{m}_{\theta,1} \leftarrow \frac{m_{\theta,1}}{\sqrt{1-\beta_1}} \tag{1.57}$$

$$\hat{m}_{\theta,2} \leftarrow \frac{m_{\theta,2}}{\sqrt{1-\beta_2}} \tag{1.58}$$

$$\theta \leftarrow \theta - \frac{\eta}{\sqrt{\hat{m}_{\theta,2}} + \epsilon}\hat{m}_{\theta,1}. \tag{1.59}$$

where β_1 and β_2 are arbitrarily chosen, ϵ a small number to avoid division by zero and $m_{\theta,1}$ and $m_{\theta,2}$ are set to 0 at the start. An additional advantage of *Adam* is that step updates are invariant to the scaling of the loss (i.e. the parameter updates remain unchanged when changing the loss function from $\mathcal{L}_{b,\Theta}$ to $\alpha\mathcal{L}_{b,\Theta}$). This property will be used in Section 4.1.2 when discussing multi-task learning.

Recurrent neural networks

The feedforward neural network assumes input of a fixed size and is therefore not suited for a sequential input $\mathbf{X} = [\mathbf{x}_0, \dots, \mathbf{x}_t, \dots, \mathbf{x}_{T-1}]$, where examples have varying length T. Even if all sequences in the data set are equally long, the number of parameters of the model still scales with input length T and quickly causes overfitting or computational issues. That is why RNNs where introduced. In RNNs the hidden units are not only dependent on the hidden units of the layer below, but also on the hidden units of the previous time step.

RNNs give state-of-the-art performance in many speech [11, 97] and non-speech [165, 157] related tasks.

There are many different type of RNN *cells*. For the standard RNN cell (1.40) is expanded to

$$\mathbf{h}_t^l = f^l(\boldsymbol{\xi}_t^l) = f^l(\mathbf{W}^l\mathbf{x}_t^l + \mathbf{R}^l\mathbf{h}_{t-1}^l + \mathbf{b}^l) \quad \text{for} \quad l = 1 \dots L, \tag{1.60}$$

with $\mathbf{R}$ the weights for the recurrent connection. (1.41)–(1.43) are kept, but we add the time subindex for clarity

$$\mathbf{x}_t^l = \mathbf{h}_t^{l-1} \quad \text{for} \quad l = 2 \dots L \tag{1.61}$$

$$\mathbf{x}_t^1 = \mathbf{x}_t \tag{1.62}$$

$$\hat{\mathbf{y}}_t = \mathbf{h}_t^L. \tag{1.63}$$

The parameters $\Theta = \{\mathbf{W}, \mathbf{R}, \mathbf{b}\}$ are independent of the time t. They are thus shared over all time steps and the network size is independent of the sequence length. The block diagram of an RNN can be seen in Figure 1.7. The recurrent connection $(\mathbf{R}^l\mathbf{h}_{t-1}^l)$ allows to use information from the previous time step, which in turn is dependent on the time step before. This, theoretically, allows the RNN to use any information from the past.

It is possible to also use information from the following time step by using a backward (time) RNN. The same equation as (1.60) is used but the $t-1$ subscript is changed to $t+1$. In a backward RNN the input is processed from end to start and is thus only feasible for non real-time applications or applications where a delay is allowed. A bidirectional RNN uses both forward and backward layers. We use $\overrightarrow{\mathbf{h}}_t^l$ and $\overleftarrow{\mathbf{h}}_t^l$ to denote the hidden units of the forward and backward layer, respectively. There are two ways to combine the outputs from the forward and backward direction: either after every layer or only after the last layer. If the latter is chosen (1.41) becomes

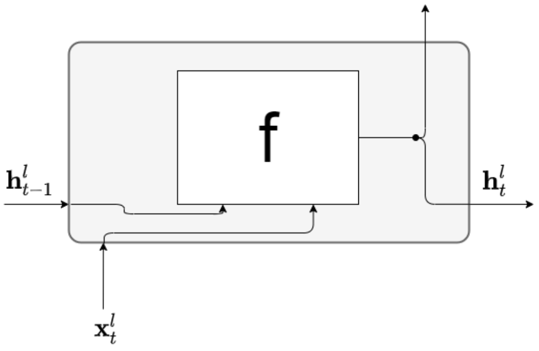
$$\overrightarrow{\mathbf{x}}_t^l = \overrightarrow{\mathbf{h}}_t^{l-1} \qquad \overleftarrow{\mathbf{x}}_t^l = \overleftarrow{\mathbf{h}}_t^{l-1}. \tag{1.64}$$

If inputs are instead combined after every layer we get

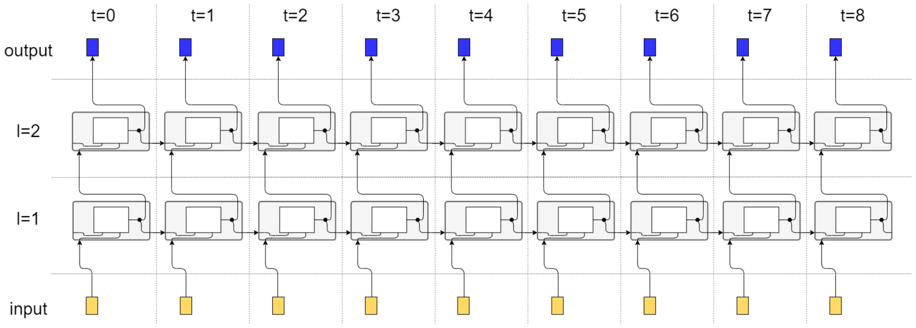
$$\overrightarrow{\mathbf{x}}_t^l = \begin{bmatrix} \overrightarrow{\mathbf{h}}_t^{l-1} \\ \overleftarrow{\mathbf{h}}_t^{l-1} \end{bmatrix} \qquad \overleftarrow{\mathbf{x}}_t^l = \begin{bmatrix} \overrightarrow{\mathbf{h}}_t^{l-1} \\ \overleftarrow{\mathbf{h}}_t^{l-1} \end{bmatrix}. \tag{1.65}$$

A block diagram of the latter can be found in Figure 1.8. In our experiments it was found that it is better to combine outputs after every layer, even if the total number of trainable parameters is kept unchanged. In both cases, for the final output of the network, (1.63) becomes

$$\hat{\mathbf{y}}_t = \begin{bmatrix} \overrightarrow{\mathbf{h}}_t^{L} \\ \overleftarrow{\mathbf{h}}_t^{L} \end{bmatrix}. \tag{1.66}$$



(a)



(b)

Figure 1.7: a) A block diagram illustration of a standard RNN cell. b) A block diagram illustration of a 2-layer RNN for a sequence of $T = 9$.

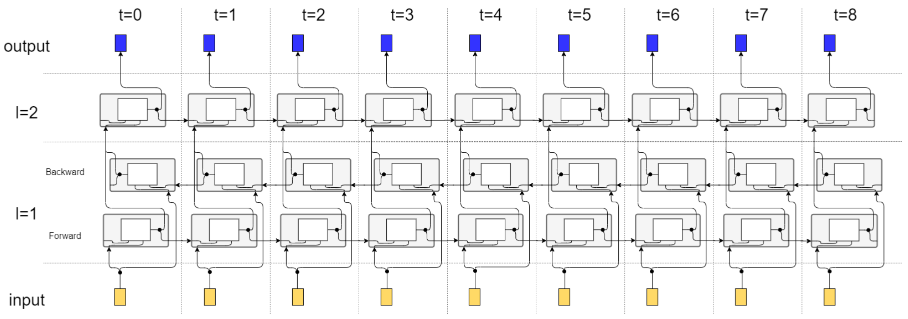


Figure 1.8: A block diagram illustration of a 2-layer bi-directional RNN for a sequence of $T = 9$. To prevent cluttering of the image, only the forward direction of the second layer is shown.

Gradient problems

The training of the RNN is done similar to a feedforward neural network. However there is an additional backpropagation requirement over the recurrent step, called Backpropagation Through Time (BPTT) [185]. It was already mentioned that RNNs are theoretically capable to remember any input from the past and use this information as it sees fit. However, it is found that their memory time span (the duration for which information is kept in memory) is actually relatively short. This is caused by vanishing and exploding gradients, which occur during BPTT [15, 124, 81], a problem also observed in very deep networks (see (1.49)). This problem can be illustrated by considering the derivative $\partial\boldsymbol{\xi}^l_t/\partial\boldsymbol{\xi}^l_{t-\tau}$ of a forward time RNN, which is required for determining the gradients of the parameters.

$$\frac{\partial\boldsymbol{\xi}^l_t}{\partial\boldsymbol{\xi}^l_{t-\tau}} = \prod_{t'=t-\tau+1}^{t} \frac{\partial\boldsymbol{\xi}^l_{t'}}{\partial\boldsymbol{\xi}^l_{t'-1}} = \prod_{t'=t-\tau+1}^{t} \mathbf{R}^l \mathrm{diag}\left[f'^l\left(\boldsymbol{\xi}^l_{t'-1}\right)\right] \tag{1.67}$$

If eigendecomposition is applied to the Jacobian $\frac{\partial\boldsymbol{\xi}^l_{t'}}{\partial\boldsymbol{\xi}^l_{t'-1}}$ and the most dominant eigenvalue $\lambda_{1,t'}$ is retained, two special cases can be considered:

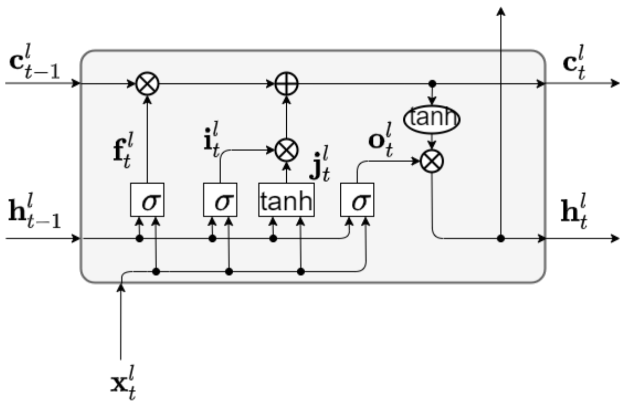
1. $\lambda_{1,t'} < 1 \quad \forall t'$: $\partial\boldsymbol{\xi}^l_{i,t}/\partial\boldsymbol{\xi}^l_{t-\tau}$ will quickly go towards 0, making it difficult to learn long-term dependencies. This problem is called vanishing gradients.
2. $\lambda_{1,t'} > 1 \quad \forall t'$: $\partial\boldsymbol{\xi}^l_t/\partial\boldsymbol{\xi}^l_{t-\tau}$ will quickly diverge, causing oscillations in parameter updates and making it difficult to converge to a satisfactory optimum. This problem is called exploding gradients.

Long short-term memory

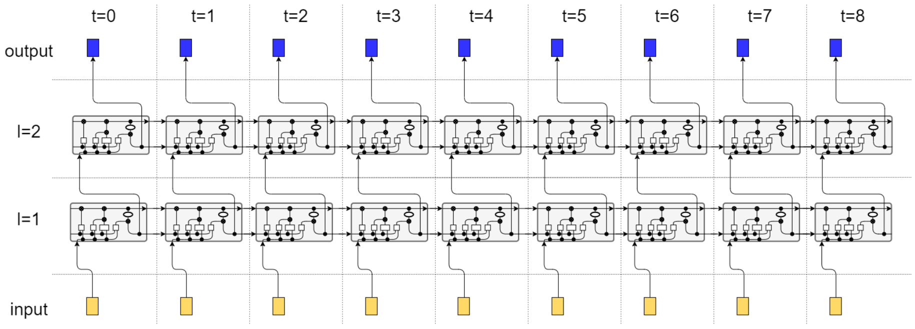
The long short-term memory RNN (LSTM-RNN or LSTM for short), was developed specifically to counter these gradient problems. The LSTM cell introduces the concept of a cell state $\mathbf{c}_t$. The output $\mathbf{h}_t$ of the cell is determined from the cell's state via the output gate $\mathbf{o}_t$. At each time step novel information $\mathbf{j}_t$ can be added to this cell state. The amount of this new information that is added to the cell state is controlled by the input gate $\mathbf{i}_t$. At the same time old cell state information that is no longer deemed relevant can be forgotten via the forget gate $\mathbf{f}_t$. The LSTM cell is defined in (1.68)–(1.73), replacing (1.60).

$$\mathbf{f}^l_t = \sigma(\mathbf{W}^l_f\mathbf{x}^l_t + \mathbf{R}^l_f\mathbf{h}^l_{t-1} + \mathbf{b}^l_f), \tag{1.68}$$

$$\mathbf{i}^l_t = \sigma(\mathbf{W}^l_i\mathbf{x}^l_t + \mathbf{R}^l_i\mathbf{h}^l_{t-1} + \mathbf{b}^l_i), \tag{1.69}$$



(a)



(b)

Figure 1.9: a) A block diagram illustration of a LSTM cell. Based on [36]. b) A block diagram illustration of a 2-layer LSTM-RNN for a sequence of $T = 9$.

$$\mathbf{o}_t^l = \sigma(\mathbf{W}_o^l\mathbf{x}_t^l + \mathbf{R}_o^l\mathbf{h}_{t-1}^l + \mathbf{b}_o^l), \tag{1.70}$$

$$\mathbf{j}_t^l = \tanh(\mathbf{W}_j^l\mathbf{x}_t^l + \mathbf{R}_j^l\mathbf{h}_{t-1}^l + \mathbf{b}_j^l), \tag{1.71}$$

$$\mathbf{c}_t^l = \mathbf{c}_{t-1}^l \odot \mathbf{f}_t^l + \mathbf{j}_t^l \odot \mathbf{i}_t^l, \tag{1.72}$$

$$\mathbf{h}_t^l = \tanh(\mathbf{c}_t^l) \odot \mathbf{o}_t^l. \tag{1.73}$$

(1.61)–(1.66) are kept. The block diagram of an LSTM-RNN can be seen in Figure 1.9.

Constant error carousel

To counter the exploding and vanishing gradients encountered in standard RNNs, the LSTM-RNN uses the principle of the *constant error carousel*[82]. To illustrate this $\mathbf{i}_t^l$, $\mathbf{j}_t^l$, $\mathbf{o}_t^l$ and $\mathbf{f}_t^l$ are assumed independent of $\mathbf{c}_{t-1}^l$. Then, similarly to (1.67), the following gradient is considered

$$\frac{\partial \mathbf{c}_t^l}{\partial \mathbf{c}_{t-\tau}^l} = \prod_{t'=t-\tau+1}^{t} \left(\frac{\partial \mathbf{c}_{t'}^l}{\partial \mathbf{c}_{t'-1}^l} \right) = \prod_{t'=t-\tau+1}^{t} \operatorname{diag}\left[\mathbf{f}_{t'}^l\right] \tag{1.74}$$

If $\mathbf{f}_{t'}^l$ would be set to $\mathbf{1}$ the gradient would further simplify to $\mathbf{I}$, hence the name *constant error*. In this case the problem of vanishing or exploding gradients is avoided. Typically the bias $\mathbf{b}_f^l$ is initalized to $\mathbf{1}$ to encourage a $\mathbf{f}_{t'}^l$ close to $\mathbf{1}$ (since a sigmoid activation is used, $\mathbf{0} \leqslant \mathbf{f}_{t'}^l \leqslant \mathbf{1}$).

Although some major assumptions were made to show the principle of the *constant error carousel*, experimental analysis has shown that LSTM-RNNs are able to use information from the far past, much better than standard RNNs do. In fact, for theoretical experiments the LSTM-RNN is able to remember input seen over a thousand time steps ago [82]. Chapter 5 will give an in-depth analysis of the LSTM memory for BSSS.

1.4.2 DNNs for BSSS

In recent years there have been many studies in DL approaches for SS. Examples in SE are [188, 184, 182, 201, 3] and for SSS there have been studies for male - female SSS [84] and speaker dependent SSS [85]. All these works handle an inter-class separation problem since distinctive classes can be defined: speech and noise; male and female; Alex and Bob. In blind or unsupervised (gender independent) SSS, no assumptions on the sources can been made and only a single class can be defined, namely a speaker class. This makes BSSS an intra-class separation problem, which is intrinsically harder than the other problems discussed. While beamforming approaches using multi-channel data can be very useful for this problem, it also imposes hardware requirements on the recording system. To be as general as possible, this section handles single microphone problems.

To use DNNs for BSSS, a differentiable loss function can be defined to assess the quality of the speech estimates

$$\mathcal{L} = \sum_{s=1}^{S} \sum_{t,f} \mathcal{D}(|\hat{x}_s(t,f)|, |x_s(t,f)|), \tag{1.75}$$

with $\mathcal{D}$ some discrepancy measure. In Section 1.1.1 it was discussed that typically only the modulus of $x_s(t,f)$ is estimated (and the phase of the input signal is retained). This is reflected in (1.75). However, phase-aware loss functions exist as well [123]. Instead of directly estimating $\hat{x}_s(t,f)$, one can also estimate masks $\hat{m}_s(t,f)$ which are then used to obtain $\hat{x}_s(t,f)$ via (1.10), a technique often used in SSS methods. When using the Frobenius norm for $\mathcal{D}$ and combining (1.75) and (1.10) the following loss function is obtained

$$\mathcal{L} = \sum_{s=1}^{S} \sum_{t,f} (\hat{m}_s(t,f)|y(t,f)| - |x_s(t,f)|)^2. \tag{1.76}$$

However, since an intra-class separation task is executed and no prior information on the speakers is assumed to be known, there is no guarantee that the network's assignment of speakers is consistent with the speaker labels of the targets. This is referred to as the label ambiguity or permutation problem [78]. To cope with this ambiguity, a loss function has to be defined that is independent of the order of the target speakers (i.e. the loss function does not change when the ordering of the speaker labels is changed). During evaluation time the same permutation problem occurs and should be taken into account.

Different approaches to solve this BSSS problem have been proposed, all of which try to cope with the permutation problem. In Deep Clustering (DC), every time-frequency bin of a mixture's spectrogram is mapped to an embedding vector and these are then clustered per speaker using K-means [78, 88]. Another approach is to directly estimate each source signal via masks estimates and use utterance Permutation Invariant Training (uPIT) to cope with the permutation problem [97]. Deep Attractor Networks (DANet) can be seen as a combination of DC and uPIT as they, similar to DC, estimate embeddings, but the clustering is done in the netwerk itself via attractors, so that masks can be estimated directly [30], as is done in uPIT.

Deep Clustering

In DC, a D-dimensional embedding vector $\mathbf{v}_{tf}$ is constructed for every time-frequency bin as $\mathbf{v}_{tf} = g_{tf}(|\mathcal{Y}|)$, where $\mathbf{v}_{tf}$ has unit length and g_{tf} is typically a DNN. A $(TF \times D)$-dimensional matrix $\mathbf{V}$ is then constructed from these embedding vectors. Furthermore, a $(TF \times S)$-dimensional target matrix $\mathbf{Z}$ is defined. If target speaker s is the dominant speaker for bin (t,f), then $z_{s,tf} = 1$, otherwise $z_{s,tf} = 0$. Speaker s is dominant in a bin (t,f) if $s = \arg\max_{s'} |\mathcal{X}_{s'}(t,f)|$. An illustration of DC is given in Figure 1.10. A permutation invariant loss

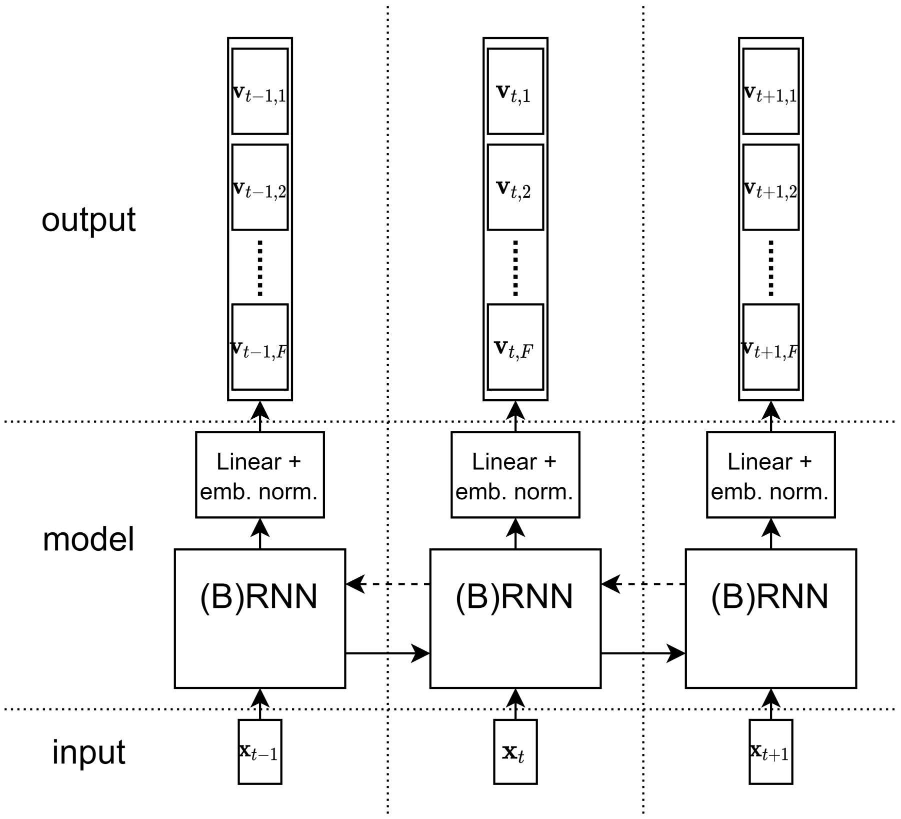


Figure 1.10: An illustration of DC when using a (B)RNN with a linear output layer.

function is then defined as

$$\mathcal{L} = \|\mathbf{V}\mathbf{V}^T - \mathbf{Z}\mathbf{Z}^T\|_F^2 = \sum_{t_1,f_1,t_2,f_2} (\mathbf{v}_{t_1 f_1} \cdot \mathbf{v}_{t_2 f_2} - \mathbf{z}_{t_1 f_1} \cdot \mathbf{z}_{t_2 f_2})^2, \tag{1.77}$$

where $\cdot$ is the inner product or dot product operation and $\|.\|_F^2$ is the squared Frobenius norm. Since $\mathbf{z}_{tf}$ is a one-hot vector,

$$\mathbf{z}_{t_1 f_1} \cdot \mathbf{z}_{t_2 f_2} = \begin{cases} 1, & \text{if } \mathbf{z}_{t_1 f_1} = \mathbf{z}_{t_2 f_2} \\ 0, & \text{otherwise} \end{cases}. \tag{1.78}$$

The ideal angle $\phi_{t_1f_1,t_2f_2}$ between the normalized vectors $\mathbf{v}_{t_1f_1}$ and $\mathbf{v}_{t_2f_2}$ is thus

$$\phi_{t_1f_1,t_2f_2} = \begin{cases} 0, & \text{if } \mathbf{z}_{t_1f_1} = \mathbf{z}_{t_2f_2} \\ \pi/2, & \text{otherwise} \end{cases} . \tag{1.79}$$

At this stage a model can be trained to minimize (1.77), but it does not produce the signal masks $\hat{m}_s(t,f)$ directly. After estimating $\mathbf{V}$, all embedding vectors are clustered into S clusters using K-means. The mask estimates are then constructed as follows

$$\hat{m}_s(t,f) = \begin{cases} 1, & \text{if } \mathbf{v}_{tf} \in c_s \\ 0, & \text{otherwise} \end{cases} , \tag{1.80}$$

with c_s a cluster from K-means. (1.10) can then be used to estimate $\hat{x}_s(t,f)$.

In practice, it is found that speech spectra are sparse [110] and therefore often no speaker is active in a certain time-frequency bin. This is not properly represented in the label $\mathbf{z}_{tf}$. Therefore, in (1.77) only the time-frequency bins were y_{tf} exceeds an arbitrary threshold are considered. Similarly for K-means, only the bins with sufficient energy are considered to find the cluster centers. However, once the cluster centers have been found, the low energy bins can still be assigned to a cluster for (1.80).

The network architecture is independent of the number of speakers present in the mixture (the number of output nodes is dependent on the embedding dimension D, which is independent of S) and can thus be used for a mixture of any number of speakers. However, for the K-means clustering, the number of clusters (or speakers) has to be known. This property will be used in Section 6.1.

Utterance-level Permutation Invariant Training

uPIT has a loss function directly based on (1.76), but it has been adjusted to cope with the label ambiguity. The loss is defined as

$$\mathcal{L} = \min_{p \in \mathcal{P}_S} \sum_{s=1}^{S} \sum_{t,f} (\hat{m}_s(t,f)|y(t,f)| - |x_s(t,f)|)^2 \tag{1.81}$$

with $\mathcal{P}_S$ the set of all possible permutations for S members. An illustration of uPIT is given in Figure 1.11.

Since every mask requires its own output nodes, the network is dependent on the number of speakers. The number of permutations to check in equation

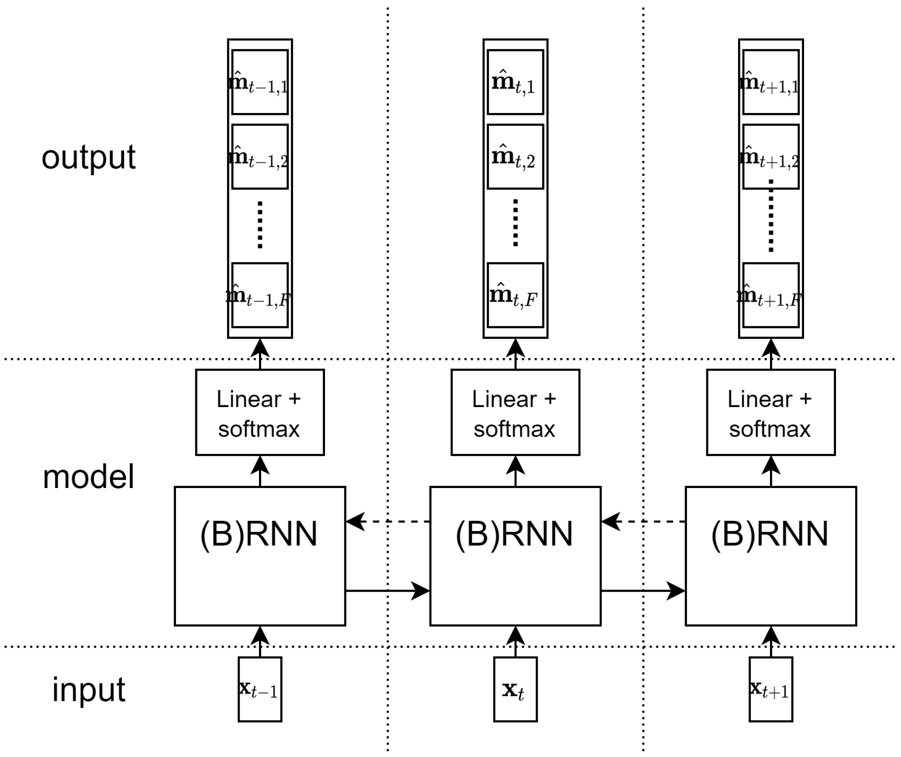


Figure 1.11: An illustration of uPIT when using a (B)RNN.

(1.81) is $S!$, but can be implemented to have a computational complexity of S^2. This could still give computational problems for large S, but that seems unrealistic for the task of BSSS.

Deep Attractor Networks

DANets also use (1.76). At train time the label ambiguity problem for DANets is solved by using oracle information to cluster the embeddings per dominant speaker. The embeddings are again estimated by a DNN as was done in DC. The mask of the corresponding speaker is then used to estimate the reconstructions. The cluster centers, called attractors $\mathbf{a}$, are the average of the embeddings of

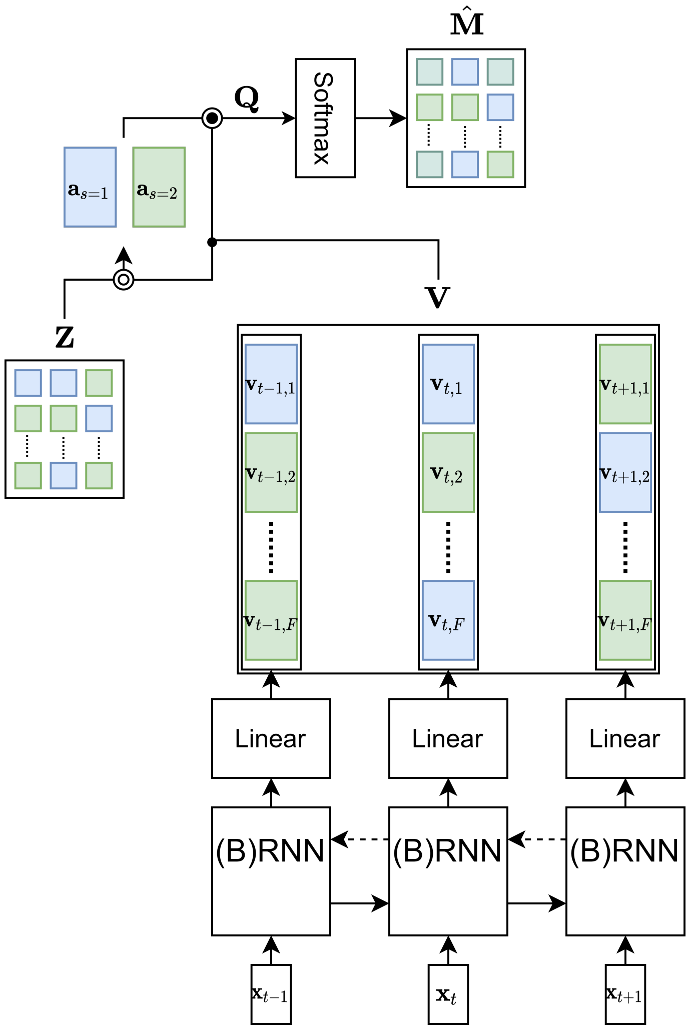


Figure 1.12: An illustration of DANet when using a (B)RNN for $S = 2$ and using softmax for mask estimation. $\mathbf{a}_{s=1}$ and $\mathbf{a}_{s=2}$ are determined via (1.82) and $\mathbf{Q}$ is determined via (1.83)

the time-frequency bins dominated by a particular speaker:

$$\mathbf{a}_s = \frac{\sum_{t,f} \mathbf{v}_{tf} z_{s,tf}}{\sum_{t,f} z_{s,tf}}. \tag{1.82}$$

An embedding closer to a speaker's attractor is considered more likely to be dominated by this speaker. The intermediate variable $q_{s,tf}$ is defined as the dot product or cosine similarity of attractor $\mathbf{a}_s$ and embedding $\mathbf{v}_{tf}$:

$$q_{s,tf} = \mathbf{a}_s \cdot \mathbf{v}_{tf} \tag{1.83}$$

The masks for (1.76) are based on this $q_{s,tf}$ and a non-linearity such that its range is between 0 and 1. For the non-linearity, usually the softmax function of (1.39) is taken, such that the masks for each time-frequency bin sum to 1. An alternative non-linearity is the sigmoid function of (1.36). An illustration of DANet is given in Figure 1.12.

By minimising loss function (1.75), using the masks derived from $q_{s,tf}$, the network learns to form an attraction point $\mathbf{a}_s$ in the embedding space for each speaker, that attracts embedding vectors $\mathbf{v}_{tf}$ associated with this source.

At evaluation time, the labels $z_{s,tf}$ are not available and thus (1.82) cannot be found directly. Instead, the embeddings $\mathbf{V}$ are estimated from the trained network and are once again clustered using K-means. The found clusters are then used as the attractor points $\mathbf{a}_s$ in (1.83).

For DC, the loss function and the K-means clustering only considered time-frequency bins with sufficient energy. This also applies for DANet when determining the attractors $\mathbf{a}_s$ in (1.82) and when determining the cluster centers using K-means.

Similarly to DC, DANets estimate embeddings where embeddings that are closer together are more likely to belong to the same speaker. However it uses the concept of attractors to be able to directly minimize the reconstruction loss of (1.75), like in uPIT. Identical to DC, the network architecture is independent of the number of speakers present in the mixture and can thus be used for a mixture of any number of speakers. However, for the K-means clustering, the number of clusters has to be known.

1.4.3 DNNs for SR

Before DNNs were introduced in the field of SR, the i-vector paradigm detailed in Section 1.3 was the most used solution. New DL approaches replaced parts of the i-vector system with DNNs [167]. In one of the first of such approaches,

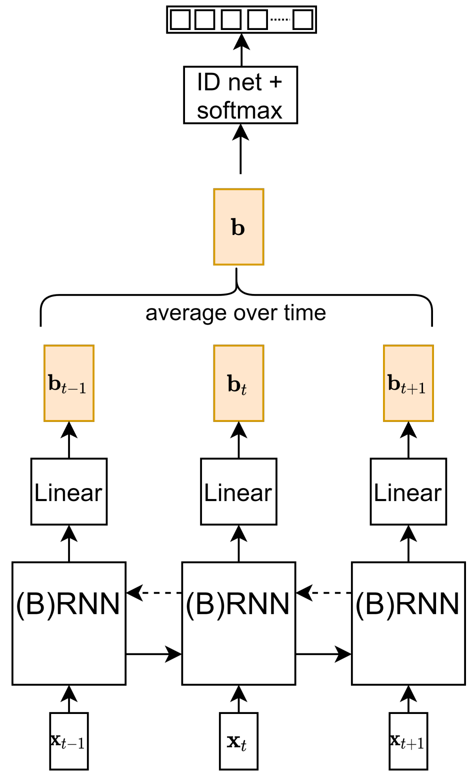


Figure 1.13: Illustration of the d-vector approach. The d-vector, here indicated with $\mathbf{b}$, is the averaged output of the main network. The speaker probabilities are estimated from this d-vector using an identification network, which is typically a single layer with softmax.

bottleneck features of a DNN-based Automatic Speech Recognition (ASR) system were used to replace the supervector $\mathbf{s}$ (see (1.30)) in the i-vector framework [105, 61, 143].

The d-vector approach went one step further and replaced the i-vector ($\mathbf{w}$ in (1.30)) itself by the units of the last hidden layer of a DNN, trained using a speaker classification cross-entropy loss [175, 112]. This approach is illustrated in Figure 1.13.

Finally, end-to-end DNN approaches exist, where the network takes as input an evaluation utterance and some enrollment utterances and is trained to directly predict whether they belong to the same speaker or not [76].

1.5 Thesis overview

To conclude the first chapter of this text, the thesis's main research question is recapitulated: do (B)SSS and SR rely on one another in the presence of overlapping speech and is a joint model for both problems the most efficient? Chapters 2 and 4 of the thesis will look for such joint models and indeed find that they outperform sequential approaches. In Chapter 2 this is done using NMF, while in Chapter 4 DNNs are used. Given that they reach related conclusions, it is expected that the dependencies between BSSS and SR are independent of the model choice and rather intrinsic to the tasks themselves. Chapters 3 and 5 will not explicitly search for a joint model, but will rather show that indeed speaker characteristic information is inherently required when performing BSSS. Finally, Chapters 6 and 7 will consider more applied and practical aspects of BSSS. In the remainder of this section, each chapter is introduced shortly.

- **Chapter 2: Joint Speaker Separation and Recognition using NMF**

  The NMF method seemed the most suited to build a joint model for BSSS and SR. For years it had been used in many SS problems. Furthermore, the speaker-dependent spectral characteristics contained in the NMF dictionaries, seemed directly applicable to SR, as was shown in [57]. However, NMF requires multi-channel data for a BSSS task, as otherwise there is no direct way to assign each basis vector to a specific speaker. This is in contrast with the DNN solutions used in Chapters 3-7 which can handle single-channel setups. An existing multi-channel NMF model for BSSS is extended to allow for joint BSSS and SR.

  This chapter is based on: Zegers, J., and Van hamme, H. Joint sound source separation and speaker recognition. In *Interspeech 2016* (2016), ISCA, pp. 2228–2232.

- **Chapter 3: Improving Source Separation via Speaker Representations**

  With the use of DNNs for single-channel BSSS in recent years, an attempt was made to exploit the hypothesised dependencies between BSSS and SR. This was done by blind multi-speaker adaptation by appending the i-vectors (Section 1.3) of the speakers to the input of the DNN. This way an explicit speaker characterization is presented to be used for BSSS. This can be done using oracle i-vectors (derived from the single speaker utterances, normally not present during inference), or the i-vectors can be derived from the estimated signals by a first-pass BSSS network. The main question is whether the DNN would benefit from this explicit speaker characterization or whether its internal speakers representation is sufficient for the BSSS task.

This chapter is based on: Zegers, J., and Van hamme, H. Improving source separation via multi-speaker representations. In *Interspeech 2017* (2017), ISCA, pp. 1919–1923.

- **Chapter 4: Joint Speaker Separation and Recognition using Deep Learning**

  The postulation of this chapter is similar to that of Chapter 2, only DNNs will be used instead of NMF. The choice of a new model also causes a different approach to building a *joint* model. The speaker dependent BSSS dictionaries of NMF are directly applicable to SR, due to the spectral speaker characteristics contained in these dictionaries. This is not directly possible with DNNs. More precisely, the bin embeddings $\mathbf{v}_{tf}$ for BSSS cannot be directly used for SR. Instead, a multi-task learning approach is used, common in many situations where DNNs are applied. In this case, some or most of the hidden representations in the network are shared, believing that the tasks rely on partly the same information.

- **Chapter 5: Analysis of Memory in RNNs for Blind Speech Source Separation**

  One of the major drawbacks of DNNs is that due to their nested and non-linear structure, it is difficult to understand what makes them arrive at their prediction and therefore DNNs lack explainability [149]. This chapter will look for insights in the memory of the LSTM-RNN. Novel methods will be presented that are generally applicable to study the memory of RNNs. These methods will be applied to the task of BSSS to determine to which extent SR and other factors play a role in a successful separation.

  This chapter is based on: Zegers, J., and Van hamme, H. Memory time span in LSTMs for multi-speaker source separation. In *Interspeech 2018* (2018), ISCA, pp. 1477–1481
  and on: Zegers, J., and Van hamme, H. Analysis of memory in LSTM-RNNs for source separation, 2020.

- **Chapter 6: Increasing Separation Robustness in Realistic Conditions**

  After reading Chapters 3, 4 and 5, one might rightfully wonder if the data set used in these chapters is representative. Until now only two-speaker English mixtures were used without any background noise. This chapter addresses this limitation, as it was generally missing in the DL domain for BSSS. Models will be developed that can cope with a variable number of speakers in the mixture. The performance of the DL algorithms of Section 1.4.2, until now only evaluated on English data, will be measured for different languages. The generalization of these models to unseen languages will be studied. Finally, extensions to the DL methods will be presented to cope with background noise. Although this chapter focuses on BSSS and no SR experiments are performed, it was considered important to add such a chapter to the thesis to increase the viability and the relevance of the discussed matter.

This chapter is based on: Zegers, J., and Van hamme, H. Multi-scenario deep learning for multi-speaker source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 5379–5383
and on: Appeltans, P., Zegers, J., and Van hamme, H. Practical applicability of deep neural networks for overlapping speaker separation. In *Interspeech 2019* (2019), ISCA, pp. 1353–1357.

- **Chapter 7: Speech Separation for EEG-informed Attention Decoding**

  There are many application were BSSS plays a role. These can be roughly divided into two categories:

  1. As a preprocessing step. At the start of the text it was mentioned that many speech technologies, such as ASR, SR, languague recognition, emotion recognition , etc. assume only a single speaker to be present in the audio recording. If this is not the case, BSSS is needed as a preprocessing step.
  2. As a speech enhancement method. Listening to a recording with multiple speakers can hamper speech intelligibility. This is particularly the case for hearing-aid users that struggle in cocktail party scenarios [40].

  This chapter addresses a particular application that fits in both categories as it studies the concept of a neuro-steered hearing aid. In summary, brain activity of a hearing-aid user is measured via Electroencephalography (EEG), where the user is asked to focus on a single target speaker in a multi-speaker scenario. This EEG data is used to estimate the speech envelope of the attended target speaker using an Auditory Attention Decoder (AAD). Until recently, this speech envelope was then compared with the oracle envelopes extracted from the single speaker audio recordings to decide which of the active speakers the listener wants to attend to. In a realistic setting, these single speaker audio envelopes should be estimated using BSSS. Once the attended speaker is selected, the estimated speech signal can be sent to the hearing-aid user. This chapter compares linear and DL approaches for BSSS.

This chapter is based on: Das, N., Zegers, J., Van hamme, H., Francart, T., and Bertrand, A. Linear versus deep learning methods for noisy speech separation for EEG-informed attention decoding. *Journal of Neural Engineering 17*, 4 (August 2020).

- **Chapter 8: Conclusion**

  This chapter concludes the thesis by listing the original contributions and directions for future research.

Chapter 2

Joint Speaker Separation and Recognition using NMF

This chapter is based on the following publication:
Zegers, J., and Van hamme, H. Joint sound source separation and speaker recognition. In *Interspeech 2016* (2016), ISCA, pp. 2228–2232.

The goal of the thesis, as described in the previous chapter, is to find a model that jointly performs (B)SSS and SR. In this chapter it will be shown that NMF is inherently capable of solving these two problems jointly. The dictionaries found in Section 1.2.1 for BSSS and the dictionaries of Section 1.2.2 for SR can be shared over both tasks.

The remainder of this chapter is organized as follows. Section 2.1 shows how NMF can be used to jointly solve these SSS and SR problems. Experiments are shown in Section 2.2 and a conclusion is given in Section 2.3.

2.1 Joint approach

To build a joint multi-channel NMF model for BSSS and SR, we retake (1.20) of Section 1.2.1.

$$\hat{\mathbf{X}}_{ft} = \sum_{k=1}^{K_{\text{mix}}} \sum_{s=1}^{S} c_{sk} t_{fk} q_{kt} \mathbf{H}_{fs}, \tag{2.1}$$

which allowed for multi-channel BSSS. Here t_{fk} models the value of the k^{th} basis vector for the f^{th} frequency index. The activation of a basis vector k at time t is given by q_{kt}, $\mathbf{H}_{fs}$ models the frequency response for the s^{th} speaker to the different microphones and c_{sk} is a latent speaker-indicator which links the k^{th} basis vector to the s^{th} speaker if c_{sk} is close to 1. K_{mix} is the total number of basis vectors, to be partitioned over the S speakers. We would like $\hat{\mathbf{X}}_{ft}$ to be close to $\mathbf{X}_{ft}$ where $\mathbf{X}_{ft} = \boldsymbol{y}_{ft}\boldsymbol{y}_{ft}^{\mathrm{H}}$.

For the joint model, this formulation is kept during the enrollment phase. For each enrollment mixture, the k^{th} basis vector is then assigned to the s^{th} latent speaker for which c_{sk} is maximal. The dictionaries of all speakers are collected in the library $\mathbf{T}_{\text{tot}} = [\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_I]$. The total number of basis vectors in the library is referred to as K_{tot}. For instance, consider N enrollment mixtures, with S active speakers each. If all S speakers in all N enrollment mixtures are unique, then $I = NS$. If each mixture has K_{mix} basis vectors, to be partitioned over its S speakers, then $K_{\text{tot}} = K_{\text{mix}}N$.

While testing, a novel source separation algorithm is used, which is similar to (2.1). $\mathbf{T}_{\text{tot}}$ remains fixed and SSS is no longer blind. The subscripts of $\mathbf{C}$ are changed to c_{ik} (compared to c_{sk} in (2.1) and Section 1.2.1) since all speakers I in the library are considered. As stated in Section 1.2.1, not all I speakers seen during enrollment have to be present in the test mixture $\mathbf{X}^{\text{test}}$ ($S \leqslant I$). A new latent variable indicator z_{si} is introduced, which maps a complete dictionary $\mathbf{T}_i$ and its corresponding speaker identity to an active test speaker s, under the constraints $0 \leqslant z_{si} \leqslant 1$ and $\sum_s z_{si} = 1$. This allows to detect which of the I speakers are active in the test mixture. Furthermore, it also links a speaker identity i, to a speech signal estimate from the test mixture. This will be detailed later on.

The variable $\hat{\mathbf{X}}_{ft}$ from (2.1) is then reformulated as follows

$$\hat{\mathbf{X}}_{ft} = \sum_{k=1}^{K_{\text{tot}}} \sum_{i=1}^{I} \sum_{s=i}^{S} z_{si} c_{ik} t_{fk} q_{kt} \mathbf{H}_{fs}. \tag{2.2}$$

It is shown in Appendix A.1 that the update formulas of (1.22)-(1.27) are then extended to

$$t_{fk} \leftarrow t_{fk} \sqrt{\frac{\sum_t \sum_i \sum_s z_{si} c_{ik} q_{kt} \operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{X}_{ft} \hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs}\right)}{\sum_t \sum_i \sum_s z_{si} c_{ik} q_{kt} \operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs}\right)}} \tag{2.3}$$

$$q_{kt} \leftarrow q_{kt} \sqrt{\frac{\sum_f \sum_i \sum_s z_{si} c_{ik} t_{fk} \operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{X}_{ft} \hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs}\right)}{\sum_f \sum_i \sum_s z_{si} c_{ik} t_{fk} \operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs}\right)}} \tag{2.4}$$

$$z_{si} \leftarrow z_{si} \sqrt{\frac{\sum_f \sum_t \sum_k c_{ik} t_{fk} q_{kt} \mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{X}_{ft} \hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs}\right)}{\sum_f \sum_t \sum_k c_{ik} t_{fk} q_{kt} \mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs}\right)}} \tag{2.5}$$

$$c_{ik} \leftarrow c_{ik} \sqrt{\frac{\sum_f \sum_t \sum_s z_{si} t_{fk} q_{kt} \mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{X}_{ft} \hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs}\right)}{\sum_f \sum_t \sum_s z_{si} t_{fk} q_{kt} \mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1} \mathbf{H}_{fs}\right)}} \tag{2.6}$$

To update $\mathbf{H}_{fs}$ an algebraic Riccati equation is solved

$$\mathbf{H}_{fs} \mathbf{A} \mathbf{H}_{fs} = \mathbf{B} \tag{2.7}$$

$$\mathbf{A} = \sum_t \sum_k \sum_i z_{si} c_{ik} t_{fk} q_{kt} \hat{\mathbf{X}}_{ft}^{-1} \tag{2.8}$$

$$\mathbf{B} = \mathbf{H}'_{fs} \left(\sum_t \sum_k \sum_i z_{si} c_{ik} t_{fk} q_{kt} \hat{\mathbf{X}}_{ft}^{-1} \mathbf{X}_{ft} \hat{\mathbf{X}}_{ft}^{-1} \right) \mathbf{H}'_{fs} \tag{2.9}$$

where $\mathbf{H}^{'}_{fs}$ is the $\mathbf{H}_{fs}$ of the previous update. To avoid scale ambiguity these normalizations should follow: $\mathbf{H}_{fs} \leftarrow \mathbf{H}_{fs}/\mathrm{tr}(\mathbf{H}_{fs})$, $t_{fk} \leftarrow t_{fk}/\sum_{f'} t_{f'k}$, $z_{si} \leftarrow z_{si}/\sum_{i'} z_{si'}$ and $c_{ik} \leftarrow c_{ik}/\sum_{i'} c_{i'k}$.

The above equations give the general update rules, but when used in the test phase of the considered application, $\mathbf{T}$ (or rather $\mathbf{T}_{\mathrm{tot}}$) will be kept fixed. Furthermore, c_{ik} is also kept fixed as a basis vector k should stay in the dictionary i it was trained for[1]. Therefore, $c_{ik} = 1$ only if the k^{th} basis vector belongs to the j^{th} dictionary, otherwise $c_{ik} = 0$.

The separation masks, similar to (1.28), can then be found as

$$\hat{\mathbf{m}}_{s,ft} = \left(\sum_i^I \sum_k^{K_{\mathrm{tot}}} z_{si} c_{ik} t_{fk} q_{kt} \right) \mathbf{H}_{fs} \hat{\mathbf{X}}_{ft}^{-1}. \tag{2.10}$$

The estimated ID for speaker s is i for which z_{si} is maximal

$$\hat{ID}_s = \arg\max_i z_{si}. \tag{2.11}$$

Through $\mathbf{H}_{fs}$, $\hat{\mathbf{m}}_{s,ft}$ and z_{si}, the SR of (2.11) links together the identity, the spatial position and the signal estimate of speaker s in the test mixture.

It was previously mentioned that not all enrolled speakers need to be active in the test mixture ($S \leqslant I$). However, in the experiments of Section 2.2, this will

[1]Otherwise z_{si} would no longer hold any meaning and no SR would be possible.

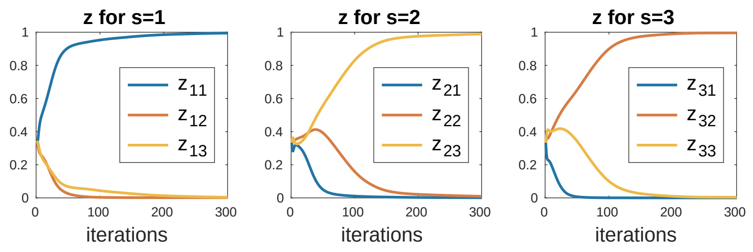


Figure 2.1: Example of z_{si} of the iterative process during the test phase.

be the case ($S = I$, $K_{\text{tot}} = K_{\text{mix}}$). Therefore, the latent variable indicator z_{si} no longer indicates whether an enrolled speaker i is present in the text mixture, as they all are. However, the link between the identity, the spatial position and the signal estimate of speaker s still holds in case of a successful SR in (2.11).

In Figure 2.1 an illustration is given of the values z_{si} during the iterative process in the test phase. For each speaker s in the mixture (or rather the speaker fixed to the position s via $\mathbf{H}_{fs}$ and the signal estimate via $\hat{\mathbf{m}}_{s,ft}$), a different plot is made. Although the reader is reminded that these are found jointly in the iterative process. When looking at the plot for $s = 1$, it becomes clear, after some iterations, that the algorithm believes that for this example the identity of speaker $s = 1$ of the mixture corresponds to the one belonging to dictionary $i = 1$ in the library as $z_{11} > z_{12}$ and $z_{11} > z_{13}$. Similarly, $s = 2$ will be mapped to $i = 3$ and $s = 3$ to $i = 2$.

2.2 Experiments

2.2.1 Data and set-up

The CHiME corpus [31] has been used to perform the experiments in this chapter. It contains 34 speakers with 500 spoken utterances per speaker and a vocabulary size of 52 words. Each utterance is about 1.5 seconds long. A mixture in the time domain is simulated with a microphone array of $J = 2$ microphones and three randomly chosen speakers ($I = S = 3$) on randomly chosen spatial positions that are at least 20° apart. The RIR for each speaker to the microphones is determined via its spatial angle to the microphone array and some mild reverberation ($\text{RT}_{60} = 280$ ms) [21]. The microphones are placed 15 cm apart and sample at 16kHz. A Generalized Cross Correlation with Phase Transform (GCC-PHAT) is used to estimate the DOA and to perform a source count [95]. Thus, aside from SSS and SR, also information on the number of sources and an initial estimate of their location will be known. It has already been shown that such a GCC-PHAT is useful to initialize the multi-channel NMF [120]. Here it will be used to set up the initial off-diagonal elements of $\mathbf{H}_{fs}$.

For mixtures of 1.5 seconds, the GCC-PHAT finds the correct number of sources in about 85% of the cases. This goes to 100% for longer mixtures. Since this chapter is not about source counting, mixtures with a false estimate of the number of sources will not be included in the results of the experiments in this chapter. A spectrogram for each microphone is calculated using an STFT with a window length of 64 ms and an overlap of 32 ms.

During the enrollment/training phase each person speaks U_{tr} utterances, without moving. The library $\mathbf{T}_{\text{tot}}$ is learned according to Section 1.2.1 in 1000 iterations. For each speaker $K_s = K$ basis vectors are assigned, giving a total of $K_{\text{tot}} = K_{\text{mix}} = KS$ basis vectors. Random initializations are used for $\mathbf{T}$ and $\mathbf{Q}$. Basis vectors are fixed in a dictionary, belonging to a speaker on a certain location, by setting the speaker-indicators $\mathbf{C}$ as follows.

$$c_{sk} = \begin{cases} 1 & \text{if } k \in \kappa^s \\ 0 & \text{otherwise} \end{cases} \tag{2.12}$$

Through c_{sk} and the spatial information in $\mathbf{H}_{fs}$, a basis vector is then fixed to a speaker on a certain location. In preliminary experiments, it was found that when $\mathbf{C}$ is not fixed but updated via (1.24), it naturally converges to a solution where $\mathbf{c}_k$ is one-hot encoded. The major difference of this solution compared to (2.12) is that the number of basis vectors K_s in a dictionary can vary per speaker. While it could be advantageous to allow this parameter to be speaker dependent (e.g. for speakers with more enrollment data or a larger speech variety), for practical reasons it was chosen to keep this fixed and follow (2.12).

In the test phase the same speakers as in the training phase are used ($I = S = 3$), but placed on different virtual locations. Each person speaks $U_{\text{test}} = 1$ utterance, without moving. The algorithm in Section 2.1 is applied and 1000 iterations are used. The library $\mathbf{T}_{\text{tot}}$ is taken from the training phase, $\mathbf{Q}$ is initialized randomly and $\mathbf{H}$ is again initialized using GCC-PHAT. The estimated IDs are determined by using (2.11). 50 such test mixtures are created and for each three speakers have to be estimated. This gives a total of 150 recognition tasks per training set. In total 20 independent training sets are created and their recognition accuracies are averaged to cope with the training variability.

Pure SR performance is analyzed in the single speaker scenario, where no SSS takes place and dictionaries are learned from non-reverberated speech. The *joint* approach (Section 2.1), which can be applied during either or both, testing and

$\mathbf{N}_{\text{comp}}$	$\mathbf{D}_{\text{ivec}}$					
	5	10	15	20	50	100
256	68.3	73.7	74.6	74.5	62.5	62.4
512	66.1	74.8	74.8	73.4	68.8	66.5
1024	60.9	75.2	**75.4**	75.2	71.9	68.4
2048	55.4	68.5	70.6	72.7	73.8	72.2

Table 2.1: i-vector based SR accuracies (in %) for the *full simultaneous* scenario.

training, is compared with a single speaker scenario and a sequential scenario. In the *single speaker* scenario, the dictionaries are trained or tested on non-reverberated speech of a single speaker, without applying SSS. In the *sequential* approach, SSS is first applied on speaker mixture data, then dictionaries are trained or tested on the segregated streams. In the remainder of the chapter, the scenario where both training and testing are performed on simultaneous speech, will be referred to as the *full simultaneous* scenario. In the *full single* scenario both training and testing are performed on a single speaker.

2.2.2 i-vector baseline

A second baseline for a *sequential* approach is also presented which uses i-vectors for the SR part. The MSR Toolbox is used to set up the speaker recognizer [147]. First a UBM and the total variability space are determined using 25 speakers not used in the enrollment phase. For every universal background speaker 500 single speaker utterances are used. Source separation was performed on mixtures using the multi-channel NMF with dictionary size $K = 10$. The STFT features of the separated signals are transformed to MFCC features with differentials (MFCCΔ) and accelerations (MFCC$\Delta\Delta$). i-vectors are determined from the separated signals, the total variability space and the UBM. A test segment is classified using the cosine distance between the test i-vector and the training i-vectors. Table 2.1 shows how the speaker recognition accuracies in the *full simultaneous* scenario depend on the number of UBM components N_{comp} and the dimensionality of the i-vector D_{ivec}. The optimal values were found to be $N_{\text{comp}} = 1024$ and $D_{\text{ivec}} = 15$. This is comparable to the values found in [57].

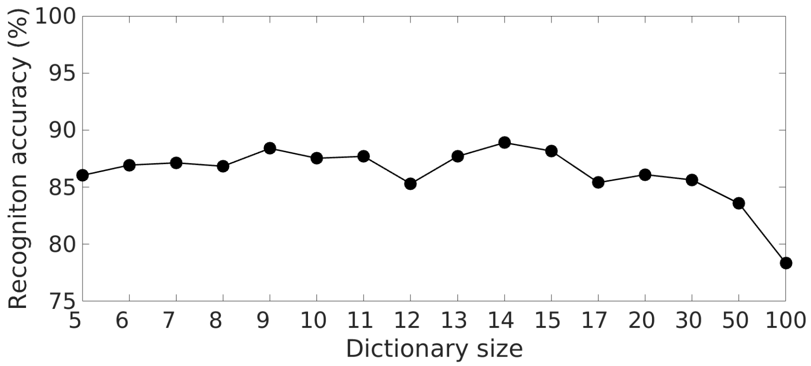


Figure 2.2: Average speaker recognition accuracy (in %), depending on the dictionary size, using the *joint* approach in the *full simultaneous* scenario.

2.2.3 Results

In Figure 2.2 SR accuracies relative to the dictionary size K are plotted for the *full simultaneous* scenario. The number of training utterances per speaker U_{tr} was taken at 20 utterances. Results were obtained using the *joint* solution. There is little influence of the dictionary size on the recognition accuracy if K is between 8 and 30. In the remainder of the chapter the dictionary size is taken at $K = 10$.

Speaker recognition accuracies for the different scenarios and used methods, using the above found optimal parameters ($K = 10$, $N_{\text{comp}} = 1024$ and $D_{\text{ivec}} = 15$), are shown in Table 2.2(a) for NMF-based speaker recognition and Table 2.2(b) for i-vector based speaker recognition. In the *full single* scenario i-vector based SR performs slightly better then NMF based SR. However, in *simultaneous* scenarios, i-vectors struggle with the crosstalk after SSS. The same problem occurs for NMF when a sequential approach is used in, either or both, training and testing. This problem is circumvented when the *joint* approach for NMF in the *full simultaneous* scenario is used. The Speaker Error Rate (SER) for the *joint* approach (12%) is substantially lower than a *sequential* approach using i-vectors for the SR (24%).

Figure 2.3 shows how the recognition accuracy increases with the amount of training data. The optimal parameters above are used again. The recognition accuracy only decreases for very low amount of training utterances. The sufficiency of a limited amount of training data is probably due to the limited vocabulary size in the CHiME corpus. Notice that increasing the amount of training is beneficial to the recognition accuracy in two ways. SSS is improved

Train	**Test**		
	single	seq	joint
single	98.0	**86.4**	73.2
seq	88.8	76.1	62.5
joint	80.2	61.3	**87.9**

(a) NMF based SR.

Train	**Test**	
	single	seq
single	**98.4**	78.5
seq	**91.2**	76.0

(b) i-vector based SR.

Table 2.2: SR accuracies (in %) for different training and testing scenarios. In *seq* and *joint* 3 speakers are active simultaneously. *Single* uses data from one speaker at a time.

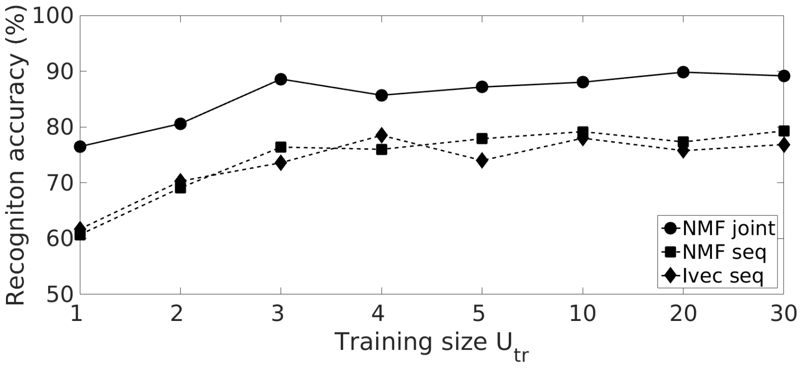


Figure 2.3: Average speaker recognition accuracy, depending on the training size U_{tr}, in the *full simultaneous case.* Results are shown for the *joint* approach (circles) and for a *sequential* approach using NMF (squares) or i-vectors (diamonds).

(see Table 2.4) and the amount of patterns seen in training data is increased, which allows the speakers to be better characterized and more easily recognized. To cope better with the increasing amount of seen patterns, the dictionary size could also be increased. This way the extra detected patterns can be stored in the dictionary. A system is chosen that can cope with any training size and thus the dictionary size is fixed.

As this chapter analyzes joint SR and SSS, an evaluation metric for the SSS quality is also considered. SDR, SIR and SAR between the original source signal and the signal separated from the mixture, are calculated and are shown in Table 2.4 [176]. Three different situations are shown: a mixture where each person speaks 20 utterances, a mixture where each person speaks 1 utterance and a mixture where each person speaks 1 utterance that is separated using previously learned dictionaries. 20 independent runs are used to calculate the ratios and the average is shown. As expected, the SSS results are better when the mixture is longer. NMF has more examples to recognize recurring patterns and build distinctive dictionaries per speaker which enhances the separation quality. However, performing SSS using learned and fixed dictionaries decreases performance. If a pattern is not seen in the training stage or is not frequent enough to fit in the dictionary, it cannot be used during testing for the reconstruction of the signal and will thus degrade the reconstructed signal. A solution is to allow flexibility in the trained dictionaries when testing, but this would interfere with the speaker recognition.

	SDR	SIR	SAR
SSS on 20 utts	6.93	12.50	9.36
SSS on 1 utt	4.28	7.99	8.26
SSS on 1 utt using learned dicts	3.08	5.62	8.81

Table 2.4: SSS performance for different scenarios measured in SDR, SIR and SAR (dB).

2.3 Conclusion

In this chapter it was shown that in simultaneous speech environments for three overlapping speakers, a joint approach for BSSS and SR outperforms sequential approaches for both NMF and i-vector based SR. This benefit is inherent to the (multi-channel) NMF as the same speaker patterns that are learned to perform segregation can be used to recognize speakers. Nonetheless, a similar goal will be set in the next chapters where DNNs will be used.

Chapter 3

Improving Source Separation via Speaker Representations

This chapter is based on the following publication:
ZEGERS, J., AND VAN HAMME, H. Improving source separation via multi-speaker representations. In *Interspeech 2017* (2017), ISCA, pp. 1919–1923.

In this chapter an attempt is made to improve BSSS by informing the model on the identities and characteristics of the speakers via blind speaker adaptation, thereby again attempting to show the link between BSSS and speaker characterization. It is also the first chapter that uses DNNs. All subsequent chapters will also be using DNNs.

3.1 Introduction

In speaker adaptation a system is adapted to better suit the characteristics of the target speaker. Speaker adaptation has been successfully applied in ASR tasks using DL. There are several ways to incorporate speaker information in a network.

- One can apply a space transform to the input features depending on the speaker identity, such as feature-space Maximum Likelihood Linear

Regression (fMLLR) [154, 190]. The network can then be trained as usual to learn from this speaker information.

- In model-based adaptation, the whole network or parts of the network are adapted to a specific speaker by retraining the model to adaptation data of that specific speaker [166, 107, 193, 189]. It is possible to perform so-called blind speaker adaptation by clustering speakers together via their i-vector representation [202]. For each cluster a network is adapted. At test time an utterance is first assigned to a cluster and then decoded with the according network. The term blind is used, since the identity of the speaker is not known a priori, but still a network is used that is thought to be more adapted to the unknown speaker.
- Another way to adapt the network is to add speaker characterizing features, such as an i-vector to the input [150]. In fact, this can also be seen as a speaker dependent space transform in a broad sense. The advantage of such an i-vector is that it models speaker variability and can also be determined blindly, without any prior knowledge of the speaker [72].

In this chapter an attempt is made to perform blind speaker adaptation for BSSS. There are two main challenges. The first challenge is that the network is to be adapted to more than a single speaker. Secondly, there is no direct way to extract an i-vector for all speakers, since they are speaking simultaneously. A multi-speaker representation is sought that can be added to the input of the network. The general idea is to first perform blind source separation, then extract i-vectors on all estimated sources and use these to adapt a second network, extract the i-vectors on the new estimates of the second network and so on. A similar idea was used by Zhang *et al.* [200] for noise suppression in presence of speech. They used speech enhancement to get a better estimate of the pitch, which they in turn fed in a subsequent network for better speech enhancement, which allowed for better pitch estimation and so on. If the i-vector extraction procedure is performed by a neural network, it is possible to do a final end-to-end training of the complete network. However, this will not be implemented in this chapter.

After original publication of this chapter, another approach called *SpeakerBeam* was introduced [52]. It uses single speaker enrollment data of a target speaker to adapt a network to filter out the speech of that specific speaker in a multi-speaker test mixture. Since it uses enrollment data, it is not considered a blind adaptation. Furthermore, it is not a multi-speaker adaptation, since only a single speaker is of interest.

While i-vector extraction on signal estimates could allow for SR in multi-speaker scenarios, the focus of this chapter is on source separation quality. The rest of this chapter is organized as follows. In Section 3.2 a method is proposed to perform blind multi-speaker adaption for source separation. Experiments are presented in Section 3.3 and a conclusion is given in Section 3.4.

3.2 Blind multi-speaker adaptation

To perform blind multi-speaker adaptation, first an estimate of the source signals is found by estimating a binary mask using DC, as explained in Section 1.4.2. Subsequently, an i-vector is extracted from each speech estimate. Optionally, the dimensionality of the i-vector can be reduced using LDA (see Section 1.3). All (LDA) i-vectors are then repeated over all time frames of the mixture and, together with the mixture spectrogram, are then fed into another neural network, which is thus trained with explicit speaker information. The last two steps can be repeated iteratively. The proposed iterative architecture is shown in figure 3.1. The network of level 0 is trained on the mixture spectrogram and is the baseline for this chapter. The network of level l is trained on the mixture spectrogram and the (LDA) i-vectors of level $l - 1$.

A level $l + 1$ network can only be expected to bring further improvement compared to a level l network if the i-vectors presented at the input are less noisy (extracted from a level l network and level $l - 1$ network, respectively). The extracted i-vectors can only be expected to be of better quality if the following two conditions are met:

1. The separation performance of the level l network is better than the level $l - 1$ network. In that case it can be expected that the extracted i-vector from level l will be less noisy than those extracted from the level $l - 1$ network.
2. The separation performance of the level l network is worse than a network that is trained with oracle i-vectors, determined on the single speaker utterances, as the latter can be seen as an upper bound. If the level l network reaches this upper bound, no further improvements can be expected.

In the experiments that were performed, it was usually concluded that the separation performance of the level 1 network matches that of the upper bound and therefore no results for $L > 1$ networks are given in this chapter.

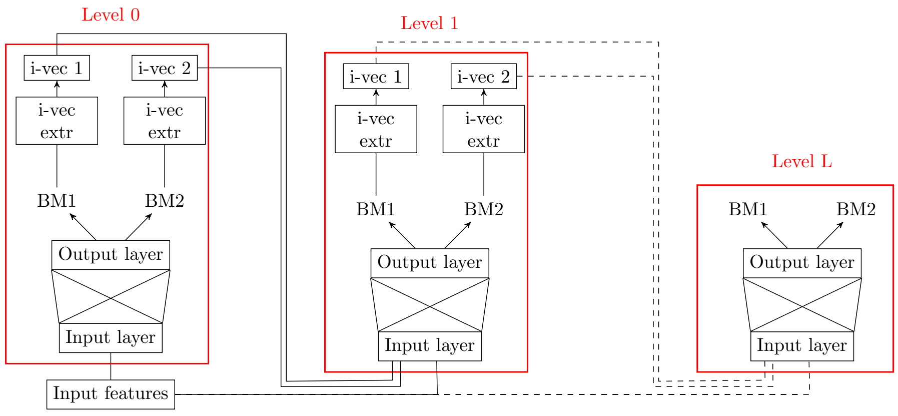


Figure 3.1: Proposed iterative architecture. *BM* refers to the binary masks estimated by DC.

3.3 Experiments

3.3.1 Experimental set-up

The experimental set-up for this chapter will be detailed quite extensively as it is the first chapter that uses DNNs. The experimental set-ups in the following chapters will try to mirror the set-up detailed in this section.

For the BSSS task, mixtures of two speakers ($S = 2$) were used from the corpus introduced in [78]. These mixtures were artificially created by mixing single speaker utterances from the Wall Street Journal 0 (WSJ0) corpus [64]. A gain for the first speaker compared to the second speaker was randomly chosen between 0 and 5 dB. Utterances were sampled at 8 kHz and the length of the mixture was chosen equal to the shortest single speaker utterance in the mixture to maximize the overlap. The training and validation sets contained 20 000 mixtures ($\sim$30h) and 5 000 mixtures, respectively from 101 speakers, while the test set contained 3 000 mixtures from 16 held-out speakers. An STFT with a 32 ms window length and a hop size of 8 ms was used.

DC was used as BSSS algorithm (see Section 1.4.2). The DNN consists of two fully connected bidirectional LSTM-RNN layers, with 600 hidden units each in each direction. Hidden units of both directions are concatenated before being passed to the next layer, as was expressed by (1.65). The DC embedding dimension was chosen at $D = 20$ and since the frequency dimension was $F = 129$, the total number of output nodes was $DF = 20 * 129 = 2580$. According to the principles of *curriculum learning* [14, 78], the networks were first trained on segments of 100 time frames, before training on the full mixture. The weights and biases were optimized with the Adam learning algorithm described in (1.55)-(1.59), with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. For the 100-frame stage the learning rate was kept constant at 10^{-3}. For the full mixture training stage the learning rate decreased exponentially, starting at $5 * 10^{-4}$ and reducing with a factor 0.85 each epoch. The log-magnitudes of the STFT coefficients were used as input features and was mean and variance normalized. Zero mean Gaussian noise with standard deviation 0.2 was applied to the training features to avoid local optima. Unlike in [88], dropout on the feedforward weights did not improve results, so it was not used in the experiments. Time-frequency bins with magnitude lower than -40 dB, compared to the maximum of the utterance, were omitted in the loss function of (1.77) to prevent the network from learning on empty or low-energy bins. All networks were trained using TensorFlow [2] and the code for all the experiments can be found here: `https://github.com/JeroenZegers/Nabu-MSSS`.

The K-means clustering, after estimating the embeddings $\mathbf{V}$, was done with the Scikit-learn library [135]. For each clustering, 10 random initializations were done, choosing the version with lowest total sum of distances. Again, time-frequency bins with magnitude lower than -40 dB, compared to the maximum of the utterance were omitted.

Performance for BSSS is measured by the average SDR improvements on the test set, using the `bss_eval` toolbox also used in Chapter 2 [176]. When comparing the average SDR improvements between two models, the following statistical consideration can be taken into account. The sample standard deviation is roughly 4.0 dB, determined on 6,000 samples ($S = 2$ evaluations for each of the 3,000 test mixtures). When assuming these samples to be independently and identically distributed, according to Student's t-test a sample mean difference of 0.09 dB is then significant for a significance level of $\alpha = 0.05$ [164].

In this chapter, each experiment was run 3 times, and the median score of the sample means is reported. This was done to cope with some variability due to the applied input noise and the random initialization of the networks.

For the i-vectors, the `si_tr_s` set of WSJ1 [1] was used as development data to train the UBM, the total variability matrix $\mathbf{T}$ and the LDA projection matrix $\mathbf{A}$ (see Section 1.3). 13-dimensional MFCCs are used as features and a VAD was used to leave out the silence frames. The UBM has 256 Gaussian mixtures. If LDA is applied, w is 400-dimensional. The MATLAB MSR Identity Toolbox v1.0 [148] was used to determine the UBM, $\mathbf{T}$ and $\mathbf{A}$ and to obtain the (LDA) i-vectors.

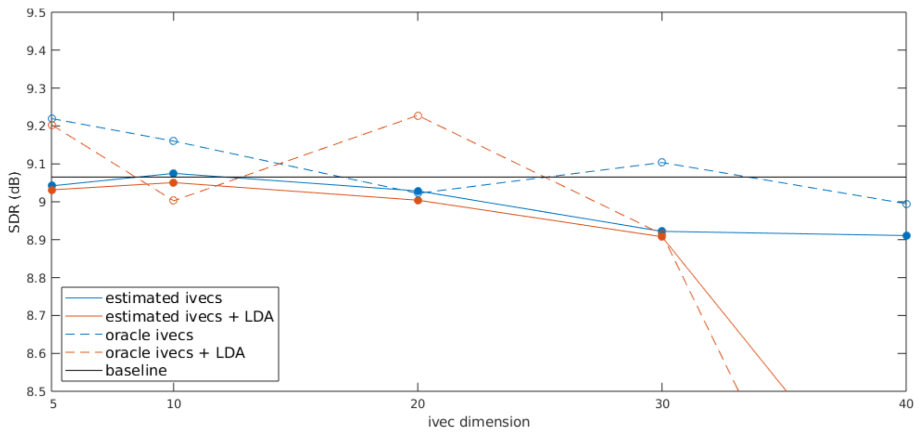


Figure 3.2: SDR improvements (in dB) for oracle and realistic experiments

3.3.2 Results

Results for the multi-speaker adapted networks are shown in figure 3.2. The baseline DC system did not use any speaker adaptation and was based on Section 1.4.2. The oracle i-vectors were determined from the single speaker utterances before mixing. This information is normally not present in realistic applications, but allows to find an upper bound for the performance of the presented system. Estimated i-vectors were achieved via the procedure of Figure 3.1.

In general it can be concluded from Figure 3.2 that the addition of i-vectors to the input of the BSSS network has little or no beneficial effect to the separation performance, compared to the baseline. This is in contrast with the observation made in the original publication [195]. The baseline there was roughly 3 dB lower than in Figure 3.2, although comparable with other papers at that time[1]. Including i-vectors in the old baseline model did improve separation performance. There it was concluded that the baseline DNN's internal speaker characterization for BSSS was not optimal. Apparently, as the baseline model improved, so did its internal speaker characterization, and the addition of external speaker vectors has little effect.

Taking this in consideration, the addition of external speaker vectors might still be useful in cases where the DNN struggles more. For example, in the case of very high background noise or when the DNN is much smaller. The latter

[1]Over the years many different hyper parameters have been tuned to achieve this 3 dB difference in baseline performance.

baseline	oracle ivecs dim 10	est. ivecs dim 10	shared model
6.24	6.72	6.68	6.56

Table 3.1: SDR improvements (in dB) where the networks have 50 hidden units per directional layer.

is tested by considering a baseline model with 50 units per directional layer (compared to the 600 units used for the results in Figure 3.2). The performance of this model with and without (oracle) i-vectors of dimension 10 is given in table 3.1. When comparing the baseline result with the network were the estimated i-vectors were added, an improvement of 0.44 dB is found. However, it can be argued that this comparison is not fair as the latter uses roughly twice the amount of trainable parameters ($l = 0$ and $l = 1$ networks), compared to the baseline (only $l = 0$ network). Therefore, an additional experiment was performed with a joint model where the parameters of the $l = 0$ and $l = 1$ networks were shared[2]. The result for this shared model, when in $l = 1$ mode, is still better than the baseline (+0.32 dB). It is therefore concluded that speaker characterization is relevant for the task of BSSS and if the model does not sufficiently succeed in building an internal speaker characterization, it can be helped by providing explicit external speaker vectors.

3.4 Conclusion

In this chapter it was shown that indeed BSSS is dependent on the SR subtask, when using DNNs. State-of-the-art BSSS models succeed in building a sufficient internal speaker characterization and adding external i-vectors does not bring gains in separation performance. However, it is noted that these models work in highly idealized settings, e.g. no reverberation and no background noise as well as assuming there is sufficient data to train a large number of parameters and sufficient computational capabilities to allow inference with a large model. It was found that a small model did have trouble with finding the optimal speaker representations and adding external speaker characterizations, did substantially help in this case. In Chapter 5 it will be shown that the same applies to RNNs with limited memory. This suggests that in (very) challenging scenario's (high background noise and/or reverberation) models might struggle to sufficiently

[2]To prevent the need for i-vector extraction during each training step, i-vectors were extracted from the baseline model. These extracted i-vectors were then fed to the joint model when in $l = 1$ mode. For the results in Table 3.1, the shared model is in $l = 1$ mode, with i-vectors extracted from the $l = 0$ mode.

tackle the SR subtask and putting an external focus on this can be beneficial. This is a suggestion for future research. In that case it could also be relevant to reconsider the use of $L > 1$ networks.

Chapter 4

Joint Speaker Separation and Recognition using Deep Learning

In Chapter 2 a joint NMF model for BSSS and SR was found that had better SR performance compared to two sequential NMF models, thereby supporting the main hypothesis of the thesis that indeed BSSS and SR for overlapping speech are best tackled jointly. In this chapter, the same hypothesis will be tested, using DNN approaches instead of NMF. If the hypothesis is again confirmed, it is expected that the dependencies between BSSS and SR are independent of the model choice and rather intrinsic to the tasks themselves.

For this, the model from [58] will be used that allows for both BSSS and SR, where the main part of the model is shared over both tasks. A modification will be proposed that improves performance of the model, as well as an extension that allows the SR part to be used for a combination of overlapping and non-overlapping speech fragments. Furthermore experiments will be used to show how best to perform the joint learning.

The remainder of this chapter is organized as follows. Section 4.1 will describe how a joint approach to BSSS and SR can be achieved with DNNs. In Section 4.2 some experiments will be reported and in Section 4.3 a conclusion will be given.

4.1 Joint approach

4.1.1 Basic joint model

The basic joint BSSS and SR model will be based on [58], which is an extension to DANet described in 1.4.2. An illustration of this model is given in Figure 4.1. The BSSS part of this model is identical to DANet (see Figure 1.12): BSSS embeddings $\mathbf{v}_{tf}$ are estimated using a linear output layer and BSSS attractors $\mathbf{a}_s$ are formed using (1.82). For the SR part, a separate output layer is used to estimate SR embeddings $\mathbf{u}_{tf}$. Then SR attractors $\mathbf{b}_s$ are estimated, similarly to (1.82) as

$$\mathbf{b}_s = \frac{\sum_{t,f} \mathbf{u}_{tf} z_{s,tf}}{\sum_{t,f} z_{s,tf}}. \tag{4.1}$$

The SR attractor per speaker $\mathbf{b}_s$ can then be processed similarly as in Figure 1.13 in Section 1.4.3: Pass $\mathbf{b}_s$ to an identification network which estimates speaker posteriors using a softmax which allows to determine the cross-entropy loss. Notice that since S speakers are present, S cross-entropy losses are determined and averaged. The SR attractor $\mathbf{b}_s$ can then be used as speaker identification vector (or d-vector, see Section 1.4.3).

In Section 1.4.2 it was mentioned that when determining $\mathbf{a}_s$ using (1.82), only time-frequency bins with sufficient energy were considered. The same applies for $\mathbf{b}_s$ when using (4.1).

Notice that the permutation problem discussed in Section 1.4.2 also applies to the (cross-entropy) SR loss when multiple speakers are active. The permutation problem is solved similarly as in DANet (see Section 1.4.2).

In a true joint spirit, one might wish to use a single type of embedding (and attractor) for BSSS and SR, i.e. $\mathbf{v}_{tf} = \mathbf{u}_{tf}$ (and $\mathbf{a}_s = \mathbf{b}_s$). However, it was experimentally found that this does not work well and it is much better to have a different embedding space per task[1]. This conclusion will also be made in Section 6.1, where a separate embedding space for 2-speaker and 3-speaker mixtures is preferred.

[1] The same conclusion is made in [58].

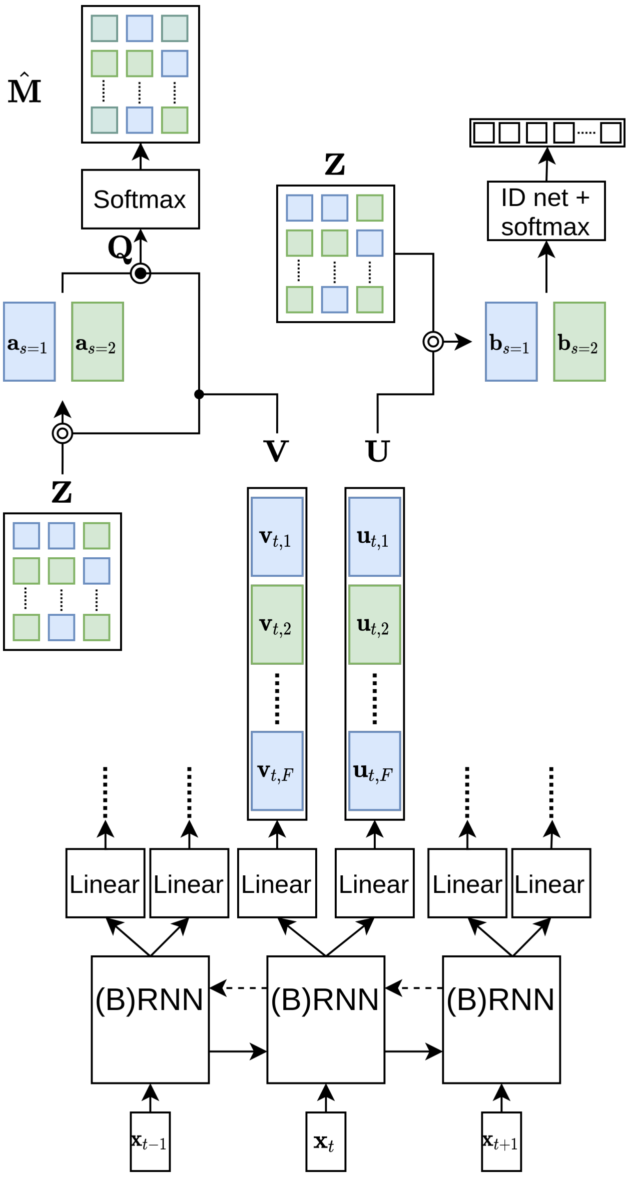


Figure 4.1: An illustration of the basic joint model when using a (B)RNN for $S = 2$. $\mathbf{a}_{s=1}$ and $\mathbf{a}_{s=2}$ are determined via (1.82) and $\mathbf{Q}$ is determined via (1.83). $\mathbf{b}_{s=1}$ and $\mathbf{b}_{s=2}$ are determined via (4.1).

4.1.2 Multi-task learning

To train the joint model, multi-task learning has to be applied. The advantages of multi-task learning for DNNs are well known [23, 22]. The BSSS loss of (1.76) and the cross-entropy loss for SR should both be optimized.

For single task learning a loss $\mathcal{L}_\Theta(\mathbf{Y}, \hat{\mathbf{Y}}(\mathbf{X}, \Theta))$ as in (1.44) is defined, based on the model inputs $\mathbf{X}$, the model targets $\mathbf{Y}$ and the model parameters Θ. After each training batch, a parameter θ is updated as follows

$$\Delta\theta = g\left(\frac{\partial \mathcal{L}_\Theta(\mathbf{Y}, \hat{\mathbf{Y}}(\mathbf{X}, \Theta))}{\partial\theta}\right) \tag{4.2}$$

where $g()$ is specific to the learning algorithm. For instance for SGD, according to (1.52)

$$g_{\text{SGD}}\left(\frac{\partial \mathcal{L}_\Theta}{\partial\theta}\right) = -\eta\frac{\partial \mathcal{L}_\Theta}{\partial\theta}. \tag{4.3}$$

For the Adam learning algorithm we find according to (1.59)

$$g_{\text{Adam}}\left(\frac{\partial \mathcal{L}_\Theta}{\partial\theta}\right) = -\frac{\eta}{\sqrt{\hat{m}_{\theta,2}\left(\frac{\partial \mathcal{L}_\Theta}{\partial\theta}\right)} + \epsilon}\hat{m}_{\theta,1}\left(\frac{\partial \mathcal{L}_\Theta}{\partial\theta}\right) \tag{4.4}$$

with $\hat{m}_{\theta,1}\left(\frac{\partial \mathcal{L}_\Theta}{\partial\theta}\right)$ and $\hat{m}_{\theta,2}\left(\frac{\partial \mathcal{L}_\Theta}{\partial\theta}\right)$ as defined in (1.55)-(1.58).

For multi-task learning, consisting of J tasks, a solution would be to define a joint loss $\mathcal{L}_{\Theta,m}$ as

$$\mathcal{L}_{\Theta,m}(\mathbf{Y}_m, \hat{\mathbf{Y}}_m(\mathbf{X}_m, \Theta)) = \sum_{j=1}^{J} \alpha_j \mathcal{L}_{\Theta,j}(\mathbf{Y}_j, \hat{\mathbf{Y}}_j(\mathbf{X}_j, \Theta)) \tag{4.5}$$

and take $\alpha_1 = 1$, with $\mathcal{L}_{\Theta,j}(\mathbf{Y}_j, \hat{\mathbf{Y}}_j(\mathbf{X}_j, \Theta))$ the loss of task j. In the remainder of this section $\mathcal{L}_{\Theta,j}(\mathbf{Y}_j, \hat{\mathbf{Y}}_j(\mathbf{X}_j, \Theta))$ is referred to as $\mathcal{L}_{\Theta,j}$, to simplify notations. (4.2) can then be applied as usual. However, $J-1$ meta parameters α have to be chosen, e.g. by tuning them using a validation set.

A good choice of α_j can be important, as is also pointed out in [92]. For example, the joint model in Section 4.1.1 uses the loss of (1.76) and the cross-entropy loss, which are of a totally different scale. Neglecting to find the proper α_j by simply setting them to 1, could mean that one loss would be dominant over the other (in scale) and therefore effectively training on one loss rather than both. To alleviate part of the problem, one could roughly guess the magnitude $\hat{m}_j$ of the optimal loss of all tasks and use this as normalization

$$\mathcal{L}_{\Theta,m}(\mathbf{Y}_m, \hat{\mathbf{Y}}_m(\mathbf{X}_m, \Theta)) = \sum_{j=1}^{J} \frac{\alpha_j}{\hat{m}_j} \mathcal{L}_{\Theta,j}(\mathbf{Y}_j, \hat{\mathbf{Y}}_j(\mathbf{X}_j, \Theta)). \tag{4.6}$$

This might give a better solution for the $\alpha_j = 1$ case but still seems far from optimal.

Alternatively, to circumvent all this, a parameter update can be found for each task individually and then summed as follows

$$\Delta_m \theta = \sum_{j=1}^{J} \Delta_j \theta = \sum_{j=1}^{J} g\left(\frac{\partial \alpha_j \mathcal{L}_{\Theta,j}}{\partial \theta}\right) = \sum_{j=1}^{J} g\left(\frac{\alpha_j \partial \mathcal{L}_{\Theta,j}}{\partial \theta}\right). \tag{4.7}$$

If a learning algorithm is chosen that is independent of the scale of the gradients, or in other words $g(\alpha \partial \mathcal{L}_\Theta / \partial \theta) = g(\partial \mathcal{L}_\Theta / \partial \theta)$, like the Adam algorithm, (4.7) simplifies to

$$\Delta_m \theta_i = \sum_{j=1}^{J} g\left(\frac{\partial \mathcal{L}_j}{\partial \theta_i}\right) \tag{4.8}$$

which is independent of the loss scale factors α so that they no longer have to be tuned. If each task calculates a parameter update using the Adam algorithm, the relative magnitudes of the losses between tasks no longer play a role due to the invariance to the magnitude of the loss for Adam. Notice, if a learning algorithm like SGD would be chosen, the parameter updates would scale to the magnitude of the losses and the global update would be the same as using (4.5).

If requested, it it still possible to use a tuneable parameter α_j in (4.8) to give one task more importance than the other. This would give

$$\Delta_m \theta_i = \sum_{j=1}^{J} \alpha_j g\left(\frac{\partial \mathcal{L}_j}{\partial \theta_i}\right). \tag{4.9}$$

However in this case parameter updates are weighted, rather than the losses or the gradients. Setting $\alpha_j = 1$ will then give equal importance to all tasks, while this might not be the case when doing this for (4.5).

It is possible that some losses $\mathcal{L}_j$ are independent of some parameters θ_i. For example, the BSSS loss will not depend on the parameters in the identification network of the joint model of Section 4.1.1. As a consequence, $g\left(\frac{\partial \mathcal{L}_j}{\partial \theta_i}\right) = 0$. If all $\alpha_{j' \neq j}$ were taken very small, for instance to give much importance to the BSSS task in our example, then $\Delta_m \theta_i \approx 0$. It can therefore be preferred to make the scale factors dependent on the parameters and use the following normalization

$$\Delta_m \theta_i = \frac{1}{\sum_{j'}^{J} \alpha_{j'} \delta_{ij'}} \sum_{j=1}^{J} \alpha_j \delta_{ij} g\left(\frac{\partial \mathcal{L}_j}{\partial \theta_i}\right) \tag{4.10}$$

where $\delta_{ij} = 1$ if loss $\mathcal{L}_j$ is dependent on parameter θ_i and $\delta_{ij} = 0$ otherwise.

In most multi-task works, the sum of the task losses (or the task gradients) using (4.5) is taken [191, 28, 27, 39, 83]. Other work sums the parameter task updates using (4.7), but chooses SGD as the learning algorithm, making it equivalent to (4.5) [106, 96]. In Section 4.2.3, an experimental comparison between (4.5), (4.9) and (4.10) will be made for the joint BSSS-SR model. It will be shown that when performing a full search for the optimal α_j all three approaches give similar results. However, $\alpha_j = 1$ for (4.9) and (4.10) will give a performance close to the optimal α_j, while this might not be the case when summing the task losses. This makes a hyperparameter search required for the latter. It is noteworthy that in [92] a way was found to optimize the α_j during training.

4.1.3 Modifications to the basic joint model

In this section two modifications to the basic joint model of Section 4.1.1 will be presented. The first will aim at a better general performance, while the second modification will allow to include single speaker audio as well as overlapping speech.

Coping with BSSS uncertainty

In Section 4.1.1 the bin target labels $z_{s,tf}$ were used to obtain the SR attractors $\mathbf{b}_s$ in (4.1), similar to obtaining the BSSS attractors $\mathbf{a}_s$ in (1.82). This information is only present during training. During evaluation $z_{s,tf}$ is estimated by setting it to 1 if the embedding $\mathbf{v}_{tf}$ is assigned to K-means cluster c_s and 0 otherwise. However, this creates a train-test mismatch and the model will not take the uncertainty on $z_{s,tf}$ into account during training. Therefore, it is proposed to also use estimates of $z_{s,tf}$ during training for (4.1) instead of $z_{s,tf}$ itself.

$$\mathbf{b}_s = \frac{\sum_{t,f} \mathbf{u}_{tf} \hat{z}_{s,tf}}{\sum_{t,f} \hat{z}_{s,tf}} \tag{4.11}$$

$$\hat{z}_{s,tf} = \begin{cases} 1, & \text{if } s = \arg\max_{s'} \hat{m}_{s',tf} \\ 0, & \text{otherwise} \end{cases} \tag{4.12}$$

where $\hat{m}_{s,tf}$ can be found by applying the softmax activation to (1.83). The extended training algorithm is visualized in Figure 4.2.

It was found in initial experiments that a similar idea also helps for BSSS. In that case, the masks $\hat{m}_{s,tf}$ are estimated in a first pass, by using $z_{s,tf}$ for (1.82) to find $\mathbf{a}_s$ and then applying the softmax activation to (1.83) with the found

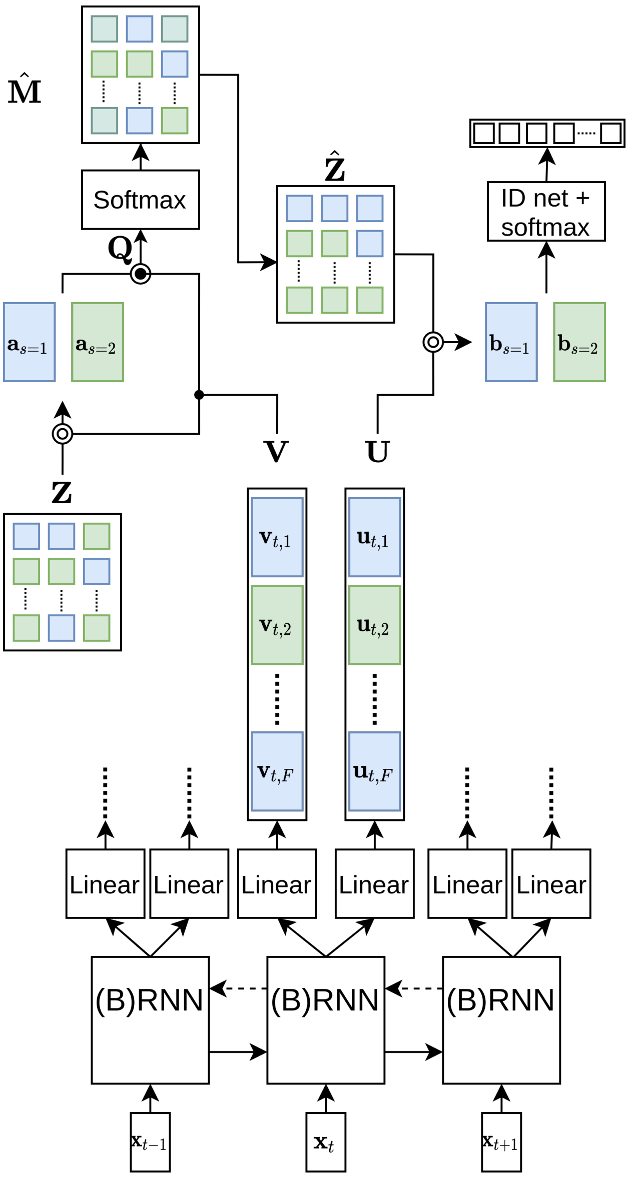


Figure 4.2: An illustration of the extended joint model when using a (B)RNN for $S = 2$.

$\mathbf{a}_s$. In a second pass, $\hat{z}_{s,tf}$ is determined via (4.12) which then replaces $z_{s,tf}$ in (1.82) to find the second pass $\mathbf{a}_s$. This second pass $\mathbf{a}_s$ are used to find the second pass masks $\hat{m}_{s,tf}$ by applying the softmax activation to (1.83).

Allowing single speaker audio

While overlapping speech is a significant problem for speech processing, as explained in Chapter 1, in most applications it can be expected that a large part of audio recordings will contain single speaker audio. In fact, these single speaker segments can be advantageous to better model the speakers for SR. Therefore, a modification to the use of the basic model of Section 4.1.1 is proposed to allow for single speaker processing. This is visualized in Figure 4.3. When handling an audio fragment with a single speaker, the BSSS part of the model is not used. SR embeddings $\mathbf{u}_{tf}$ for each time-frequency bin are still estimated. However, since only one speaker is active, they can simply be averaged to obtain $\mathbf{b}$. The same identification network as for the basic joint model is then used to estimate the speaker posteriors.

By training the model with examples of both $S = 1$ and $S = 2$, the SR attractors or speaker identification vectors $\mathbf{b}$ can be used for both situations. For example, the enrollment of a speaker could be done using single speaker audio and the obtained $\mathbf{b}_{\text{enr}}$ could then directly be compared to a $\mathbf{b}_{\text{test}}$ found from an overlapping speech segment.

It is proposed to estimate $\mathbf{b}$ for a single speaker audio fragment by averaging over all $\mathbf{u}_{tf}$. However, since only one speaker is active, there is no longer a need to estimate this on the time-frequency level. In fact, a separate linear output layer could be trained to estimate a single embedding every time step and these could then be averaged over all time steps to obtain $\mathbf{b}$ (or simply take the embedding of the last time step). Since the identification network, to estimate the speaker posteriors, would still be shared for both $S = 1$ and $S = 2$, $\mathbf{b}$ is expected to hold the same speaker information in both cases. However, this did not give improvements in our experiments. In the experiments of Section 4.2 $\mathbf{b}$ will thus be determined as in Figure 4.3 for $S = 1$ when using the joint model.

We do wish to clarify that in this section it is assumed that it is known whether a segment contains one or more speakers. Either this has to be provided as metadata, or a module has to be designed that estimates this. We leave it to further research to find such a model.

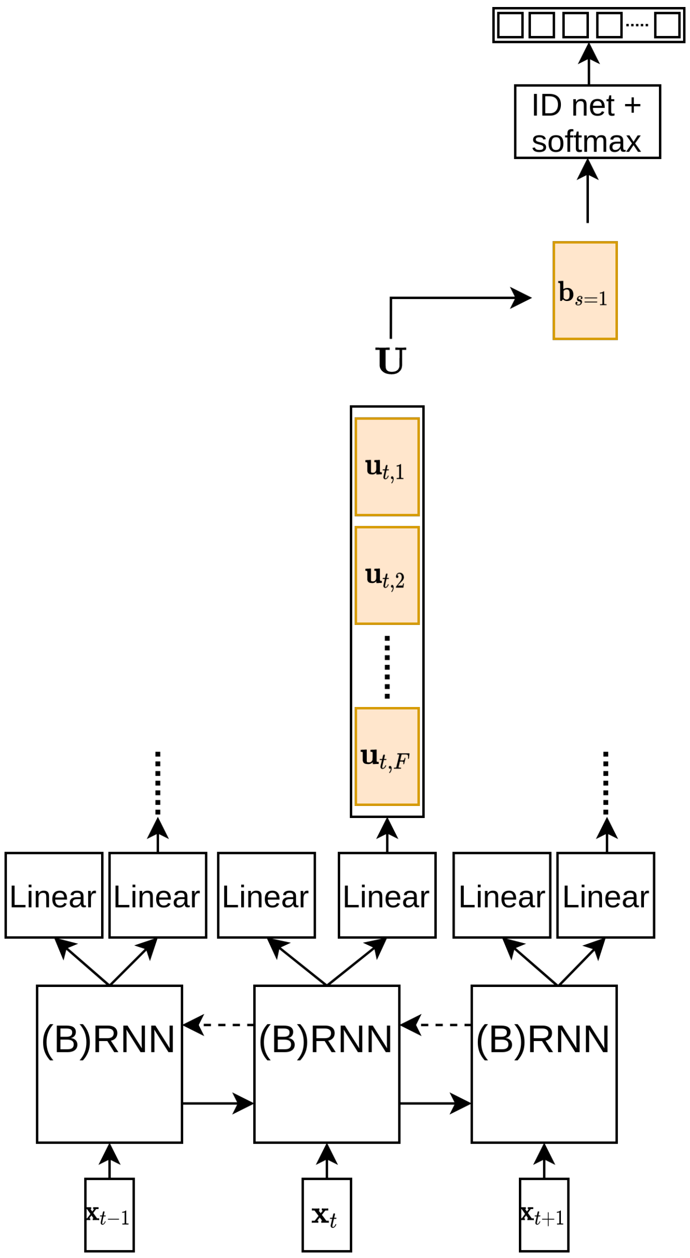


Figure 4.3: An illustration when using the joint model for $S = 1$.

4.2 Experiments

4.2.1 Experimental set-up

The BSSS part of the experimental set-up for the WSJ experiments (Sections 4.2.2 and 4.2.3) is the same as in the previous chapter (Section 3.3.1). For the SR part, the dimension of $\mathbf{u}_{tf}$ was chosen equal to $\mathbf{v}_{tf}$ ($D = D_v = D_u = 20$). For the identification network 2 feedforward layers with 100 units each were chosen. The output layer of this small identification network had $I_{\text{tr}} = 101$ units, one for each speaker in the training set.

In Section 4.2.4, the SRE data set will be used. This contains data from the NIST SRE 2004, 2005 and 2006 challenges, as well as the Fisher dataset and the Switchboard Cellular dataset[2]. It was specifically developed for SR experiments. The mixing of single speaker utterances to a mixture was done similarly as in Section 3.3.1. The single speaker utterances of SRE are typically a couple of minutes long, including large parts of silences. Therefore, they were first divided into segments with length between 7 and 15 seconds. Furthermore, mixtures were chosen with at least 60 % speech overlap. The training set consisted of 150 000 mixtures from $I_{\text{tr}} = 7702$ speakers. 20 000 mixtures were used as validation set and 15 000 mixtures were used for testing on held-out speakers.

To measure SR accuracy, the cosine similarity between an enrollment and an evaluation $\mathbf{b}$ is computed and thresholded to obtain the EER metric as discussed in Section 1.1.2. For all test mixtures[3], all $\mathbf{b}_s$ are determined. Random target and non-target trials are then generated such that each $\mathbf{b}_s$ has 5 positive trials and 5 negative trials[4]. It was found that mean and variance normalizing the 10 cosine similarities before thresholding, improved results. This is quite similar to the approach taken in [8]. However, this requires more enrollment data for each speaker and is therefore not always possible. The EER score obtained using score normalization will be given in brackets in Section 4.2.2.

After finishing all experiments it was found that excluding the SR loss for early stopping, improved results[5]. For Section 4.2.2, the experiments were redone by excluding the SR loss for early stopping. For Sections 4.2.3 and 4.2.4 the experiments were not redone due to time constraints. However, it is assumed that all conclusions still hold.

[2]This was based on the Kaldi [139] implementation for the NIST SRE2008 challenge [90]. The Linguistic Data Consortium codes of all datasets used are: LDC2006S44, LDC2011S01, LDC2011S04, LDC2011S09, LDC2004S13, LDC2005S13, LDC2001S13 and LDC2004S07

[3]3 000 for the WSJ experiments and 20 000 for the SRE experiments.

[4]Note that the same $\mathbf{b}_s$ can be used for enrollment and for evaluation in different SV trials.

[5]this was due to early stopping caused by the SR loss during the pretraining on 100 frame segments for the curriculum learning.

4.2.2 Joint BSSS and SR experiments

The following models have been trained and evaluated:

1. *Single speaker SR model*: A model trained on single speaker utterances using the speaker identification cross-entropy loss. At each time step t one SR embedding $\mathbf{u}_t$ is estimated by the linear output layer, which is then averaged over time to obtain $\mathbf{b}$.

2. *Sequential baseline*: Two models have been trained for the sequential baseline: a BSSS model and an SR model. The BSSS model is trained on overlapping speech using the DANet loss of (1.76). The SR model is trained on the speech estimates $|\hat{\boldsymbol{\mathcal{X}}}_s|$ of the BSSS model. Together, the two models of the sequential baseline, perform both BSSS and SR, but possibly in a less efficient manner than the joint model, although having roughly twice the amount of trainable parameters.

3. *Parallel baseline*: Similar to the *Sequential baseline* but the input of the SR model is the mixture spectrogram $|\boldsymbol{\mathcal{Y}}|$, rather than the speech estimates $|\hat{\boldsymbol{\mathcal{X}}}_s|$. $\mathbf{u}_{tf}$ is then estimated and used to determine $\mathbf{b}_s$, where $\hat{\mathbf{M}}_s$ from the BSSS part is used during evaluation. An alternative way of viewing this model is by comparing it to the *Basic joint model*, but without any shared parameters. Again, this baseline has roughly twice the amount of trainable parameters compared to the joint model.

4. *Basic joint model*: The model as explained in Section 4.1.1.

5. *Extended joint model*: The extended joint model as explained in the first paragraph of Section 4.1.3.

6. *$S = 1$ and $S = 2$ model*: The joint model that also allows for single speaker utterances as explained in the second paragraph of Section 4.1.3.

The SR and BSSS scores for these models are shown in Table 4.1. First, the SR performance (in EER) will be discussed. Only the raw scores without score normalization will be considered, but the same conclusions can be made for the normalized scores.

The most important thing to note is that in terms of EER, the *basic joint model* indeed performs better than the *sequential baseline* (-2.96% absolute and -23.5% relative), even though the sequential approach has roughly twice the amount of parameters compared to the joint model and the two parts of the sequential baseline are optimized individually. When comparing with the *parallel baseline*, it is found that the *basic joint model* again performs better (-0.39% absolute

Model	**enr. - test scenario**			**SDR**
	single - single	single - multi	multi - multi	
Single speaker SR	7.51 (4.87)	-	-	-
Sequential	-	-	12.57 (9.61)	8.60
Parallel	-	-	10.00 (6.94)	8.70
Basic joint	-	-	9.61 (6.25)	8.50
Extended joint	-	-	8.89 (6.04)	8.62
$S = 1$ and $S = 2$	4.34 (2.39)	8.44 (4.44)	10.34 (7.00)	8.63

Table 4.1: EER (in %) for different enrollment and testing scenarios. The results in brackets are obtained after score normalization. The last column shows the BSSS SDR performance (in dB), evaluated on the test scenario with multiple speakers ($S = 2$). The joint models were trained by excluding the SR loss for early stopping.

and -3.9% relative). It is also noted that the proposed *extended joint model* substantially outperforms the *basic joint model* (-0.72% absolute and -7.5% relative). It thus seems indeed important to consider the imperfect BSSS mask estimates for SR. As new BSSS approaches become better and $\hat{\mathbf{Z}}$ more closely resembles $\mathbf{Z}$, the difference between the *extended joint model* and the *basic joint model* (as well as the *sequential baseline* and the *parallel baseline* for a matter of fact), is expected to decrease. However, as state-of-the-art BSSS performance improves, more challenging scenarios like noisy or reverberant conditions can be considered and the difference between $\hat{\mathbf{Z}}$ and $\mathbf{Z}$ will again become larger.

None of the considered models in a multi-speaker enrollment setting with a multi-speaker test setting, perform as well as the *single speaker SR model* in a single-speaker enrollment setting with a single-speaker test setting. This is expected as in the multi-speaker scenario the interfering speaker makes it more difficult to estimate the identity of the target speaker.

When using the $S = 1$ *and* $S = 2$ *model* in a multi-speaker enrollment setting with a multi-speaker test setting, an increase in EER is noted compared to the *basic joint model* (+0.73% absolute and +7.6% relative). However, this model allows enrollment in a single-speaker setting to then later be tested in a multi-speaker setting. If single-speaker enrollment is possible, this indeed substantially improves EER performance (-1.17% absolute and -12.2% relative).

It is remarkable that the $S = 1$ *and* $S = 2$ *model* considerably outperforms the *single speaker SR model* in a single-speaker enrollment setting with a single-speaker test setting. This might indicate overfitting of the *Single speaker SR model.* In fact, by tuning some model hyper parameters, to reduce the number of trainable parameters, more than 1% absolute EER reduction could be obtained for the *single speaker SR model*, which is still worse than the *$S = 1$ and $S = 2$ model*. Possibly, further EER reduction would be possible for the *single speaker SR model* with more tuning or regularization. Nonetheless, it remains an interesting observation that including multi-speaker data in the model helps regularize and improve performance for the single-speaker case.

When considering the SDR measure in Table 4.1, it is noticed that joint models do not improve compared to the baseline models.

The joint model clearly improves SR performance compared to the sequential approach. An extension to this model gives even further gains. It seems important for SR in a multi-speaker setting to know which parts of the audio signal belong to the same speaker and have to be considered jointly to estimate that speaker's identity. Apparently, an estimate of this does not suffice and it is better to approach the problem jointly. However, the same cannot be said for BSSS performance, where the performance of the joint model is on-par with the model that was trained towards BSSS only. This is consistent with the observation of Chapter 3 where explicitly adding speaker information to the input of the BSSS model, had little impact.

Similarly as in Chapter 3 one can question whether this conclusion still holds in more challenging scenarios. Therefore, the analysis is redone where the BLSTM part of the model only has 50 units per layer, compared to the 600 units for the results in Table 4.1. It was found that the baseline 50 units model achieves an SDR of 6.27 dB, while the basic joint 50 units model achieves an SDR of 6.71 dB or a 0.44 dB absolute improvement. We thus conclude, consistent with Chapter 3, that in a more challenging case the BSSS can be improved with an auxiliary SR task.

4.2.3 Joint learning experiments

In this section the design choices in multi-task learning (Section 4.1.2) will be considered. Three alternatives were discussed in Section 4.1.2: summing the losses of the tasks (equivalent to summing the gradients of the tasks) via (4.5), summing the individual parameter updates of the tasks via (4.9) and an additional normalization and dependency check via (4.10). In all cases a meta parameter α can be used to indicate the relative importance between the tasks. Due to the scaling ambiguity, the α for the SR loss was set to 1 and the α for the BSSS loss was optimized. A higher α will thus give more importance to the BSSS task. Instead of using (4.5) directly, (4.6) was used where $\hat{m}_j$ was set equal to the validation task losses of a model trained with (4.9) and with $\alpha = 1$.

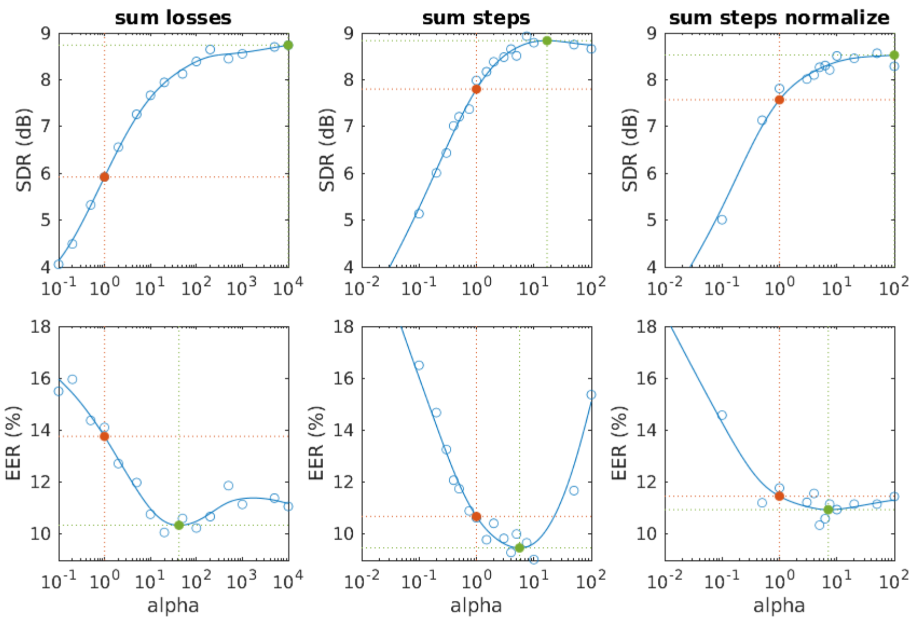


Figure 4.4: SDR and EER performance for the three training methods when training the *basic joint model*. ‘sum losses’, ‘sum steps’ and ‘sum steps normalize’ correspond to (4.6), (4.9) and (4.10), respectively. The values obtained for $\alpha = 1$ are indicated in red, while the values obtained for the optimal α are indicated in green.

SDR and EER performances, for all three methods, are shown in Figure 4.4 when training the *basic joint model*, using different α. The same is done in Figure 4.5 for the *extended joint model*. The values obtained for $\alpha = 1$ are indicated in red, while the values obtained for the optimal α are indicated in green. The names ‘sum losses’, ‘sum steps’ and ‘sum steps normalize’ are used to refer to (4.6), (4.9) and (4.10), respectively.

Performance in Figure 4.4 (*basic joint model*) and Figure 4.5 (*extended joint model*) is similar but the EER is generally better for the *extended joint model*, as was expected from Table 4.1. When comparing the three multi-task learning methods, using for instance Figure 4.5, it is noted that they achieve similar performance for optimal α. Furthermore, for ‘sum steps’ and ‘sum steps normalize’, the performance when $\alpha = 1$ (red) is close to the performance of the optimal α (green). However this is clearly not the case for ‘sum losses’. This shows that summing losses of the tasks together, makes the joint model much more dependent on a good choice for α, while this is much less the case

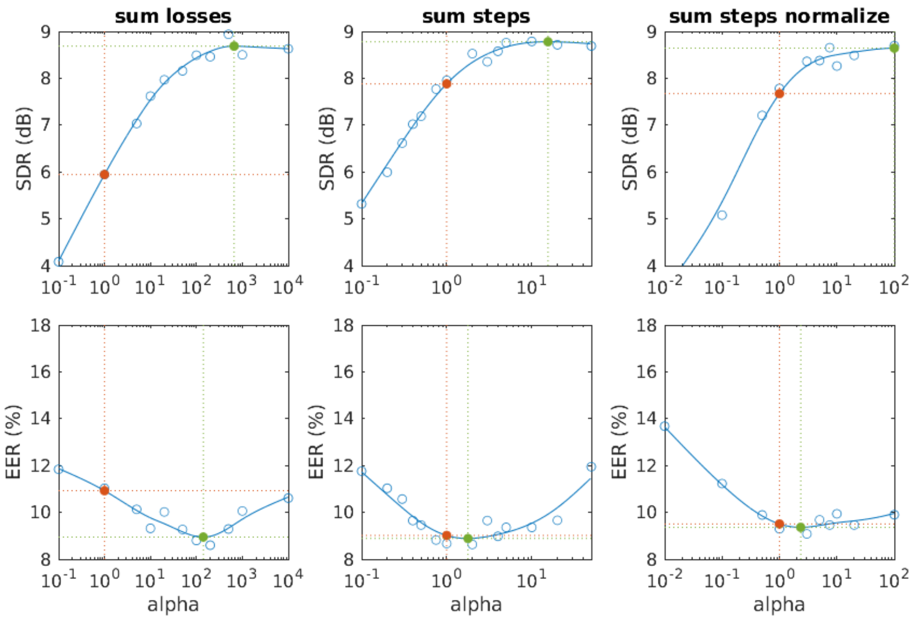


Figure 4.5: SDR and EER performance for the three training methods when training the *extended joint model*. 'sum losses', 'sum steps' and 'sum steps normalize' correspond to (4.6), (4.9) and (4.10), respectively. The values obtained for $\alpha = 1$ are indicated in red, while the values obtained for the optimal α are indicated in green.

for 'sum steps' and 'sum steps normalize'. Finally, it is noted that for high α performance substantially degrades for 'sum steps', while this is not the case for 'sum steps normalize', making the latter preferred when losses are independent on parts of the model's parameters.

When further focusing, on for example Figure 4.5 'sum steps normalize', similar observation as in Table 4.1 can be made. It is beneficial for SR to be trained together with BSSS, since for low α (more importance towards the SR task) an increase in EER is observed. Note that when $\alpha \rightarrow 0$, the shared part of the model over both tasks, is only optimized towards SR. However, it again seems that BSSS does not need to be jointly trained with SR for optimal performance, as the best SDRs are achieved for $\alpha \rightarrow \infty$.

Data set	**I_{tr}**	**# tr.**	**Model**	**EER**		**SDR**
				single	multi	
WSJ-2spk	101	20k	Single speaker SR	7.51	-	-
			Seq. baseline	-	12.57	8.60
			Basic joint	-	10.43	8.07
SRE-2spk	101	20k	Single speaker SR	13.53	-	-
			Basic joint	-	16.64	7.45
	7702	20k	Basic joint	-	18.79	6.92
		150k	Single speaker SR	8.49	-	-
			Seq. baseline	-	11.67	11.78
			Basic joint	-	11.39	10.92
			Basic joint $D_u = 150$	-	10.06	10.90
			Ext. joint $D_u = 150$	-	9.78	10.74

Table 4.2: EER (in %) and SDR (in dB) for different data sets and training set-ups.

4.2.4 Experiments for generalization of SR

The results in Section 4.2.2 were obtained from models trained on a speaker set of $I_{tr} = 101$ speakers. This is considered quite small for SR [90]. Nevertheless, decent EER scores were obtained. However, it must be noted that the WSJ0 data was recorded in an ideal studio environment from read speech. The NIST SRE challenges [90] include spontaneous speech from telephone conversations and generally include more speech and speaker diversity compared to WSJ. In this section, experiments are done using the NIST SRE dataset to study the capabilities of the joint models in more realistic settings. Experiments were done for different number of training speakers I_{tr} and different number of training mixtures and are shown in Table 4.2.

Firstly, when keeping the number of training speakers ($I_{tr} = 101$) and the number of training mixtures (20 000) fixed, it is observed that the SRE-2spk models perform much worse than the WSJ-2spk models in terms of EER (+6.21% absolute and +59.5% relative)[6]. Simply including many more speakers ($I_{tr} = 7702$) does not give performance improvement. Only when also the number of training mixtures is increased (150 000), do the SRE-2spk models achieve similar SR performance as the WSJ-2spk models. From this it is concluded that, when sufficient and diverse training data is provided, the joint approach is also suitable for more challenging and realistic scenarios. Further gains in EER can be achieved by increasing the $\mathbf{u}_{tf}$ embedding dimension to $D_u = 150$ and by using the extended joint model from Section 4.1.3.

[6]The test sets for both models are also different.

4.3 Conclusion

A joint DNN model was built for BSSS and SR. The SR performance improved compared to a sequential and a parallel baseline. By including separation uncertainty for the SR task, further improvements were obtained. However, the joint model was not beneficial in terms of separation performance, consistent with the observations in Chapter 3. It is again noted that BSSS is performed in highly idealized settings, e.g. no reverberation, no background noise and a large amount of trainable parameters. It was found that for a small model, considerable improvements in separation performance could be found. Again, this is consistent with the observation in Chapter 3, where it was found that a small BSSS model does not sufficiently build an internal characterization of the active speakers.

Different styles of joint learning were compared and it was concluded that aggregating the weight updates of the tasks is preferred to summing the losses or the gradients. Finally, SR experiments were done on the more challenging SRE-2spk dataset and with sufficient training data, similar performance can be achieved compared to the WSJ dataset.

Chapter 5

Analysis of Memory in RNNs for Blind Speech Source Separation

This chapter is based on the following publications:
ZEGERS, J., AND VAN HAMME, H. Memory time span in LSTMs for multi-speaker source separation. In *Interspeech 2018* (2018), ISCA, pp. 1477–1481.
ZEGERS, J., AND VAN HAMME, H. Analysis of memory in LSTM-RNNs for source separation, 2020.

One of the major drawbacks of DNNs is that due to their nested and non-linear structure, it is difficult to understand what makes them arrive at their predictions and therefore they lack explainability [149]. This chapter will look for insights in the memory of the LSTM-RNN. Two novel methods will be presented that are generally applicable to study the memory of RNNs, or more specifically memory time spans. The first approach, called *the leaky approach*, is easy to implement and the additional computational cost compared to a standard LSTM-RNN is negligible. The disadvantage of this method is that timings of memory spans are only approximated. This is in contrast with *the reset approach*, where timings are exact, but at the cost of higher computation complexity. Both methods will be applied to the task of BSSS to determine to which extent SR and other factors play a role in a successful separation.

5.1 Introduction

In Section 1.4.1 it was explained how RNNs are suited for sequential processes like speech. It was also mentioned that standard RNNs have difficulty with learning long-term dependencies due to the exploding and vanishing gradient problem, making their memory time span (the duration for which information is kept in memory) relatively short. This problem is countered with the *constant error carousel* principle in LSTM-RNNs. It has been shown that LSTM-RNNs can learn long-term dependencies of over a thousand time steps for simple, artificial tasks [82]. However, it can be expected that this memory time span depends on the network architecture as well as on the complexity of the data and the task. It is thus unknown how long the memory time span for LSTM cells is for real and complex tasks like speech processing.

The aim of this chapter is to examine the time span and importance of internal dynamics in the LSTM memory. This will be done by removing information from the LSTM memory over time. To the best of our knowledge, three different approaches exist that allow for the removal of information over time: the leaky approach, the reset approach and the segment approach. The difference between these approaches will be discussed in Section 5.2.

By gradually reducing this memory removal frequency, or reset frequency, bigger memory time spans are allowed. The networks are evaluated for different reset frequencies and the task performance differences can be used to assess the importance of the different memory spans. Furthermore, it is possible to use different memory spans for different layers in the LSTM-RNN. This allows to confirm or reject the hypothesis that deeper layers in RNNs bring higher level abstractions of the data and therefore use a bigger time span [33]. Finally, different memory spans can be used for the forward and backward direction of a bidirectional LSTM-RNN which allows to distinguish between the importance of the directions.

The task of BSSS seems well suited for this analysis as it has been shown that both long-term and short-term effects are important [196]. Nonetheless, the proposed methodology can be applied to any task using RNNs. Specifically, we would like to answer the following research questions with regards to BSSS:

- Which order of time spans are important when using an LSTM-RNN for BSSS and can time spans be linked with descriptions of speech like phonetics, phonotactics, lexicon, prosody and grammar?
- Since it is has been shown in Chapters 3 and 4 that speaker characterization is relevant for the task, can we find the amount of context necessary for

the LSTM-RNN to sufficiently characterize the speakers in overlapping speech?

- For BSSS, do we observe the same hierarchical property that deeper layers have longer time dependencies, as was found by [33]?
- In bidirectional LSTM-RNNs, would either direction be more important than the other for BSSS?

The rest of this chapter is organized as follows. In Section 5.2 an overview of memory constraining methods for RNNs will be given. The leaky approach is described in Section 5.3, followed by some leaky experiments in Section 5.4. Similarly, the memory reset LSTM cell is described in Section 5.5 and is again followed by the relevant experiments in Section 5.6. A final conclusion is given in Section 5.7.

5.2 Memory constraining methods

Three different approaches exist that allow for the removal of memory information over time in an RNN: the leaky approach, the reset approach and the segment approach.

The leaky approach is a novel method to assess the relative importance of different memory span lengths. A fraction of the LSTM cell state is leaked, on purpose, at every time step. This forces the LSTM to forget information of the past over time. The leaky approach can be seen as a soft reset compared to the hard reset of the reset approach discussed below. The amount of computations in a leaky LSTM cell remains unchanged, while the computational load, for both the reset and segment approach, scale with the width of the memory span. The leaky approach is thus more interesting from a computational standpoint, but since the reset is soft, the timings found will not be exact.

In the reset approach, the state of the LSTM cell is reset at particular time intervals. It has a similar concept as the leaky approach, but rather than gradually leaking memory over time, it applies a hard reset at a certain reset frequency. This has the advantage that the timings found are exact. To our knowledge [160] is the only work where a similar reset approach to ours is given. In their paper a multi-stream system with an LSTM component for video action detection is described. However, they only consider a single unidirectional LSTM layer. In this chapter we extend to bidirectional multi-layer LSTMs where we allow layer dependent reset periods (Section 5.5.2) and a method to reduce computational burden for longer reset periods (Section 5.5.3). This makes the state reset algorithm far more complex, yet flexible compared to the unidirectional single layer reset. Furthermore, the memory reset in [160] was done during testing only, causing a train-test mismatch. In this work memory reset will be done during training and testing.

A final approach, called the segment approach in this chapter, has a similar aim to restrict the LSTM memory span [122, 26, 170]. Segments are created by shifting a window by one time step over the input data. Each segment is passed through the LSTM and the output of the LSTM at the last time step within the segment is retained. The output of each segment has been produced with an LSTM with a memory span equal to the length of the window. It has been verified in our experiments that the reset approach and the segment approach give the same results.

The segment approach was first applied in [122] on the spectral input of an ASR task. They found that the Word Error Rate (WER) of the acoustic LSTM-RNN saturated relatively quickly. Therefore it was concluded that the main strength of the RNN is the frame-by-frame processing rather than the ability to have a large memory span. However, we found that for the task of BSSS long-term dependencies were, in fact, important for the separation quality.

The segment approach was also used on a language modeling task in [26] and [170], where it was claimed that perplexity scores did not improve by increasing the memory span over 40 time steps (words). Similarly, WER converged at 20 time steps. In both works the number of different memory lengths evaluated was rather limited and they were mainly interested in the performance saturation point. There was no analysis on the importance of different time scales within the model.

The above approaches focus on analysis of the memory span by limiting memory capabilities. There are multiple works that give no explicit in-depth time analysis but adapt or restrict the memory span of the RNN in order to improve performance on a task. In [33] and [60] the soft reset of the forget gate was replaced with a hard reset implementation. The network tries to learn the optimal reset frequency. Other approaches, like the clockwork RNN, have different, fixed update frequencies for different cells in the network. Memory restrictions are made on a cell level, rather than on a network or layer level. The idea is that some cells should focus on long-term effects and some should focus on short-term [99, 5, 128].

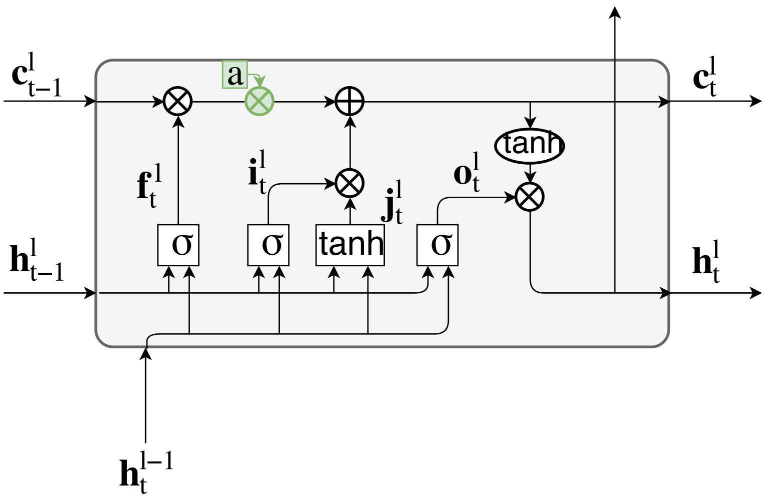


Figure 5.1: Schematic of a leaky LSTM. The only difference with Figure 1.9a, is the addition of the leaky scalar a.

5.3 Deep Leaky LSTM

A leaky LSTM is an LSTM that is designed to leak cell state memory. First, the architecture of the leaky LSTM is explained. Afterwards, a description is given of the memory flow in a deep network with LSTM-RNNs and how this flow can be controlled.

5.3.1 The leaky LSTM

A leaky LSTM is a regular LSTM where the forget gate is multiplied by a positive constant a smaller than 1 such that not all cell state information from the previous time steps can be retained, as $\mathbf{0} \leqslant a\mathbf{f}_t^l \leqslant a\mathbf{1} < \mathbf{1}$, and thus part of the memory is leaked. (1.72) changes to

$$\mathbf{c}_t^l = \mathbf{c}_{t-1}^l \odot a\mathbf{f}_t^l + \mathbf{j}_t^l \odot \mathbf{i}_t^l, \tag{5.1}$$

as is indicated in Figure 5.1.

In (5.1) only the first term seems to contain temporal information since only $\mathbf{c}_{t-1}^l$ has the $t-1$ subscript. The component of the cell state containing temporal information from time $t-1$ can thus be described as

$$\bar{\mathbf{c}}^l_{t,t-1} = \mathbf{c}^l_{t-1} \odot a\mathbf{f}^l_t. \tag{5.2}$$

The maximal information transfer will be achieved when $\mathbf{f}^l_t = \mathbf{1}$. In this case the contribution from time $t - \Delta T$ in the current cell state is given by

$$\begin{aligned} \bar{\mathbf{c}}^l_{t,t-\Delta T} &= a^{\Delta T}\mathbf{c}^l_{t-\Delta T} \\ &= e^{\Delta T/\tau}\mathbf{c}^l_{t-\Delta T}, \end{aligned} \tag{5.3}$$

where the last equality is written in exponential decay form with $\tau = -1/\log(a)$. τ is called the lifetime and when $\Delta T = \tau$ 69% of the initial value has been lost, which will be used as a rough estimate for the allowed time span of the memory.

5.3.2 Temporal information flow

It was claimed that temporal information in (5.1) was only present in the $\bar{\mathbf{c}}^l_{t,t-1}$ term. However, the second term in this equation has two factors that indirectly depend on previous time steps via (1.69) and (1.71). For instance, if $\mathbf{c}^l_{t-1} = \mathbf{1}$, $\mathbf{f}^l_t = \mathbf{1}$ and the network would like $\mathbf{c}^l_t = \mathbf{c}^l_{t-1} + \Delta\mathbf{c}^l_t$, it could obtain this via (5.1) by making $\mathbf{j}^l_t \odot \mathbf{i}^l_t = (1 - a) + \Delta\mathbf{c}^l_t$ (with the restriction that $-\mathbf{1} \leqslant \mathbf{j}^l_t \odot \mathbf{i}^l_t \leqslant \mathbf{1}$). This information flow is indicated with the red arrow in Figure 5.2. This is an unwanted flow since it bypasses the memory leakage. This flow can be stopped by cutting the red dotted connections and thus removing the temporal information in $\mathbf{j}^l_t$ and $\mathbf{i}^l_t$. (1.69) and (1.71) then become

$$\mathbf{i}^l_t = \sigma(\mathbf{W}^l_i\mathbf{h}^{l-1}_t + \mathbf{b}^l_i) \tag{5.4}$$

$$\mathbf{j}^l_t = \tanh(\mathbf{W}^l_j\mathbf{h}^{l-1}_t + \mathbf{b}^l_j), \tag{5.5}$$

respectively.

However, in these equations $\mathbf{h}^{l-1}_t$ can also contain temporal information for $l \geqslant 2$ via (1.73) and (1.69,1.71,1.70,1.72) as is indicated by the blue arrow in Figure 5.2. This second order flow can be stopped by also cutting the blue dotted connections. (5.4) and (5.5) are retained and (1.68) and (1.70) become

$$\mathbf{f}^l_t = \sigma(\mathbf{W}^l_f\mathbf{h}^{l-1}_t + \mathbf{b}^l_f) \tag{5.6}$$

$$\mathbf{o}^l_t = \sigma(\mathbf{W}^l_o\mathbf{h}^{l-1}_t + \mathbf{b}^l_o) \tag{5.7}$$

respectively.

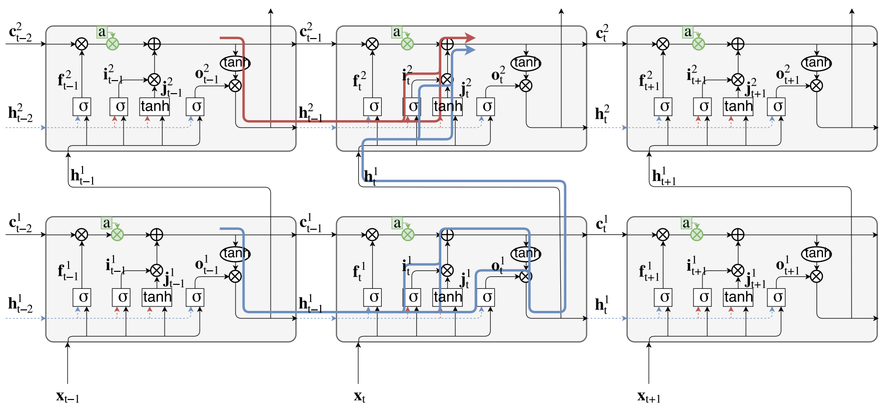


Figure 5.2: Schematic of flow of temporal information in a bidirectional deep leaky LSTM-RNN and how it can be interrupted by cutting connections. The red line shows how recurrent information can bypass the leakage and how this can be countered by removing the red dashed connections. Similar for blue.

Memory leakage has only been considered on $\mathbf{c}_t^l$ while a standard RNN has no cell state and is also able to memorize (even though less effective) via it cell's output $\mathbf{h}_t^l$. Cutting the blue and red connection also removes this temporal flow as $\mathbf{h}_{t-1}^l$ is never used.

Note that the cutting of recurrent connections is only done to gain more control over the memory leaking process to better define the memory time span. It is not intended to gain performance for the presented task. The research question could be generalized to exploring temporal information used in RNNs with LSTM-like cells.

5.4 Leaky LSTM experiments

Networks were trained and tested for different values of a and for the different architectures as depicted by the dotted cuts in Figure 5.2. The same experimental set-up as in Section 3.3.1 is used. There it was mentioned that an STFT with a 32 ms window length and a hop size of 8 ms were used. This means that $\tau = -1/\log(a) * 8\text{ms}$, when expressed in milliseconds. Some networks were given the 10-dimensional oracle i-vectors of both single speech signals used in the mixture (see Chapter 3). Results are shown in Figure 5.3.

For all curves a rapid decrease in performance is found for $\tau < 100$ ms. Here the

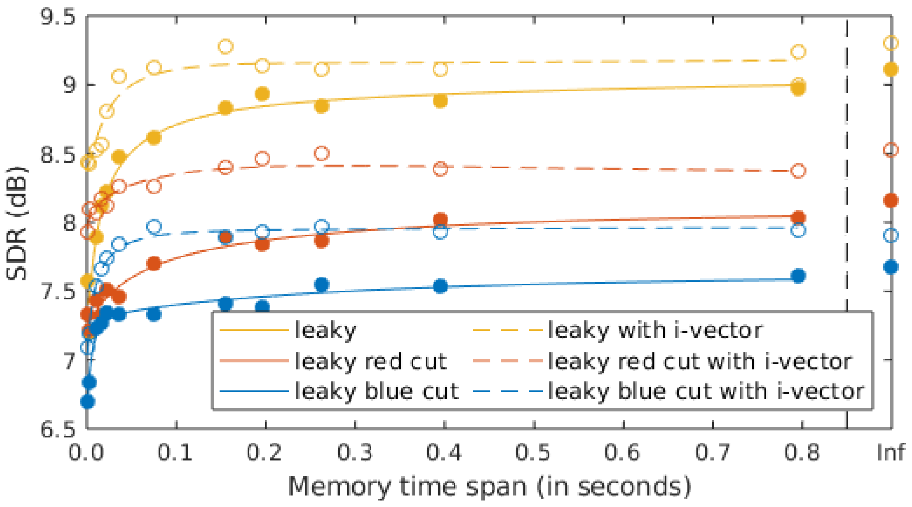


Figure 5.3: BSSS results with LSTM leakage. The memory time span is given by the lifetime $\tau = -1/\log(a) * 8$ ms.

leaky LSTMs might have difficulty with tracking the formants of the 2 speakers which span roughly 100 ms [65, 171]. For $\tau > 100$ ms the networks without i-vectors steadily keep increasing in performance, while the networks with i-vectors have a higher performance which is roughly constant. This seems to indicate that with a bigger time span, the networks without i-vectors manage better to find their own internal speaker representations. For no leakage ($\tau = \infty$) an i-vector is still a better speaker representation than the internal representation in the LSTM. This small difference was also observed in Chapter 3 for oracle i-vectors of dimension 10. When leakage is absolute ($\tau = 0$), performance is better when i-vectors are used, indicating that speaker information aids separation if no context is given and assignment is more consistent over frames.

Since the curves for the LSTMs with i-vectors are flat between $\tau = 100$ ms and $\tau = 300$ ms the network seems to give little importance to phonotactic and lexical information for the task of BSSS.

The shapes of the curves for all three types of LSTMs are quite similar, indicating that the bypass mechanisms of temporal information flow as described in section 5.3.2 are limited. The performance drop when cutting connections is expected as the model capacity is reduced by removing parameters (see Table 5.1) and LSTM gates are more difficult to control.

Experiments were also done where no leakage was applied during training but only during testing. Results differed from those shown in Figure 5.3. For

	Basic	Red cut	Blue cut
$a > 0$	15.25	12.37	9.49
$a = 0$	12.21	9.33	7.89

Table 5.1: The number of trainable parameters in the network in millions. Note that for $a = 0$, the forget gate is effectively removed and so are its trainable parameters.

example, no temporal information can be used if the LSTM with blue cuts and full leakage ($\tau = 0$) is used at test time. However, if the network was trained without leakage the SDR was found to be 4.7 dB which is 2.0 dB below the performance obtained when it is also trained with full leakage, although both networks have no memory during testing. This indicates it is indeed necessary to apply leakage also at training time to avoid mismatch.

5.5 Memory reset LSTM

In this section, the memory reset LSTM cell is introduced. Afterwards, it is described how this memory reset LSTM cell can be used in an RNN. Derivations are given in Appendix B. Finally, we discuss how computational costs for experiments can be reduced by using the grouped memory reset approach.

5.5.1 Memory reset LSTM cell

To limit the recurrent information, the cell state $\mathbf{c}_t^l$ and hidden units $\mathbf{h}_t^l$ will be reset using a fixed reset period T_{reset}. This assures that only information of the last T_{reset} frames can be used (or $T_{\text{reset}} - 1$ context frames). However, the time frame after such a reset would see no recurrent information at all. In fact, at every time step, an output should be produced based on T_{reset} frames of information. To achieve this, $K = T_{\text{reset}}$ different instances of the cell state and hidden units are kept, each reset at different moments in time. The instance that will be reset at time t is the instance k_t^* for which

$$k_t^* = t \bmod K, \tag{5.8}$$

with $t = 0, \dots, T - 1$.

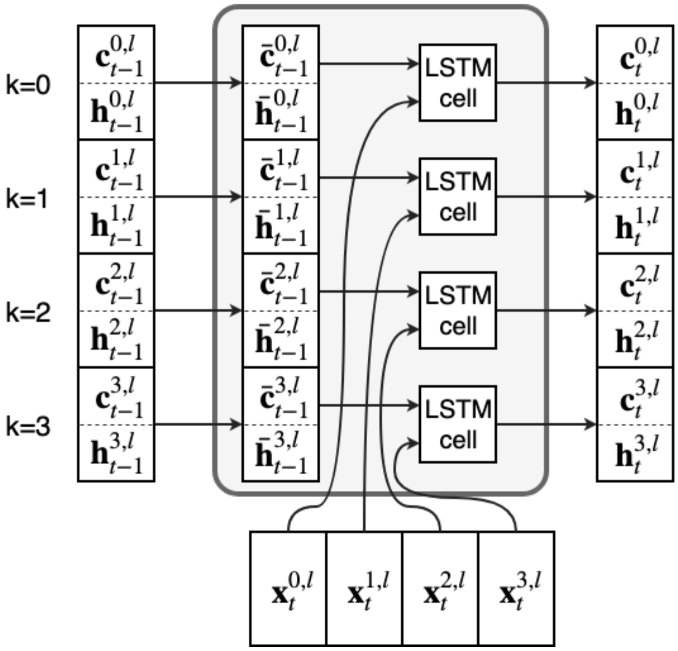


Figure 5.4: Illustration of a memory reset LSTM cell with $T_{\text{reset}} = 4$. It keeps $K = 4$ instances of the hidden units and cell state, which can be reset according to (5.9). The LSTM cell is used to process each instance.

The reset operation can be formulated as

$$\left(\bar{\mathbf{h}}_{t-1}^{k,l}, \bar{\mathbf{c}}_{t-1}^{k,l}\right) = \begin{cases} (\mathbf{0}, \mathbf{0}), & \text{if } k = k_t^* \text{ or } t = 0 \\ \left(\mathbf{h}_{t-1}^{k,l}, \mathbf{c}_{t-1}^{k,l}\right), & \text{otherwise} \end{cases}, \tag{5.9}$$

with $k = 0, \ldots, K-1$. (1.68)–(1.73) are updated to (5.10)–(5.15).

$$\mathbf{f}_t^{k,l} = \sigma(\mathbf{W}_f^l \mathbf{x}_t^{k,l} + \mathbf{R}_f^l \bar{\mathbf{h}}_{t-1}^{k,l} + \mathbf{b}_f^l), \tag{5.10}$$

$$\mathbf{i}_t^{k,l} = \sigma(\mathbf{W}_i^l \mathbf{x}_t^{k,l} + \mathbf{R}_i^l \bar{\mathbf{h}}_{t-1}^{k,l} + \mathbf{b}_i^l), \tag{5.11}$$

$$\mathbf{o}_t^{k,l} = \sigma(\mathbf{W}_o^l \mathbf{x}_t^{k,l} + \mathbf{R}_o^l \bar{\mathbf{h}}_{t-1}^{k,l} + \mathbf{b}_o^l), \tag{5.12}$$

$$\mathbf{j}_t^{k,l} = \tanh(\mathbf{W}_j^l \mathbf{x}_t^{k,l} + \mathbf{R}_j^l \bar{\mathbf{h}}_{t-1}^{k,l} + \mathbf{b}_j^l), \tag{5.13}$$

$$\mathbf{c}_t^{k,l} = \bar{\mathbf{c}}_{t-1}^{k,l} \odot \mathbf{f}_t^{k,l} + \mathbf{j}_t^{k,l} \odot \mathbf{i}_t^{k,l}, \tag{5.14}$$

$$\mathbf{h}_t^{k,l} = \tanh(\mathbf{c}_t^{k,l}) \odot \mathbf{o}_t^{k,l}. \tag{5.15}$$

A visualization for $K = 4$ is given in Figure 5.4.

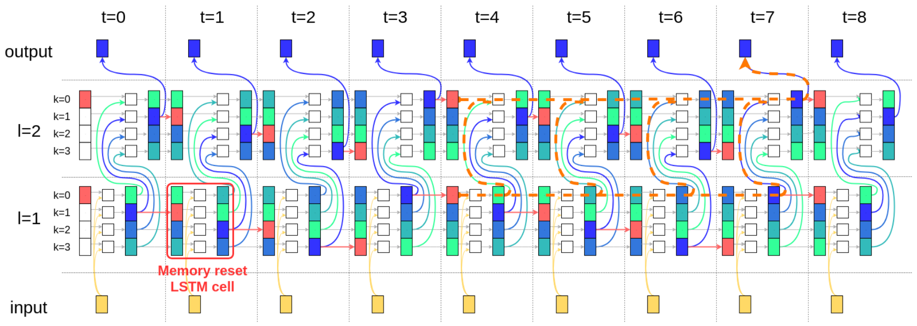


Figure 5.5: Example of a unidirectional memory reset LSTM-RNN with $L = 2$, $K = 4$ and $T = 9$. Color coding is according to (5.16). An instance is colored red when it is reset ($k = k_t^*$, according to (5.8)). Connections between layers are according to (5.17). The final output follows (5.18). When following the orange dashed line, indicating the data dependencies, backward, one can verify that indeed exactly K frames are used to produce an output.

5.5.2 Deep memory reset LSTM-RNN

For multi-layer memory reset LSTM-RNNs, instances need input from instances of the layer below. In other words, an equivalent for (1.61) has to be found for the memory reset LSTM-RNNs. We introduce a new variable $\tau_t^{k,l}$ which is equal to how long ago instance k of layer l was last reset (or how many context frames instance k of layer l considers at time t). (B.2) shows that[1]

$$\tau_t^{k,l} = (t - k) \bmod K. \tag{5.16}$$

The value of $\tau_t^{k,l}$ is color coded in Figure 5.5 for $K = 4$. If an instance is colored light green, then $\tau_t^{k,l} = 0$. If an instance is colored dark blue, then $\tau_t^{k,l} = 3$ $(= K - 1)$. An instance should receive input from the instance of the layer below with the same number of context frames $\tau_t^{k,l}$. The orange dashed line in Figure 5.5 shows that this way no information further than K frames can be used. (B.3) shows that for a unidirectional memory reset LSTM-RNN, this is obtained when instance k receives input from instance k from the layer below. (1.61) generalizes to

$$\mathbf{x}_t^{k,l} = \mathbf{h}_t^{k,l-1}. \tag{5.17}$$

We introduce a new simplified notation $k' \leftarrow k''$, stating that instance k' of layer l receives input from instance k'' of layer $l - 1$. In this notation (5.17)

[1]The context is restricted at the edges of the data sequence. $\tau_t^{k,l}$ can never exceed t. This is not a restriction of the memory reset approach but intrinsic to the data sequence.

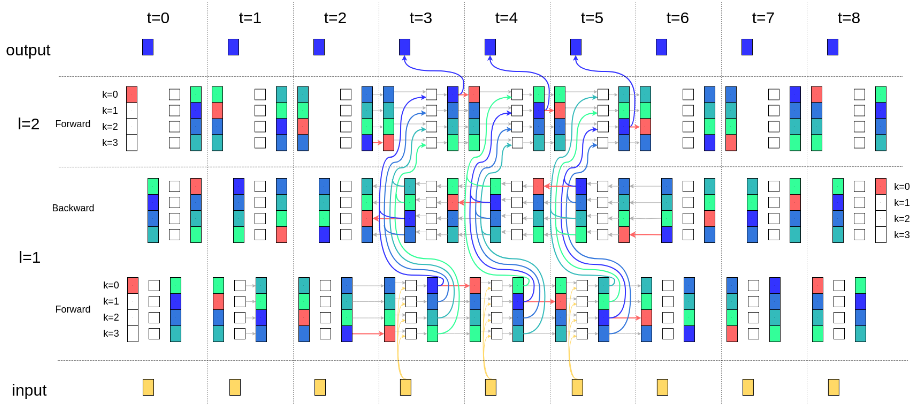


Figure 5.6: Similar to Figure 5.5 but for bidirectional memory reset LSTM-RNN. To prevent cluttering of the image, only the forward direction of the second layer is shown and connections are only drawn for $t = \{3, 4, 5\}$.

becomes $k \leftarrow k$. The final output of the network at time t is the instance with the maximum number of context frames at that time. This is the instance that will be reset at time $t+1$ (see (B.4)). (1.63) generalizes to

$$\mathbf{h}_t = \mathbf{h}_t^{k^*_{t+1},L}. \tag{5.18}$$

For bidirectional LSTM-RNNs we apply the same equal context method to find the following connections, replacing (1.65)

$$\overrightarrow{k} \leftarrow \begin{bmatrix} \overrightarrow{k} \\ ((T-1) - 2t + \overleftarrow{k}) \bmod K \end{bmatrix}, \tag{5.19}$$

$$\overleftarrow{k} \leftarrow \begin{bmatrix} (-(T-1) + 2t + \overrightarrow{k}) \bmod K \\ \overleftarrow{k} \end{bmatrix}. \tag{5.20}$$

(5.19)-(5.20) are derived in (B.13)-(B.14) and can be verified in Figure 5.6.

The reset period T_{reset} needs not be the same for every layer. For instance, we could allow the lower levels to operate on short-term information and let the higher layers cope with the long-term dependencies. By connecting an instance with the correct instance of the previous layer at every time t, we can still make sure the the number of context frames per layer is limited to a chosen value. (5.19)-(5.20) generalize to (B.19)-(B.21).

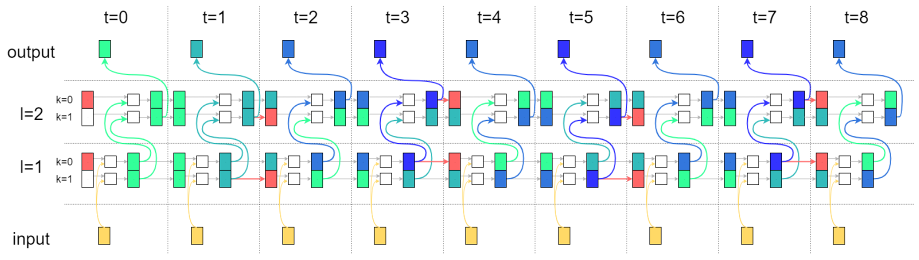


Figure 5.7: Example of a unidirectional memory reset LSTM-RNN using groups with $L = 2$, $K = 2$, $G = 2$ and $T = 9$. $T_{\text{reset}} = 4$, just as in Figure 5.5. Color coding is according to (B.23). An instance is colored red when it is reset ($k = k_t^*$, according to (5.21)).

5.5.3 Grouped memory reset LSTM

Until now, a different instance is reset at every time step such that each instance is reset every K time steps. The number of instances K is therefore equal to the reset period T_{reset} and the computational requirements for experiments grow as the reset period (or the memory span) becomes larger. By using the reset operation only every G time steps, an instance will only be reset every KG time steps and thus the reset period becomes $T_{\text{reset}} = KG$. Then the number of instances can be reduced with a factor G, for the same T_{reset}. (5.8) is changed to

$$\begin{cases} k_t^* = (t/G) \bmod K, & \text{if } t \equiv 0 \pmod{G} \\ \text{no reset} & \text{otherwise} \end{cases} \tag{5.21}$$

A visualization of the grouped memory reset approach is given in Figure 5.7. The downside is that instead of allowing the LSTM to use $T_{\text{reset}} = KG$ frames of input, it will use between $KG - (G - 1)$ and KG frames of input, as shown in (B.32)-(B.33) (also see output layer in Figure 5.7). However this need not be a concern, since the computational problems arise only for large number of instances and then $KG >> G$. A similar approach was taken in [60] where *frame grouping* for the segment approach was used to reduce the computational burden. Derivation for the connections between grouped memory reset LSTM layers is given in Appendix B.2.

5.6 Memory reset experiments

The memory reset LSTM-RNN is used to gain insights in the importance of the memory span of the LSTM on the BSSS task performance. The first experiment (Section 5.6.1) is solely to verify that indeed for large T_{reset} we can allow $G > 1$ and still measure the memory effect correctly. This allows us to use the grouping technique in the other experiments for computational efficiency. The next experiment (Section 5.6.2) analyses the importance of different memory spans, with and without externally provided speaker information. This gives insights into what the LSTM tries to remember. The third experiment (Section 5.6.3) uses a short memory span for the first layer and a large one for the second layer. We find little difference with a network where both layers have large memory spans. This confirms the existence of hierarchy in memorization. Finally, in Section 5.6.4 we look at the differences between the effect of the forward memory and the backward memory on the task performance.

The experimental set-up is the same as in Section 3.3.1. There it was mentioned that an STFT with a 32 ms window length and a hop size of 8 ms was used. This means that the context span is defined as $T_{\text{span}} = (T_{\text{reset}} - 1) * 8$ ms. Notice that for bidirectional networks, this memory span is used for both the left and right context. When estimating the performance of a network for a certain reset frequency T_{reset}, always two networks, with different initializations, were trained an tested to cope with variance on the evaluated performance. Some networks were given the 10-dimensional oracle i-vectors of both single speech signals used in the mixture (see Chapter 3).

5.6.1 Verification of grouping technique

In Figure 5.8 the separation performance for networks with different memory time spans (without grouping) are given in blue. Notice that no networks were trained for $T_{\text{span}} > 400$ ms since the computational memory requirements became too large for $K > 50$. In orange, a group factor of $G = 5$ (= 40 ms) was applied.

It is clear that a group factor of $G = 5$ can be used for $T_{\text{span}} > 100$ ms (since 100 ms $>>$ 40 ms), without loss of performance. Thus, the grouping method is a valid way to break the linear dependence of the computational memory requirements on the reset period for large reset periods. In the remainder of the section, results for $T_{\text{span}} \leqslant 400$ ms will be given without the grouping approach and results for $T_{\text{span}} > 400$ ms will be given with a group factor of $G = 5$.

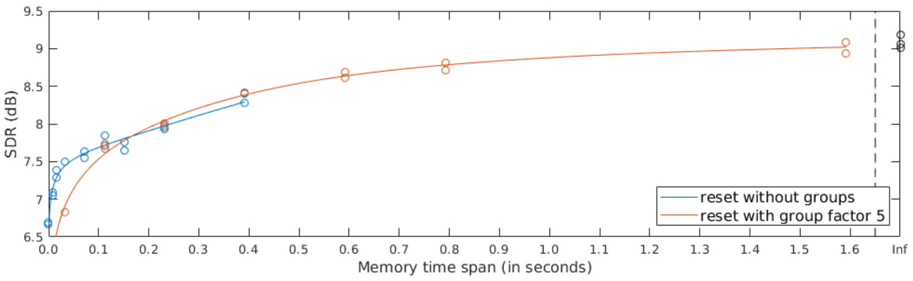


Figure 5.8: Average separation results for networks trained and evaluated for different memory time spans. The blue curve uses no grouping ($G = 1$), the orange curve uses a grouping factor of 5 ($G = 5$). Every experiment was performed twice to cope with variance on the evaluated performance.

5.6.2 Memory span with and without speaker information

Figure 5.9 shows the average separation performance for same gender (male-male and female-female) mixtures, with and without the i-vectors of both speakers appended to the input. Figure 5.10 shows the male-female results for the same networks. Since the results are clearly different, the figures will be discussed separately.

In Figure 5.9 the blue curve quickly rises when the memory span is extended from 0 ms to 400 ms. This effect is also noted for the models where i-vectors were appended to the input (orange curve). For the orange curve, the increase in performance cannot be explained by a better speaker characterization, since the information is already present in the i-vectors. Therefore, the increase in performance of the blue curve cannot solely be explained by a better speaker characterization. The separation task seems to take phonetic information (about 100 ms [65, 171]) into account. Features like common onset, common offset and harmonicity playing a central role in auditory grouping in humans [20] are compatible with this observed time scale as well. Information spanning several 100 ms also seems important. In this range, effects like phonotactics, lexicon and prosody can play a role but further research is necessary to determine to which extent each of these are individually important for BSSS. In Section 6.2.2 it will be shown that models trained on one language generalize to some extent to a different language, making it unlikely that lexical information is key for BSSS. If the memory span is restricted to $T_{\text{span}} < 400$ ms, there is a clear difference between the networks with and without i-vectors at the input. If the memory is restricted, it is difficult to characterize and separate speakers with the same gender. Using i-vectors helps to solve this problem.

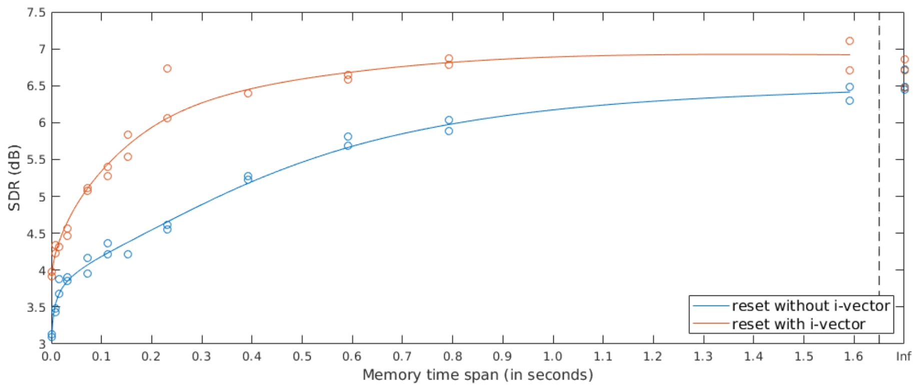


Figure 5.9: Average separation results for networks using different memory time spans, evaluated on same gender mixtures. In blue, the results are given without using i-vectors. In orange, the results when the i-vector of both speakers are appended to the input of the network.

For $T_{\text{span}} > 400$ ms, the SDR without i-vectors keeps increasing with increasing time span, while the curve with i-vectors remains approximately flat. This indicates that when further increasing the memory span at this point, the LSTM-RNN without i-vector can only learn to model the speakers better. On the other hand, grammatical information, which is expected to span much longer than 400 ms [117], is not considered in the LSTM for the BSSS task (otherwise, the orange curve would increase for $T_{\text{span}} > 400$ ms). In [122], using the segment approach, it was found that performance for an ASR task did not improve for $T_{\text{span}} > 250$ ms (500 ms is mentioned, but this includes both left and right context). We notice that this is not the case for BSSS and conclude that the need for a longer time span is mainly caused by the subtask of speaker characterization.

Both curves for the male-female mixtures (Figure 5.10) quickly converge to the result without memory restrictions. There is also a limited difference between results with and without i-vectors at the input, indicating that it is indeed easy to distinguish a male from a female speaker. The result for $T_{\text{span}} = 0$ ms is far above the optimal result for same gender mixtures (Figure 5.9). Instantaneous pitch and formant information seems to achieve most of the effect. Male and female speakers are easily separable, even without any context. Since we are interested in how the LSTM-RNN uses this context, only same gender separation results will be reported in the remainder of this section. However, it is observed that while there is no substantial difference between the reset with and without i-vectors for $T_{\text{span}} < 30$ ms, there is a slight

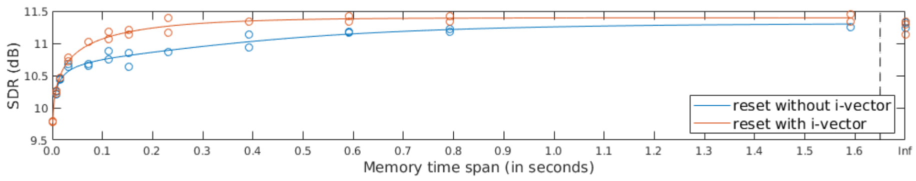


Figure 5.10: Average separation results for networks using different memory time spans, evaluated on male-female mixtures. In blue, the results are given without using i-vectors. In orange, the results when the i-vector of both speakers are appended to the input of the network.

improvement to be found when including the i-vectors for $T_{\text{span}} > 30$ ms. This might indicate that most different-gender mixtures can be separated based on local information (pitch, formants), while for some cases, more sophisticated speaker characterization at longer time span is required. In the latter case, unsurprisingly, i-vectors help.

To conclude this section, the results between the leaky approach (Section 5.4) and the reset approach can be compared. While the leaky approach was more interesting from a computational point-of-view, the timings found were not exact. In fact, when comparing Figure 5.3 (yellow curve) and Figure 5.8[2], it is noted that τ in (5.3) is not a good estimate of the memory span. A better approximation would be 3τ [3].

5.6.3 Layer wise reset

The orange curve in Figure 5.11 shows the performance when memory reset is only applied to the first layer of the network. Naturally, performance is better compared to resetting both layers (blue curve). However, it is interesting to note that optimal performance is already approximately achieved with a memory time span of less than 50 ms. This confirms the hypothesis in [33] that it is sufficient to allow larger time spans only in the deeper layers to model the higher level abstractions.

[2]both figures include same gender and different gender mixtures.

[3]When $\Delta T = 3\tau$ in (5.3), 95% of the initial value of $\mathbf{c}^l_{t-\Delta T}$ is lost. This can be compared to the 69% when $\Delta T = \tau$

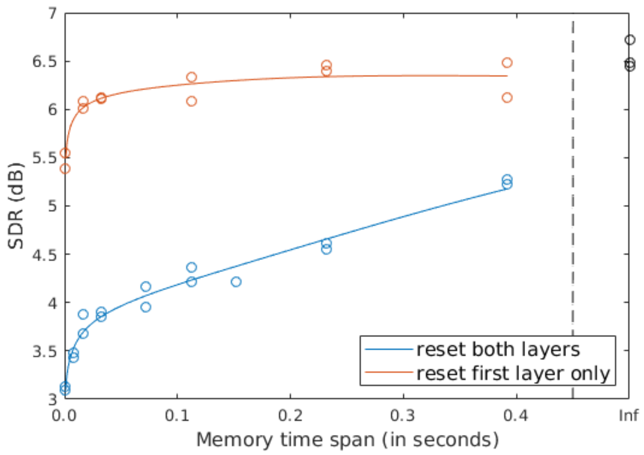


Figure 5.11: Average separation results for networks using different memory time spans, evaluated on same gender mixtures. The blue curve is identical to the blue curve in Figure 5.9. The orange curve shows the result when memory reset is only applied to the first layer and the second layer has no memory restrictions. The results at infinity, colored in black, use no memory reset.

5.6.4 Forward and backward reset

To our knowledge, there has been very little analysis on the relative importance between the forward direction and backward direction of a bidirectional RNN. Early results for a forward-only and backward-only model are given in [153] and [71]. Figure 5.12 shows the difference in performance between a bidirectional LSTM-RNN when memory reset is applied only on the forward direction (blue curve) compared to only on the backward direction (orange curve). For the blue curve, the networks evaluated at $T_{\text{span}} = 0$ ms, essentially correspond to a backward-only RNN. As T_{span} is increased, more forward information is allowed but the backward direction remains dominant since it has no memory restrictions. It is noted that at $T_{\text{span}} = 0$ ms, the backward-only LSTM-RNN slightly outperforms the forward-only LSTM-RNN. This small but consistent difference is kept as the time span for the non-dominant direction increases to 400 ms.

While looking for reasons that could explain this difference, we found that speakers in WSJ0 ended their utterance with “period”, often taking a short break before pronouncing it[4]. This leads to some asymmetry in the speech activity as is shown in Figure 5.13, when measuring with a VAD [169]. After

[4]For instance, in the 4th CHiME challenge [177] this part is removed from the utterance.

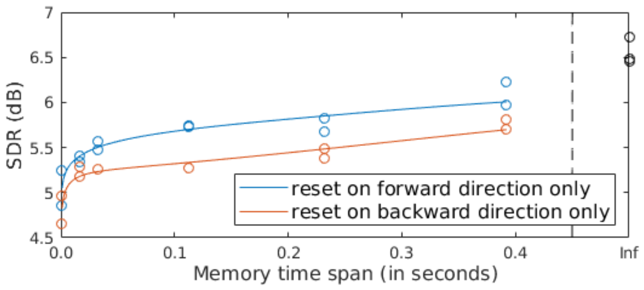


Figure 5.12: Average separation results for networks using different memory time spans, evaluated on same gender mixtures. The blue curve shows the result when memory reset is only applied to the forward direction and the backward direction layer has no memory restrictions. The orange curve applies memory reset only to the backward direction. The results at infinity, colored in black, use no memory reset.

combining single speaker utterances to mixtures, this leads to less overlapping speech near the end of the mixture and might explain the difference we observe in Figure 5.12. Furthermore, as “period” is pronounced at the end of every utterance, it might behave as a prompt for text-dependent speaker recognition [175]. To exclude these unwanted effects, the LibriSpeech (LS) data set [132], which does not contain verbal punctuation, was used to artificially create mixtures[5]. To ensure symmetry in speech activity, leading and trailing silence in the single speaker utterances were cut (see Figure 5.14 bottom). The forward-backward experiment was repeated on the newly created data set and results are shown in Figure 5.15. We see a similar trend as for Figure 5.12 and retain our conclusion that the backward direction is slightly more important than the forward direction for BSSS.

However, the question of what causes this difference remains unanswered. It seems to suggest that cues in speech for BSSS are partly asymmetric. It has been found that voice onset time (VOT) is a predictive cue for post-aspiration [94, 109], while similar conclusions have been drawn for voice offset time (VoffT) [161, 138]. Furthermore, it has also been observed that there is an acoustical asymmetry in vowel production [133]. Finally, reverberation could also play a role in a realistic cocktail party scenario, but this is expected not to be relevant in our experiments, considering the recording set-up for WSJ0 and LS. We leave it to further research to indicate to which extent these asymmetric cues help in BSSS.

[5]This data set has also been used by other papers for BSSS [162, 121]

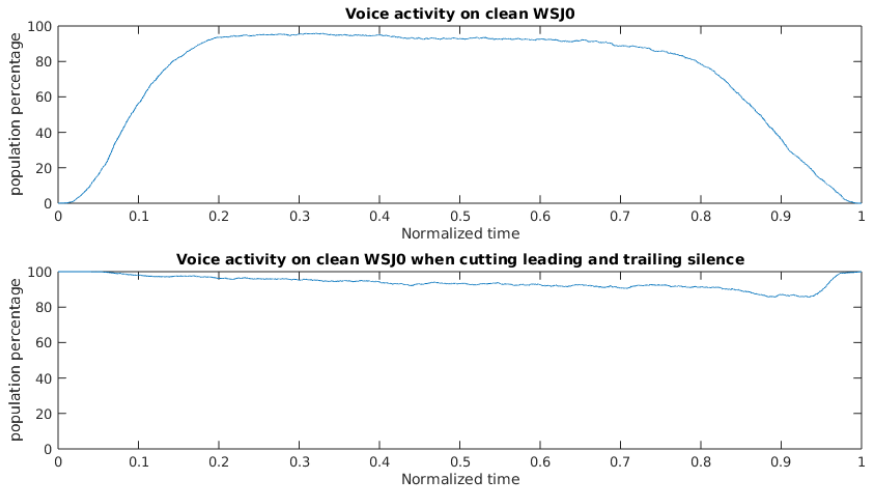


Figure 5.13: Speech activity percentage of clean WSJ0 utterance when audio length normalized is to 1. For the original utterances (top) and when cutting leading and trailing silence (bottom).

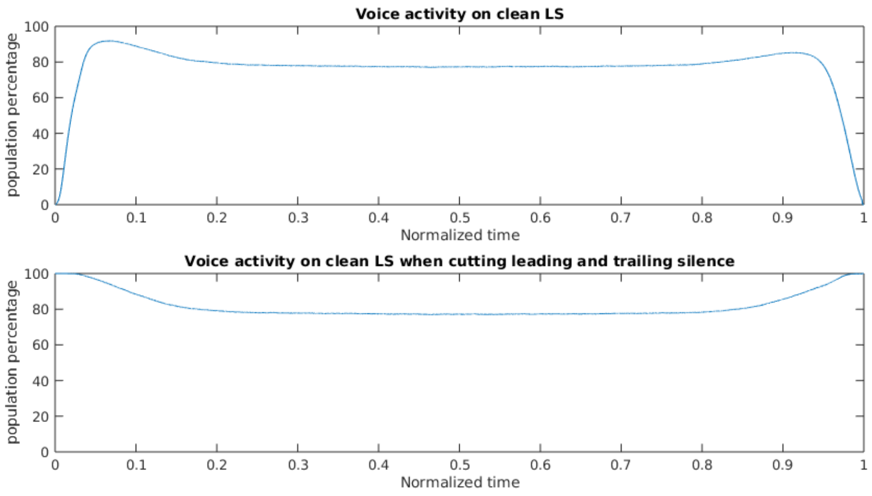


Figure 5.14: Speech activity percentage of clean LS utterance when audio length is normalized to 1. For the original utterances (top) and when cutting leading and trailing silence (bottom).

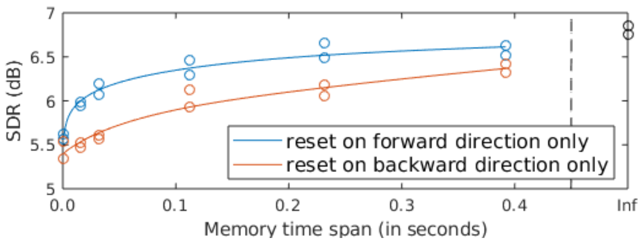


Figure 5.15: Similar to Figure 5.12, but on the LS mixture data set.

Comparing the result without memory restrictions (black colored results at $T_{\text{span}} = \infty$) with the backward constrained results (orange curve), gives an indication on the SDR drop when only limited backward data is available. This can be relevant for near real-time applications with limited allowed delay. It is noted that online implementations with limited delay (shorter than 100 ms) loose roughly 1.5 dB in SDR for same gender mixtures, compared to offline implementations.

5.7 Conclusion

A leakage approach was developed to be applied to the LSTM memory cell to find the importance in different time spans for the task of BSSS. A short-term effect for formant tracking and a long-term effect for speaker characterization were found. However, the lifetime τ was later found not to be representative for the memory span and 3τ would be a better estimate.

The memory reset approach was developed and applied to the same problem. Compared to the leaky approach, it provides exact timings, although being much more computationally expensive. Short-term linguistic processes (time spans shorter than 100ms) have a strong impact on the separation performance. Above 400ms the network can only learn better speaker characterization and other possible separation cues like grammar are not exploited by the LSTM.

Furthermore, the reset method allowed us to verify that performance-wise it is sufficient to implement longer memory in deeper layers. Finally, we found that the backward direction is slightly more important than the forward direction for a bidirectional LSTM-RNN.

The next step of this research would be to use the insights we have gained to adapt the architecture of the (LSTM-)RNN. We would like to encourage other researchers to apply a similar timing analysis for RNNs in their field.

Either with the leaky approach (straightforward implementation, but no exact timings) or the memory reset or segment approach (less trivial implementation with higher computational burdens, but assuring exact timings). Moreover, these methods allow to assess the memory implications on the RNN for a certain subtask. For our task, the importance of speaker characterization was determined by comparing results with and without adding oracle i-vectors to the input of the network. This technique is generalizable to other tasks. For instance, in language modeling for e.g. French, the gender of the subject must be remembered, possibly over many words, to conjugate the perfect tense accordingly. An oracle binary input (male/female) depending on the gender of the relevant subject could be provided. Comparing results with and without this additional binary input, could give an idea on the importance of this subtask on the memory of the RNN.

Chapter 6

Increasing Separation Robustness in Realistic Conditions

This chapter is based on the following publications:
Zegers, J., and Van hamme, H. Multi-scenario deep learning for multi-speaker source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 5379–5383.
Appeltans, P., Zegers, J., and Van hamme, H. Practical applicability of deep neural networks for overlapping speaker separation. In *Interspeech 2019* (2019), ISCA, pp. 1353–1357.

In Chapters 3 and 4 DC and DANet were used for BSSS on artificially created 2-speaker mixtures. While this type of data is excellent for fundamental research on BSSS, it can also be very different from realistic application data. In this chapter an attempt is made to see how well these DL based BSSS approaches will work in more realistic scenarios, something that is generally lacking in the DL field for BSSS. This allows to expand the application range for BSSS and increases the viability and the relevance of DL for BSSS.

There are of course many aspects to consider in realistic scenarios. In this chapter we will consider how well DC, DANet and uPIT work for mixtures with more than 2 speakers (Section 6.1), other languages besides English (Section 6.2) and in the presence of background noise (Section 6.3). While it

algorithm	**train task**	**test task**				
		2spk			3spk	2+3spk
		SG	BG	av	av	av
DC	2spk	6.7	11.3	9.1	2.0	5.6
	3spk	6.0	10.7	8.4	6.1	7.3
	2+3spk	5.9	10.8	8.5	5.6	7.1
DC sep out	2+3spk	7.0	11.4	**9.4**	**6.4**	**7.9**
uPIT	2spk	5.6	10.8	8.4	-	-
	3spk	-	-	-	6.4	-
	2+3spk	6.0	11.0	**8.6**	**6.6**	**7.6**

Table 6.1: SDR improvement results for DC and uPIT for different train/test sets. For the 2spk case, the results are broken down into same gender mixtures (SG), mixtures with both genders (BG) and all mixtures (av). Entries with '-' refer to a result that was not plausible.

is important to consider these 3 in realistic settings, there are of course many more relevant aspects. The author suggests that in particular reverberation should be considered in future research.

6.1 Variable number of speakers

In Section 1.4.2 DNN methods have been described that use a model for mixtures of S speakers. If a solution is requested for multiple scenarios (e.g. both mixtures of S_1 and S_2 speakers), a simple solution would be to train separate models for each task. However, this has the disadvantage that a model can only be trained on data for a specific scenario, while data of the other scenarios could still be relevant. Furthermore, the solution would contain as many models as there are tasks and for each presented mixture, the requested model would have to be selected.

Therefore a multi-task solution, based on Section 4.1.2, is proposed for two methods of Section 1.4.2: DC and uPIT. This allows the model to learn from more data and only a single model has to be retained for all tasks. In [88] a small initial experiment in this context has already been done by evaluating a model, trained to 2-speaker mixtures, on 3-speaker mixtures. Finally it is noted that this section does not consider the special case of $S = 1$, but a joint solution for $S = 1$ and $S = 2$ has already been discussed in Chapter 4. The experimental set-up for Sections 6.1.1 and 6.1.2 are the same as described in Section 3.3.1.

6.1.1 Single task learning

First, some single task learning experiments are performed using DC. Models are trained on 2 or 3 speakers and testing is done on 2 and 3 speakers and the results are shown in Table 6.1. It is noticed that testing on 3 speakers when only seen mixtures of 2 speakers during training, drastically degrades performance compared to training on 3 speakers (-4.1 dB). However, when reversing the training and test task, the drop in performance is much lower (-0.7 dB). Being able to separate 3 speakers seems to partly rely on the subtask of separating 2 speakers.

Secondly, the performance of uPIT and DC are compared and it is concluded that DC outperforms uPIT for 2 speaker mixtures and uPIT is better for 3 speaker mixtures. Differences are rather small and possibly dependent on the chosen network architecture.

6.1.2 Multi-task learning

The DC multi-task learning model (train 2+3spk), just falls short in comparison with the task specific models (-0.6 dB for test 2spk and -0.5 dB for test 3spk). However, when a separate output layer is implemented per task and thus a task specific embedding space is used, denoted by *DC sep out*, the joint model even outperforms the task specific models (+0.3 dB for test 2spk and +0.3 dB for test 3spk). Not only has the model succeeded to use training data from a task different from the test task, it has also managed to create a mostly shared model for both test tasks. The latter is emphasized to make a distinction with the experiment where the BLSTM layers would be fine-tuned to the specific test task or the task weight parameter α in (4.10) would be fine-tuned towards the specific test task. In that case possibly better performance could be achieved, but this would lead again to the need of a model per test task (even though data from different tasks has been used). The same conclusion can be made for the uPIT method where the joint model improves over the task specific models (+0.2 dB for test 2spk and +0.2 dB for test 3spk). Note that because the output layer of uPIT is dependent on S (see Section 1.4.2), a separate output layer per task is always required in uPIT.

6.1.3 Conclusion

Generally, a model trained on one type of mixture performs suboptimally on another type of mixture. In this section it was shown that it is useful for a single task to include data from another task. Furthermore, we conclude that a single model can be used to cope with different tasks without substantial loss in performance. We hope that other researchers in the BSSS field will consider multi-task learning and testing in the feature, instead of being restricted to a single type of mixture.

6.2 Different languages

In the original papers where DC and DANet were described [78, 30], the separation performance is only evaluated on English speakers. It is therefore unknown how well these BSSS algorithms are suited for different languages. This is studied in Section 6.2.1.

Furthermore, generalization of these algorithms to unseen languages is relevant from a practical point of view. For example if the target language cannot be determined, if (sufficient) training data is not available for that language or if it is undesirable to train and store a BSSS model for every individual language. In such cases it is interesting to know whether a model, that is trained on one language or one set of languages, can be used for other languages. This is studied in Section 6.2.2. The experiments in this section will also give an indication on the robustness against different accents and dialects of a language. Furthermore, they might give some information on what cues, such as phonetic, phonotactic, lexical or grammatical, the methods exploit to separate speakers.

6.2.1 Target languages

This section presents experiments with six languages, including a tonal language. The mixtures are generated using the Global Phone corpus [152] by summing utterances of two different speakers. The experimental set-up is similar as in Section 3.3.1, with the difference that $I_{tr} = 70$ (instead of 101) speakers and $I_{test} = 20$ (instead of 16).

Table 6.2 gives the average SDR for DC and DANet for mixtures of two speakers in respectively Arabic, French, Mandarin, Portuguese, Spanish, and Swedish. Both methods obtain their best score for Mandarin, which is the only tonal language in our test set. This might indicate that tonality is a useful feature for speaker separation but more research with other tonal languages is needed to support this.

language	**DC**	**DANet**
Arabic	7.50	7.97
French	7.46	8.20
Mandarin	8.54	8.86
Portuguese	7.24	8.27
Spanish	6.72	7.76
Swedish	6.93	7.83

Table 6.2: The average SDR (in dB) when trained and tested on the same language.

6.2.2 Generalization to unseen languages

This section will examine how well a network can separate mixtures of a language not seen during training. The French and Swedish networks from Section 6.2.1, are reused. The French network is tested on Portuguese mixtures and Mandarin mixtures. The Swedish network is tested on Arabic and Spanish mixtures. Table 6.3 gives the average separation performance of the methods for the non-target languages. Also the difference with the score of the network trained with the considered language (Table 6.2) is given.

It is noticed that for three out of the four test languages (Portuguese, Arabic and Spanish) the loss in performance compared to the target language model is relatively mild (roughly -1 dB) and a decent separation quality is retained. For Mandarin on the other hand the decrease is more substantial, around -4 dB. This seems to indicate that the performance for non-target languages depends on the relation of the training and test language.

The fact that the methods do not break completely implies that they do not create grammatical or lexical models, but at most phonotactic or phonetic models. They do seem to do more than tracking formants or pitch, which would make them almost language independent. This is consistent with the observations that were made in Chapter 5.

Table 6.4 gives the separation quality for the networks trained with multiple languages, when keeping the total number of training mixtures fixed. The average SDR is reported for both trained (t) languages and untrained (u) languages. Again the difference with scores of the networks trained with the language itself (Table 6.2) is given as well as the difference when training with a single non-target language, if applicable (Table 6.3). From the results it is observed that for trained languages (t) it is in most cases disadvantageous to replace a part of the training data with mixtures in other languages. For

language	**DC**		**DANet**	
	French network			
Mandarin	4.59	(-3.95)	4.86	(-4.00)
Portuguese	6.33	(-1.13)	7.22	(-0.98)
	Swedish network			
Arabic	5.98	(-1.52)	7.01	(-0.96)
Spanish	6.01	(-0.71)	7.20	(-0.55)

Table 6.3: The average SDR in dB for DC and DANet for an unseen test language and the difference with the SDR for matched language training in Table 6.2.

language	**DC**		**DANet**	
	{French, Turkish} network			
French (t)	6.92	(-0.55)	7.75	(-0.45)
Mandarin (u)	4.89	(-3.65; +0.30)	5.32	(-3.54; +0.46)
Portuguese (u)	6.31	(-1.15; -0.02)	7.24	(-0.96; +0.02)
	{French, Turkish, Japanese} network			
French (t)	6.35	(-1.11)	7.27	(-0.93)
Mandarin (u)	4.57	(-3.97; -0.02)	5.34	(-3.52; +0.48)
Portuguese (u)	5.94	(-1.53; -0.39)	6.77	(-1.44; -0.45)
	{Swedish, Turkish} network			
Swedish (t)	7.03	(0.10)	6.97	(-0.87)
Arabic (u)	6.45	(-1.02; +0.47)	6.71	(-1.26; -0.30)
Spanish (u)	6.39	(-0.33; +0.38)	6.99	(-0.77; -0.21)
	{Swedish, Turkish, Japanese} network			
Swedish (t)	6.75	(-0.18)	7.58	(-0.25)
Arabic (u)	6.49	(-1.02; +0.51)	7.31	(-0.66; +0.30)
Spanish (u)	6.16	(-0.56; +0.15)	7.34	(-0.42; +0.14)

Table 6.4: The average SDR in dB for DC and DANet trained with multiple languages for trained and untrained languages. In brackets the difference with the SDR for matched language training (Table 6.2) as well as the difference with training on a single non-target language (Table 6.3) if applicable.

untrained languages (u) on the other hand , it is in some cases advantageous to include multiple training languages instead of one.

6.2.3 Conclusion

DC and DANet are applicable to BSSS in a wide variety of languages, including tonal languages. Training models with (a combination of) related languages yields relatively minor performance degradation compared to training on the target language. This observation supports the results in Chapter 5, which showed that LSTM models for BSSS mainly exploit information within the time span of a couple of phones and long span information is limited to speaker identification while grammatical patterns are ignored.

6.3 Background noise

This section discusses the applicability of the DNN models of Section 1.4.2 in the presence of background noise. This was lacking in the research field at the time the experiments were done. A noisy mixture takes the form of (1.5).

Two approaches will be considered for tackling the BSSS problem in the presence of background noise. The first approach will simply regard the noise signal as an additional speech signal. I.e. $S \leftarrow S + 1$ in (1.77) or (1.76). The second approach proposes some modifications to the network and this will improve the performance compared to the first approach.

6.3.1 Modified network architectures

Figure 6.1 shows the proposed modified network architecture. Each time-frequency bin now not only estimates an embedding vector $\mathbf{v}_{tf}$, but also a scalar output α_{tf} with the sigmoid non-linearity such that $0 \leqslant \alpha_{tf} \leqslant 1$. This scalar is an estimated ratio mask to suppress the noise in that bin. Before, to train a model that learns to separate S speech signals, a loss function was required that was invariant to the order of the labels of the speech signals, as an intra-class separation task has to be performed (see Section 1.4.2). However, noise can be considered of a different class than speech and the auxiliary loss function to optimize α_{tf} does not need to be permutation independent. Taking this into account, the loss function for DC and DANet will be modified.

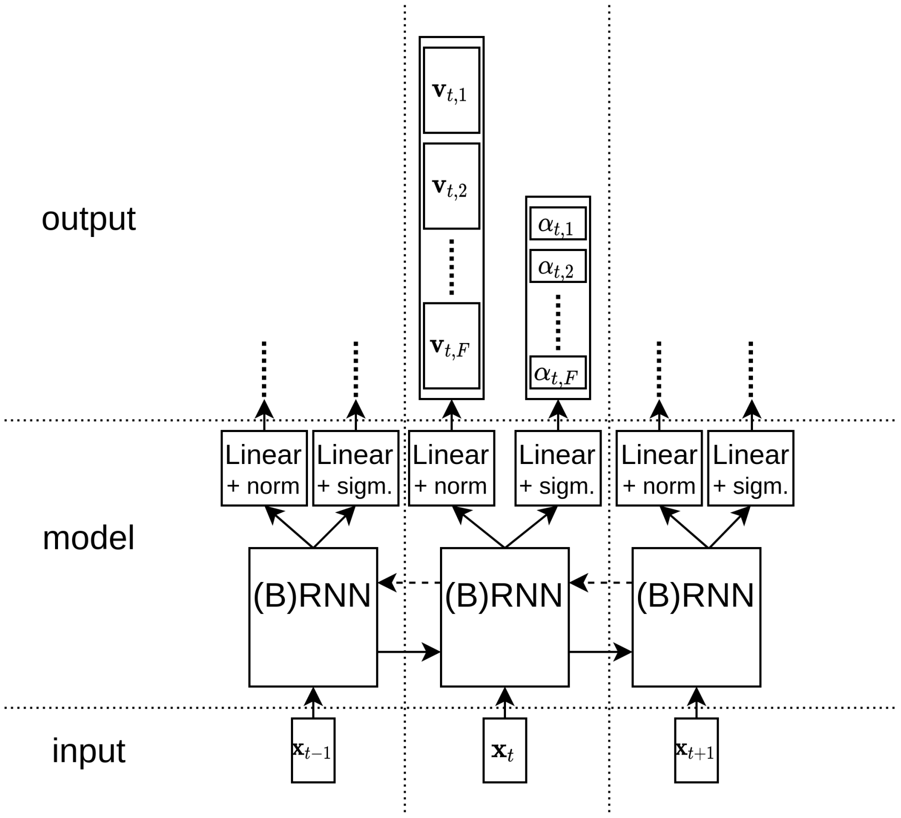


Figure 6.1: An illustration of a modified network architecture to better cope with background noise. Compared to Figure 1.10 it has an additional output layer to estimate α_{tf}.

Deep clustering

The loss function in (1.77) is modified to:

$$\mathcal{L} = \frac{1}{T^2F^2} \sum_{t_1,f_1,t_2,f_2} (\mathbf{v}_{t_1f_1} \cdot \mathbf{v}_{t_2f_2} - \mathbf{z}_{t_1f_1} \cdot \mathbf{z}_{t_2f_2})^2 + \gamma \frac{1}{TF} \sum_{t,f} (\alpha_{tf} - \alpha_{tf}^{\text{ideal}})^2, \quad (6.1)$$

with α^{ideal} the optimal ratio mask to filter the noise. The first term is the same as (1.77). The second term trains the network to generate ratio masks to filter out the noise by penalizing the distance between the estimated and the optimal noise suppression mask. The latter is set to be

$$\alpha_{tf}^{\text{ideal}} = \left(\sum_{s}^{S} |x_{s,tf}|^2\right)^{1/2} \left(|n_{tf}|^2 + \sum_{s}^{S} |x_{s,tf}|^2\right)^{-1/2}, \tag{6.2}$$

as it was shown to perform well in many noisy scenarios [181]. The hyper-parameter γ weighs the importance of separating the speakers and filtering out noise. γ is arbitrarily set to one in the experiments in Section 6.3.2.

The procedure to find the mask estimates $\hat{m}_{s,tf}$ during evaluation time is modified. As before, K-means is used to cluster the embeddings $\mathbf{v}_{tf}$ per speaker. However, only the embeddings where the associated α_{tf} is greater than 0.75 are used to find the cluster centers[1]. Then, instead of setting the mask to 1 for speaker s that is estimated to be dominant, the mask is set to α_{tf}. This is equivalent to generalizing (1.80) to

$$\hat{m}_s(t,f) = \begin{cases} \alpha_{tf}, & \text{if } \mathbf{v}_{tf} \in c_s \\ 0, & \text{otherwise} \end{cases}. \tag{6.3}$$

Deep attractor networks

For DANet the loss function (1.76) is modified to:

$$\mathcal{L} = \sum_{s=1}^{S} \sum_{t,f} (\alpha(t,f)\hat{m}_s(t,f)|y(t,f)| - |x_s(t,f)|)^2. \tag{6.4}$$

During training (1.82) is used to obtain the attractors $\mathbf{a}_s$, but modified to

$$\mathbf{a}_s = \frac{\sum_{t,f} \mathbf{v}_{tf} \tilde{z}_{s,tf}}{\sum_{t,f} \tilde{z}_{s,tf}}. \tag{6.5}$$

with $\tilde{z}_{s,tf} = 1$ if $z_{s,tf} = 1$ and $\alpha(t,f) > 0.75$ and $\tilde{z}_{s,tf} = 0$ in all other cases. Although this hard cut-off introduces discontinuities and local optima in the cost function, an alternative (smoother) penalty for noisy bins did not lead to improved performance.

To separate new mixtures, a similar strategy as in Section 1.4.2 is applied, but $\mathbf{a}_s$ is now estimated as the cluster centers after using K-means when only using embedding if the corresponding $\alpha_{tf} > 0.75$.

[1] All embeddings $\mathbf{v}_{tf}$ are assigned to a cluster. Only the embeddings for which $\alpha_{tf} > 0.75$ are used to find the cluster centers.

6.3.2 Experiments

Similar to the experimental set-up from Section 3.3.1, the utterances were sampled from WSJ0 data set. The noise signals were chosen from the *3rd CHiME speech separation and recognition challenge* data set [13], which contains recordings of realistic environment noise. Four new two-speaker mixture sets were used:

- A noise free training (20 000 mixtures) and development set (5 000 mixtures). The signals are normalised such that the individual speakers have the same power.
- A noisy training (100 000 mixtures) and development set (5 000 mixtures). The training set reuses each mixture of the noise free training set five times, each time with different noise. The new development set is similar to the noise free variant, only with noise added. The signals of the speakers and the noise are normalised such that they have the same power.
- A noisy test set of 3 000 mixtures with different speakers and utterances than in the training and development sets. The noise comes from different parts of the same recordings as the training and development sets (for the training and development sets noise is sampled from the first 10 minutes of the recording, for the test set from the leftover part). The signals of the speakers and the noise are normalised such that they have equal power.
- A second noisy test set of 3 000 mixtures. Similar to the previous test set but now the signals are normalised such that both speakers have equal power and the noise is 3 dB lower than each speaker.

Table 6.5 compares the performance of the following five methods for the two noisy test sets described above:

- DC trained without noise (DC no noise).
- DANet trained without noise (DANet no noise).
- DC with noise (DC with noise). During training the noise was considered as third speaker and the network was trained to form three clusters: two associated with speakers and one associated with the noise. During testing three reconstructions were created but only the two that most resembled a speaker were used for scoring.
- Modified DC described in Section 6.3.1 (modified DC).
- Modified DANet described in Section 6.3.1 (modified DANet).

The recurrent part of the networks trained without noise consisted of two layers with 800 bidirectional LSTM cells each. For the networks trained with noise this consisted of four layers with each 800 bidirectional LSTM cells.

Separation performance of the five different models are shown on the two test sets. The models trained without noise break down on noisy data. Including noise during training as a third speaker already leads to improved performance. The best SDRs are obtained with the modified methods of Section 6.3.1. The SDR improvement w.r.t. 'DC with noise' comes at a cost of a few dB in SNR, which seems less important since noise is not the main source of distortion.

	0 dB		3 dB	
Method	SDR	SNR	SDR	SNR
DC no noise	-1.75	5.38	1.99	11.5
DC with noise	4.85	16.5	6.81	19.4
modified DC	5.11	12.8	7.43	17.2
DAN no noise	-0.37	5.83	2.67	10.8
modified DAN	5.27	13.5	7.33	17.4

Table 6.5: The average SDR and SNR in dB for the test sets with respectively the two speakers and the noise equally loud (0 dB) and the two speakers 3 dB louder than the noise (3 dB).

6.3.3 Conclusion

In this section, DC and DANet were extended with an estimated spectral mask to cope with noisy mixtures and this showed to give substantial improvement over the baselines. A limitation of the current experiments is that they only examine how well the methods perform for noise types for which training data is available. Future work should consider *unseen* noise types.

This work was published in the conference proceedings of *Interspeech 2019*. At the same conference, a different paper tried to tackle the same problem [186]. They also tried our first approach 'DC with noise' where the noise signal is regarded as a speech signal for the DC algorithm. They proposed an extension to this where, still the embeddings of the noise dominated bins should be far away from the embeddings dominated by speech, but they did not request that the embeddings of the noise dominated bins should be close to each other. However, their best performance was obtained by simply treating the noise signal as a speech signal without further extension. Furthermore, in this section it was found that applying network and loss modifications improves performance over simply regarding the noise as an additional speech signal. It is therefore concluded that the modified architecture and loss is currently the best approach for BSSS in the presence of background noise.

6.4 Conclusion

Most of the current research on DL approaches towards BSSS consider only data in highly idealized settings. Therefore many aspects in practical settings are ignored. This chapter studied the impact of some of these aspects and proposed modifications to cope with them.

First, a network was presented that can deal with a variable number of speakers, rather than using a different network per number of active speakers. Then, state-of-the-art BSSS methods were evaluated on many languages and it was concluded that these BSSS methods are widely applicable. It was even noticed that a model trained on one language, generally manages to separate mixtures form a different language to some extent. Finally, the impact of background noise on the separation performance was studied. It was found that traditional approaches suffer considerably in the case of background noise. Using an additional spectral mask for denoising shows substantial improvements.

While these three aspects are very relevant in realistic settings, there are of course many more to study. In particular, reverberation seems to be challenging and seems an interesting direction for further research.

Chapter 7

Linear versus Deep Learning Methods for Noisy Speech Separation for EEG-informed Attention Decoding

This chapter is based on the following publication:
Das, N., Zegers, J., Van hamme, H., Francart, T., and Bertrand, A. Linear versus deep learning methods for noisy speech separation for EEG-informed attention decoding. *Journal of Neural Engineering 17*, 4 (August 2020).

The contribution of the author of the manuscript is in the use of DNNs for BSS. The AAD module and the M-NICA were implemented by the first author of the above publication.

In this last chapter one of the many applications where BSSS plays a role, is studied in depth: A neuro-steered hearing aid. In summary, brain activity of a hearing-aid user is measured via Electroencephalography (EEG) in a multi-speaker scenario, where the user is asked to focus on a single target speaker. This EEG data is used to estimate the speech envelope of the attended target speaker using an Auditory Attention Decoder (AAD). Until recently, this speech envelope was then compared with the oracle envelopes extracted from the single speaker audio recordings to decide which of the active speakers the listener wants to attend to. In a realistic setting, these single speaker audio envelopes should be estimated using BSSS, possibly followed by an audio envelope extractor. Once the attended speaker is selected, the estimated speech signal can be sent to the hearing-aid user.

To make relevant conclusions on the feasibility of such a hearing-aid, background noise will be present in most experiments. Therefore, the term BSS will be used instead of BSSS, as the latter presumes only speech sources to be active. In most works on AAD, linear BSS techniques are used to estimate the speaker audio envelopes, and the use of DNN can only be found in one or two papers. This chapter will give an extensive comparison between linear and DL methods for BSS for EEG-informed attention decoding.

7.1 Introduction

In a noisy environment with multiple speakers talking simultaneously, i.e., the so-called cocktail-party problem, a person with normal hearing has the ability to focus attention on one speaker and ignore the other speakers and surrounding noise in a seemingly effortless manner. People with a hearing impairment, on the other hand, find such situations extremely challenging. Although modern hearing aids allow to suppress background noise, a major unsolved problem is to determine which of the speakers is the desired one, and which speakers should be treated as noise. Existing approaches use unreliable heuristics based on speaker loudness or other acoustic features or simply assume that the target speaker is always in the frontal direction.

Recent developments in the field of neuroscience have shown that it is possible to decode the auditory attention of a listener in a multi-talker environment from brain signals recorded with magneto- or electro-encephalography (M/EEG) [55, 130, 4]. This opens up new opportunities to design smarter hearing devices, augmented with electrodes to record neural signals, that decode to which speaker a listener is attending, thereby assisting the hearing device to determine and enhance the attended speaker.

The envelope tracking of speech streams in a listener's cortical responses, and more so of the attended speech stream, is well established in the literature [54, 69, 116]. Furthermore, this differential tracking of attended and unattended streams is also present in hearing impaired listeners [136, 50, 140]. This fact can be exploited to design algorithms that perform AAD. Various methods have been developed to achieve AAD, addressing different kinds of decoders [130, 118, 48, 49, 34], stimulus features [18, 4, 53], data acquisition [119, 203, 62], electrode selection and miniaturization strategies [119, 62, 127], etc.

The aforementioned AAD studies in [130, 118, 48, 49, 18, 4, 53, 119, 203, 62, 127, 54, 69, 116, 136, 50, 140, 34] assume access to the clean speech signals to perform AAD, which are not available in a practical setting. Indeed, a hearing aid only has access to noisy microphone recordings in which multiple speech sources and noise sources are mixed. The first study that aimed for AAD using BSS was published in [174], where a Multiplicative Nonnegative ICA (M-NICA) algorithm [16] was used to extract the individual speech envelopes from a noisy multi-microphone input. The AAD method in [18] was then used to select the attended speaker, which was then extracted from the microphone array using a Multi-channel Wiener Filter (MWF) [174]. This pipeline was later extended in [43] with a twofold MWF (one for each speaker) leading to better decoding performance and less variation across different acoustic conditions. However, while in both of these studies, the performance of the speaker separation algorithm was evaluated for a range of background noise levels and speaker positions, the EEG signals used for AAD were collected from subjects who listened to a 2-speaker scenario with 180° speaker separation and without background noise. Therefore, there was a mismatch between acoustic conditions in the AAD experiments and the audio processing modules. In [7], BSS was achieved using binaural Direction of Arrival (DOA) estimators with Linearly Constrained Minimum Variance (LCMV) beamformers. The AAD module used EEG signals recorded in two relatively high SNR conditions and two reverberation conditions.

The BSS audio processing in [174, 43, 7] is based on linear beamforming and linear source separation techniques, which do not require any training data and have the advantage of being computationally cheap. In recent years, the DNN-based approaches of Section 1.4.2, have become a popular alternative to solve the speaker separation problem, particularly for the challenging single-microphone scenario [78, 97, 30] and the even more challenging scenario with additional background noise [6, 186]. When using multiple microphones, separation performance can be increased by making use of the spatial information of the sources [183, 29]. In [129, 74], a single-microphone DNN-based approach was used for speaker separation in an AAD context. However, both of these studies involved training and testing in noise-free conditions. Furthermore, AAD was performed on Electro-Corticography (ECoG) data, which is an invasive approach involving surgery. This is a limiting factor for hearing-aid users. Non-invasive techniques like EEG are more preferable in this regard.

It is evident that a detailed analysis of neuro-steered BSS under challenging acoustic conditions is necessary to draw conclusions about its feasibility. Despite the promising results in the aforementioned studies, all of them have important caveats, which may lead to overoptimistic conclusions in a context of demonstrating viability of neuro-steered BSS algorithms. To summarize:

The audio processing of [174, 43] was tested in various noisy conditions with different angles between speakers, but the EEG recordings were performed in a noise-free condition with 180° separation, resulting in a mismatch between the acoustic conditions of the microphone and neural signals. This mismatch does not occur in [7], but only mild noisy conditions are considered and the speaker positions are the same in all conditions. Finally, it is noticed that in [74] ECoG is used instead of EEG. Furthermore, the authors only report results in noiseless conditions, using single-microphone recordings(while most state-of-the-art hearing aids contain multiple microphones), and with competing speakers of different gender[1].

In addition to these practical limitations, the different settings across the aforementioned studies make it impossible to draw conclusions about the impact on AAD performance of non-linear (DNN-based) methods like [129, 74], and linear methods like [174, 43, 7], as well as the effect of using spatial information from multiple microphones versus single-microphone methods.

The contribution of this chapter is twofold. First, the aforementioned caveats are avoided by focusing on realistic scenarios involving microphone recordings from a binaural hearing aid in challenging noisy conditions with same-gender competing talkers at various relative speaker positions. In each acoustic condition, the audio and neural data are matched to each other, i.e., the audio signals presented to the subjects during the EEG recording are the same as those used for the audio signal processing. Secondly, the potential improvement of using a DNN-based BSS algorithm compared to a computationally cheap and training-free linear signal processing algorithm is investigated. The performance of the extended linear approach is shown to be at par or better than the purely DNN-based approach. The potential advantage of using a combination of both approaches, i.e. a DNN-based speech separation to support a linear beamformer, is also investigated. The superior performance of such a system that combines *the best of both worlds* is demonstrated in terms of improvement in Signal-to-Interference-plus-Noise Ratio (SINR), Perceptual evaluation of speech quality (PESQ) [144] and AAD accuracy. The study focuses on two representative state-of-the-art algorithms, i.e., DC as the DNN-based approach and M-NICA+MWF [43] as the linear method. In both cases, a multi-microphone input is used, which will substantially improve speech separation compared to a single-microphone input.

The outline of this chapter is as follows. The different neuro-steered BSS pipelines are presented in Section 7.2. The various experiments used to validate the different approaches are presented in Section 7.3. In Section 7.4, the results of the experiments are shown, and the various implications of these findings are

[1]Note that both AAD and speaker separation are expected to be more difficult when the competing speakers are of the same gender.

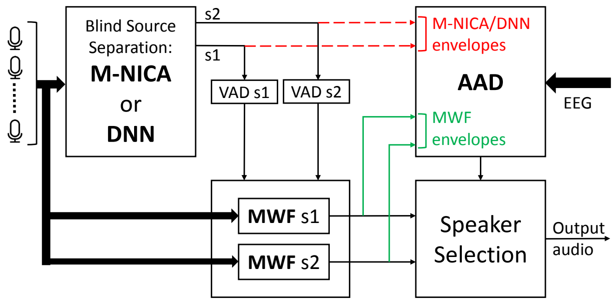


Figure 7.1: Block diagram of the proposed neuro-steered BSS algorithm for a 2-speaker scenario, consisting of modules for BSS, with optional Multi-channel Wiener Filtering, and AAD. The red (dashed) track refers to using the output of the BSS block directly for AAD. The green track corresponds to additional filtering.

discussed in Section 7.5. The chapter is concluded in Section 7.6.

7.2 Methods

The proposed neuro-steered BSS pipeline is shown in the block diagram of Figure 7.1 for a 2-speaker scenario. Although the analysis in this chapter focuses on a 2-speaker scenario, the BSS procedure in Figure 7.1 can straightforwardly be generalized to a larger number of speakers (denoted by S in the text). The pipeline consists of three steps:

1. In the BSS module, M-NICA or DNN is applied. Optionally, the estimated speech envelopes obtained from the BSS module can be used as side information for the semi-supervised MWF module, which typically achieves a better separation and denoising performance. This will be discussed in Section 7.2.2.

2. In the AAD module, the attended speech envelope from the listener's EEG is reconstructed using a pre-trained decoder. This reconstructed speech envelope is correlated with the speech envelopes of the individual speakers extracted in step 1, where the speaker with the highest correlation is selected as the attended speaker. To this end, either the envelopes from the M-NICA or DNN algorithm can be used directly (red dashed lines in Figure 7.1), or a new set of envelopes can be extracted from the MWF outputs (green lines in Figure 7.1). Both cases will be investigated, where the envelopes in the red dashed lines will be referred to as M-NICA or

DNN envelopes (depending on which algorithm is used in the BSS block) and the ones in the green lines as MWF envelopes. More details about the AAD are given in Section 7.2.1.

3. Once the attended speaker has been estimated, the corresponding MWF output signal is selected as the final output, which ideally consists of a denoised version of the speech signal corresponding to the attended speaker. Optionally, the DNN output of the attended speaker can directly be chosen as final output.

In the remainder of this chapter, it is assumed that the number of speakers S is known. In practice this would require a supporting algorithm to estimate S from the microphone recordings, e.g., based on subspace analysis, which is beyond the scope of this chapter.

7.2.1 Auditory attention decoding

In order to choose which enhanced speech estimate will be presented as output of the neuro-steered BSS pipeline, it is necessary to know which speech signal is being attended to by the listener. Towards this goal, an AAD module is used which consists of a pre-trained spatio-temporal decoder (subject-specific training) which takes the listener's EEG data as an input to reconstruct the speech envelope of the attended speaker. In the training data, the ground truth about the subject's attention is known, and the decoder is designed such that the difference between its output and the attended speech envelope is minimized in the MSE sense. The algorithm described in [18] was used for our decoder design. The reconstructed attended speech envelope is given by

$$\tilde{p}(k) = \sum_{\tau=0}^{L-1} \sum_{c=1}^{C} d_c(\tau)\, m_c(k+\tau), \tag{7.1}$$

where $m_c(k)$ is the value of the c-th EEG channel at sample time k, and $d_c(\tau)$ denotes the decoder weight at the c-th channel at time lag τ. All decoder weights are stacked in a vector $\mathbf{d} = [d_1(0), d_1(1), ..., d_1(L-1), d_2(0), ..., d_2(L-1), ..., d_C(0), ..., d_C(L-1)]^T$. Stacking EEG samples of all C channels for L time lags, we get $\mathbf{m}(k) = [m_1(k), m_1(k+1), ..., m_1(k+L-1), m_2(k), ..., m_2(k+L-1), ..., m_C(k), ..., m_C(k+L-1)]^T \in \mathbb{R}^{LC}$. Using this notation, (7.1) can be written as $\tilde{p}(k) = \mathbf{d}^T\mathbf{m}(k)$. The optimal decoder that minimizes the MSE between the reconstructed attended envelope $\tilde{p}(k)$ and the attended speech envelope $p_a(k)$ is given by

$$\hat{\mathbf{d}} = \mathbf{R}_{mm}^{-1}\mathbf{R}_{mp_a}, \tag{7.2}$$

where $\mathbf{R}_{mm} = E\{\mathbf{m}(k)\mathbf{m}(k)^T\} \in \mathbb{R}^{LC\times LC}$ is the EEG covariance matrix, $\mathbf{R}_{mp_a} = E\{\mathbf{m}(k)p_a(k)\} \in \mathbb{R}^{LC}$ is the cross-correlation vector between the EEG data and the attended speech envelope, which is known during the training of the decoder and $E\{.\}$ denotes the expected value operator. At test time, the correlation coefficients between $\tilde{p}(k)$ and the estimated speech envelope of each speaker are then computed. The speech envelope that results in the higher correlation is selected as the attended speech stream.

7.2.2 Blind source separation

The problem of (B)SS has been described in Section 1.1.1. Given a noisy multi-speaker input $y(t, f)$, as in (1.5), the goal is to find estimates of the speech source signals $\hat{x}_s(t, f)$, which is typically done by estimating a mask $\hat{m}_s(t, f)$ for each speech source, as in (1.10). Multi-channel extensions are given in (1.8) and (1.11).

Two methods are presented to estimate the masks in (7.6): one that is fully linear, based on the M-NICA algorithm, and one that makes use of a DNN. Their use for AAD is summarized here and is detailed further below.

1. The linear multi-channel signal processing approach using M-NICA provides an estimate of the energy envelope of each individual speaker. These envelopes can be directly fed to the AAD module (red dashed track in Figure 7.1) as in [17], or alternatively, they can be used as a VAD mechanism to design an S-fold MWF which estimates per-speaker masks $\hat{\mathbf{m}}_s(t, f)$ in (1.11). The envelope of $\hat{x}_s(t, f)$ from (1.11) can then be fed to the AAD module as in [43] (green track in Figure 7.1).
2. On the other hand, a DNN can be trained to separate speech mixtures into individual speech streams. As opposed to M-NICA, the DNN directly estimates the masks $\hat{\mathbf{m}}_s(t, f)$ to compute (1.11), after which the speech envelopes can be extracted and directly fed to the AAD module as in [129, 74] (red dashed track in Figure 7.1). Alternatively, the envelopes obtained from the DNN can also be used as a VAD to design MWF masks (green track in Figure 7.1).

Multiplicative non-negative independent component analysis

The M-NICA algorithm was introduced in [16], and has been used to blindly separate speech envelopes from a multi-microphone input [17, 174]. The M-NICA algorithm operates in the short-term energy domain, thereby transforming the problem into a (non-negative) energy envelope separation problem. To this end, the short-term energy signal from each microphone signal is computed over a window of K samples. The M-NICA is applied on the resulting M energy signals (note that M-NICA operates at a sampling rate that is K times lower than the microphone signal sampling rate). For more details on the implementation of M-NICA for demixing speech envelopes, we refer to [16, 17].

M-NICA exploits the inter-channel differences in the energy distributions of the speech sources in order to extract the energy envelopes of each individual speaker. In the case of a binaural hearing aid, these differences are mostly due to head shadow effects. The non-negativity of the underlying sources facilitates the use of only second order statistics in its computations, in comparison to other source separation algorithms that use higher order statistics. This, together with the K-fold reduction in sampling rate, leads to a low computational complexity which is a desirable factor to incorporate such a source separation scheme in an actual hearing prosthesis.

MWFs for source separation

The MWF is a data-driven and adaptive linear beamforming algorithm which allows to efficiently extract a single speaker from a mixture by computing the mask in (1.11) which results in the linear estimate with the smallest MSE, i.e., the Linear minimum MSE (LMMSE) estimate. It is a semi-supervised method as it requires a VAD for the target speaker, i.e., it needs to know at which times the target speaker is not active in order to estimate the second-order statistics of the background noise and interfering speakers. The VAD information can be straightforwardly extracted by thresholding the speech envelopes obtained from the M-NICA algorithm (or alternatively using the deep learning algorithm). The s-th MWF computes the LMMSE mask $\hat{\mathbf{m}}_s(t,f)$ in (1.11) that yields the estimate $\hat{x}_s(t,f)$ that is closest to $\boldsymbol{h}_{s,j_{\mathrm{ref}}}(f)x_s(t,f)$, i.e., the s-th speaker's contribution in an arbitrarily-chosen reference microphone j_{ref}. The MWF $\hat{\mathbf{m}}_s(t,f)$ is defined as

$$\hat{\mathbf{m}}_s(t,f) = \underset{\mathbf{m}_s}{\arg\min}\, E\{|\boldsymbol{h}_{s,j_{\mathrm{ref}}}(f)x_s(t,f) - \mathbf{m}_s(t,f)^{\mathrm{H}}\boldsymbol{y}(t,f)|^2\}. \tag{7.3}$$

The solution for this LMMSE problem is given by [56]:

$$\hat{\mathbf{m}}_s(t,f) = \mathbf{R}_{\boldsymbol{y}\boldsymbol{y}}^{-1}\mathbf{R}_{x_s x_s}\mathbf{e}_{j_{\mathrm{ref}}}, \tag{7.4}$$

where $\mathbf{R}_{\boldsymbol{y}\boldsymbol{y}} = E\{\boldsymbol{y}(t,f)\boldsymbol{y}(t,f)^{\mathrm{H}}\}$, $\mathbf{R}_{x_s x_s} = E\{\boldsymbol{h}_s(f)\boldsymbol{h}_s(f)^{\mathrm{H}}x_s(t,f)^2\}$, and $\mathbf{e}_{j_{\mathrm{ref}}}$ denotes the j_{ref}-th column of a $J \times J$ identity matrix, which selects the column of $\mathbf{R}_{x_s x_s}$ corresponding to the reference microphone. The 'speech plus interference' autocorrelation matrix $\mathbf{R}_{\boldsymbol{y}\boldsymbol{y}}$ can be estimated during the time periods when the s-th speaker is active[2]. Assuming independence between all sources, $\mathbf{R}_{\boldsymbol{x}_s\boldsymbol{x}_s}$ can be estimated as $\mathbf{R}_{\boldsymbol{x}_s\boldsymbol{x}_s} = \mathbf{R}_{\boldsymbol{y}\boldsymbol{y}} - \mathbf{R}_{\boldsymbol{v}\boldsymbol{v}}$. Here, $\mathbf{R}_{\boldsymbol{v}\boldsymbol{v}} = E\{\boldsymbol{n}(t,f)\boldsymbol{n}(t,f)^{\mathrm{H}}\} + \sum_{s'\neq s} P_{s'}(f)\boldsymbol{h}_{s'}(f)\boldsymbol{h}_{s'}(f)^{\mathrm{H}}$ is the interference autocorrelation matrix, where $P_s(f) = E\{|\boldsymbol{x}_s(t,f)|^2\}$. $\mathbf{R}_{\boldsymbol{v}\boldsymbol{v}}$ can be estimated by averaging over all STFT frames in which speaker s is not active. Estimating $\mathbf{R}_{\boldsymbol{x}_s\boldsymbol{x}_s}$ as $\mathbf{R}_{\boldsymbol{y}\boldsymbol{y}} - \mathbf{R}_{\boldsymbol{v}\boldsymbol{v}}$ can result in poor filters, particularly, if $\mathbf{R}_{vv}$ contains non-stationary sources. Therefore, in our computations, $\mathbf{R}_{\boldsymbol{x}_s\boldsymbol{x}_s}$ was estimated based on a GEVD of $\mathbf{R}_{\boldsymbol{y}\boldsymbol{y}}$ and $\mathbf{R}_{\boldsymbol{v}\boldsymbol{v}}$ (details in [155]) to ensure robustness. Note that, while speech is known to be highly non-stationary, the expected values here are calculated over a long-term window, thereby capturing the average long-term power spectrum of the speech. Therefore, the MWF will not react to short-term changes in the speech spectra, it mainly focuses on the (fixed or slowly varying) spatial coherence across the microphones. To also exploit the fast spectral changes in the speech, a single-channel Wiener filter which uses short-term statistics can be added as a postfilter [37], yet this is beyond the scope of this chapter.

To separately estimate $\mathbf{R}_{\boldsymbol{y}\boldsymbol{y}}$ and $\mathbf{R}_{\boldsymbol{v}\boldsymbol{v}}$, the MWFs require the voice activity information of the speaker it must enhance. The energy envelopes from the output of the source separation algorithm (M-NICA or a DNN approach, see subsections 7.2.2 and 7.2.2, respectively) are used to identify the active and silent periods of each speaker, for e.g., through a thresholding operation[3], and hence form the VAD tracks that each MWF needs.

Note that the S MWFs work in parallel to enhance the speech streams of S different speakers. It is important to note that these MWFs can share a large part of the computations, as the estimation of the matrix inverse $\mathbf{R}_{\boldsymbol{y}\boldsymbol{y}}^{-1}$ is a common requirement across them. The real-time aspects of such a system will be discussed in Section 7.5.4.

Deep clustering for source separation

DC for BSSS has been explained in Section 1.4.2. In Section 6.3.1, a modification to the DC architecture has been proposed to cope with background noise using (6.1). DC has been developed for single microphone setups, but can easily be extended to include spatial information $\boldsymbol{\mathcal{Y}}_{\mathrm{sp}}$ from a multi-microphone setup [183]. This spatial information is simply appended to the input of the model to estimate the embeddings (and the noise mask): $[\mathbf{v}(t,f), \alpha(t,f)] = g_{tf}\left(|\boldsymbol{\mathcal{Y}}|, \boldsymbol{\mathcal{Y}}_{\mathrm{sp}}\right)$.

[2]For the sake of conciseness, the frequency variable f is omitted for autocorrelation matrices in this section.

[3]There is also the possibility of using other parameters, speech presence probability [67] for instance, to make softer VADs, but this is not within the scope of this chapter.

The IPD between two microphones j_1 and j_2 is chosen to represent the spatial information [183]:

$$y_{\mathrm{sp}}(t,f) = [\cos(\theta_{j_1,j_2}(t,f)), \sin(\theta_{j_1,j_2}(t,f))] \tag{7.5}$$

with $\theta_{j_1,j_2}(t,f) = \angle y_{j_1}(t,f) - \angle y_{j_2}(t,f)$ and $\mathcal{Y}_{\mathrm{sp}}$ is the matrix containing the spatial information $y_{\mathrm{sp}}(t,f)$ over all time-frequency bins (t,f). In case there are more than two microphones ($J > 2$), the spatial features can be determined for all microphone pairs and stacked as a single feature representation.

A single mask for each speaker s is produced, in the same way as was done in (1.80) or (6.3) which is then copied and applied to each microphone. The mask is applied to the average of all microphone spectrograms, or equivalently[4]

$$\hat{\mathbf{m}}_s(t,f) = \left[\frac{\hat{m}_s(t,f)}{J}, \frac{\hat{m}_s(t,f)}{J}, \ldots, \frac{\hat{m}_s(t,f)}{J}\right]^T \tag{7.6}$$

when using (1.11).

So far we have focused on estimating masks to filter out the interfering speaker (and noise). The estimated $\hat{x}_s(t,f)$ can be further enhanced in a secondary stage as in [88]. To this end, a new DNN is trained, denoted by the function g', which estimates an enhanced mask for speaker s based on $|\mathcal{Y}|$ and the previous estimate $\hat{x}_s(t,f)$: $\hat{m}_s^{\mathrm{enh}}(t,f) = g'_{tf}\left(|\mathcal{Y}(t,f)|, \hat{x}_s(t,f)\right)$. Notice that the network used for g' is shared over the speakers. In the multi-microphone case, the IPD is again included to obtain $\hat{\mathbf{m}}_s^{\mathrm{enh}}(t,f)$ similarly to (7.5)-(7.6).

In the next step, the envelopes of the speech sources as estimated from the mask (7.6) can be used for AAD (red dashed lines in Figure 7.1). There is also the option of using these envelopes to first compute the VADs for the S-fold MWFs, and then using the envelopes from these MWF outputs for attention decoding instead (green lines in Figure 7.1). This would correspond to the algorithm proposed in [43], where the M-NICA block is replaced with a DNN. A final option, which will not be further discussed in this chapter, is to use the masks estimates $\hat{m}_s(t,f)$ as a weight when determining $\mathbf{R}_{yy}$ and $\mathbf{R}_{vv}$, similarly as was done in [192]. Initial experiments with this method gave no significant improvements in AAD accuracies.

[4]This corresponds to applying the mask to the output of a forward-steering beamformer. Other choices are possible (e.g. select only one microphone, or summing magnitude spectra and adding the phase of one of the microphones), but these were empirically found to not have a significant impact on the results.

7.3 Experiments

7.3.1 Experiments setup

AAD setup

18 normal-hearing subjects (of which 6 males) between 20 and 25 years old participated in the experiment, where each subject had to focus on the speech stream from a particular direction, in the presence of a competing speaker from another direction and background noise. Their 64-channel EEG was recorded at a sampling rate of 8192 Hz. This dataset consists of 138 minutes of EEG recordings per subject. Further details can be found in [41]. The stimuli and the corresponding acoustic conditions used in this experiment are described below, in Section 7.3.1.

RBM Dataset

The RadioBoeken Mix (RBM) dataset was built by mixing parts of 8 Dutch stories narrated by 8 different female narrators [142], taken 2 at a time as competing speech streams in the presence of different levels of background multi-talker ('babble') noise. The background babble noise consisted of 9 different 4-talker babble streams (2-male and 2-female) perceived to be coming from 9 equidistant positions (from -180° to 140° in steps of 40°) around the listener. The babble was constructed from 36 audiobooks (18 male and 18 female narrators) from LibriVox (a public domain collection of audiobooks) [108]. Anechoic head-related transfer functions for 6 behind-the-ear microphones ($M = 6$) in a binaural hearing aid set-up [91] (with approximately 7.5 mm distance between neighbouring microphones) were used for making directional audio. All audio was sampled at 44.1 kHz. The long term spectrum of the babble sources was matched with the average spectrum of all the target speech streams. The experiments included a condition without background babble noise (referred to as the noise-free case) and two noisy conditions with SNRs[5] of -1.1 dB and -4.1 dB. Note that both SNRs are negative, resulting in challenging conditions where the signal power of the background babble noise is higher than the signal power of the attended speaker. Different scenarios regarding the relative position of both speakers are simulated, namely (-90° (utmost left), 90° (utmost right)), (30°, 90°), (-30°, -90°), and (-5°, 5°) with angle between speakers 180°, 60°, 60° and 10° respectively.

[5]In this chapter (input) SNR is defined as the ratio of the power of the target speaker to the power of background babble. Note that the actual SNR is even lower due to the presence of an interfering speaker, which is not included in this SNR metric.

This RBM data set is used in all evaluation experiments throughout this chapter, unless mentioned otherwise.

DNN training dataset

A separate audio dataset was needed to train the model for the speaker independent deep learning based source separation as explained in 7.2.2. A novel dataset, called Corpus Gesproken Nederlands Mix (CGNM), was created. It uses the *Corpus Gesproken Nederlands component-o-Flemish* (CGN-o-VL) [173], which contains 150 Dutch speakers of which 102 (51 female and 51 male) were randomly chosen to build up the training set, containing 27 hours of unique single-speaker speech in total (16 minutes per speaker on average). 20 000 2-speakers mixtures were artificially created using randomly selected single-speaker audio from CGN-o-VL. These mixtures totalled 60 hours of overlapping speech, to be used as training set. Even though in the RBM dataset (Section 7.3.1) only female-female mixtures are used, male-female and male-male mixtures are included in the training set as well for better generalizability. Furthermore, both speakers can be located anywhere between -100° and 100°, again to make the network more general. Two types of datasets were made: one containing the 20 000 mixtures without background babble noise (CGNM-noisefree), and the same mixtures with background babble noise (CGNM-noise). The SNR for the background noise for each mixture was uniformly sampled between -4 dB and 4 dB. Babble speakers were taken from the LibriSpeech corpus [132] and were positioned in the same way as in 7.3.1. All DNN models in this chapter are trained on this CGNM dataset. Both CGNM-noisefree and CGNM-noise were used to train the DNN models, unless specified otherwise.

To assess separation quality, 12 held-out female speakers (totalling 3.2 hours of unique speech) were chosen to create a test set. The test set consisted of 728 2-speaker mixtures (totalling 2.2 hours of mixed speech). While the RBM set remains the main dataset for evaluation throughout this chapter, this CGNM test set is introduced as it is more closely related to the CGNM train set, which was used to train the DNN models. Comparing DNN separation performance on the CGNM test set and the RBM set, will allow to address some generalization issues of the DNN in Section 7.4.1.

7.3.2 Data preprocessing and design choices

EEG preprocessing

All EEG data was filtered using an equiripple bandpass filter with passband between 0.5 Hz and 10 Hz, and with passband attenuation of at most 0.5 dB and stopband attenuation of 20 dB (lower) and 15 dB (upper). Previous studies [69, 55, 42] have shown that cortical envelope tracking is best within this frequency range. EEG data was then downsampled to 40 Hz.

M-NICA audio preprocessing

The microphone signals were lowpass filtered with an 800 Hz cut-off and downsampled to 8 kHz before computing the energy envelopes, since it was found to be beneficial for the source separation process under low SNR conditions [43]. Energy was computed every 200 samples (25 ms), bringing down the sampling rate to 40 Hz.

Deep learning design choices

Most of the experimental setup was taken the same as in Section 3.3.1. The enhancement network g' is used for the first time in the text and was chosen to be a BLSTM-RNN with 2 layers using 300 hidden units each. The output layer had $F = 129$ nodes and a sigmoid function was applied on top to estimate the final masks.

For the single-channel scenario a single microphone was randomly selected. For the multi-channel scenario the input consisted of the average of the left-ear and the average of the right-ear microphone signals, as well as the spatial features (see (7.5)) from 3 microphone pairs (with one pair using a left-ear and a right-ear microphone and pair 2 and 3 using two left-ear and two right-ear microphones, respectively).

MWF design choices

For the estimation of MWF coefficients an STFT with a 64 ms window length (512 samples) and a hop size of 32 ms (256 samples) was applied to the microphone signals, after they were downsampled to 8 kHz. The thresholds for estimating the VAD tracks for the different approaches were determined empirically. For energy envelopes extracted from the M-NICA and DNN-based source separation, the 25th percentile of the per sample amplitude of the envelope was used as the threshold, above which the speaker was considered to be active.

Audio envelope extraction

The envelope extraction method from [18] was used. Each of the speech streams used as stimulus were filtered with a gammatone filterbank [134], splitting the signal into 15 filter bands. In each gammatone band, the absolute value of each sample was taken after which a power law compression with an exponent of 0.6 was applied. The power law-compressed samples were then bandpass filtered in the same manner as was done for the EEG signals resulting in subband envelopes. The subband envelopes were added together and downsampled to 40 Hz to form a single 'powerlaw subband' envelope. Power law compression aims to replicate the non-linear transformation of the stimulus in the human auditory system, with relatively higher attenuation of higher amplitude signals [163]. This method of auditory inspired envelope extraction has been found to result in significantly better attention decoding accuracies than other envelope extraction methods[18].

Power law subband envelopes were only used in cases where broadband speech streams where available, i.e., the DNN and MWF envelopes built from the speech streams obtained with the DNN-based mask (7.6) or the MWF-based mask (7.4), respectively. Since M-NICA directly extracts energy envelopes from the broadband speech streams, the gammatone filterbank could not be applied here. When applying AAD on the M-NICA-envelopes, the square-root of the extracted energy envelopes was used to transform them to an amplitude envelope, since energy envelopes are sub-optimal to perform AAD [18].

AAD design choices

In order to perform AAD, EEG data was split into trials of 30 s, resulting in 276 trials per subject for each of the 3 different angles between speakers (separations of 180° for (-90°,90°), 60° for (30°,90°) and (-30°,-90°), and 10° for (-5°,5°) respectively). Decoders were trained (using leave-one-trial-out (LOO) cross validation) to reconstruct the envelope of the attended speech stream. To improve decoding accuracy and eliminate the need for regularization, a single decoder was computed over the entire training set data as in [18], instead of averaging over per-trial decoders in the training set as in [130].

	noise-free CGNM test	noise-free RBM
single-channel	10.0 dB	3.8 dB
multi-channel	12.3 dB	12.3 dB

Table 7.1: SDR improvement results for a single-channel and multi-channel DNN without noise.

	10°	60°	180°	avg.
single-channel	2.5 dB	1.7 dB	7.3 dB	3.8 dB
multi-channel	12.7 dB	11.6 dB	12.5 dB	12.3 dB

Table 7.2: SDR improvement results for a single-channel and multi-channel DNN for the noise-free RBM set, depending on the difference in speaker positions.

7.4 Results

7.4.1 Single-channel and multi-channel source separation

In some cases, for hardware reasons, a single-channel set-up can be preferred over a multi-channel set-up. The deep learning approaches to noise-free source separation, both for single-channel and multi-channel setup, were trained on the CGNM-noisefree train set and evaluated on the CGNM-noisefree test set and the RBM noise-free set. Results are expressed in SDR [176] improvements and are shown in Table 7.1.

Notice that for the single-channel case, performance on the noise-free RBM dataset is much worse than on the CGNM-noisefree test set, even though both consist of held-out female speakers. The DNN generalizes poorly to other datasets. A possible reason for this could be differences in the recording set-up, and the fact that the CGNM-noisefree test set consists of volunteers, while the RBM dataset consists of children stories told by professional story-tellers. It is concluded that the single-channel setup does not generalize well across data sets. Multi-channel separation quality is considerably better than the single-channel counter part. This was expected as the network can make use of the spatial information. Furthermore, the SDR results on the CGNM-noisefree test set and the RBM dataset are also more alike, compared to the single-channel results. The network uses spatial information to characterize the speakers and does not need to solely rely on the difference in speaker characteristics to separate

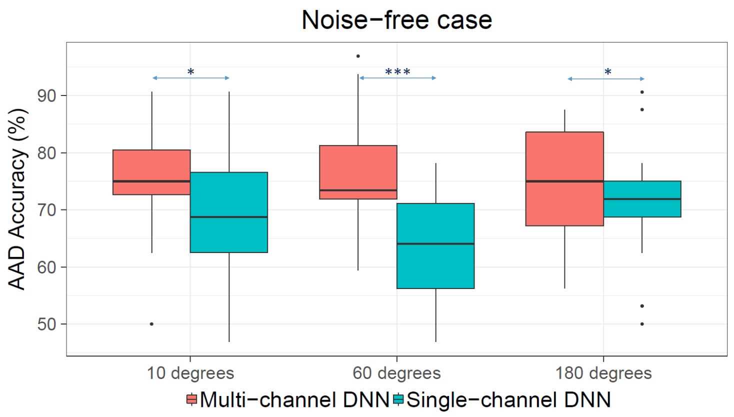


Figure 7.2: AAD accuracies (on 30 s trials) of 18 subjects listening in noise-free conditions when envelopes from the output of multi-channel DNN and single-channel DNN were used. Comparisons between the different approaches were done using Wilcoxon's signed-rank test [187] with Holm correction. : '***' for $p < 0.001$, '*' for $p < 0.05$.

the speakers, as is necessary in the single-channel case [196]. In Table 7.2, the SDR results on the RBM dataset are split up per angle between speakers. For the single-channel setting, it is observed that the model performs much better for the 180° case compared to the other cases. For the 180° case the speech amplitudes of both speakers will differ more due to head shadow effects, which may explain why it is easier for the network to separate both. We believe the network uses this difference in amplitude to distinguish between the speakers and once again does not need to solely rely on the spectral characteristics. This head shadow effect might also happen, although less pronounced, for the 10° (speakers at -5° and 5°) case, which could explain why it performs slightly better than the 60° (speakers at 30° and 90°) case, where both speakers are on the same side of the head. Further research is needed to support these claims/assumptions.

The absolute envelopes, extracted from the outputs of both single-channel and multi-channel DNNs performing speaker separation of the stimuli in the RBM dataset for the noise-free condition, were used for auditory attention decoding of the EEG data collected under matching acoustic conditions and stimuli. Figure 7.2 shows AAD accuracies over 18 subjects under the noise-free condition, for the 3 different angles between the speakers. For all angles between speakers, the multi-channel DNN resulted in significantly higher AAD accuracies compared to the single-channel DNN, as was expected from the results in Table 7.1. We therefore continue with the multi-channel approach for all further analyses.

7.4.2 Linear versus non-linear source separation for AAD

The per-subject AAD accuracies on the RBM dataset were computed for each of the 3 different angles between speakers and the 3 noise conditions. For each 30 s trial, the correlation of the reconstructed attended envelope was computed with 5 pairs of envelopes extracted using different BSS methods:

- ‘Oracle’ - powerlaw subband envelopes extracted from the original speech signals (not mixed)
- ‘M-NICA’ - amplitude envelopes extracted from the output of the M-NICA source separation algorithm (red dashed track in Figure 7.1)
- ‘DNN’ - powerlaw subband envelopes extracted from the output of the DNN-based source separation algorithm (also red dashed track in Figure 7.1).
- ‘M-NICA+MWF’ - powerlaw subband envelopes extracted from the output of MWFs when using the output of the M-NICA based source separation algorithm to build VAD tracks (green track in Figure 7.1).
- ‘DNN+MWF’ - powerlaw subband envelopes extracted from the output of MWFs when using the output of the DNN based source separation algorithm to build VAD tracks (also green track in Figure 7.1).

Figure 7.3 shows the resulting AAD accuracies for the 18 subjects for the noise-free condition, for -1.1 dB and for -4.1 dB. For each angle between speakers, comparison between the different approaches were done using Wilcoxon’s signed-rank test with Holm correction. The green boxplots correspond to the green lines (MWF envelopes), and the red box plots correspond to the dashed red lines (M-NICA/DNN envelopes) respectively in Figure 7.1.

An interesting observation from Figure 7.3 is that a training-free linear algorithm as proposed in [43] (here represented by ‘M-NICA+MWF’) is able to perform at least on par and often even better than a pre-trained complex deep model (here represented by ‘DNN’) in all speaker angle and noise conditions. When breaking down the subcomponents of [43] into M-NICA and MWF separately, or when using the DNN as support for the MWF, some other interesting observations can be made, which are briefly reported in the remainder of this subsection.

In the noise-free condition, for 180° and 10° separations, the M-NICA envelopes result in accuracies that match those when using oracle speech envelopes for AAD

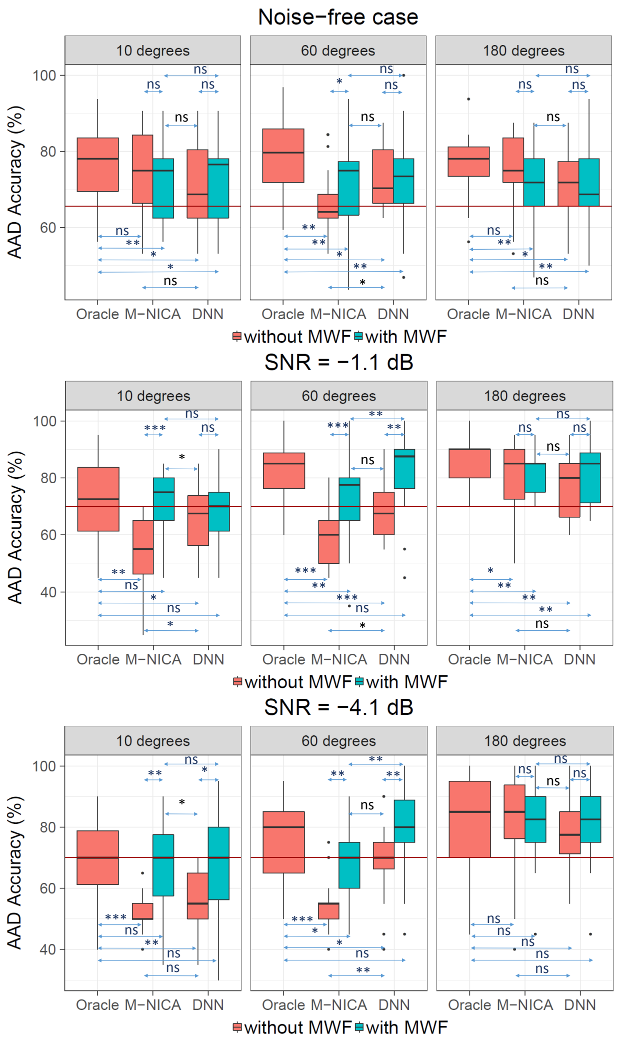


Figure 7.3: AAD accuracies (on 30 s trials) of 18 subjects for different speech separation approaches for 3 cases: the noise-free case, -1.1 dB SNR, and -4.1 dB SNR. The results of using the M-NICA and DNN approaches are compared with the results when using oracle speech envelopes for AAD. The separated streams were either directly used for AAD (red boxes here and dashed red lines in Figure 7.1), or used to generate VAD tracks for MWFs (green boxes here and green lines in Figure 7.1), the outputs of which were then used for AAD. The red line indicates the significance level. Comparison between the different approaches were done using Wilcoxon's signed-rank test with Holm correction. : '***' for $p < 0.001$, '**' for $p < 0.01$, '*' for $p < 0.05$, 'ns' for no significant difference.

while the DNN envelopes perform significantly poorer. On the other hand, for 60° speaker separation, the DNN envelopes outperform M-NICA envelopes, yet upon including the MWF filtering step, there are no more significant differences between the 2 source separation approaches. However, the results are still significantly lower than with oracle speech envelopes. It must be noted that in the 60° case, the competing speaker positions are on the same side of the listener's head ((-30°,-90°) or (30°,90°)). It has already been observed in [43] that the same 60° case was challenging for M-NICA, indicating difficulty to extract discriminative spatial features under such conditions

Figure 7.3 also shows AAD accuracies from trials in a noisy condition with -1.1 dB SNR. For 180° speaker separation, it can be seen that all source separation approaches, with or without the MWF filtering step, result in similar AAD performances, but do not match up to that of oracle speech envelopes. Nevertheless, the performance decrease is minimal, in particular when an MWF is used. For 10° and 60° speaker separation, the DNN envelopes result in better performance than the M-NICA envelopes. MWF filtering substantially improves results for both approaches for the 60° case, where the DNN+MWF combination even matches performance with the oracle speech envelopes. For the 10° case, only the M-NICA approach requires MWF filtering to match oracle speech performance.

For the -4.1 dB SNR (Figure 7.3), which is a highly challenging condition, it is observed that for 180° separation, the performances of all the approaches do not differ significantly from each other. However, for the 10° condition, the M-NICA and DNN envelopes result in accuracies below chance level, and only with the additional MWF filtering, do the results match up to those of oracle speech envelopes. In the 60° case, the MWF filtering does improve results for both the source separation approaches, however only the results of the DNN+MWF approach match up to those of oracle speech envelopes.

7.4.3 Speech separation performance

To estimate the effectiveness of the MWF for BSS, we looked at the improvement in the Signal-to-Interference-plus-Noise Ratio (SINR) and PESQ, on the RBM dataset, at the output of the MWFs when using VAD tracks generated from 3 sources: the oracle speech envelopes, the M-NICA envelopes, and the DNN envelopes. The SINR is defined as the ratio of the power of the attended speaker to the total of the power of the unattended speaker and the background noise. The reference input SINR is taken as the highest SINR among the 6 microphones (note that the achieved SINR improvement will be larger for the other microphones). For each noise condition and angle between speakers, the

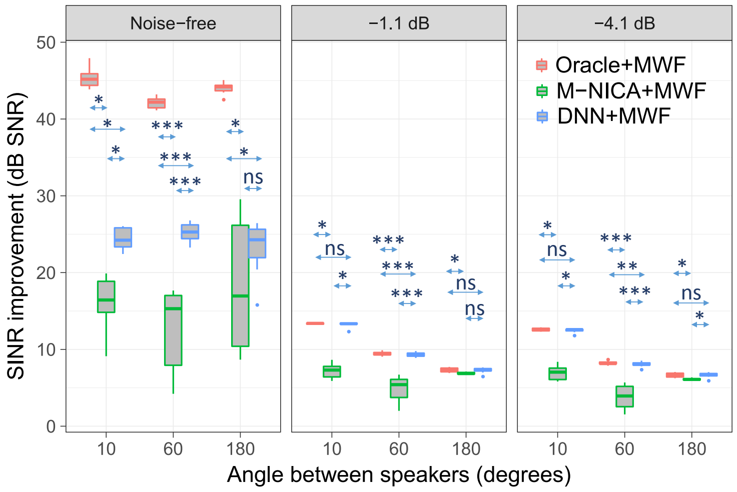


Figure 7.4: Boxplots showing improvement in SINR of 8 story parts when the MWFs use VAD tracks extracted from the different speech separation approaches. The results of using M-NICA and DNN outputs for the VAD are compared with the results when using oracle speech for the VAD. Comparison between the different approaches were done using Wilcoxon's signed-rank test with Holm correction. : '***' for $p < 0.001$, '**' for $p < 0.01$, '*' for $p < 0.05$, 'ns' for no significant difference.

improvement in SINR was computed for 8 story parts to which the subjects had to listen without interruption. Figure 7.4 shows the improvement in SINRs of the 8 story parts, for the 3 noise conditions and 3 different angles between speakers. For the noise-free case, it is observed that the oracle speech envelopes result in better BSS than M-NICA or DNN envelopes in all separation angles (although perceptually there is not much difference: the difference between 40dB and 20dB SINR is hard to notice). However, in the presence of background noise, the DNN+MWF approach significantly outperforms the M-NICA+MWF approach, and matches up to the performance with the oracle speech envelopes in the 10° and 180° cases. For the acoustically more challenging case of 60° (since the speakers are on the same side of the head), it is observed that the DNN+MWF approach performs almost similar to the oracle+MWF approach (although the small differences are still statistically significant), and both of them substantially outperform the M-NICA+MWF approach.

The results for PESQ, a measure that focuses more on speech intelligibility, are shown in Figure 7.5. The significance tests are very similar to those of SINR

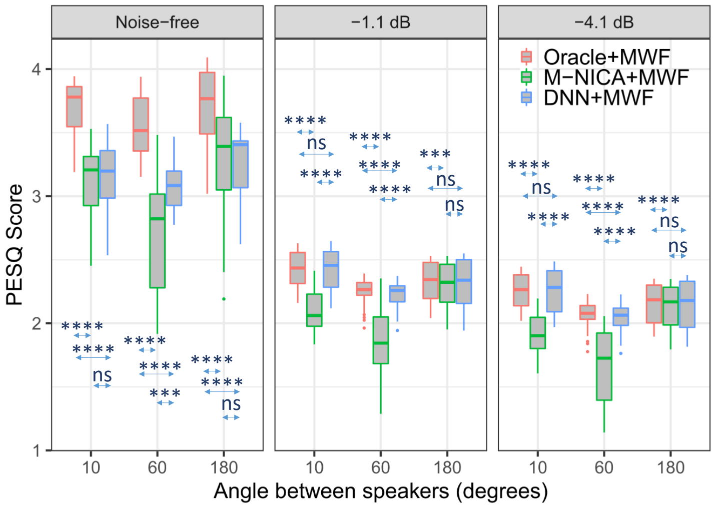


Figure 7.5: Boxplots showing PESQ scores of 8 story parts when the MWFs use VAD tracks extracted from the different speech separation approaches. The results of using M-NICA and DNN outputs for the VAD are compared with the results when using oracle speech for the VAD. Comparison between the different approaches were done using Wilcoxon's signed-rank test with Holm correction. : '****' for $p < 0.0001$, '***' for $p < 0.001$, '**' for $p < 0.01$, '*' for $p < 0.05$, 'ns' for no significant difference.

and thus the above observations for SINR also hold for PESQ.

7.5 Discussion

Previous studies that combine speech (B)SS with AAD investigated only one type of algorithm (either linear [174, 43, 7] or DNN-based [129, 74]), mostly in relatively mild acoustic conditions. Furthermore, each study has its own specific set-up (single- vs. multi-microphone, ECoG vs. EEG, matched or mismatched acoustic conditions during EEG recordings, etc.) which does not allow to directly compare the results achieved in these studies. The study in this chapter was designed to investigate the feasibility of a neuro-steered BSS pipeline in challenging acoustic conditions with negative SNRs and various speaker positions, while also comparing different source separation approaches to produce the speech envelopes used by the attention decoding block. We investigated the AAD performance of a training-free and purely linear audio signal processing algorithm (M-NICA+MWF) on the one hand, and a non-linear DNN on the other hand, as well as a combination of a linear with a non-linear approach (DNN+MWF), where the latter acts as a supporting voice activity detector for the former. In the remainder of this section, we will discuss the main findings of our study.

7.5.1 Multi-microphone outperforms single-microphone

Existing AAD studies using DNNs for the acoustic source separation were based on single-microphone recordings [129, 74]. We observed that a single-channel approach generalizes less well across datasets and that a multi-channel approach is more robust in terms of separation quality. It was concluded that single-microphone solutions are reliant on spectral information for speaker characterization and this might generalize poorly across datasets. This is another examples were the DNN does not sufficiently learn to internally model speaker characteristics (see Chapters 3 and 4).

In addition, envelopes extracted from the output of the multi-channel neural network resulted in AAD accuracies that were more robust to varying speaker positions compared to those from the single-channel neural network. It is concluded that a multi-microphone set-up is crucial to obtain a sufficiently high AAD accuracy in practical settings. Thus, we chose the multi-channel neural network approach for our comprehensive analysis of the neuro-steered source separation.

7.5.2 Linear algorithm outperforms DNN

In the literature, several algorithms have been proposed to combine AAD with speech separation methods. Concerning the speech separation part, two main strategies can be distinguished; those based on traditional (linear) beamforming approaches [174, 43, 7] and those based on (non-linear) neural networks [129, 74]. However, these two strategies have never been compared to each other in terms of the resulting AAD performance. The results in Section 7.4 demonstrate that the AAD performance with a linear speech separation algorithm (represented by 'M-NICA+MWF' in figure 7.3) is at least on par and often outperforms a non-linear DNN approach (represented by 'DNN' in figure 7.3) in all investigated conditions. This holds both in terms of AAD performance (Fig. 7.3) and BSS performance (Fig. 7.4). In addition, the linear method is computationally cheaper than the DNN approach (in terms of operations per second and memory size), while also being training-free thereby not depending on representative training data. Furthermore, implementing the MWF as an adaptive, causal, low-latency filter is relatively straightforward, and latencies can be limited to less than 5 ms [172] (see also Section 7.5.4 below). Speech separation based on deep clustering on the other hand, usually uses a non-causal network where the computation of the embedding vector at present time is based on all past and future time samples, although recently also causal DNNs for BSS have been proposed [73, 74].

7.5.3 Combining the best of both worlds: DNN and MWF

While linear and non-linear approaches both individually result in good speech separation, a combination of both - as in the DNN+MWF approach - resulted in the best AAD performance, particularly in challenging acoustic conditions. In this case the (non-linear) DNN is used as a VAD mechanism to inform the (linear) MWF which performs the actual speaker separation and denoising, thereby combining the best of both worlds. This results in the best AAD performance as well as the highest SINR improvement.

7.5.4 Real-time aspects

When using a weighted overlap-add (WOLA) procedure [38], the algorithmic delay of the MWF is equal to the window length used in the STFT computations, i.e., the number of samples over which the discrete Fourier transform is computed. For example, for a hearing aid using a sampling rate of 20480 Hz and a 96-point STFT windowing [172], the algorithmic delay is less than 5 ms. Note that a delay in the VAD information (e.g. due to possible non-causality of the BSS block in Figure 7.1), does not have an impact on the overall input-output delay of the stimulus presented to the user. This is because the adaptation of the MWF filter coefficients can be decoupled from the actual filtering operation itself, where the former can lag behind on the latter. This implies that any delay or non-causality in the M-NICA or DNN block will not create a bottleneck towards real-time speech processing. It will only add some inertia in the updating of the MWF coefficients to adapt to changes in the acoustic scenario.

7.5.5 EEG-based AAD is feasible in challenging low-SNR conditions

The SNRs in this study were chosen such that the speech intelligibility varied across conditions. The estimated speech recognition threshold (SRT), at which a normal-hearing subject can understand 50% of the attended story on average, was found to be -7.1 dB (see [41] for details of the estimation procedure). The SNRs -4.1 dB and -1.1 dB were chosen by adding 3 dB steps from the 50% speech intelligibility point. The -7.1 dB condition was excluded from this study due to the difficulty subjects faced in focusing on the attended speaker (as reflected in the AAD accuracies under this condition reported in [41]). Subjective speech intelligibility reported during the recording sessions for the different angles between speakers show that -4.1 dB and -1.1 dB SNRs also are not easy listening conditions. Compared to [7] where the noise conditions are relatively mild (4 dB and 9 dB SNR), [74] where there is no background noise, and [43] where the EEG recording conditions did not match the acoustic conditions, our current study provides a holistic view on the performance of a neuro-steered BSS pipeline under realistic to challenging acoustic conditions. It is found that not only EEG-based AAD with no access to clean-speech sources is feasible under challenging acoustic conditions, but also that the AAD accuracies and improvement in SINRs for DNN+MWF get remarkably close to oracle performance.

7.5.6 Future outlook

The proposed neuro-steered BSS and its analysis over a range of acoustic conditions is a step towards effectively incorporating attention decoding and source separation algorithms in neuro-steered hearing prostheses. With the field of deep learning expanding and exerting its presence in various scientific domains, we have tried to address the possibility of deep learning based source separation in this context and systematically compare it with a classical signal processing approach. While the realization of neuro-steered hearing prostheses is still far from being a reality, there are ongoing advancements that can accelerate this process. In addition to hardware requirements such as miniaturization of neural recording equipment, better computational and storage capabilities in hearing devices etc., also newer approaches for improved attention decoding itself based on DNNs [48, 49, 34], canonical correlation analysis [46] or state-space modelling [4, 118] can contribute to faster and more robust attention decoding and hence better neuro-steered BSS. Finally, a limitation of this study is that it does not include reverberant conditions, which can both affect speech separation algorithms and the AAD performance.

7.6 Conclusion

In a multi-speaker *cocktail party* scenario, a hearing aid's noise reduction algorithm does not have the knowledge of which speaker the user intends to listen to. Incorporating AAD algorithms to detect the attention of the user, leads to the concept of so-called neuro-steered hearing aids. In this study a neuro-steered BSS pipeline is presented, without access to clean speech signals in challenging acoustic scenarios.

The performance of a linear as well as DNN-based speaker separation approach was analyzed. It was found that a purely linear approach often outperforms the DNN-based approach. However, the best performance, under challenging scenarios, was obtained when combining the best of both worlds, i.e., when using the DNN to provide VAD information to a linear data-driven beamformer. In addition, for the DNN-based speaker separation, the benefit of using a multi-microphone system compared to a single-microphone system was demonstrated. It was concluded that single-microphone solutions are reliant on spectral information for speaker characterization and this might generalize poorly across datasets. An additional advantage of using a linear data-driven beamformer is that it decouples the algorithmic delay in the blind source separation of the audio signals from the total system delay, thus facilitating robust real-time speech processing. With this study, we present a proof-of-concept of feasibility of a neuro-steered BSS pipeline in challenging acoustic conditions.

Chapter 8

Conclusion

To end the thesis an overview of the original contributions is given, as well as some directions for future research.

8.1 Original contributions

Joint Speaker Separation and Recognition using NMF

A state-of-the-art multi-channel NMF BSSS model was extended to allow for joint BSSS and SR in the presence of overlapping speech. This was done by introducing a latent variable that linked an enrolled speaker's dictionary to a speech signal. Estimating this latent variable links the speech signal, the speaker identity and the speaker's location. The model outperformed NMF and i-vector baselines in terms of SR performance.

Improving Source Separation via Speaker Representations

To study the importance of speaker characterization for BSSS when using DNNs, a blind multi-speaker adaptation model was proposed. This model iteratively applies BSSS and i-vector estimation. This allowed to verify whether state-of-the-art DNN based BSSS models build a sufficient internal speaker representation. It was concluded that large models in scenarios without background noise and reverberation indeed manage this. However, for a smaller model it was found that the internal speaker representations were not sufficient and applying the blind multi-speaker adaptation model improved performance. This blind multi-speaker adaptation technique was also used in Chapter 5 to study the importance of the SR subtask in the memory of an LSTM-RNN for BSSS.

Joint Speaker Separation and Recognition using Deep Learning

A joint BSSS and SR model using DL in the presence of overlapping speech was studied. Different ways to apply multi-task learning were considered and it was concluded that summing the weight updates over the tasks was preferred compared to summing the tasks' losses. Furthermore, it was found that optimal SR performance is achieved when giving sufficient importance to the BSSS task. However, the BSSS task was not helped by adding the auxiliary SR task. This observation is consistent with Chapter 3. Again, a comparison with a smaller model was made and it was concluded that for this smaller model, the SR task did help the BSSS task.

An extension to the joint model was made to allow for single speaker audio. If clean enrollment data for a speaker is available, this improves the SR performance. Furthermore, an additional extension to the training method was proposed such that the SR better takes into account the uncertainty on the BSSS. This improved the SR performance.

Analysis of Memory in RNNs for Blind Speech Source Separation

Two methods, the leaky approach and the reset approach, were developed to study the importance of different memory time spans in RNNs. If large computational complexity is allowed, the reset approach is favored. It was found that for the task of BSSS, short-term linguistic processes have a strong impact on the separation performance. Above 400 ms the network can only learn better speaker characterization and other possible separation cues like grammar are not considered by the LSTM-RNN. Furthermore, it was found that it is sufficient to only implement longer memory in the deeper layers. Finally, it was found that the backward direction is slightly more important than the forward direction for a bidirectional LSTM-RNN for BSSS.

Increasing Separation Robustness in Realistic Conditions

Analysis has been done on practical aspects of BSSS that are relevant in realistic settings:

1. A joint model for 2-speaker and 3-speaker mixtures is built, that even outperforms the task specific models. This avoids the need to build a different model for each mixture type.
2. The DL based BSSS models are evaluated on different languages and seem generally applicable to any language. Furthermore, it is noted that a BSSS model trained on one language generalizes to some extent to a different language.
3. Extensions to DL based BSSS models are proposed to cope with additional background noise. An improvement is found compared to simply treating the noise as an additional speech signal.

Speech Separation for EEG-informed Attention Decoding

The concept of a neuro-steered AAD was studied. The use of BSSS in this application is twofold. First, the envelopes extracted from the speech reconstructions can be compared with the envelope estimates of the AAD to estimated the attended speaker of the subject. Secondly, once the attended speaker is estimated, the speech reconstruction of this speaker should be sent to the hearing aid. A traditional linear BSSS approach, M-NICA, is compared to DL. It was found that the DL based solution, when combined with a MWF, achieved the best performance for noisy same gender mixtures. However, the computational complexity of DL compared to M-NICA is an important drawback for the application. It was also found that the single-channel DNN had trouble generalizing to a different dataset, while the multi-channel version was robust to this.

8.2 Future outlook

This thesis showed that a joint model for BSSS and SR is beneficial for SR performance (Chapters 2 and 4). Furthermore, the importance of the SR subtask for BSSS was shown in Chapter 5.

These insights are interesting from an academic point of view, but they can also be used in order to obtain performance improvement in speech technology applications. It has been shown in Chapters 2 and 4 that the SR problem in overlapping speech is addressed more successfully when explicitly taking the BSSS-SR dependency into account. The same is found for BSSS in Chapters 3 and 4 when considering smaller models. The analysis in Chapter 5 has shown that both short-term and long-term effects play a role, where the latter focuses only on SR. This fact can be exploited in two ways. From a computational complexity point, one could argue that the hidden units focusing on the long-term dependencies, do not necessarily need to be updated at every time step. This point is similar to the one being made in clockwork RNNs [99]. On the other hand, this insight also allows to propose new architectures that use this information to obtain better BSSS performance. Such an attempt was made in [198], written by the same author as this thesis. There, a hybrid CNN-LSTM model was designed with the idea that the CNN part should focus more on the short-term dependencies and the LSTM should focus on the long-term dependencies. An improvement over an LSTM-only model was found.

However, it was also concluded in Chapters 3 and 4 that state-of-the-art BSSS models do not benefit from an auxiliary SR input or target. It was concluded that these models build a sufficient internal speaker representation for BSSS. However, it must be noted that these models were studied in highly idealized settings, without background noise or reverberation, and that no practical limitation was set on the size of the models. It was found that for a smaller network, BSSS is helped by an auxiliary SR input or target. Future research should focus on whether this conclusion can be generalized to (very) challenging scenarios, such as background noise or reverberation.

In fact, very recent work on time-domain solutions for BSSS [158, 159] have surpassed the performance of optimal mask based time-frequency domain solutions. However, as more realistic and challenging scenarios are considered (noisy, reverberant, real-time-constraints, . . .), it remains unknown how these time-domain solutions will cope as the upper bound of mask based solutions will be harder to achieve [12, 77]. Furthermore, one can question the added benefit of achieving a separation performance that exceeds the optimal mask based solution. In general, two further directions for research in BSSS can be considered. The first one being to further continue to push the estimated speech signals to near-perfection in highly idealized, studio-like and artificial scenarios. Alternatively, future research could focus on more challenging scenarios. This thesis focused mostly on the first direction (apart from Chapters 6 and 7), since it allowed fundamental research on DL for BSSS and most BSSS researchers focus on this domain. However, now that a good understanding and good separation performance is achieved in these idealized scenarios, it is time to focus more on the second direction. It is expected that in this case approaches that help the BSSS, such as a joint approach and adding external speaker representations, once again become more important.

Appendix A

NMF Derivations

This appendix will derive the update formulas found in Chapter 2. The derivation is based on [151]. The update formulas will be derived in Appendix A.1, which will use some basic matrix derivatives that are described in Appendix A.2.

A.1 Derivation of multi-channel NMF for joint BSSS and SR

The multi-channel IS divergence of (1.19) is retaken but constant terms are omitted,

$$\begin{aligned} f\left(\mathbf{T}, \mathbf{Q}, \mathbf{H}, \mathbf{Z}, \mathbf{C}\right) &= D_{\mathrm{IS}}(\mathbf{X}, \{\mathbf{T}, \mathbf{Q}, \mathbf{H}, \mathbf{C}\}) \\ &= \sum_f \sum_t \mathrm{tr}(\mathbf{X}_{ft}\hat{\mathbf{X}}_{ft}^{-1}) + \mathrm{logdet}(\hat{\mathbf{X}}_{ft}), \end{aligned} \tag{A.1}$$

where

$$\hat{\mathbf{X}}_{ft} = \sum_k \sum_i \sum_s z_{si} c_{ik} t_{fk} q_{kt} \mathbf{H}_{fs}, \tag{A.2}$$

as in (2.2). To minimize $f(\mathbf{T},\mathbf{Q},\mathbf{H},\mathbf{Z},\mathbf{C})$, the optimization scheme of majorization is used [24, 47, 151]. An auxiliary function is defined as

$$f^{+}(\mathbf{T},\mathbf{Q},\mathbf{H},\mathbf{Z},\mathbf{C},\mathbf{R},\mathbf{U}) =$$
$$\sum_f\sum_t\left[\sum_k\sum_i\sum_s\frac{\mathrm{tr}\left(\mathbf{X}_{ft}\mathbf{R}^{\mathrm{H}}_{ftkis}\mathbf{H}^{-1}_{fs}\mathbf{R}_{ftkis}\right)}{z_{si}c_{ik}t_{fk}q_{kt}}+\mathrm{logdet}(\mathbf{U}_{ft})+\frac{\det(\hat{\mathbf{X}}_{ft})-\det(\mathbf{U}_{ft})}{\det(\mathbf{U}_{ft})}\right], \tag{A.3}$$

where $\mathbf{R}_{ftkis}$ and $\mathbf{U}_{ft}$ are auxiliary variables that satisfy positive definiteness, $\mathbf{R}_{ftkis}$ is constrained to $\sum_k\sum_i\sum_s\mathbf{R}_{ftkis}=\mathbf{I}$ with $\mathbf{I}$ the identity matrix of size J, and $\mathbf{R}_{ftkis}$ and $\mathbf{U}_{ft}$ are Hermitian symmetric. It will be shown that f^{+} has two properties

1. $f(\mathbf{T},\mathbf{Q},\mathbf{H},\mathbf{Z},\mathbf{C}) \leqslant f^{+}(\mathbf{T},\mathbf{Q},\mathbf{H},\mathbf{Z},\mathbf{C},\mathbf{R},\mathbf{U})$
2. $f(\mathbf{T},\mathbf{Q},\mathbf{H},\mathbf{Z},\mathbf{C}) = \min_{\mathbf{R},\mathbf{U}}f^{+}(\mathbf{T},\mathbf{Q},\mathbf{H},\mathbf{Z},\mathbf{C},\mathbf{R},\mathbf{U})$

The function f can then be minimized by iteratively repeating these two steps:

1. Minimizing f^{+} with respect to $\mathbf{R}$ and $\mathbf{U}$, which brings $f^{+}(\mathbf{T},\mathbf{Q},\mathbf{H},\mathbf{Z},\mathbf{C},\mathbf{R},\mathbf{U})$ closer to $f(\mathbf{T},\mathbf{Q},\mathbf{H},\mathbf{Z},\mathbf{C})$ (see Section A.1.1).
2. Minimizing f^{+} with respect to $\mathbf{T}$,$\mathbf{Q}$ and $\mathbf{H}$, which also minimizes f (see Section A.1.2).

A.1.1 Minimizing f^{+} with respect to R and U

To minimize f^{+} with respect to $\mathbf{R}$, with the constraint that $\sum_k\sum_i\sum_s\mathbf{R}_{ftkis}=\mathbf{I}$, Lagrange multipliers $\mathbf{\Lambda}_{ft}$ are introduced such that

$$\mathcal{F} = f^{+} + \sum_f\sum_t\mathrm{Re}\left\{\mathrm{tr}\left[\left(\sum_k\sum_i\sum_s\mathbf{R}_{ftkis}-I\right)^{\mathrm{H}}\mathbf{\Lambda}_{ft}\right]\right\}. \tag{A.4}$$

The partial derivative of $\mathcal{F}$ with respect to $\mathbf{R}_{ftkis}$ is then

$$\begin{aligned}\frac{\partial \mathcal{F}}{\partial \mathbf{R}_{ftkis}} &= \frac{1}{z_{si}c_{ik}t_{fk}q_{kt}} \frac{\partial \text{tr}\left(\mathbf{X}_{ft}\mathbf{R}_{ftkis}^{\text{H}}\mathbf{H}_{fs}^{-1}\mathbf{R}_{ftkis}\right)}{\partial \mathbf{R}_{ftkis}} + \\ &\quad \frac{\partial \text{Re}\left\{\text{tr}\left[\left(\sum_k \sum_i \sum_s \mathbf{R}_{ftkis} - I\right)^{\text{H}} \mathbf{\Lambda}_{ft}\right]\right\}}{\partial \mathbf{R}_{ftkis}} \qquad (A.5) \\ &= \frac{2\mathbf{H}_{fs}^{-1}\mathbf{R}_{ftkis}\mathbf{X}_{ft}}{z_{si}c_{ik}t_{fk}q_{kt}} + \mathbf{\Lambda}_{ft},\end{aligned}$$

where the equality for the first term follows from (A.35) and for the second term from [137, eq. (103)]. Setting this partial derivative to zero gives

$$\frac{\partial \mathcal{F}}{\partial \mathbf{R}_{ftkis}} = \frac{2\mathbf{H}_{fs}^{-1}\mathbf{R}_{ftkis}\mathbf{X}_{ft}}{z_{si}c_{ik}t_{fk}q_{kt}} + \mathbf{\Lambda}_{ft} = \mathbf{0} \qquad (A.6)$$

$$\mathbf{R}_{ftkis} = -z_{si}c_{ik}t_{fk}q_{kt}\mathbf{H}_{fs}\mathbf{\Lambda}_{ft}\mathbf{X}_{ft}^{-1}. \qquad (A.7)$$

Since $\sum_k \sum_i \sum_s \mathbf{R}_{ftkis} = \mathbf{I}$ and applying (A.7) leads to

$$\sum_k \sum_i \sum_s \mathbf{R}_{ftkis} = \left(\sum_k \sum_i \sum_s -z_{si}c_{ik}t_{fk}q_{kt}\mathbf{H}_{fs}\right)\mathbf{\Lambda}_{ft}\mathbf{X}_{ft}^{-1} = -\hat{\mathbf{X}}_{ft}\mathbf{\Lambda}_{ft}\mathbf{X}_{ft}^{-1} = \mathbf{I} \qquad (A.8)$$

$$\mathbf{\Lambda}_{ft} = -\hat{\mathbf{X}}_{ft}^{-1}\mathbf{X}_{ft}. \qquad (A.9)$$

Substituting (A.9) in (A.7) gives

$$\mathbf{R}_{ftkis} = z_{si}c_{ik}t_{fk}q_{kt}\mathbf{H}_{fs}\hat{\mathbf{X}}_{ft}^{-1}. \qquad (A.10)$$

The partial derivative of f^+ with respect to $\mathbf{U}_{ft}$ is given by

$$\begin{aligned}\frac{\partial f^+}{\partial \mathbf{U}_{ft}} &= \frac{\partial \text{logdet}(\mathbf{U}_{ft})}{\partial \mathbf{U}_{ft}} + \det(\hat{\mathbf{X}}_{ft})\frac{\partial \frac{1}{\det(\mathbf{U}_{ft})}}{\partial \mathbf{U}_{ft}} \\ &= \mathbf{U}_{ft}^{-1} + \frac{\det(\hat{\mathbf{X}}_{ft})}{\det(\mathbf{U}_{ft})}\mathbf{U}_{ft}^{-1},\end{aligned} \qquad (A.11)$$

where (A.32) and (A.33) are used in the second equation in the first and second term, respectively. Setting this partial derivative to zero gives

$$\frac{\partial f^+}{\partial \mathbf{U}_{ft}} = \mathbf{U}_{ft}^{-1} + \frac{\det(\hat{\mathbf{X}})}{\det(\mathbf{U}_{ft})}\mathbf{U}_{ft}^{-1} = \mathbf{0} \qquad (A.12)$$

$$\det(\mathbf{U}_{ft}) = \det(\hat{\mathbf{X}}). \tag{A.13}$$

Substituting (A.10) and (A.13) in (A.3) , indeed shows that $f\left(\mathbf{T},\mathbf{Q},\mathbf{H},\mathbf{Z},\mathbf{C}\right) = \min_{\mathbf{R},\mathbf{U}} f^{+}\left(\mathbf{T},\mathbf{Q},\mathbf{H},\mathbf{Z},\mathbf{C},\mathbf{R},\mathbf{U}\right)$.

A.1.2 Minimizing f^+ with respect to $\mathbf{T}$, $\mathbf{Q}$, $\mathbf{Z}$, $\mathbf{C}$ and $\mathbf{H}$

Minimizing f^+ with respect to t_{fk} gives

$$\begin{aligned}
\frac{\partial f^+}{\partial t_{fk}} &= \left[\sum_t\sum_i\sum_s \frac{-\mathrm{tr}\left(\mathbf{X}_{ft}\mathbf{R}^{\mathrm{H}}_{ftkis}\mathbf{H}_{fs}\mathbf{R}_{ftkis}\right)}{z_{si}c_{ik}t_{fk}^2 q_{kt}}\right] + \sum_t \frac{1}{\det(\mathbf{U}_{ft})}\frac{\partial \det(\hat{\mathbf{X}}_{ft})}{\partial t_{fk}} \\
&= \frac{-1}{t_{fk}^2}\left[\sum_t\sum_i\sum_s \frac{\mathrm{tr}\left(\mathbf{X}_{ft}\mathbf{R}^{\mathrm{H}}_{ftkis}\mathbf{H}_{fs}\mathbf{R}_{ftkis}\right)}{z_{si}c_{ik}q_{kt}}\right] + \\
&\qquad \sum_t \frac{\det(\hat{\mathbf{X}}_{ft})}{\det(\mathbf{U}_{ft})}\mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\sum_i\sum_s z_{si}c_{ik}q_{kt}\mathbf{H}_{fs}\right) \\
&= \frac{-1}{t_{fk}^2}\left[\sum_t\sum_i\sum_s \frac{\mathrm{tr}\left(\mathbf{X}_{ft}\mathbf{R}^{\mathrm{H}}_{ftkis}\mathbf{H}_{fs}\mathbf{R}_{ftkis}\right)}{z_{si}c_{ik}q_{kt}}\right] + \\
&\qquad \sum_t\sum_i\sum_s \frac{z_{si}c_{ik}q_{kt}\det(\hat{\mathbf{X}}_{ft})}{\det(\mathbf{U}_{ft})}\mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)
\end{aligned} \tag{A.14}$$

where (A.28) was used in the second equation in the second term. Setting this partial derivative to zero gives

$$\begin{aligned}
\frac{\partial f^+}{\partial t_{fk}} &= \frac{-1}{t_{fk}^2}\left[\sum_t\sum_i\sum_s \frac{\mathrm{tr}\left(\mathbf{X}_{ft}\mathbf{R}^{\mathrm{H}}_{ftkis}\mathbf{H}_{fs}^{-1}\mathbf{R}_{ftkis}\right)}{z_{si}c_{ik}q_{kt}}\right] + \\
&\qquad \sum_t\sum_i\sum_s \frac{z_{si}c_{ik}q_{kt}\det(\hat{\mathbf{X}}_{ft})}{\det(\mathbf{U}_{ft})}\mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right) = \mathbf{0}
\end{aligned} \tag{A.15}$$

$$t_{fk} = \sqrt{\frac{\sum_t\sum_i\sum_s \frac{\mathrm{tr}\left(\mathbf{X}_{ft}\mathbf{R}^{\mathrm{H}}_{ftkis}\mathbf{H}_{fs}^{-1}\mathbf{R}_{ftkis}\right)}{z_{si}c_{ik}q_{kt}}}{\sum_t\sum_i\sum_s \frac{z_{si}c_{ik}q_{kt}\det(\hat{\mathbf{X}}_{ft})}{\det(\mathbf{U}_{ft})}\mathrm{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)}}. \tag{A.16}$$

When substituting (A.10) and (A.13) in (A.16) this gives

$$t_{fk} = \sqrt{\frac{\sum_t \sum_i \sum_s \frac{\left(z_{si}c_{ik}t'_{fk}q_{kt}\right)^2 \operatorname{tr}\left(\mathbf{X}_{ft}\hat{\mathbf{X}}_{ft}^{-1^{\mathrm{H}}}\mathbf{H}_{fs}^{\mathrm{H}}\mathbf{H}_{fs}^{-1}\mathbf{H}_{fs}\hat{\mathbf{X}}_{ft}^{-1}\right)}{z_{si}c_{ik}q_{kt}}}{\sum_t \sum_i \sum_s z_{si}c_{ik}q_{kt}\operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)}}$$

$$= t'_{fk}\sqrt{\frac{\sum_t \sum_i \sum_s z_{si}c_{ik}q_{kt}\operatorname{tr}\left(\mathbf{X}_{ft}\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\hat{\mathbf{X}}_{ft}^{-1}\right)}{\sum_t \sum_i \sum_s z_{si}c_{ik}q_{kt}\operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)}} \qquad \text{(A.17)}$$

$$= t'_{fk}\sqrt{\frac{\sum_t \sum_i \sum_s z_{si}c_{ik}q_{kt}\operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{X}_{ft}\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)}{\sum_t \sum_i \sum_s z_{si}c_{ik}q_{kt}\operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)}}.$$

where the Hermitian symmetric properties of $\mathbf{X}_{ft}$, $\hat{\mathbf{X}}_{ft}$ and $\mathbf{U}_{ft}$ were used and t'_{fk} is used to denote t_{fk} before the update. With similar derivations the iterative update formulas for q_{kt}, z_{si} and c_{ik} can be found to be

$$q_{kt} = q'_{kt}\sqrt{\frac{\sum_f \sum_i \sum_s z_{si}c_{ik}t_{fk}\operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{X}_{ft}\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)}{\sum_f \sum_i \sum_s z_{si}c_{ik}t_{fk}\operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)}} \qquad \text{(A.18)}$$

$$z_{si} = z'_{si}\sqrt{\frac{\sum_f \sum_t \sum_k c_{ik}t_{fk}q_{kt}\operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{X}_{ft}\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)}{\sum_f \sum_t \sum_k c_{ik}t_{fk}q_{kt}\operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)}} \qquad \text{(A.19)}$$

$$c_{ik} = c'_{ik}\sqrt{\frac{\sum_f \sum_t \sum_s z_{si}t_{fk}q_{kt}\operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{X}_{ft}\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)}{\sum_f \sum_t \sum_s z_{si}t_{fk}q_{kt}\operatorname{tr}\left(\hat{\mathbf{X}}_{ft}^{-1}\mathbf{H}_{fs}\right)}}. \qquad \text{(A.20)}$$

Finally, the partial derivative of f^+ with respect to $\mathbf{H}_{fs}$ if found to be

$$\frac{\partial f^+}{\partial \mathbf{H}_{fs}} = \left[\sum_t\sum_k\sum_i \frac{-\mathbf{H}_{fs}^{-1}\mathbf{R}_{ftkis}\mathbf{X}_{ft}\mathbf{R}_{ftkis}^{\mathrm{H}}\mathbf{H}_{fs}^{-1}}{z_{si}c_{ik}t_{fk}q_{kt}}\right] + \sum_t \frac{1}{\det(\mathbf{U}_{ft})}\frac{\partial \det(\hat{\mathbf{X}}_{ft})}{\partial \mathbf{H}_{fs}}$$

$$= -\mathbf{H}_{fs}^{-1}\left[\sum_t\sum_k\sum_i \frac{\mathbf{R}_{ftkis}\mathbf{X}_{ft}\mathbf{R}_{ftkis}^{\mathrm{H}}}{z_{si}c_{ik}t_{fk}q_{kt}}\right]\mathbf{H}_{fs}^{-1}$$

$$+\sum_t\sum_k\sum_i z_{si}c_{ik}t_{fk}q_{kt}\frac{\det(\hat{\mathbf{X}}_{ft})}{\det(\mathbf{U}_{ft})}\hat{\mathbf{X}}_{ft}^{-1}, \qquad \text{(A.21)}$$

where (A.36) was used in the first equality and (A.34) in the second equality. Setting this partial derivative to zero gives

$$\frac{\partial f^+}{\partial \mathbf{H}_{fs}} = -\mathbf{H}_{fs}^{-1}\left[\sum_t\sum_k\sum_i \frac{\mathbf{R}_{ftkis}\mathbf{X}_{ft}\mathbf{R}_{ftkis}^{\mathrm{H}}}{z_{si}c_{ik}t_{fk}q_{kt}}\right]\mathbf{H}_{fs}^{-1} \tag{A.22}$$

$$+\sum_t\sum_k\sum_i z_{si}c_{ik}t_{fk}q_{kt}\frac{\det(\hat{\mathbf{X}}_{ft})}{\det(\mathbf{U}_{ft})}\hat{\mathbf{X}}_{ft}^{-1} = \mathbf{0}$$

$$\sum_t\sum_k\sum_i z_{si}c_{ik}t_{fk}q_{kt}\frac{\det(\hat{\mathbf{X}}_{ft})}{\det(\mathbf{U}_{ft})}\hat{\mathbf{X}}_{ft}^{-1} = \tag{A.23}$$

$$\mathbf{H}_{fs}^{-1}\left[\sum_t\sum_k\sum_i \frac{\mathbf{R}_{ftkis}\mathbf{X}_{ft}\mathbf{R}_{ftkis}^{\mathrm{H}}}{z_{si}c_{ik}t_{fk}q_{kt}}\right]\mathbf{H}_{fs}^{-1}.$$

$$\mathbf{H}_{fs}\left[\sum_t\sum_k\sum_i z_{si}c_{ik}t_{fk}q_{kt}\frac{\det(\hat{\mathbf{X}}_{ft})}{\det(\mathbf{U}_{ft})}\hat{\mathbf{X}}_{ft}^{-1}\right]\mathbf{H}_{fs} = \tag{A.24}$$

$$\sum_t\sum_k\sum_i \frac{\mathbf{R}_{ftkis}\mathbf{X}_{ft}\mathbf{R}_{ftkis}^{\mathrm{H}}}{z_{si}c_{ik}t_{fk}q_{kt}}.$$

Substituting (A.10) and (A.13) in (A.24) gives

$$\mathbf{H}_{fs}\left[\sum_t\sum_k\sum_i z_{si}c_{ik}t_{fk}q_{kt}\hat{\mathbf{X}}_{ft}^{-1}\right]\mathbf{H}_{fs} = \tag{A.25}$$

$$\sum_t\sum_k\sum_i \frac{z_{si}c_{ik}t_{fk}q_{kt}\mathbf{H}'_{fs}\hat{\mathbf{X}}_{ft}^{-1}\mathbf{X}_{ft}z_{si}c_{ik}t_{fk}q_{kt}\hat{\mathbf{X}}_{ft}^{-1^{\mathrm{H}}}\mathbf{H}'^{\mathrm{H}}_{fs}}{z_{si}c_{ik}t_{fk}q_{kt}}.$$

$$\mathbf{H}_{fs}\left[\sum_t\sum_k\sum_i z_{si}c_{ik}t_{fk}q_{kt}\hat{\mathbf{X}}_{ft}^{-1}\right]\mathbf{H}_{fs} = \tag{A.26}$$

$$\mathbf{H}'_{fs}\left[\sum_t\sum_k\sum_i z_{si}c_{ik}t_{fk}q_{kt}\hat{\mathbf{X}}_{ft}^{-1}\mathbf{X}_{ft}\hat{\mathbf{X}}_{ft}^{-1}\right]\mathbf{H}'_{fs},$$

where $\mathbf{H}'_{fs}$ is used to indicate $\mathbf{H}_{fs}$ before the update.

A.2 Matrix derivatives

This section will give some matrix derivatives that were used in Appendix A.1.

According to [137, eq. (46)]

$$\frac{\partial \det(\mathbf{Y}(x_{ij}))}{\partial x_{ij}} = \det(\mathbf{Y})\mathrm{tr}\left(\mathbf{Y}^{-1}\frac{\partial \mathbf{Y}}{\partial x_{ij}}\right). \tag{A.27}$$

For $\mathbf{Y} = x_{ij}\mathbf{A}$, with $\mathbf{A}$ a constant matrix, this gives

$$\frac{\partial \det(\mathbf{A}x_{ij})}{\partial x_{ij}} = \det(\mathbf{A})\mathrm{tr}\left(\mathbf{A}^{-1}\right). \tag{A.28}$$

For $\mathbf{Y} = \mathbf{X}$, (A.27) gives

$$\frac{\partial \det(\mathbf{X})}{\partial x_{ij}} = \det(\mathbf{X})\mathrm{tr}\left(\mathbf{X}^{-1}\mathbf{P}_{ij}\right), \tag{A.29}$$

with $\mathbf{P}_{ij}$ The single-entry matrix, 1 at (i,j) and 0 elsewhere. Since the matrix product is defined as $[\mathbf{AB}]_{kl} = \sum_n a_{kn}b_{nl}$ this gives $\mathrm{tr}\left(\mathbf{AB}\right) = \sum_j\sum_n a_{jn}b_{nj}$ and $\mathrm{tr}\left(\mathbf{AP}_{ij}\right) = a_{ji}$. Using this last equation in (A.29) gives

$$\frac{\partial \det(\mathbf{X})}{\partial x_{ij}} = \det(\mathbf{X})\left[\mathbf{X}^{-1}\right]_{ji} = \det(\mathbf{X})\left[\mathbf{X}^{-1}\right]_{ij}, \tag{A.30}$$

where the last equation only holds if $\mathbf{X}$ is symmetric. Then we get

$$\frac{\partial \det(\mathbf{X})}{\partial \mathbf{X}} = \det(\mathbf{X})\mathbf{X}^{-1}. \tag{A.31}$$

This allows to find the following partial derivatives when using the chain rule

$$\frac{\partial \mathrm{logdet}(\mathbf{X})}{\mathbf{X}} = \frac{1}{\det(\mathbf{X})}\det(\mathbf{X})\mathbf{X}^{-1} = \mathbf{X}^{-1} \tag{A.32}$$

$$\frac{\partial \frac{1}{\det(\mathbf{X})}}{\mathbf{X}} = \frac{1}{\det(\mathbf{X})^2}\det(\mathbf{X})\mathbf{X}^{-1} = \frac{\mathbf{X}^{-1}}{\det(\mathbf{X})}. \tag{A.33}$$

Using the same deviation as (A.29)-(A.31) would show that

$$\frac{\partial \det(a\mathbf{X}+\mathbf{B})}{\partial \mathbf{X}} = a\det(a\mathbf{X}+\mathbf{B})(\mathbf{aX}+\mathbf{B})^{-1}, \tag{A.34}$$

with a a constant scalar and $\mathbf{B}$ a constant matrix.

Finally, it can be shown that

$$\frac{\partial \mathrm{tr}\left(\mathbf{AX}^{\mathrm{H}}\mathbf{BX}\right)}{\partial \mathbf{X}} = \mathbf{B}^{\mathrm{H}}\mathbf{XA}^{\mathrm{H}} + \mathbf{BX}^{\mathrm{H}}\mathbf{A} = 2\mathbf{BXA}, \tag{A.35}$$

where the first equality is found in [137, eq. (117)] and the second equality only holds if $\mathbf{X}$,$\mathbf{A}$ and $\mathbf{B}$ are Hermitian symmetric. Furthermore in [137, eq. (124)] it is stated that

$$\frac{\partial \mathrm{tr}\left(\mathbf{AX}^{-1}\mathbf{B}\right)}{\partial \mathbf{X}} = -\mathbf{X}^{-1^{\mathrm{H}}}\mathbf{A}^{\mathrm{H}}\mathbf{B}^{\mathrm{H}}\mathbf{X}^{-1^{\mathrm{H}}} \tag{A.36}$$

Appendix B

Layer connections for reset LSTM-RNN

This appendix will derive the layer connections for reset LSTM-RNNs. Appendix B.1 will give the general derivations and Appendix B.2 will show the derivations when using the grouping approach.

B.1 Inter-layer connections

In this section the general inter-layer connections rules for reset LSTM-RNNs will be derived. First, this will be done for unidirectional LSTM-RNNs in Appendix B.1.1. Then this will be extended to bidirectional LSTM-RNNs in Appendix B.1.2. Finally, the connection rules for layer dependent resets will be derived in Appendix B.1.3.

B.1.1 Unidirectional reset LSTM-RNN

Given (5.8), instance k of layer l will be reset at times $t_{k,l}$ given by

$$t_{k,l} = k + \alpha K, \tag{B.1}$$

with α a natural number. Therefore, the number of time steps $\tau_t^{k,l}$ between time t and the last time instance k was reset before time t is given by

$$\tau_t^{k,l} = (t - t_{k,l}) \bmod K = (t - k - \alpha K) \bmod K = (t - k) \bmod K. \tag{B.2}$$

A time t instance k of layer l contains $\tau_t^{k,l}$ frames of context. We would like this instance to receive input from an instance of the layer below with the same number of context frames $\tau_t^{k,l}$. In other words, we would like to find the instance that was reset at time $t - \tau_t^{k,l}$ in layer $l-1$. Using (5.8) we find this to be

$$k^{*,l-1}_{t-\tau_t^{k,l}} = (t-\tau_t^{k,l}) \bmod K = (t-(t-k) \bmod K) \bmod K = k \bmod K = k. \quad \text{(B.3)}$$

This simply means that instance k from layer l should receive input from instance k from layer $l-1$. Or in simplified notation $k \leftarrow k$. Finally, the last layer of the LSTM-RNN should output a single instance which will be the final output of the network. We choose this to be the instance with the maximum number of context frames $\tau_t^{max,L} = K-1$. This means that the instance was reset at $t - \tau_t^{max,L} = t - K + 1$. Thus the instance to select is

$$k^{*,L}_{t-K+1} = (t-K+1) \bmod K = (t+1) \bmod K = k^{*,L}_{t+1}. \quad \text{(B.4)}$$

B.1.2 Bidirectional reset LSTM-RNN

For bidirectional LSTM-RNNs, we take a similar approach. Equivalent to (5.8), (B.1) and (B.2), we define

$$\overrightarrow{k}^{*,l}_t = t \bmod K_l, \quad \text{(B.5)}$$

$$t_{\overrightarrow{k},l} = \overrightarrow{k} + \alpha K_l, \quad \text{(B.6)}$$

$$\tau_t^{\overrightarrow{k},l} = (t - \overrightarrow{k}) \bmod K_l. \quad \text{(B.7)}$$

For the backward direction we find

$$\overleftarrow{k}^{*,l}_t = ((T-1)-t) \bmod K_l, \quad \text{(B.8)}$$

$$t_{\overleftarrow{k},l} = (T-1) - (\overleftarrow{k} + \alpha K_l), \quad \text{(B.9)}$$

$$\begin{aligned}\tau_t^{\overleftarrow{k},l} &= (t_{\overrightarrow{k},l} - t) \bmod K_l = ((T-1) - t - \overleftarrow{k} - \alpha K_l) \bmod K_l \\ &= ((T-1) - t - k) \bmod K_l.\end{aligned} \quad \text{(B.10)}$$

Again, we want an instance to receive input from an instance in the layer below with the same number of context frames. For the instances in the forward direction these are $\overrightarrow{k}^{*,l-1}_{t-\tau_t^{\overrightarrow{k},l}}$ and $\overleftarrow{k}^{*,l-1}_{t+\tau_t^{\overrightarrow{k},l}}$. When we use the same reset period for all layers ($K = K_l = K_{l-1}$), these are

$$\begin{aligned}\overrightarrow{k}^{*,l-1}_{t-\tau_t^{\overrightarrow{k},l}} &= (t - \tau_t^{\overrightarrow{k},l}) \bmod K = (t - (t - \overrightarrow{k}) \bmod K) \bmod K \\ &= \overrightarrow{k} \bmod K = \overrightarrow{k}.\end{aligned} \tag{B.11}$$

and

$$\begin{aligned}\overleftarrow{k}^{*,l-1}_{t+\tau_t^{\overrightarrow{k},l}} &= ((T-1) - (t + \tau_t^{\overrightarrow{k},l})) \bmod K \\ &= ((T-1) - (t + t - \overrightarrow{k}) \bmod K) \bmod K \\ &= ((T-1) - 2t + \overrightarrow{k}) \bmod K,\end{aligned} \tag{B.12}$$

respectively. As per (1.65), in simplified notation this becomes

$$\overrightarrow{k} \leftarrow \begin{bmatrix} \overrightarrow{k} \\ ((T-1) - 2t + \overleftarrow{k}) \bmod K \end{bmatrix}. \tag{B.13}$$

Similarly for the backward direction we find the inputs to be

$$\overleftarrow{k} \leftarrow \begin{bmatrix} (-(T-1) + 2t + \overrightarrow{k}) \bmod K \\ \overleftarrow{k} \end{bmatrix}. \tag{B.14}$$

As per (1.66), the final output of the network is a concatenation of the output of both directions of the last layer. We again choose these to be the instance with the maximum number of context frames $\tau_t^{\overrightarrow{max},L} = K-1$ and $\tau_t^{\overleftarrow{max},L} = K-1$ for the forward and backward direction, respectively. This means that corresponding instances were reset at $t - \tau_t^{\overrightarrow{max},L} = t - K + 1$ and $t + \tau_t^{\overleftarrow{max},L} = t + K - 1$. Thus the corresponding instances to select are

$$\overrightarrow{k}^{*,L}_{t-K+1} = (t - K + 1) \bmod K = (t+1) \bmod K = \overrightarrow{k}^{*,L}_{t+1} \tag{B.15}$$

and

$$\overleftarrow{k}^{*,L}_{t+K+1} = ((T-1) - (t+K-1)) \bmod K = ((T-1) - (t-1)) \bmod K = \overleftarrow{k}^{*,L}_{t-1}. \tag{B.16}$$

Thus (1.66) generalizes to

$$\mathbf{h}_t = \begin{bmatrix} \overrightarrow{\mathbf{h}}_t^{\overrightarrow{k}^{*,L}_{t+1},L} \\ \overleftarrow{\mathbf{h}}_t^{\overleftarrow{k}^{*,L}_{t-1},L} \end{bmatrix}. \tag{B.17}$$

B.1.3 Layer dependent reset period

If K_l and K_{l-1} are different, we would still like an instance to receive input from the layer below with the same number of context frames. However, this is not possible when the number of context frames exceeds the maximum number of context frames in the layer below (bounded by $K_{l-1} - 1$). therefore, a new variable $\bar{\tau}$ is introduced, which is defined as

$$\bar{\tau}_t^{\overrightarrow{k},l} = min(\tau_t^{\overrightarrow{k},l}, K_{l-1} - 1). \tag{B.18}$$

When replacing $\tau_t^{\overleftarrow{k},l}$ with $\bar{\tau}_t^{\overleftarrow{k},l}$ in (B.11) and (B.12), we get

$$\overrightarrow{k} \leftarrow \begin{bmatrix} (t - \bar{\tau}_t^{\overrightarrow{k},l}) \bmod K_{l-1} \\ ((T-1) - t - \bar{\tau}_t^{\overrightarrow{k},l}) \bmod K_{l-1} \end{bmatrix}. \tag{B.19}$$

Similarly, for the backward direction we define

$$\bar{\tau}_t^{\overleftarrow{k},l} = min(\tau_t^{\overleftarrow{k},l}, K_{l-1} - 1), \tag{B.20}$$

$$\overleftarrow{k} \leftarrow \begin{bmatrix} (t - \bar{\tau}_t^{\overleftarrow{k},l}) \bmod K_{l-1} \\ ((T-1) - t - \bar{\tau}_t^{\overleftarrow{k},l}) \bmod K_{l-1} \end{bmatrix}. \tag{B.21}$$

With the constraint $K_l \geqslant K_{l-1}$ (otherwise, layer l would be allowed less context than layer $l-1$. Instances with $\tau_t^{k,l-1} > K_l - 1$ would not be connected to the higher layer and effectively K_{l-1} would be set to K_l).

B.2 Grouped inter-layer connections

Using (5.21), instance k of layer l will be reset at times $t_{k,l}$ given by

$$t_{k,l} = (k + \alpha K_l)G_l. \tag{B.22}$$

Therefore, the number of time steps $\tau_t^{k,l}$ between time t and the last time instance k was reset before time t is given by

$$\begin{aligned} \tau_t^{k,l} &= (t - t_{k,l}) \bmod T_{\text{reset}}^l = (t - kG_l - \alpha K_l G_l) \bmod K_l G_l \\ &= (t - kG_l) \bmod K_l G_l. \end{aligned} \tag{B.23}$$

As before, we would like an instance to receive input from an instance in the layer below, with the same number of context frames. However, this cannot be guaranteed as $\tau_t^{k,l}$ increases with steps of G. For the forward input we therefore select the first instance in layer $l-1$ to be reset at $t-\tau_t^{k,l}$ or afterwards. (B.22) shows that resets happen at multiples of G_{l-1}. Therefore the requested reset in the forward direction will happen at time

$$\overrightarrow{\gamma}_t^{\overrightarrow{k},l} = \left\lceil \frac{t-\bar{\tau}_t^{\overrightarrow{k},l}}{G_{l-1}} \right\rceil G_{l-1}, \tag{B.24}$$

where $\bar{\tau}$ is defined as

$$\bar{\tau}_t^{\overrightarrow{k},l} = min(\tau_t^{\overrightarrow{k},l}, K_{l-1}G_{l-1}-1). \tag{B.25}$$

Using (5.21), we find that the instance in the forward direction in layer $l-1$ that will be reset at time $\overrightarrow{\gamma}_t^{\overrightarrow{k},l}$ is given by

$$\begin{aligned} \overrightarrow{k}^{*,l-1}_{\overrightarrow{\gamma}_t^{\overrightarrow{k},l}} &= \frac{\overrightarrow{\gamma}_t^{\overrightarrow{k},l}}{G_{l-1}} \text{ mod } K_{l-1} = \left(\left\lceil \frac{t-\bar{\tau}_t^{\overrightarrow{k},l}}{G_{l-1}} \right\rceil \frac{G_{l-1}}{G_{l-1}} \right) \text{ mod } K_{l-1} \\ &= \left\lceil \frac{t-\bar{\tau}_t^{\overrightarrow{k},l}}{G_{l-1}} \right\rceil \text{ mod } K_{l-1}. \end{aligned} \tag{B.26}$$

In the backward direction (5.21) is changed to

$$\overleftarrow{k}_t^{*,l} = \left(\frac{(T-1)-t}{G_l} \right) \text{ mod } K_l \text{ if } (T-1)-t \equiv 0 \pmod{G_l}. \tag{B.27}$$

The requested reset in the backward direction will happen at time

$$\overleftarrow{\gamma}_t^{\overrightarrow{k},l} = (T-1) - \left(\left\lceil \frac{(T-1)-(t+\bar{\tau}_t^{\overrightarrow{k},l})}{G_{l-1}} \right\rceil G_{l-1} \right). \tag{B.28}$$

Combining (B.27) and (B.28) gives the instance in the backward direction in layer $l-1$ that will be reset at time $\overleftarrow{\gamma}_t^{\overrightarrow{k},l}$.

$$\overleftarrow{k}^{*,l-1}_{\overleftarrow{\gamma}_t^{\overrightarrow{k},l}} = \left(\frac{(T-1) - \overleftarrow{\gamma}_t^{\overrightarrow{k},l}}{G_{l-1}}\right) \bmod K_{l-1}$$
$$= \left(\frac{(T-1) - \left((T-1) - \left(\left\lceil \frac{(T-1)-(t+\bar{\tau}_t^{\overrightarrow{k},l})}{G_{l-1}} \right\rceil G_{l-1}\right)\right)}{G_{l-1}}\right) \bmod K_{l-1}$$
$$= \left\lceil \frac{(T-1)-(t+\bar{\tau}_t^{\overrightarrow{k},l})}{G_{l-1}} \right\rceil \bmod K_{l-1}. \tag{B.29}$$

In shorthand notation this becomes

$$\overrightarrow{k} \leftarrow \begin{bmatrix} \left\lceil \frac{t-\bar{\tau}_t^{\overrightarrow{k},l}}{G_{l-1}} \right\rceil \bmod K_{l-1} \\ \left\lceil \frac{(T-1)-t-\bar{\tau}_t^{\overrightarrow{k},l}}{G_{l-1}} \right\rceil \bmod K_{l-1} \end{bmatrix}. \tag{B.30}$$

Similarly, for the backward direction we find the inputs to be

$$\overleftarrow{k} \leftarrow \begin{bmatrix} \left\lceil \frac{t-\bar{\tau}_t^{\overleftarrow{k},l}}{G_{l-1}} \right\rceil \bmod K_{l-1} \\ \left\lceil \frac{(T-1)-t-\bar{\tau}_t^{\overleftarrow{k},l}}{G_{l-1}} \right\rceil \bmod K_{l-1} \end{bmatrix}. \tag{B.31}$$

Finally, we would like to find the final output of the network or a generalization of (B.17). Ideally, we would like to select the instances with $T_{reset,L}-1 = K_L G_L - 1$ number of context frames. Again this cannot be guaranteed as $\tau_t^{\overrightarrow{k},L}$ and $\tau_t^{\overleftarrow{k},L}$ increase with steps of G_L. Instead we will be looking for time $\overrightarrow{\gamma}_t^{\overrightarrow{max},L}$ and $\overleftarrow{\gamma}_t^{\overleftarrow{max},L}$ defined as

$$\overrightarrow{\gamma}_t^{\overrightarrow{max},L} = \left\lceil \frac{t-(K_L G_L - 1)}{G_L} \right\rceil G_L \tag{B.32}$$

and

$$\overleftarrow{\gamma}_t^{\overleftarrow{max},L} = (T-1) - \left(\left\lceil \frac{(T-1)-(t+(K_L G_L - 1))}{G_L} \right\rceil G_L\right). \tag{B.33}$$

The corresponding instances are thus

$$\begin{aligned}\overrightarrow{k}^{*,L}_{\overrightarrow{\gamma}_t^{\overrightarrow{max},L}} &= \frac{\overrightarrow{\gamma}_t^{\overrightarrow{max},L}}{G_L} \bmod K_L = \left(\left\lceil \frac{t-(K_L G_L - 1)}{G_L} \right\rceil \frac{G_L}{G_L}\right) \bmod K_L \\ &= \left\lceil \frac{t-(K_L G_L - 1)}{G_L} \right\rceil \bmod K_L.\end{aligned} \tag{B.34}$$

and

$$\begin{aligned}\overleftarrow{k}^{*,L}_{\overleftarrow{\gamma}_t^{\overleftarrow{max},L}} &= \left(\frac{(T-1) - \overleftarrow{\gamma}_t^{\overleftarrow{max},L}}{G_L}\right) \bmod K_L \\ &= \left(\frac{(T-1) - \left((T-1) - \left(\left\lceil \frac{(T-1)-(t+(K_L G_L-1))}{G_L}\right\rceil G_L\right)\right)}{G_L}\right) \bmod K_L \\ &= \left\lceil \frac{(T-1)-(t+(K_L G_L - 1))}{G_L} \right\rceil \bmod K_L.\end{aligned} \tag{B.35}$$

Bibliography

[1] Csr-ii (wsj1) complete ldc94s13a. `https://catalog.ldc.upenn.edu/LDC94S13A`, 1994.

[2] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[3] Adiloğlu, K., and Vincent, E. Variational bayesian inference for source separation and robust feature extraction. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 24*, 10 (2016), 1746–1758.

[4] Akram, S., Presacco, A., Simon, J. Z., Shamma, S. A., and Babadi, B. Robust decoding of selective auditory attention from MEG in a competing-speaker environment via state-space modeling. *NeuroImage 124* (2016), 906–917.

[5] Alpay, T., Heinrich, S., and Wermter, S. Learning multiple timescales in recurrent neural networks. In *International Conference on Artificial Neural Networks* (2016), Springer, pp. 132–139.

[6] Appeltans, P., Zegers, J., and Van hamme, H. Practical applicability of deep neural networks for overlapping speaker separation. In *Interspeech 2019* (2019), ISCA, pp. 1353–1357.

[7] Aroudi, A., and Doclo, S. Cognitive-driven binaural beamforming using EEG-based auditory attention decoding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2020).

[8] Auckenthaler, R., Carey, M., and Lloyd-Thomas, H. Score normalization for text-independent speaker verification systems. *Digital Signal Processing 10*, 1-3 (2000), 42–54.

[9] Baby, D. *Non-negative sparse representations for speech enhancement and recognition.* PhD thesis, Ph. D. dissertation, University of Leuven, 2016.

[10] Bahari, M. H., and Van hamme, H. Speaker age estimation and gender detection based on supervised non-negative matrix factorization. In *In proc. IEEE Workshop on Biometric Measurements and Systems for Security and Medical Applications (BIOMS)* (2011).

[11] Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P., and Bengio, Y. End-to-end attention-based large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), IEEE, pp. 4945–4949.

[12] Bahmaninezhad, F., Wu, J., Gu, R., Zhang, S.-X., Xu, Y., Yu, M., and Yu, D. A comprehensive study of speech separation: Spectrogram vs waveform separation. *Interspeech 2019* (2019), 4574–4578.

[13] Barker, J., Marxer, R., Vincent, E., and Watanabe, S. The third CHiME speech separation and recognition challenge: Dataset, task and baselines. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (Dec 2015), pp. 504–511.

[14] Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning* (2009), ACM, pp. 41–48.

[15] Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks 5*, 2 (1994), 157–166.

[16] Bertrand, A., and Moonen, M. Blind separation of non-negative source signals using multiplicative updates and subspace projection. *Signal Processing 90*, 10 (2010), 2877–2890.

[17] Bertrand, A., and Moonen, M. Energy-based multi-speaker voice activity detection with an ad hoc microphone array. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing* (Dallas, Texas, USA, 2010), IEEE, pp. 85–88.

[18] Biesmans, W., Das, N., Francart, T., and Bertrand, A. Auditory-inspired speech envelope extraction methods for improved EEG-based auditory attention detection in a cocktail party scenario. *IEEE Transactions on Neural Systems and Rehabilitation Engineering 25*, 5 (2017), 402–412.

[19] Boll, S. Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on acoustics, speech, and signal processing 27*, 2 (1979), 113–120.

[20] Bregman, A. S. *Auditory scene analysis: The perceptual organization of sound.* MIT press, 1994.

[21] Campbell, D., Palomäki, K., and Brown, G. A MATLAB simulation of "shoebox" room acoustics for use in research and teaching. *Computing and Information Systems 9*, 3 (2005), 48.

[22] Caruana, R. A dozen tricks with multitask learning. In *Neural networks: tricks of the trade.* Springer, 1998, pp. 165–191.

[23] Caruana, R. Multitask learning. *Learning to learn* (1998), 95–133.

[24] Cemgil, A. T. Bayesian inference for nonnegative matrix factorisation models. *Computational intelligence and neuroscience 2009* (2008).

[25] Chakravarty, P., Zegers, J., Tuytelaars, T., and Van hamme, H. Active speaker detection with audio-visual co-training. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction* (New York, NY, USA, 2016), ICMI '16, Association for Computing Machinery, p. 312–316.

[26] Chelba, C., Norouzi, M., and Bengio, S. N-gram language modeling using recurrent neural network estimation. Tech. rep., Google, 2017.

[27] Chen, N., Qian, Y., and Yu, K. Multi-task learning for text-dependent speaker verification. In *Sixteenth annual conference of the international speech communication association* (2015).

[28] Chen, W., Chen, X., Zhang, J., and Huang, K. A multi-task deep network for person re-identification. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[29] Chen, Z., Li, J., Xiao, X., Yoshioka, T., Wang, H., Wang, Z., and Gong, Y. Cracking the cocktail party problem by multi-beam deep attractor network. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)* (Okinawa, Japan, 2017), IEEE, pp. 437–444.

[30] Chen, Z., Luo, Y., and Mesgarani, N. Deep attractor network for single-microphone speaker separation. In *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on* (2017), IEEE, pp. 246–250.

[31] Christensen, H., Barker, J., Ma, N., and Green, P. The CHiME corpus: a resource and a challenge for computational hearing in multisource environments. In *INTERSPEECH* (2010), pp. 1918–1912.

[32] Christlein, V., Bernecker, D., Hönig, F., Maier, A., and Angelopoulou, E. Writer identification using gmm supervectors and exemplar-svms. *Pattern Recognition 63* (2017), 258–267.

[33] Chung, J., Ahn, S., and Bengio, Y. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704* (2016).

[34] Ciccarelli, G., Nolan, M., Perricone, J., Calamia, P. T., Haro, S., O'Sullivan, J., Mesgarani, N., Quatieri, T. F., and Smalt, C. J. Comparison of two-talker attention decoding from EEG with nonlinear neural networks and linear methods. *Scientific reports 9*, 1 (2019), 1–10.

[35] Cichocki, A., Zdunek, R., Phan, A. H., and Amari, S. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation.* John Wiley & Sons, 2009.

[36] Colah, C. Understanding LSTM Networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015. [Online; accessed 3-Juli-2019].

[37] Cornelis, B., Moonen, M., and Wouters, J. Reduced-bandwidth multi-channel Wiener filter based binaural noise reduction and localization cue preservation in binaural hearing aids. *Signal Processing 99* (2014), 1–16.

[38] Crochiere, R. A weighted overlap-add method of short-time Fourier analysis/synthesis. *IEEE Transactions on Acoustics, Speech, and Signal Processing 28*, 1 (1980), 99–102.

[39] Dai, J., He, K., and Sun, J. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 3150–3158.

[40] Das, N. Eeg-based auditory attention decoding: Towards neuro-steered hearing devices.

[41] Das, N., Bertrand, A., and Francart, T. EEG-based auditory attention detection: boundary conditions for background noise and speaker positions. *Journal of neural engineering 15*, 6 (2018), 066017.

[42] Das, N., Biesmans, W., Bertrand, A., and Francart, T. The effect of head-related filtering and ear-specific decoding bias on auditory attention detection. *Journal of neural engineering 13*, 5 (2016), 056014.

[43] Das, N., Van Eyndhoven, S., Francart, T., and Bertrand, A. EEG-based attention-driven speech enhancement for noisy speech mixtures using N-fold multi-channel Wiener filters. In *25th European Signal Processing Conference (EUSIPCO)* (Kos, Greece, 2017), pp. 1660–1664.

[44] Das, N., Zegers, J., Van hamme, H., Francart, T., and Bertrand, A. Multi-microphone speaker separation for neuro-steered hearing aids: neural networks versus linear methods. In *Auditory EEG Signal Processing symposium (AESoP)* (September 2019).

[45] Das, N., Zegers, J., Van hamme, H., Francart, T., and Bertrand, A. Linear versus deep learning methods for noisy speech separation for EEG-informed attention decoding. *Journal of Neural Engineering 17*, 4 (August 2020).

[46] de Cheveigné, A., Wong, D. D., Liberto, G. M. D., Hjortkjaer, J., Slaney, M., and Lalor, E. Decoding the auditory brain with canonical component analysis. *NeuroImage 172* (2018), 206 – 216.

[47] De Leeuw, J. Block-relaxation algorithms in statistics. In *Information systems and data analysis*. Springer, 1994, pp. 308–324.

[48] de Taillez, T., Kollmeier, B., and Meyer, B. T. Machine learning for decoding listeners' attention from electroencephalography evoked by continuous speech. *European Journal of Neuroscience* (2017).

[49] Deckers, L., Das, N., Ansari, A. H., Bertrand, A., and Francart, T. EEG-based detection of the attended speaker and the locus of auditory attention with convolutional neural networks. *bioRxiv* (2018), 475673.

[50] Decruy, L., Vanthornhout, J., and Francart, T. Hearing impairment is associated with enhanced neural tracking of the speech envelope. *bioRxiv* (2019).

[51] Dehak, N., Kenny, P. J., Dehak, R., Dumouchel, P., and Ouellet, P. Front-end factor analysis for speaker verification. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 19*, 4 (2011), 788–798.

[52] Delcroix, M., Zmolikova, K., Kinoshita, K., Ogawa, A., and Nakatani, T. Single channel target speaker extraction and recognition with speaker beam. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 5554–5558.

[53] Di Liberto, G. M., O'Sullivan, J. A., and Lalor, E. C. Low-frequency cortical entrainment to speech reflects phoneme-level processing. *Current Biology 25*, 19 (2015), 2457–2465.

[54] Ding, N., and Simon, J. Z. Emergence of neural encoding of auditory objects while listening to competing speakers. *Proc. National Academy of Sciences 109*, 29 (2012), 11854–11859.

[55] Ding, N., and Simon, J. Z. Neural coding of continuous speech in auditory cortex during monaural and dichotic listening. *Journal of neurophysiology 107*, 1 (2012), 78–89.

[56] Doclo, S., and Moonen, M. GSVD-based optimal filtering for single and multimicrophone speech enhancement. *IEEE Transactions on Signal Processing 50*, 9 (Sept. 2002), 2230 – 2244.

[57] Drgas, S., and Virtanen, T. Speaker verification using adaptive dictionaries in non-negative spectrogram deconvolution. In *Latent Variable Analysis and Signal Separation*. Springer, 2015, pp. 462–469.

[58] Drude, L., von Neumann, T., and Haeb-Umbach, R. Deep attractor networks for speaker re-identification and blind source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 11–15.

[59] Duong, N. Q., Vincent, E., and Gribonval, R. Under-determined reverberant audio source separation using a full-rank spatial covariance model. *IEEE Transactions on Audio, Speech, and Language Processing 18*, 7 (2010), 1830–1840.

[60] El Hihi, S., and Bengio, Y. Hierarchical recurrent neural networks for long-term dependencies. In *Advances in neural information processing systems* (1996), pp. 493–499.

[61] Ferrer, L., Lei, Y., McLaren, M., and Scheffer, N. Study of senone-based deep neural network approaches for spoken language recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 24*, 1 (2015), 105–116.

[62] Fiedler, L., Woestmann, M., Graversen, C., Brandmeyer, A., Lunner, T., and Obleser, J. Single-channel in-ear-EEG detects the

focus of auditory attention to concurrent tone streams and mixed speech. *Journal of Neural Engineering 14*, 3 (2017), 036020.

[63] GAO, Y., AND CHURCH, G. Improving molecular cancer class discovery through sparse non-negative matrix factorization. *Bioinformatics 21*, 21 (2005), 3970–3975.

[64] GAROFOLO, J. S., GRAFF, D., PAUL, D., AND PALLETT, D. Csr-i (wsj0) complete ldc93s6a. `https://catalog.ldc.upenn.edu/LDC93S6A`, 1993.

[65] GAY, T. Effect of speaking rate on diphthong formant movements. *The Journal of the Acoustical Society of America 44*, 6 (1968), 1570–1573.

[66] GEMMEKE, J. F., VIRTANEN, T., AND HURMALAINEN, A. Exemplar-based sparse representations for noise robust automatic speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing 19*, 7 (2011), 2067–2080.

[67] GERKMANN, T., AND HENDRIKS, R. C. Noise power estimation based on the probability of speech presence. In *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (New Paltz, NY, USA, 2011), IEEE, pp. 145–148.

[68] GLEMBEK, O., BURGET, L., MATĚJKA, P., KARAFIÁT, M., AND KENNY, P. Simplification and optimization of i-vector extraction. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2011), pp. 4516–4519.

[69] GOLUMBIC, E. M. Z., DING, N., BICKEL, S., LAKATOS, P., SCHEVON, C. A., MCKHANN, G. M., GOODMAN, R. R., EMERSON, R., MEHTA, A. D., SIMON, J. Z., ET AL. Mechanisms underlying selective neuronal tracking of attended speech at a "cocktail party". *Neuron 77*, 5 (2013), 980–991.

[70] GOODFELLOW, I., BENGIO, Y., AND COURVILLE, A. *Deep learning.* MIT press, 2016.

[71] GRAVES, A., AND SCHMIDHUBER, J. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks 18*, 5-6 (2005), 602–610.

[72] GUPTA, V., KENNY, P., OUELLET, P., AND STAFYLAKIS, T. I-vector-based speaker adaptation of deep neural networks for french broadcast audio transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2014), pp. 6334–6338.

[73] Han, C., Luo, Y., and Mesgarani, N. Online deep attractor network for real-time single-channel speech separation. In *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (Brighton, UK, 2019), IEEE, pp. 361–365.

[74] Han, C., O'Sullivan, J., Luo, Y., Herrero, J., Mehta, A. D., and Mesgarani, N. Speaker-independent auditory attention decoding without access to clean speech sources. *Science advances 5*, 5 (2019), eaav6134.

[75] Hecht-Nielsen, R. Theory of the backpropagation neural network. In *International 1989 Joint Conference on Neural Networks* (1989), pp. 593–605 vol.1.

[76] Heigold, G., Moreno, I., Bengio, S., and Shazeer, N. End-to-end text-dependent speaker verification. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), IEEE, pp. 5115–5119.

[77] Heitkaemper, J., Jakobeit, D., Boeddeker, C., Drude, L., and Haeb-Umbach, R. Demystifying tasnet: A dissecting approach. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2020), IEEE, pp. 6359–6363.

[78] Hershey, J. R., Chen, Z., Le Roux, J., and Watanabe, S. Deep clustering: Discriminative embeddings for segmentation and separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), IEEE, pp. 31–35.

[79] Hershey, J. R., Rennie, S. J., Olsen, P. A., and Kristjansson, T. T. Super-human multi-talker speech recognition: A graphical modeling approach. *Computer Speech & Language 24*, 1 (2010), 45–66.

[80] Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., and Sainath, T. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine 29* (November 2012), 82–97.

[81] Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Neural Networks*, J. F. Kolen and S. C. Kremer, Eds. IEEE Press, 2001, ch. 14, pp. 237–244.

[82] Hochreiter, S., and Schmidhuber, J. Long short-term memory. *Neural computation 9*, 8 (1997), 1735–1780.

[83] Huang, J.-T., Li, J., Yu, D., Deng, L., and Gong, Y. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (2013), IEEE, pp. 7304–7308.

[84] Huang, P.-S., Kim, M., Hasegawa-Johnson, M., and Smaragdis, P. Deep learning for monaural speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2014), pp. 1562–1566.

[85] Huang, P.-S., Kim, M., Hasegawa Johnson, M., and Smaragdis, P. Joint optimization of masks and deep recurrent neural networks for monaural source separation. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 23*, 12 (2015), 2136–2147.

[86] Hulens, D., Aerts, B., Chakravarty, P., Diba, A., Goedemé, T., Roussel, T., Zegers, J., Tuytelaars, T., Van Eycken, L., Van Gool, L., Van hamme, H., and Vennekens, J. The cametron lecture recording system: High quality video recording and editing with minimal human supervision. In *International Conference on Multimedia Modeling* (2018), Springer, pp. 518–530.

[87] Hurmalainen, A., Saeidi, R., and Virtanen, T. Group sparsity for speaker identity discrimination in factorisation-based speech recognition. In *INTERSPEECH* (2012), pp. 2138–2141.

[88] Isik, Y., Le Roux, J., Chen, Z., Watanabe, S., and Hershey, J. R. Single-channel multi-speaker separation using deep clustering. *Interspeech 2016* (2016), 545–549.

[89] Joder, C., and Schuller, B. Exploring nonnegative matrix factorization for audio classification: Application to speaker recognition. In *Speech Communication* (2012), VDE.

[90] Kajarekar, S. S., Scheffer, N., Graciarena, M., Shriberg, E., Stolcke, A., Ferrer, L., and Bocklet, T. The sri nist 2008 speaker recognition evaluation system. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing* (2009), pp. 4205–4208.

[91] Kayser, H., Ewert, S. D., Anemüller, J., Rohdenburg, T., Hohmann, V., and Kollmeier, B. Database of multichannel in-ear and behind-the-ear head-related and binaural room impulse responses. *EURASIP Journal on Advances in Signal Processing 2009* (2009), 6.

[92] Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *The*

IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2018).

[93] Kingma, D., and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[94] Klatt, D. H. Voice onset time, frication, and aspiration in word-initial consonant clusters. *Journal of Speech and Hearing Research 18*, 4 (1975), 686–706.

[95] Knapp, C. H., and Carter, G. C. The generalized correlation method for estimation of time delay. *IEEE Transactions on Acoustics, Speech and Signal Processing 24*, 4 (1976), 320–327.

[96] Kokkinos, I. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (July 2017), IEEE.

[97] Kolbæk, M., Yu, D., Tan, Z., and Jensen, J. Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 25*, 10 (2017), 1901–1913.

[98] Koldovsky, Z., and Tichavsky, P. Time-domain blind separation of audio sources on the basis of a complete ica decomposition of an observation space. *IEEE transactions on audio, speech, and language processing 19*, 2 (2010), 406–416.

[99] Koutník, J., Greff, K., Gomez, F., and Schmidhuber, J. A clockwork rnn. In *Proceedings of the 31st International Conference on Machine Learning* (2014), E. P. Xing and T. Jebara, Eds., vol. 32, pp. 1863–1871.

[100] Kumar, N., and Karunavathi, R. Estoi for predicting the intelligibility of speech. *International Journal of Advance Research, Ideas and Innovations in Technology 4* (2018), 793–799.

[101] Le Roux, J., Weninger, F. J., and Hershey, J. R. Sparse nmf–half-baked or well done? *Mitsubishi Electric Research Labs (MERL), Cambridge, MA, USA, Tech. Rep., no. TR2015-023* (2015).

[102] Le Roux, J., Wisdom, S., Erdogan, H., and Hershey, J. R. Sdr–half-baked or well done? In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2019), IEEE, pp. 626–630.

[103] Lee, D. D., and Seung, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature 401* (1999), 788–791.

[104] Lefèvre, A., Bach, F., and Févotte, C. Itakura-saito nonnegative matrix factorization with group sparsity. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2011), pp. 21–24.

[105] Lei, Y., Scheffer, N., Ferrer, L., and McLaren, M. A novel scheme for speaker recognition using a phonetically-aware deep neural network. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2014), IEEE, pp. 1695–1699.

[106] Li, X., Zhao, L., Wei, L., Yang, M.-H., Wu, F., Zhuang, Y., Ling, H., and Wang, J. Deepsaliency: Multi-task deep neural network model for salient object detection. *IEEE transactions on image processing 25*, 8 (2016), 3919–3930.

[107] Liao, H. Speaker adaptation of context dependent deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2013), pp. 7947–7951.

[108] Librivox. `https://librivox.org/`. [Online; accessed: May-2020].

[109] Lisker, L., and Abramson, A. S. Some effects of context on voice onset time in english stops. *Language and Speech 10*, 1 (1967), 1–28.

[110] Loizou, P. C. *Speech enhancement: theory and practice.* CRC press, 2013.

[111] Lu, Z., Pu, H., Wang, F., Hu, Z., and Wang, L. The expressive power of neural networks: A view from the width. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 6231–6239.

[112] Marchi, E., Shum, S., Hwang, K., Kajarekar, S., Sigtia, S., Richards, H., Haynes, R., Kim, Y., and Bridle, J. Generalised discriminative transform via curriculum learning for speaker recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), pp. 5324–5328.

[113] May, T., van de Par, S., and Kohlrausch, A. A binaural scene analyzer for joint localization and recognition of speakers in the presence of interfering noise sources and reverberation. *IEEE Transactions on Audio, Speech, and Language Processing 20*, 7 (2012), 2016–2030.

[114] McAulay, R., and Malpass, M. Speech enhancement using a soft-decision noise suppression filter. *IEEE Transactions on Acoustics, Speech, and Signal Processing 28*, 2 (1980), 137–145.

[115] McCulloch, W. S., and Pitts, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*, 4 (1943), 115–133.

[116] Mesgarani, N., and Chang, E. F. Selective cortical representation of attended speaker in multi-talker speech perception. *Nature 485*, 7397 (2012), 233–236.

[117] Miller, J. L., Grosjean, F., and Lomanto, C. Articulation rate and its variability in spontaneous speech: A reanalysis and some implications. *Phonetica 41*, 4 (1984), 215–225.

[118] Miran, S., Akram, S., Sheikhattar, A., Simon, J. Z., Zhang, T., and Babadi, B. Real-time tracking of selective auditory attention from M/EEG: A bayesian filtering approach. *Frontiers in neuroscience 12* (2018).

[119] Mirkovic, B., Bleichner, M. G., De Vos, M., and Debener, S. Target speaker detection with concealed EEG around the ear. *Frontiers in neuroscience 10* (2016), 349.

[120] Mirzaei, S., Van hamme, H., and Norouzi, Y. Blind audio source separation of stereo mixtures using bayesian non-negative matrix factorization. In *European Signal Processing Conference (EUSIPCO)* (2014), IEEE, pp. 621–625.

[121] Mobin, S., Cheung, B., and Olshausen, B. Convolutional vs. recurrent neural networks for audio source separation.

[122] Mohamed, A.-r., Seide, F., Yu, D., Droppo, J., Stoicke, A., Zweig, G., and Penn, G. Deep bi-directional recurrent networks over spectral windows. In *Automatic Speech Recognition and Understanding (ASRU), 2015 IEEE Workshop on* (2015), IEEE, pp. 78–83.

[123] Mowlaee, P., Saeidi, R., and Stylianou, Y. Advances in phase-aware signal processing in speech communication. *Speech communication 81* (2016), 1–29.

[124] Mozer, M. C. Induction of multiscale temporal structure. In *Advances in neural information processing systems* (1992), pp. 275–282.

[125] Mysore, G. J., Smaragdis, P., and Raj, B. Non-negative hidden markov modeling of audio with application to source separation. In *International Conference on Latent Variable Analysis and Signal Separation* (2010), Springer, pp. 140–148.

[126] Nakano, M., Kameoka, H., Le Roux, J., Kitano, Y., Ono, N., and Sagayama, S. Convergence-guaranteed multiplicative algorithms for nonnegative matrix factorization with β-divergence. *In Proceedings of the 2010 IEEE International Workshop on Machine Learning for Signal Processing (MLSP), Kittila, Finland, 29 August – 1 September 2010* (2010), 283–288.

[127] Narayanan, A. M., and Bertrand, A. Analysis of miniaturization effects and channel selection strategies for EEG sensor networks with application to auditory attention detection. *IEEE Transactions on Biomedical Engineering 67*, 1 (January 2020), 234–244.

[128] Neil, D., Pfeiffer, M., and Liu, S.-C. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Advances in Neural Information Processing Systems* (2016), pp. 3882–3890.

[129] O'Sullivan, J., Chen, Z., Herrero, J., McKhann, G. M., Sheth, S. A., Mehta, A. D., and Mesgarani, N. Neural decoding of attentional selection in multi-speaker environments without access to clean sources. *Journal of neural engineering 14*, 5 (2017), 056001.

[130] O'Sullivan, J. A., Power, A. J., Mesgarani, N., Rajaram, S., Foxe, J. J., Shinn-Cunningham, B. G., Slaney, M., Shamma, S. A., and Lalor, E. C. Attentional selection in a cocktail party environment can be decoded from single-trial EEG. *Cerebral Cortex* (2014), 1697–1706.

[131] Ozerov, A., and Févotte, C. Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing 18*, 3 (2010), 550–563.

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

[133] Patel, R. R., Forrest, K., and Hedges, D. Relationship between acoustic voice onset and offset and selected instances of oscillatory onset and offset in young healthy men and women. *Journal of Voice 31*, 3 (2017), 389.e9–389.e17.

[134] Patterson, R. D., Allerhand, M. H., and Giguere, C. Time-domain modeling of peripheral auditory processing: A modular architecture and a software platform. *The Journal of the Acoustical Society of America 98*, 4 (1995), 1890–1894.

[135] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[136] Petersen, E. B., Wöstmann, M., Obleser, J., and Lunner, T. Neural tracking of attended versus ignored speech is differentially affected by hearing loss. *Journal of neurophysiology 117*, 1 (2016), 18–27.

[137] Petersen, K. B., and Pedersen, M. S. The matrix cookbook, nov 2012. Version 20121115.

[138] Pind, J. Rate dependent perception of aspiration and pre aspiration in icelandic. *The Quarterly Journal of Experimental Psychology: Section A 49*, 3 (1996), 745–764.

[139] Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., and Vesely, K. The kaldi speech recognition toolkit. In *IEEE 2011 Workshop on Automatic Speech Recognition and Understanding* (Dec. 2011), IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

[140] Presacco, A., Simon, J. Z., and Anderson, S. Speech-in-noise representation in the aging midbrain and cortex: Effects of hearing loss. *PloS one 14*, 3 (2019), e0213899.

[141] Qian, N. On the momentum term in gradient descent learning algorithms. *Neural networks 12*, 1 (1999), 145–151.

[142] Radioboeken. `http://www.radioboeken.eu/radioboeken.php?lang=NL`. [Online; accessed: May-2020].

[143] Richardson, F., Reynolds, D., and Dehak, N. Deep neural network approaches to speaker and language recognition. *IEEE signal processing letters 22*, 10 (2015), 1671–1675.

[144] Rix, A. W., Beerends, J. G., Hollier, M. P., and Hekstra, A. P. Perceptual evaluation of speech quality (pesq)-a new method for speech

quality assessment of telephone networks and codecs. In *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)* (2001), vol. 2, IEEE, pp. 749–752.

[145] Rosenblatt, F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review 65*, 6 (1958), 386.

[146] Ruder, S. An overview of gradient descent optimization algorithms. *CoRR abs/1609.04747* (2016).

[147] Sadjadi, S. O., Slaney, M., and Heck, L. MSR identity toolbox v1.0: A MATLAB toolbox for speaker-recognition research. *Speech and Language Processing Technical Committee Newsletter* (November 2013).

[148] Sadjadi, S. O., Slaney, M., and Heck, L. MSR identity toolbox v1.0: A MATLAB toolbox for speaker-recognition research. *Speech and Language Processing Technical Committee Newsletter 1*, 4 (november 2013).

[149] Samek, W., Wiegand, T., and Müller, K.-R. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296* (2017).

[150] Saon, G., Soltau, H., Nahamoo, D., and Picheny, M. Speaker adaptation of neural network acoustic models using i-vectors. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (2013), pp. 55–59.

[151] Sawada, H., Kameoka, H., Araki, S., and Ueda, N. Multichannel extensions of non-negative matrix factorization with complex-valued data. *IEEE Transactions on Audio, Speech, and Language Processing 21*, 5 (2013), 971–982.

[152] Schultz, T. Globalphone: a multilingual speech and text database developed at Karlsruhe University. In *Seventh International Conference on Spoken Language Processing* (2002).

[153] Schuster, M., and Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing 45*, 11 (1997), 2673–2681.

[154] Seide, F., Li, G., Chen, X., and Yu, D. Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (2011), pp. 24–29.

[155] Serizel, R., Moonen, M., Van Dijk, B., and Wouters, J. Low-rank approximation based multichannel Wiener filter algorithms for noise reduction with application in cochlear implants. *IEEE/ACM Trans. Audio, Speech, and Language Processing 22*, 4 (2014), 785–799.

[156] Shao, Y., Srinivasan, S., Jin, Z., and Wang, D. A computational auditory scene analysis system for speech segregation and robust speech recognition. *Computer Speech & Language 24*, 1 (2010), 77–93.

[157] Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-k., and WOO, W.-c. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 802–810.

[158] Shi, Z., Lin, H., Liu, L., Liu, R., Han, J., and Shi, A. Deep Attention Gated Dilated Temporal Convolutional Networks with Intra-Parallel Convolutional Modules for End-to-End Monaural Speech Separation. In *Interspeech 2019* (2019), pp. 3183–3187.

[159] Shi, Z., Lin, H., Liu, L., Liu, R., Hayakawa, S., Harada, S., and Han, J. End-to-End Monaural Speech Separation with Multi-Scale Dynamic Weighted Gated Dilated Convolutional Pyramid Network. In *Interspeech 2019* (2019), pp. 4614–4618.

[160] Singh, B., Marks, T. K., Jones, M., Tuzel, O., and Shao, M. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *Computer Vision and Pattern Recognition (CVPR)* (2016), IEEE, pp. 1961–1970.

[161] Singh, R., Keshet, J., Gencaga, D., and Raj, B. The relationship of voice onset time and voice offset time to physical age. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016), vol. 2016-, IEEE, pp. 5390–5394.

[162] Stephenson, C., Callier, P., Ganesh, A., and Ni, K. Monaural audio speaker separation with source contrastive estimation. *arXiv preprint arXiv:1705.04662* (2017).

[163] Stevens, S. S. The measurement of loudness. *The Journal of the Acoustical Society of America 27*, 5 (1955), 815–829.

[164] Student. The probable error of a mean. *Biometrika* (1908), 1–25.

[165] Sundermeyer, M., Ney, H., and Schluter, R. From feedforward to recurrent lstm neural networks for language modeling. *IEEE/ACM*

Transactions on Audio, Speech, and Language Processing (TASLP) 23, 3 (2015), 517–529.

[166] Swietojanski, P., and Renals, S. Learning hidden unit contributions for unsupervised speaker adaptation of neural network acoustic models. In *IEEE Spoken Language Technology Workshop (SLT)* (2014), pp. 171–176.

[167] Sztahó, D., Szaszák, G., and Beke, A. Deep learning methods in speaker recognition: a review. *arXiv preprint arXiv:1911.06615* (2019).

[168] Taghia, J., Martin, R., and Hendriks, R. C. On mutual information as a measure of speech intelligibility. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2012), IEEE, pp. 65–68.

[169] Tan, Z.-H., and Lindberg, B. Low-complexity variable frame rate analysis for speech recognition and voice activity detection. *IEEE Journal of Selected Topics in Signal Processing 4*, 5 (2010), 798–807.

[170] Tüske, Z., Schlüter, R., and Ney, H. Investigation on lstm recurrent n-gram language models for speech recognition. In *Interspeech 2018* (2018), ISCA, pp. 3358–3362.

[171] Umeda, N. Vowel duration in american english. *The Journal of the Acoustical Society of America 58*, 2 (1975), 434–445.

[172] Van den Bogaert, T., Doclo, S., Wouters, J., and Moonen, M. Speech enhancement with multichannel wiener filter techniques in multimicrophone binaural hearing aids. *The Journal of the Acoustical Society of America 125*, 1 (2009), 360–371.

[173] Van Eerten, L. Over het corpus gesproken nederlands. *Nederlandse Taalkunde 12*, 3 (2007), 194–215.

[174] Van Eyndhoven, S., Francart, T., and Bertrand, A. EEG-informed attended speaker extraction from recorded speech mixtures with application in neuro-steered hearing prostheses. *IEEE Transactions on Biomedical Engineering 64*, 5 (2017), 1045–1056.

[175] Variani, E., Lei, X., McDermott, E., Moreno, I. L., and Gonzalez-Dominguez, J. Deep neural networks for small footprint text-dependent speaker verification. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2014), IEEE, pp. 4052–4056.

[176] Vincent, E., Gribonval, R., and Févotte, C. Performance measurement in blind audio source separation. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 14*, 4 (2006), 1462–1469.

[177] Vincent, E., Watanabe, S., Barker, J., and Marxer, R. The 4th chime speech separation and recognition challenge. *URL: http://spandh. dcs. shef. ac. uk/chime challenge {Last Accessed on 1 August, 2018}* (2016).

[178] Virtanen, T. Speech recognition using factorial hidden markov models for separation in the feature space. In *Ninth International Conference on Spoken Language Processing* (2006).

[179] Virtanen, T. Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria. *IEEE Transactions on Audio, Speech, and Language Processing 15*, 3 (2007), 1066–1074.

[180] Wang, E. B. D., Brown, G. J., and Darwin, C. Computational auditory scene analysis: Principles, algorithms and applications. *Acoustical Society of America Journal 124* (2008), 13.

[181] Wang, Y., Narayanan, A., and Wang, D. On training targets for supervised speech separation. *IEEE/ACM transactions on audio, speech, and language processing 22*, 12 (2014), 1849–1858.

[182] Wang, Y., and Wang, D. Towards scaling up classification-based speech separation. *IEEE Transactions on Audio, Speech, and Language Processing 21*, 7 (2013), 1381–1390.

[183] Wang, Z.-Q., Le Roux, J., and Hershey, J. R. 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)* (Calgary, Canada, 2018), IEEE, pp. 1–5.

[184] Weninger, F., Eyben, F., and Schuller, B. Single-channel speech separation with memory-enhanced recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2014), pp. 3709–3713.

[185] Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE 78*, 10 (1990), 1550–1560.

[186] Wichern, G., Antognini, J., Flynn, M., Zhu, L. R., McQuinn, E., Crow, D., Manilow, E., and Roux, J. L. Wham!: Extending speech separation to noisy environments. In *Interspeech* (Graz, Austria, 2019), ISCA, pp. 1368–1372.

[187] Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics Bulletin 1*, 6 (1945), 80–83.

[188] Xu, Y., Du, J., Dai, L.-R., and Lee, C.-H. An experimental study on speech enhancement based on deep neural networks. *IEEE Signal Process. Lett. 21*, 1 (2014), 65–68.

[189] Xue, J., Li, J., Yu, D., Seltzer, M., and Gong, Y. Singular value decomposition based low-footprint speaker adaptation and personalization for deep neural network. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2014), pp. 6359–6363.

[190] Yao, K., Yu, D., Seide, F., Su, H., Deng, L., and Gong, Y. Adaptation of context-dependent deep neural networks for automatic speech recognition. In *IEEE Spoken Language Technology Workshop (SLT)* (2012), pp. 366–369.

[191] Yim, J., Jung, H., Yoo, B., Choi, C., Park, D., and Kim, J. Rotating your face using multi-task deep neural network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015), IEEE.

[192] Yoshioka, T., Erdogan, H., Chen, Z., and Alleva, F. Multi-microphone neural speech separation for far-field multi-talker speech recognition. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 5739–5743.

[193] Yu, D., Yao, K., Su, H., Li, G., and Seide, F. Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2013), pp. 7893–7897.

[194] Zegers, J., and Van hamme, H. Joint sound source separation and speaker recognition. In *Interspeech 2016* (2016), ISCA, pp. 2228–2232.

[195] Zegers, J., and Van hamme, H. Improving source separation via multi-speaker representations. In *Interspeech 2017* (2017), ISCA, pp. 1919–1923.

[196] Zegers, J., and Van hamme, H. Memory time span in LSTMs for multi-speaker source separation. In *Interspeech 2018* (2018), ISCA, pp. 1477–1481.

[197] Zegers, J., and Van hamme, H. Multi-scenario deep learning for multi-speaker source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 5379–5383.

[198] Zegers, J., and Van hamme, H. CNN-LSTM models for multi-speaker source separation using Bayesian hyper parameter optimization. In *Interspeech 2019* (2019), ISCA, pp. 4589–4593.

[199] Zegers, J., and Van hamme, H. Analysis of memory in LSTM-RNNs for source separation, 2020.

[200] Zhang, X., Zhang, H., Nie, S., Gao, G., and Liu, W. A pairwise algorithm using the deep stacking network for speech separation and pitch estimation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 24*, 6 (2016), 1066–1078.

[201] Zhang, X.-L., and Wang, D. A deep ensemble learning method for monaural speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing 24*, 5 (May 2016), 967–977.

[202] Zhang, Y., Xu, J., Yan, Z.-J., and Huo, Q. An i-vector based approach to training data clustering for improved speech recognition. In *INTERSPEECH* (2011), pp. 789–792.

[203] Zink, R., Baptist, A., Bertrand, A., Van Huffel, S., and De Vos, M. Online detection of auditory attention in a neurofeedback application. In *Proc. 8th International Workshop on Biosignal Interpretation* (Osaka, Japan, 2016), pp. 1–4.

Short Biography

Jeroen Zegers was born on 4 October 1992 in Vilvoorde, Belgium. He received his bachelor's degree in Engineering Science in 2013, as well as his master's degree in Electrical Engineering in 2015, at KU Leuven. He joined the Processing of Speech and Images (PSI) group, Department of Electrical Engineering, KU Leuven as a doctoral student in 2015.

His research interests include speech and noise separation, speaker recognition, machine learning and deep learning. In 2017 he received a four year SB grant from FWO.

List of Publications

Articles in International Journals

1. Das, N., Zegers, J., Van hamme, H., Francart, T., and Bertrand, A. Linear versus deep learning methods for noisy speech separation for EEG-informed attention decoding. *Journal of Neural Engineering 17*, 4 (August 2020)

Articles in International Conferences

1. Zegers, J., and Van hamme, H. Joint sound source separation and speaker recognition. In *Interspeech 2016* (2016), ISCA, pp. 2228–2232

2. Chakravarty, P., Zegers, J., Tuytelaars, T., and Van hamme, H. Active speaker detection with audio-visual co-training. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction* (New York, NY, USA, 2016), ICMI '16, Association for Computing Machinery, p. 312–316

3. Zegers, J., and Van hamme, H. Improving source separation via multi-speaker representations. In *Interspeech 2017* (2017), ISCA, pp. 1919–1923

4. Zegers, J., and Van hamme, H. Multi-scenario deep learning for multi-speaker source separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 5379–5383

5. Hulens, D., Aerts, B., Chakravarty, P., Diba, A., Goedemé, T., Roussel, T., Zegers, J., Tuytelaars, T., Van Eycken, L., Van Gool, L., Van hamme, H., and Vennekens, J. The cametron lecture recording system: High quality video recording and editing with

minimal human supervision. In *International Conference on Multimedia Modeling* (2018), Springer, pp. 518–530

6. Zegers, J., and Van hamme, H. Memory time span in LSTMs for multi-speaker source separation. In *Interspeech 2018* (2018), ISCA, pp. 1477–1481

7. Appeltans, P., Zegers, J., and Van hamme, H. Practical applicability of deep neural networks for overlapping speaker separation. In *Interspeech 2019* (2019), ISCA, pp. 1353–1357

8. Zegers, J., and Van hamme, H. CNN-LSTM models for multi-speaker source separation using Bayesian hyper parameter optimization. In *Interspeech 2019* (2019), ISCA, pp. 4589–4593

Abstracts

1. Das, N., Zegers, J., Van hamme, H., Francart, T., and Bertrand, A. Multi-microphone speaker separation for neuro-steered hearing aids: neural networks versus linear methods. In *Auditory EEG Signal Processing symposium (AESoP)* (September 2019)

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF ELECTRICAL ENGINEERING
ESAT-PSI
Kasteelpark Arenberg 10 box 2441
B-3001 Leuven
jeroen.zegers@esat.kuleuven.be