Deep neural networks for source separation and noise-robust speech recognition

Aditya Arie Nugraha

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

Réseaux de neurones profonds pour la séparation des sources et la reconnaissance robuste de la parole

(Deep neural networks for source separation and noise-robust speech recognition)

THÈSE

présentée et soutenue publiquement le 5 décembre 2017

pour l'obtention du

Doctorat de l'Université de Lorraine

(mention informatique)

par

Aditya Arie Nugraha

Composition du jury

Rapporteurs :	Christian Jutten	Professeur, Université Grenoble Alpes, France
	Björn Schuller	Reader in Machine Learning, Imperial College London, Royaume-Uni
Examinateurs :	Stefan Uhlich	Principal Engineer, Sony Stuttgart Technology Center, Allemagne
	Marie-Odile Berger	Directeur de recherche, Inria Nancy – Grand Est, France
Directeurs de thèse :	Emmanuel Vincent	Directeur de recherche, Inria Nancy – Grand Est, France
	Antoine Liutkus	Chargé de recherche, Inria Sophia Antipolis – Méditerranée, France

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Résumé

Dans cette thèse, nous traitons le problème de la séparation de sources audio multicanale par réseaux de neurones profonds (*deep neural networks*, DNNs). Notre approche se base sur le cadre classique de séparation par algorithme espérance-maximisation (EM) basé sur un modèle gaussien multicanal, dans lequel les sources sont caractérisées par leurs spectres de puissance à court terme et leurs matrices de covariance spatiales. Nous explorons et optimisons l'usage des DNNs pour estimer ces paramètres spectraux et spatiaux. À partir des paramètres estimés, nous calculons un filtre de Wiener multicanal variant dans le temps pour séparer chaque source. Nous étudions en détail l'impact de plusieurs choix de conception pour les DNNs spectraux et spatiaux. Nous considérons plusieurs fonctions de coût, représentations temps-fréquence, architectures, et tailles d'ensembles d'apprentissage. Ces fonctions de coût incluent en particulier une nouvelle fonction liée à la tâche pour les DNNs spectraux: le rapport signal-à-distorsion. Nous présentons aussi une formule d'estimation pondérée des paramètres spatiaux, qui généralise la formulation EM exacte. Sur une tâche de séparation de voix chantée, nos systèmes sont remarquablement proches de la méthode de l'état de l'art actuel et améliorent le rapport source-interférence de 2 dB. Sur une tâche de rehaussement de la parole, nos systèmes surpassent la formation de voies GEV-BAN de l'état de l'art de 14%, 7% et 1% relatifs en terme d'amélioration du taux d'erreur sur les mots sur des données à 6, 4 et 2 canaux respectivement.

Mots-clés: séparation de sources audio multicanale, modèle gaussien multicanal, réseaux de neurones profonds

Abstract

This thesis addresses the problem of multichannel audio source separation by exploiting deep neural networks (DNNs). We build upon the classical expectation-maximization (EM) based source separation framework employing a multichannel Gaussian model, in which the sources are characterized by their power spectral densities and their source spatial covariance matrices. We explore and optimize the use of DNNs for estimating these spectral and spatial parameters. Employing the estimated source parameters, we then derive a time-varying multichannel Wiener filter for the separation of each source. We extensively study the impact of various design choices for the spectral and spatial DNNs. We consider different cost functions, time-frequency representations, architectures, and training data sizes. Those cost functions notably include a newly proposed task-oriented signal-to-distortion ratio cost function for spectral DNNs. Furthermore, we present a weighted spatial parameter estimation formula, which generalizes the corresponding exact EM formulation. On a singing-voice separation task, our systems perform remarkably close to the current state-of-the-art method and provide up to 2 dB improvement of the source-to-interference ratio. On a speech enhancement task, our systems outperform the state-of-the-art GEV-BAN beamformer by 14%, 7%, and 1% relative word error rate improvement on 6-channel, 4-channel, and 2-channel data, respectively.

Keywords: multichannel audio source separation, multichannel Gaussian model, deep neural networks

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisors, Emmanuel Vincent and Antoine Liutkus, for all the time and energy they have provided. Their thoughts, ideas, and suggestions are highly valuable for this research. I greatly appreciate their guidance, encouragement, and patience throughout the research process.

I would also like to thank all the members of MULTISPEECH research team for providing a friendly, supportive, and stimulating research environment. Special thanks go to Imran Sheikh and Sunit Sivasankaran for countless fruitful discussions and numerous technical assistance.

I would like to acknowledge both direct and indirect support from various funding agencies. This study was partly supported by the French National Research Agency (ANR) as parts of the DYCI2 project (ANR-14-CE24-0002-01) and the KAMoulox project (ANR-15-CE38-0003-01). Experiments presented in this thesis were carried out using the Grid'5000 testbed, supported by a scientific interest group hosted by Inria and including CNRS, RENATER, and several Universities as well as other organizations (see `https://www.grid5000.fr`).

Finally, I would like to express my gratitude to my family. I dedicate this thesis to my parents who always allow and support me to pursue my ambitions. I could not thank my wife enough for her love, patience, and understanding. I also appreciate my daughter for always cheering me up.

Contents

List of Tables

List of Figures

List of Abbreviations

ASR	automatic speech recognition
BAN	blind analytic normalization
CHiME	CHiME Speech Separation and Recognition Challenge
DNN	deep neural network
EM	expectation-maximization
ERB	equivalent rectangular bandwidth
FNN	feedforward neural network
GEV	generalized eigenvalue
IS	Itakura-Saito
ISR	source-image-to-spatial-distortion ratio
KL	Kullback-Leibler
LSTM	long short-term memory
MSE	mean squared error
NMF	non-negative matrix factorization
PSD	power spectral density
RNN	recurrent neural network
SAR	signal-to-artifacts ratio
SDR	signal-to-distortion ratio
SIR	signal-to-interference ratio
SiSEC	Signal Separation Evaluation Campaign
sMBR	state-level minimum Bayes' risk
STFT	short-time Fourier transform
WER	word error rate

Résumé étendu

A Introduction

La séparation de sources audio vise à récupérer le signal d'une ou plusieurs sources audio sous-jacentes à un signal de mélange observé [Makino et al., 2007; Vincent et al., 2017a]. Il s'agit d'un problème important dans la communauté du traitement du signal audio et qui attire une attention considérable de la part de nombreux chercheurs [Ewert et al., 2014; Vincent et al., 2014]. Le signal de mélange est soit un signal monocanal, par exemple un enregistrement de parole acquis avec un seul microphone ou un enregistrement de musique mono, soit un signal multicanal, par exemple un enregistrement de parole acquis avec deux microphones ou plus ou un enregistrement de musique stéréo. Le signal multicanal comporte des informations *spatiales* qui sont précieuses pour la séparation. Des études ont montré que ces informations peuvent être exploitées dans la séparation multicanale pour obtenir de meilleurs résultats qu'en monocanal.

Par ailleurs, les réseaux neuronaux profonds (*Deep Neural Networks*, DNNs), y compris les réseaux neuronaux récurrents (*Recurrent Neural Networks*, RNNs), peuvent modéliser des fonctions complexes et bien exécuter diverses tâches [Deng, 2014; Goodfellow et al., 2016], y compris dans le domaine du traitement du signal audio [Yu & Deng, 2011] et de la reconnaissance automatique de la parole (*Automatic Speech Recognition*, ASR) [Yu & Deng, 2015].

L'objectif principal de cette thèse est d'explorer et d'optimiser l'utilisation des DNNs pour la séparation de sources audio dans le contexte multicanal. Dans ce contexte, des filtres multicanaux invariants au cours du temps sont connus pour offrir une faible distorsion de la parole. Pourtant, ils peuvent ne pas convenir à des mélanges de nombreuses sources, dont on ne peut pas présumer la stationnarité, comme dans une chanson présentant du chant et divers instruments de musique. Dans notre étude, les DNNs sont utilisés pour estimer les paramètres spectraux et spatiaux de chaque source afin de dériver un *filtre multicanal variant au cours du temps*. Notre étude vise à obtenir les sources cibles séparées en multicanal, ce qui contraste avec les approches par formation de voies sus-citées qui visent à obtenir un signal de parole monocanal. Nous pouvons appliquer les systèmes proposés à d'autres tâches que le seul rehaussement de la parole, par exemple la séparation voix-accompagnement ou la séparation des différents instruments d'un signal musical.

B Notations et contexte

Dans cette thèse et comme cela est souvent fait dans la littérature, nous restreignons le problème de la séparation de sources audio à celui de récupérer les images spatiales de chaque source à partir du mélange. Chaque image spatiale est définie comme la contribution d'une source à chaque canal du mélange observé [Vincent et al., 2006]. Le mélange est supposé en être la somme. Soient I, J, F et N le nombre de canaux, de sources, de bande fréquentielles et de trames temporelles, respectivement. Dans une représentation temps-fréquence comme la transformée de Fourier à court terme (*Short Term Fourier Transform*, STFT), on peut l'exprimer comme suit:

$$\mathbf{x}(f,n) = \sum_{j=1}^{J} \mathbf{c}_j(f,n), \tag{1}$$

où $\mathbf{x}(f,n) \in \mathbb{C}^{I\times 1}$ et $\mathbf{c}_j(f,n) \in \mathbb{C}^{I\times 1}$ représentent respectivement les coefficients de STFT du mélange observé et des images spatiales des sources. La séparation vise à récupérer $\mathbf{c}_j(f,n)$ à partir de $\mathbf{x}(f,n)$.

Notre étude a pour point de départ le cadre probabiliste gaussien multicanal, faisant intervenir un algorithme d'estimation itératif: l'algorithme d'Espérance-Maximisation (EM) [Duong et al., 2010a]. Dans ce modèle, $\mathbf{c}_j(f,n)$ sont supposés être indépendants et distribués selon une distribution gaussienne complexe isotrope centrée de matrice de covariance $\mathbf{R}_{\mathbf{c}_j}(f,n)$ variant dans le temps:

$$\mathbf{c}_j(f,n) \sim \mathcal{N}_c\left(\mathbf{0},\, \mathbf{R}_{\mathbf{c}_j}(f,n) = v_j(f,n)\mathbf{R}_j(f)\right), \tag{2}$$

où $v_j(f,n) \in \mathbb{R}_+$ désigne la *densité spectrale de puissance* (*Power Spectral Density*, PSD) de la source j pour la fréquence f et la trame n tandis que $\mathbf{R}_j(f)$ est la matrice de covariance spatiale de la source, constante au cours du temps. Tandis que $v_j(f,n)$ encode le contenu spectral de cette source, $R_j(f)$ contient les corrélations des différents canaux, rendant compte de sa disposition *spatiale* dans le champ multicanal. En supposant que les sources ne sont pas corrélées, le mélange suit également une distribution gaussienne centrée, avec comme matrice de covariance $\mathbf{R}_{\mathbf{x}}(f,n)$:

$$\mathbf{x}(f,n) \sim \mathcal{N}_c\left(\mathbf{0},\, \mathbf{R}_{\mathbf{x}}(f,n) = \sum_{j=1}^{J} \mathbf{R}_{\mathbf{c}_j}(f,n)\right). \tag{3}$$

Les matrices de covariance $\mathbf{R}_{\mathbf{c}_j}(f,n)$, $\mathbf{R}_{\mathbf{x}}(f,n)$, et $\mathbf{R}_j(f)$ sont des matrices complexes hermitiennes définies positives.

Étant données les PSDs (les paramètres spectraux) et les matrices de covariance spatiale (les paramètres spatiaux) de toutes les sources, leurs images spatiales peuvent être estimées au sens de l'erreur quadratique moyenne minimale (minimum MSE) en utilisant le filtrage de Wiener multicanal [Duong et al., 2010a] donné par

$$\widehat{\mathbf{c}}_j(f,n) = \mathbf{W}_j(f,n)\mathbf{x}(f,n) = \mathbf{R}_{\mathbf{c}_j}(f,n)\mathbf{R}_{\mathbf{x}}(f,n)^{-1}\mathbf{x}(f,n), \tag{4}$$

où $\mathbf{W}_j(f,n) \in \mathbb{C}^{I\times I}$ est le *filtre de Wiener multicanal variable dans le temps*. Les formes d'ondes correspondantes sont ensuite récupérées à partir de $\widehat{\mathbf{c}}_j(f,n)$ par STFT inverse.

L'algorithme EM suivant, introduit par Ozerov et al. [2012], est une extension de Duong et al. [2010a] qui vise à résoudre le problème de l'estimation des paramètres spectraux et spatiaux de toutes les sources au sens du maximum de vraisemblance. Sa première étape est d'initialiser tous les paramètres. Il s'agit en particulier d'*initialiser les PSDs des sources*. Les paramètres spatiaux (matrices de covariance spatiales), quant à eux, peuvent par exemple être initialisés par des matrices identités.

Étant donnés ces paramètres spectraux et spatiaux, dans l'étape E, les estimations des sources images $\widehat{\mathbf{c}}_j(f,n)$ sont obtenues par filtrage de Wiener multicanal (4) et leurs moments bruts a posteriori du second ordre $\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)$ sont donnés par

$$\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n) = \widehat{\mathbf{c}}_j(f,n)\widehat{\mathbf{c}}_j(f,n)^* + (\mathbf{I} - \mathbf{W}_j(f,n))\,\mathbf{R}_{\mathbf{c}_j}(f,n). \tag{5}$$

Lors de l'étape M, les paramètres spatiaux sont mis à jour par

$$\mathbf{R}_j(f) = \frac{1}{N}\sum_{n=1}^{N} \frac{1}{v_j(f,n)}\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n). \tag{6}$$

Pour ce qui est des paramètres spectraux, ils sont d'abord mis à jour de manière non contrainte en formant $z_j(f, n)$, défini par

$$z_j(f,n) = \frac{1}{I}\mathrm{tr}\left(\mathbf{R}_j(f)^{-1}\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)\right), \tag{7}$$

où tr$(\cdot)$ est la trace d'une matrice. Les paramètres spectraux $v_j(f, n)$ sont ensuite mis à jour à partir de $z_j(f, n)$ dans une étape d'*ajustement de spectrogramme* où les v_j sont entendus comme la meilleur approximation des z_j qui obéisse à un type de modèle choisi. Par exemple, les v_j peuvent être obtenus par factorisation matricielle positive des z_j (*Nonnegative Matrix Factorization*, NMF) [Ozerov et al., 2012], ou n'en retenir que certaines régularités locales (par exemple, par *Kernel Additive Modelling*) [Liutkus et al., 2014; Duong et al., 2011].

Dans cette thèse, nous proposons deux formalismes de séparation de sources audio multicanaux basés sur les DNNs. Le premier, dont il est question dans la partie C, utilise les DNNs uniquement pour modéliser les paramètres spectraux des sources (leurs densités spectrales de puissance). Dans le deuxième, examiné dans la partie E, nous utilisons les DNNs comme modèles spectraux mais aussi spatiaux. Une étude expérimentale approfondie a été menée pour évaluer ces deux méthodes. La performance de la séparation est mesurée par BSSEval [Vincent et al., 2007], qui fournit un ensemble de métriques exprimées en décibels (dB): le rapport signal-à-distorsion (SDR), le rapport source image à distorsion spatiale (ISR), le rapport source-à-interférence (SIR) et le rapport source-à-artefacts (SAR). Pour les expériences en ASR, la performance de reconnaissance est mesurée par le taux d'erreur sur les mots (*Word Error Rate*, WER), exprimé en pourcents de mots mal reconnus.

C Estimation des paramètres spectraux avec des DNNs

Le premier modèle proposé dans notre étude utilise des DNNs pour estimer et mettre à jour les paramètres spectraux seulement. Les mises à jour spatiales sont obtenues par application directe de la méthodologie itérative EM [Ozerov et al., 2012] décrite plus haut. En pratique, les DNNs spectraux manipulent les spectres d'amplitude des sources, définis comme la racine carrée des densités spectrales de puissance.

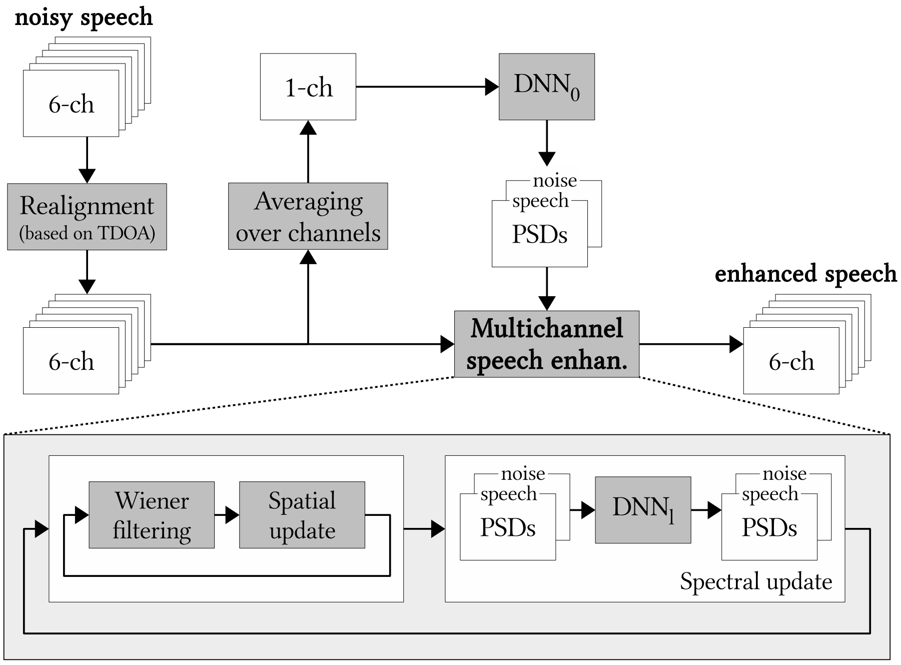


Figure C.1: Système DNN proposé pour le rehaussement multicanal de la parole.

Tout système qui implémente cette architecture se décompose en trois étapes principales: le prétraitement, l'initialisation et le filtrage multicanal, avec une étape optionnelle de post-traitement. L'étape de *pré-traitement* aligne le mélange observé et extrait les caractéristiques pertinentes à utiliser comme entrées pour les DNNs. L'alignement est nécessaire pour satisfaire le modèle de (2) qui suppose que les sources ne bougent pas dans le temps. L'étape *d'initialisation* définit ensuite les paramètres spectraux et spatiaux initiaux. L'étape de *filtrage multicanal* effectue les mises à jour (MÀJ) itératives des paramètres. Enfin, l'étape optionnelle de *post-traitement* traite les images spatiales multicanales estimées pour l'application cible. Par exemple, l'image spatiale multicanale de parole obtenue est moyennée sur plusieurs canaux pour obtenir un signal monocanal utilisable pour la reconnaissance de la parole.

Nous avons appliqué cette architecture à une tâche de rehaussement de la parole. Il s'agit de la séparation des deux sources ($J = 2$) que sont la parole et le bruit, à partir d'un mélange de 6 canaux ($I = 6$). L'ensemble de données CHiME du défi CHiME-3/4 [Barker et al., 2015; Vincent et al., 2017b] a été utilisé dans les expériences. La figure C.1 représente le système proposé.

En utilisant ce système, nous avons évalué l'impact des différents choix de conception sur la performance. Ces choix incluent notamment le nombre

Table C.1: Performance de reconnaissance vocale en termes de WER (%) en utilisant différentes méthodes de rehaussement de la parole. L'évaluation a été faite sur le jeu de signaux réels de la base. Les chiffres en gras indiquent les meilleures performances pour chaque jeu de données. Les intervalles de confiance à 95% pour les deux meilleurs WER sont $\pm$ 0,26% pour le jeu de développement et $\pm$ 0,40% pour le jeu de test. Un WER plus faible est préférable.

Méthode de rehaussement	**itér. EM**	**type de MÀJ**	**Dév.**	**Test**
Observé (bruité)	-	-	9.65	19.28
BeamformIt	-	-	6.36	13.67
NMF [Ozerov et al., 2012]	50	-	6.10	13.41
Modèle DNN proposé: KL	3	spatial	**4.88**	**10.14**

de mises à jour des paramètres spatiaux après l'initialisation des paramètres spectraux et l'utilisation de DNN multiples pour estimer les paramètres spectraux pour différentes itérations. Les expériences ont montré que le fait de mettre à jour plusieurs fois les paramètres spatiaux avant une mise à jour spectrale permet d'obtenir de meilleures performances que l'alternance simple suggérée par le cadre EM classique. Les expériences ont également montré que différents DNNs peuvent être utilisés pour la mise à jour des paramètres spectraux de différentes itérations afin d'améliorer la performance globale.

Le Tableau C.1 montre les performances de reconnaissance vocale de différentes méthodes de rehaussement. Nous avons utilisé le système d'ASR fourni par les organisateurs de CHiME-3 [Barker et al., 2015], dans lequel le modèle acoustique DNN+sMBR a été entraîné sur des données augmentées multi-conditions [Hori et al., 2015]. BeamformIt est la méthode de rehaussement de base dans CHiME-3 et le système NMF multicanal basé sur NMF [Ozerov et al., 2012] constituait l'état de l'art avant l'émergence de l'apprentissage profond. Pour les systèmes NMF et DNN, les signaux monocanaux des entrées ASR sont obtenus en calculant la moyenne des canaux des images spatiales séparées de la parole. Notre système utilise un DNN pour l'initialisation des spectrogrammes et deux autres pour leur ajustement au cours d'itérations successives. Tous les DNN suivent une architecture *MultiLayer Perceptron* (MLP) et sont appris au sens de la divergence de Kullback-Leibler (KL) minimale [Lee & Seung, 2000]. Notre système diminue le WER sur le jeu de test réel de 26% par rapport à BeamformIt et de 24 % par rapport au système NMF. De plus, notre système basé sur les DNNs a largement dépassé le système basé sur la NMF en terme de métriques de séparation des sources. Il a augmenté par exemple le SDR moyen de 5,5 dB par rapport au système par NMF.

D Sur l'amélioration des modèles spectraux profonds

Nous avons ensuite exploré des moyens d'améliorer les DNNs spectraux. Nous avons examiné différentes fonctions de coût, différentes représentations temps-fréquence, différentes architectures de DNN et différentes tailles de données pour l'apprentissage des DNNs.

Le Tableau D.2 montre les performances de la reconnaissance vocale à l'aide de DNNs appris avec différentes fonctions de coût pour l'initialisation du spectrogramme et suivis des mises à jour spatiales. Le système d'ASR est le même que dans le Tableau C.1. Outre la divergence KL utilisée précédemment, nous avons examiné la divergence d'Itakura Saito (IS) [Itakura & Saito, 1968], motivée par des considérations probabilistes, la fonction de coût de Cauchy [Liutkus et al., 2015a] et l'erreur quadratique moyenne (MSE). Le meilleur WER est obtenu par la divergence IS. Il est intéressant de noter que la divergence KL donne de moins bons scores ici, bien qu'elle ait atteint le meilleur SDR. Cependant, cette différence est modérée et compte tenu de toutes les évaluations, nous concluons que la divergence KL est le choix le plus raisonnable parce qu'elle fournit la meilleure performance en séparation des sources et qu'elle fonctionne bien en terme de reconnaissance vocale. En outre, la divergence IS fournit les meilleures performances de reconnaissance vocale, mais sa performance en séparation est bien inférieure aux autres.

Les expériences ont montré que les représentations temps-fréquence basées sur l'échelle de fréquence *Equivalent Rectangular Bandwidth* (ERB), à motivation perceptuelle, sont favorables. Ces représentations de faible dimensionnalité nous permettent d'utiliser des DNNs plus petits et ainsi de réduire la durée de l'apprentissage. Les expériences ont ensuite montré qu'une architecture de RNN bidirectionnelle à mémoire à court et long terme (*bidirectional long short-term memory*, BLSTM) gère mieux le contexte que l'approche de concaténation de trames que nous avions utilisée de prime abord. Les expériences ont également montré que l'ajout des données simulées dans le jeu d'apprentissage n'améliorait pas la performance par rapport aux données réelles seules.

Table D.2: Performance de reconnaissance vocale en terme de WER (%) en utilisant différentes fonctions de coût pour l'apprentissage. L'évaluation a été faite sur le jeu de données réelles. Les chiffres en gras indiquent les meilleures performances pour chaque jeu de données. Les intervalles de confiance à 95% pour les deux meilleurs WER sont ± 0,26% pour l'ensemble de développement et ± 0,42% pour l'ensemble de test. Un WER plus faible est préférable.

Méthode de rehaussement	EM iter.	type de MÀJ	Dév.	Test
Modèle DNN proposé: Cauchy	1	spatial	4.96	11.13
Modèle DNN proposé: IS	1	spatial	**4.88**	**10.83**
Modèle DNN proposé: KL	1	spatial	5.37	11.46
Modèle DNN proposé: MSE	1	spatial	5.80	13.01

Nous avons également appliqué le système proposé à une tâche de séparation chant-accompagnement. Dans cette tâche, nous traitons de la séparation des deux sources ($J = 2$) que sont le chant et l'accompagnement d'un morceau de musique stéréo ($I = 2$). Nous avons utilisé l'ensemble de données DSD100 de la tâche MUS de SiSEC 2016 [Liutkus et al., 2017] et avons proposé deux systèmes utilisant le même DNN pour la mise à jour des paramètres spectraux mais appris avec des fonctions de coût différentes: le premier DNN a été appris avec MSE et l'autre avec une fonction de coût discriminante que nous avons développée, qui vise à optimiser directement le SDR des sources séparées et qui tient compte du filtrage multicanal et de l'inversion de la STFT.

Le Tableau D.3 compare les performances que nous obtenons à celles du système proposé dans Uhlich et al. [2017] sur l'ensemble de données DSD100. Sur le jeu de développement, nos systèmes sont nettement meilleurs que ceux d'Uhlich pour toutes les métriques et toutes les sources. Malheureusement, cette performance supérieure n'est pas reflétée dans les performances sur le jeu de test. Nos systèmes présentent des artefacts qui nuisent à la distorsion globale. Cependant, nos systèmes sont nettement plus efficaces pour enlever l'accompagnement de la voix, ce qui peut être favorable pour une tâche de séparation du chant. En termes de SIR sur les voix, nos systèmes fournissent jusqu'à une augmentation absolue de 2 dB par rapport au système d'Uhlich. Dans l'ensemble, nous pouvons dire que ces systèmes, développés indépendamment, définissent l'état de l'art dans le domaine aujourd'hui.

De plus, nous avons tenté de créer un jeu de données d'apprentissage plus grand par génération automatique pour résoudre le problème de sur-

Table D.3: Comparaison des performances de séparation de source avec l'état de l'art pour la tâche MUS de SiSEC 2016. Le tableau montre les valeurs médianes et leurs intervalles de confiance à 95%. Les intervalles varient entre ± 0,2 dB et ± 0,5 dB. Les chiffres en gras indiquent la meilleure performance pour chaque triplet d'une métrique, d'une source et d'un jeu de données. Les chiffres en italique montrent une performance qui n'est pas statistiquement différente de la meilleure performance. Pour toutes les métriques, une valeur plus élevée est préférable.

Méthode	**Voix**				**Accompagnement**			
	SDR	**ISR**	**SIR**	**SAR**	**SDR**	**ISR**	**SIR**	**SAR**
Ensemble de dev								
Uhlich et al. [2017]	6.9	11.2	12.0	9.0	12.7	21.0	17.0	16.1
Proposé: MSE	**9.1**	**15.2**	**18.4**	**10.0**	**14.9**	**25.6**	**20.8**	**16.7**
Proposé: SDR	*8.9*	*14.8*	*18.4*	*9.9*	*14.7*	*25.1*	*20.6*	*16.6*
Ensemble de test								
Uhlich et al. [2017]	**5.4**	**10.4**	9.2	**8.3**	**11.0**	17.6	**15.1**	**14.7**
Proposé: MSE	4.7	*10.0*	*11.1*	6.1	10.3	*18.5*	*14.8*	13.2
Proposé: SDR	4.7	*9.9*	**11.2**	6.1	10.4	**18.9**	*14.9*	13.3

apprentissage de nos systèmes, mis en évidence par l'écart de performance entre les jeux de développement et les jeux de test. Cependant, il n'a fourni que des améliorations absolues de 0,3 dB sur le chant et l'accompagnement.

E Estimation des paramètres spatiaux avec des DNNs

Un dernier volet de notre travail vise à dépasser le cadre EM pour la mise à jour des paramètres spatiaux. Tout d'abord, nous proposons une généralisation pondérée de la mise à jour des paramètres spatiaux:

$$\mathbf{R}_j(f) = \left(\sum_{n=1}^{N} \omega_j(f,n)\right)^{-1} \sum_{n=1}^{N} \frac{\omega_j(f,n)}{v_j(f,n)} \widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n), \tag{8}$$

où $\omega_j(f,n)$ indique le poids de la source j pour le point temps-fréquence (f,n). Lorsque $\omega_j(f,n)$ est le même pour tous points temps-fréquence (f,n), par exemple, $\omega_j(f,n) = 1$, (8) revient à la formulation exacte d'EM dans (6). Lorsque $\omega_j(f,n) = v_j(f,n)$, (8) revient à la méthode utilisée dans Liutkus et al. [2015b].

Les expériences ont démontré que l'utilisation de $\omega_j(f,n) = v_j(f,n)$ apporte une amélioration statistiquement significative en termes de WER dans tous les cas par rapport au $\omega_j = 1$ utilisé dans (6). L'amélioration relative varie de 12 à 15%.

Outre cette contribution mineure, la seconde architecture que nous proposons repose sur l'utilisation des DNNs pour l'estimation des paramètres spectraux *et* spatiaux avec d'éventuelles mises à jour spatiales supplémentaires par EM pondéré ci-dessus. Dans la pratique, les DNNs spatiaux estiment des matrices triangulaires inférieures, qui sont la décomposition de Cholesky des matrices de covariance spatiale des sources.

Nous avons expérimenté différentes fonctions de coût dans ce contexte, différentes architectures et différents paramètres d'entrée pour ces DNNs spatiaux. Nous avons également proposé une nouvelle fonction de coût, qui combine MSE et la distance cosinus (CD), et qui est supérieure à l'une et l'autre utilisées seules. Nous avons également expérimenté différentes architectures et différents paramètres d'entrée pour les DNN spatiaux, mais nous n'avons pas trouvé une architecture ou un paramètre d'entrée qui apporte une amélioration statistiquement significative par rapport aux autres: dans tous les cas, le système proposé offre des performance très satisfaisantes à un coût assez faible en terme de calculs une fois l'apprentissage effectué. Nous avons également montré que des mises à jour spatiales itératives sont toujours utiles et complémentaires pour améliorer les performances.

Enfin, le Tableau E.4 montre les performances de reconnaissance vocale pour différents nombres de canaux par rapport à la formation de voie à valeurs propres généralisée (GEV) invariante dans le temps incluant un post-traitement de normalisation aveugle (BAN) tel que décrit dans Heymann et al. [2017]. Cette méthode définit l'état de l'art sur les données CHiME. Pour chaque tâche, nos systèmes emploient deux DNNs spectraux et deux DNNs spatiaux. Tous les DNNs suivent une architecture RNN basée sur BLSTM. Sur le jeu de test, la performance des systèmes que nous proposons — sans itération spatiale EM supplémentaire — est statistiquement similaire à celle de GEV-BAN avec 6 canaux ou 4 canaux, mais significativement moins bonne avec 2 canaux. Les performances de nos systèmes *avec* les mises à jour itératives supplémentaires par EM sont toujours meilleures que celles de GEV-BAN pour tous les nombres de canaux. La différence n'est cependant statistiquement significative que pour 6 canaux. Grâce à ces systèmes, nous pouvons obtenir des diminutions relatives de 14%, 7% et 1% du WER par rapport à GEV-BAN sur les tâches à 6, 4 et 2 canaux respectivement.

Table E.4: Comparaison des performance de reconnaissance en terme de WER (%) pour différents nombres de canaux par rapport à la formation de voie GEV-BAN. L'évaluation a été faite sur le jeu de données réelles. Les chiffres en gras indiquent la meilleure performance pour chaque paire de nombres de canaux et de jeu de données. Avec 6 canaux, les intervalles de confiance à 95% pour les deux meilleurs WER sont ± 0,25 % pour le jeu de développement et ± 0,34% pour le jeu de test. Avec 4 canaux, ce sont ± 0.26% et ± 0.37%. Avec 2 canaux, ce sont ± 0.30% et ± 0.46%. Un WER plus faible est préférable.

Système de rehaussement		**Dév.**	**Test**
ID	**Description**		
	6 canaux		
(1)	Initialisation spectrale avec $\mathrm{DNN}_0^{\mathrm{spec}}$	7.17	14.03
(2)	**(1)** + formation voie GEV-BAN	5.37	8.15
(3)	**(1)** + Initialisation spatiale avec $\mathrm{DNN}_0^{\mathrm{spat}}$ + MÀJ spectrale avec $\mathrm{DNN}_1^{\mathrm{spec}}$ + MÀJ spatiale avec $\mathrm{DNN}_{1,1}^{\mathrm{spat}}$	4.81	8.33
(4)	**(3)** + MÀJ spatiale jusqu'à convergence	**4.59**	**6.97**
	4 canaux		
(5)	Initialisation spectrale avec $\mathrm{DNN}_0^{\mathrm{spec}}$	7.30	13.91
(6)	**(5)** + formation voie GEV-BAN	5.51	9.15
(7)	**(5)** + Initialisation spatiale avec $\mathrm{DNN}_0^{\mathrm{spat}}$ + MÀJ spectrale avec $\mathrm{DNN}_1^{\mathrm{spec}}$ + MÀJ spatiale avec $\mathrm{DNN}_{1,1}^{\mathrm{spat}}$	5.07	9.12
(8)	**(7)** + MÀJ spatiale jusqu'à convergence	**4.87**	**8.47**
	2 canaux		
(9)	Initialisation spectrale avec $\mathrm{DNN}_0^{\mathrm{spec}}$	8.55	17.39
(10)	**(9)** + formation voie GEV-BAN	6.94	13.62
(11)	**(9)** + Initialisation spatiale avec $\mathrm{DNN}_0^{\mathrm{spat}}$ + MÀJ spectrale avec $\mathrm{DNN}_1^{\mathrm{spec}}$ +MÀJ spatiale avec $\mathrm{DNN}_{1,1}^{\mathrm{spat}}$	7.19	14.78
(12)	**(11)** + MÀJ spatiale jusqu'à convergence	**6.63**	**13.48**

F Conclusion et perspectives

Nous avons proposé deux systèmes de séparation multicanale de sources audio basés sur les DNNs et qui exploitent le modèle gaussien multicanal de l'état de l'art en modélisation probabiliste audio. Selon ce modèle, les sources sont caractérisées par leurs paramètres spectraux, c'est-à-dire leurs densités spectrales de puissance, et leurs paramètres spatiaux, c'est-à-dire leurs matrices de covariance spatiale, qui encodent les corrélations entre tous les canaux pour chaque fréquence. Dans le premier système, les paramètres spectraux sont modélisés par des DNNs et les paramètres spatiaux sont estimés itérativement comme dans le cadre probabiliste classique. Dans le second système, les paramètres spectraux et spatiaux sont tous modélisés par des DNNs, avec une utilisation optionnelle finale de l'algorithme EM issu de la modélisation probabiliste. Malgré quelques différences dans les détails, nous pouvons dire que ce deuxième système englobe le premier. Pour conclure, nous pouvons affirmer que la principale contribution de cette étude est un cadre unifié pour la séparation de sources audio multicanale basé à la fois sur l'utilisation de DNNs et d'un modèle probabiliste gaussien.

Nous avons étudié en profondeur l'impact des divers choix de conception des systèmes proposés. Pour l'aspect itératif, nous avons envisagé de multiples estimations des paramètres spatiaux pour chaque estimation de paramètres spectraux, ainsi que l'utilisation de DNNs différents pour différentes itérations. Pour les DNNs spectraux, nous avons considéré différentes fonctions de coût, différentes représentations temps-fréquence, différentes architectures de DNNs et différentes tailles de données d'apprentissage. Pour les DNNs spatiaux, nous avons considéré différentes fonctions de coût, différentes architectures et différents paramètres d'entrée. De plus, nous avons présenté une formule d'estimation pondérée des paramètres spatiaux, qui est une généralisation de celle du cadre EM, et dont on a montré qu'elle améliorait les performances.

En se basant sur l'étude présentée dans cette thèse, plusieurs orientations futures pourraient être envisagées. Les orientations à court et moyen terme comprennent l'apprentissage intégrée des DNNs spectraux et spatiaux, l'évaluation sur d'autres jeux de données et/ou tâches, la séparation de sources mobiles et la séparation en temps réel. Les orientations à moyen et long terme comprennent le rehaussement de la parole intégrant la déréverbération et l'annulation d'écho, la séparation de sources multiples d'un même type et la séparation avec des réseaux de microphones distribués.

CHAPTER 1

Introduction

In this chapter, we describe the motivation of our study by giving an overview of audio source separation and its increasing importance for modern real-world applications. We then define our objectives and present the organization of this thesis.

1.1 Motivation

1.1.1 Audio source separation

It is common for most people to have a good conversation with friends at the terrace of a coffee shop while overhearing the background ambient music played by the shop, the conversation in the other table nearby, the babbling sound from a crowd passing on the sidewalk, and the motor sound from the vehicles on the street. In this situation, a human listener may be able to focus on the music or even further, its guitar riffs. This capability of focusing and listening to a particular sound while overhearing many other interfering sounds seems effortless for people with normal hearing and good concentration, but it is not the case for others, especially elderly people.

This task is also a very challenging one for machines and translates into the *audio source separation* problem. Still considering the above scenario at the terrace of a coffee shop, let us imagine that someone is recording a video of a presenter talking about the shop. The audio track of this video captures not only the presenter's speech but also the other sounds mentioned above, which might be unwanted in the final version of the video. Given this audio track, an audio source separation method may be used during the video editing process to distinguish the presenter's speech from the other unwanted sounds, to preserve all sounds but the motor sound from the street, or for other purposes. Thus, in general, audio source separation aims to recover one or more underlying audio source signals from an observed audio mixture signal [Makino et al., 2007; Naik & Wang, 2014; Vincent et al., 2017a].

1.1.2 Speech and music separations

Audio source separation is a core problem in the audio signal processing community attracting a substantial attention from many researchers in the last decades [Ewert et al., 2014; Rivet et al., 2014; Vincent et al., 2014]. Most of the existing related research focuses on either *speech separation* or *music separation*, although it does not mean that the developed methods are always only applicable to one of these two areas.

Speech separation aims to recover a speech signal from a mixture containing multiple background noise sources with possibly interfering speech. The primary applications include *speech enhancement* [Loizou, 2007] and *automatic speech recognition (ASR)* [Rabiner & Juang, 1993]. Both applications demand a speech signal with as few unwanted sounds (also called *noise* or *interference*) as possible, but there is a subtle difference between them. Speech separation for enhancement targets a human as the user. In this case, the separated speech may contain more interference but should have less *distortion* to provide speech that sounds natural and fewer *artifacts*, which are other unwanted sounds induced by the separation process, to provide a good listening comfort [Doclo et al., 2015]. Both distortion and artifacts are caused by imperfect separation processes. Speech enhancement is employed in hearing aids, mobile phones, teleconference systems, etc. On the other hand, speech separation for ASR targets a machine as the 'listener', in which the reductions of interference, distortion, and artifact may have different importance than for a human. Additionally, the term 'speech enhancement' is also commonly used for this ASR-oriented speech separation [Deng & O'Shaughnessy, 2003, chap. 13]. We use this convention in this thesis, especially when we are not discussing an application unique to human perceptual listening or ASR. Furthermore, the terms 'speech enhancement' and 'speech separation' are used interchangeably. Notable commercial products include VoCon® Speech Signal Enhancement (SSE) by Nuance[1] and BeClear Speech Enhancement by Philips[2].

While speech is the sole target in almost all cases of speech separation, the targets of music separation vary from one application to another. Music

[1]See `http://www.nuance.com/mobile/speech-recognition-solutions/vocon-sse`

[2]See `http://www.ip.philips.com/licensing/program/114`

separation may aim to recover a singing voice from a mixture containing music accompaniment, e.g., a pop song. Besides the singing voice, it may aim to recover the musical instrument stems, e.g., piano, guitars, and drums, from the same mixture. Given the separated sources, the derived applications include music editing (e.g., remixing the instruments, up-mixing the recording to 3D sound formats) and music information retrieval (e.g., transcribing the melody of a particular instrument). There exist commercial products employing source separation in this context, including ADX TRAX by Audionamix[3], UNMIX::DRUMS by Zynaptiq[4], Melodyne by Celemony[5], and R-MIX by Roland[6].

An overview of research on both speech and music separation will be presented in the next chapter. It is worth mentioning that there is also a limited body of research focusing on other sounds beyond speech and music, e.g., movie soundtracks [Liutkus & Leveau, 2010], bird songs [Potamitis, 2008], and marine mammal vocalizations [Gur & Niezrecki, 2009].

1.1.3 Single-channel and multichannel separation

The observed audio mixture is in the form of either a *single-channel signal*, e.g., a speech recording acquired with a single microphone or a mono music recording (which was common in the past), or a *multichannel signal*, e.g., a speech recording acquired with two (or more) microphones or a stereo music recording. If we consider the case of speech recording in real-world environments, recording using a single microphone and performing single-channel separation on the acquired signal is practical. However, technological advances and the competitive market provide an increasing number of affordable portable recording types of equipment with multiple microphones. Thus, it is getting easier than before to obtain multichannel speech recordings. These recordings capture additional information which is valuable for separation. As an analogy, humans can predict the direction from which sounds come based on the amplitude and phase differences between the signals captured by the left and right ears. These inter-channel differences are correlated with the position of the sound sources relative to the ears (for humans) or the microphones (for machines). These differences can be exploited in multichannel separation to provide better results than

[3]See `https://audionamix.com/technology/adx-trax`
[4]See `http://www.zynaptiq.com/unmixdrums`
[5]See `http://www.celemony.com/en/melodyne`
[6]See `https://www.roland.com/us/products/r-mix`

single-channel separation [Brandstein & Ward, 2001; Benesty et al., 2008; Kumatani et al., 2012; Doclo et al., 2015; Gannot et al., 2017]. If we consider the application of separation to music recordings, multichannel separation is also preferable since most professionally-produced recordings available nowadays are in stereo (two-channel) format. Finally, if we consider the application to the soundtracks of music concerts or films, we will deal with 3D sound formats, which feature six or eight channels.

1.1.4 Deep neural networks (DNNs)

Studies in the last decade have shown that deep neural networks (DNNs), including recurrent neural networks (RNNs), can model complex functions and perform well on various tasks [Deng, 2014; LeCun et al., 2015; Schmidhuber, 2015; Juang, 2016; Goodfellow et al., 2016], including audio signal processing [Yu & Deng, 2011; Wang, 2017] and ASR [Hinton et al., 2012; Yu & Deng, 2015]. At the beginning of 2015 when our study was started, DNNs had also been applied to single-channel speech separation and music separation, especially singing voice separation [Weninger et al., 2014; Huang et al., 2015]. There were also studies about exploiting the available multichannel data, but they were limited to extracting representations, also known as *features*, from this data to estimate a single-channel separation filter [Jiang et al., 2014; Araki et al., 2015]. As a result, these studies do not fully exploit the benefits of multichannel data as achieved by multichannel filtering. Since then, there have been more studies on using DNNs for both single-channel and multichannel separation, although the single-channel case is still more popular. For the multichannel case, there have been studies on employing DNNs for doing *beamforming* [Kumatani et al., 2012]. The DNNs are used either (1) for implicitly estimating a *time-invariant beamformer* in the context of robust ASR joint training framework [Xiao et al., 2016; Sainath et al., 2017] or (2) for estimating a single-channel filter, known as a *mask*, for each channel and use the estimated filters to derive a *time-invariant beamformer* [Heymann et al., 2016; Erdogan et al., 2016; Wang et al., 2017]. Both approaches have shown to perform well for speech enhancement. However, time-invariant filtering might not be suitable for mixtures with many sources, whose mixing and stationarity cannot be assumed, such as a song with its vocals and different musical instruments. In our study, the DNNs are used to estimate the spectral and spatial parameters of each source and use the estimated parameters to derive a *time-varying multichannel filter*. Further discussions about these studies and the positioning of our study with respect to these study are presented in the next chapter.

1.2 Objectives and scope

The main objective of our study is **to explore and optimize the use of DNNs for multichannel audio source separation using time-varying multichannel filtering**. In order to do so, we want **to use the state-of-the-art multichannel source separation framework as the basis**. Consequently, we need **to investigate how DNNs should work in this probabilistic modeling framework**. The investigation includes what parameters of the probabilistic model the DNNs should estimate and how the DNNs should learn to estimate these parameters. We also need **to check whether the modeling by DNNs complies with the probabilistic point-of-view**. We consider both speech and music separation. For speech separation, we consider a speech enhancement task for ASR. For music separation, we consider both singing voice and musical instrument separation. Therefore, we need **to assess the system performance in terms of source separation metrics and also speech recognition metric**, when it is applicable. In order to increase the reproducibility of the experiments, we want **to evaluate the performance on publicly available datasets**. We use the datasets from the CHiME Speech Separation and Recognition Challenges (CHiMEs)[7] for the speech enhancement task and the Signal Separation Evaluation Campaigns (SiSECs)[8] for the music separation task. Using these datasets allows us **to be active in our research community activities**, especially when the datasets are used in the context of the challenges (by following the rules). In this case, comparison to other methods can be done easily and, most importantly, fairly.

What we want to recover is not exactly the sounds emitted by each source, but its contribution to the mixture signal, known as *source spatial image*. For speech recordings in real-world environments, the spatial image is the speech signal recorded at the microphones after propagating from the speaker's mouth to the microphones in the absence of noise. Estimating the sound produced at the speaker's mouth from this spatial image is yet another research topic called *dereverberation*, which is beyond the scope of our study. Besides, we assume that the source types are known. For speech

[7]See `http://spandh.dcs.shef.ac.uk/chime_challenge/`
[8]See `https://sisec.inria.fr/`

separation, we consider the simplest case, i.e., the mixture is composed of 'speech' and 'noise'. The 'noise' source in our model typically consists of several physical sound sources. For music separation, the mixture is composed of 'vocals', 'bass', 'drums', and 'others'. Similarly, we group all instruments which compose a drum set as 'drums' disregarding the fact that these instruments (snare drum, bass drum, cymbals, etc.) are separate physical sound sources. Other instruments which are not included in 'bass' and 'drums' are grouped and labeled as 'others'. Finally, we do not consider the problem of *multi-speaker speech separation*, which aims to separate the speech of one speaker from that of other speakers.

1.3 Contributions and organization of the thesis

Our contributions are discussed and organized in the following chapters as follows.

Chapter 2 presents the mathematical formulation of audio source separation and introduces the notations used in this thesis. It is followed by an overview of state-of-the-art separation methods, including the classical expectation-maximization (EM) based multichannel source separation framework [Duong et al., 2010a] that uses a multichannel Gaussian model, in which each source is characterized by its spectral and spatial parameters. This framework will be the basis for the two DNN based frameworks presented in Chapters 3 and 5. This chapter also presents the basics of ASR and DNNs, especially those which are relevant to our study. Additionally, this chapter briefly discusses the performance metrics which include source separation and speech recognition metrics.

Chapter 3 describes the first DNN based multichannel audio source separation framework we proposed in our study. In this framework, the spectral parameters, i.e., the source power spectral densities (PSDs), are modeled by DNNs and the spatial parameters, i.e., the source spatial covariance matrices, are estimated iteratively as in the classical EM based framework presented in Chapter 2. A time-varying multichannel Wiener filter is then derived for each source using these source spectral and spatial parameters. To the best of our knowledge, this was the first DNN based multichannel audio source separation framework ever published[9]. Extensive experiments on a multichannel speech enhancement problem have been done to assess different design choices and their impact on the performance. In this chapter, we consider several design choices that are related to the iterative aspect of the framework. The design choices notably include multiple spatial parameter updates after spectral parameter initialization and the use of multiple DNNs for estimating the spectral parameters at different iterations. Experiments show that doing multiple spatial parameter updates before a spectral parameter update provides better performance than alternating one spatial parameter update and one spectral parameter update, as in the classical EM framework. Experiments also show that different DNNs can be used for the spectral parameter update of different iterations to improve the overall performance. Finally, we show that the proposed DNN based framework outperforms multichannel non-negative matrix factorization (NMF) [Ozerov et al., 2012], which achieved state-of-the-art performance for the considered CHiME-3 dataset [Barker et al., 2015] before the emergence of deep learning, in terms of source separation and speech recognition metrics.

Chapter 4 explores ways to improve the spectral DNNs used in the framework presented in Chapter 3. In addition to the speech enhancement problem, in this chapter, we also consider a multichannel vocals-accompaniment separation problem in music. We mainly study the impact of various general-purpose and task-oriented cost functions. This notably includes the probabilistically-motivated Itakura-Saito (IS) divergence and a newly proposed task-oriented signal-to-distortion ratio (SDR) cost function. Experiments show that the Kullback-Leibler (KL) divergence is the most reasonable choice because it provides the best source separation performance and works well in terms of speech recognition performance. Meanwhile, the IS divergence provides the best speech recognition performance, but its source separation performance falls behind the others. Unfortunately, the task-oriented SDR cost function only improves the average vocals-accompaniment separation performance to some extent compared to the simple mean squared error (MSE). To our knowledge, this is one of the first uses of task-oriented discriminative training in a multichannel scenario. We then briefly study the use of perceptually-motivated equivalent rectangular bandwidth (ERB)

[9]See: Nugraha, A. A., Liutkus, A., & Vincent, E. (2015, June). *Multichannel audio source separation with deep neural networks*. Research Report RR-8740, INRIA. `https://hal.inria.fr/hal-01163369v1`.

time-frequency representations and the impact of DNN architectures, in which we consider a bidirectional long short-term memory (LSTM) based RNN architecture, in addition to the multilayer perceptron architecture we used in the beginning. We also consider data augmentation approaches to increase the DNN training data. Finally, it is worth mentioning that our vocals-accompaniment separation systems perform remarkably close in terms of overall distortion to the DNN based system of Uhlich et al. [2017], which achieves state-of-the-art performance in the context of the professionally-produced music recordings task of SiSEC 2016 [Liutkus et al., 2017]. Our systems even significantly better in terms of signal-to-interference ratio achieved.

Chapter 5 presents a weighted spatial parameter estimation formula and describes the second DNN based multichannel audio source separation framework, which is an extension of the first framework discussed in Chapter 3. In this framework, both the spectral and the spatial parameters are initialized and updated by DNNs with a possible addition of the iterative EM spatial parameter updates. We show that DNNs can provide a good estimation of the spatial parameters, i.e., the source spatial covariance matrices, by exploiting the Cholesky decomposition of the Hermitian positive-definitive covariance matrices. For these spatial DNNs, we experimented with different cost functions and different architectures. We also show that the iterative EM spatial parameter updates are still useful to improve the source spatial covariance matrices. Finally, this chapter presents a performance comparison between our proposed systems and the generalized eigenvalue (GEV) beamformer with blind analytic normalization (BAN) of Heymann et al. [2017], which achieves state-of-the-art performance for the CHiME-4 dataset [Vincent et al., 2017b]. Considering the real test set only, the performance of our proposed systems is always better than the GEV-BAN beamformer for all 6-channel, 4-channel, and 2-channel tasks, although the performance differences are not always statistically significant.

Chapter 6 concludes this thesis by summarizing the achievements accomplished in our study and presenting some future directions.

CHAPTER 2

Background

In this chapter, we formulate the problem of audio source separation mathematically and introduce the notations used in our study. This is followed by brief discussions on automatic speech recognition, deep neural networks, and the datasets used in the experiments. An overview of state-of-the-art audio source separation methods is then presented. Additionally, the considered performance measures, namely source separation metrics and a speech recognition metric, are described in the corresponding discussions.

2.1 Audio source separation

2.1.1 Sources and mixture

From a physical point of view, sound sources are typically divided into two types: *point sources*, whose sound comes from a single point in a space, and *diffuse sources*, whose sound comes from a region of space and can be regarded as a collection of point sources [Vincent et al., 2017a, chap. 1]. Point sources include a human speaker, a piano string, a water drop, etc. A singer is seen by the microphones as a point source when he/she is singing alone. But, when he/she is singing in a choir, he/she becomes part of a diffuse source. Similarly, a water drop in a kitchen sink is seen as a point source. But, water drops as parts of rain create a diffuse source.

From a perceptual point of view, we may define sound sources as they are perceived by human listeners. This has been studied in the subfield of psychoacoustics. It is called *auditory scene analysis*. It is a process that segregates the sounds reaching human ears into mental representations called *auditory streams* via segmentation and grouping mechanisms [Bregman, 1990]. The segmentation mechanism decomposes the sounds into time and frequency localized components and the grouping mechanism combines these components such that they are perceptually organized. It allows humans to listen to a piece of music as a whole, to the drums within that music, or to the distinctive cymbal sounds from the drums.

As mentioned earlier in Section 1.2, in our study, audio source separation aims to recover the *source spatial images* from the *mixture*, which is assumed to be composed of two or more spatial images. Each source spatial image is defined as the contribution of one source to the observed mixture [Vincent et al., 2006]. It is closely related to the concept of auditory stream from auditory scene analysis. This concept also justifies the grouping of some sources, e.g., group all instruments which compose a drum set as 'drums' disregarding the fact that these instruments (snare drum, bass drum, cymbals, etc.) are separate physical sound sources. This is equivalent to grouping multiple point sources into a diffuse source.

The way sources are transformed into source images, which then compose a recording, may vary. Broadly speaking, we may categorize recordings into naturally-mixed and artificially-mixed recordings. For *naturally-mixed recordings*, this transformation results from the propagation of sound through air from the source to the microphones. The propagation is affected by various factors, including the source-microphone distance, the location of recording (indoor or outdoor), the size of the room, and the sound absorption in the room [Kuttruff, 2014]. For *artificially-mixed recordings*, the transformation strongly depends on what the sound engineer wants to achieve since there exist various audio mixing techniques and they are highly adjustable [Senior, 2011]. Naturally-mixed recordings can be used as sources in artificially-mixed ones.

Figure 2.1 illustrates a studio music production where the vocals, the guitar, the violin, and the drums are separately recorded and then mixed together to compose a song. The inputs of the mixing process are either naturally-mixed or artificially-mixed recordings, and the output is an artificially-mixed recording. The resulting stereo song consists of the sum of the stereo spatial images of the vocals, the guitar, the violin, and the drums. These stereo spatial images are not necessarily the original recordings. The sound engineers are able to modify and mix the original sources by panning (distributing few channels into more channels) or down-mixing (combining several channels into fewer channels). Therefore, the number of channels of the spatial images and the mixture may be chosen freely. By contrast, in the cases of speech recording in a real-world environment or live music recording, the number of channels of the spatial images and the mixture corresponds to the number of microphones, as for the drums in Figure 2.1. In these cases, each channel

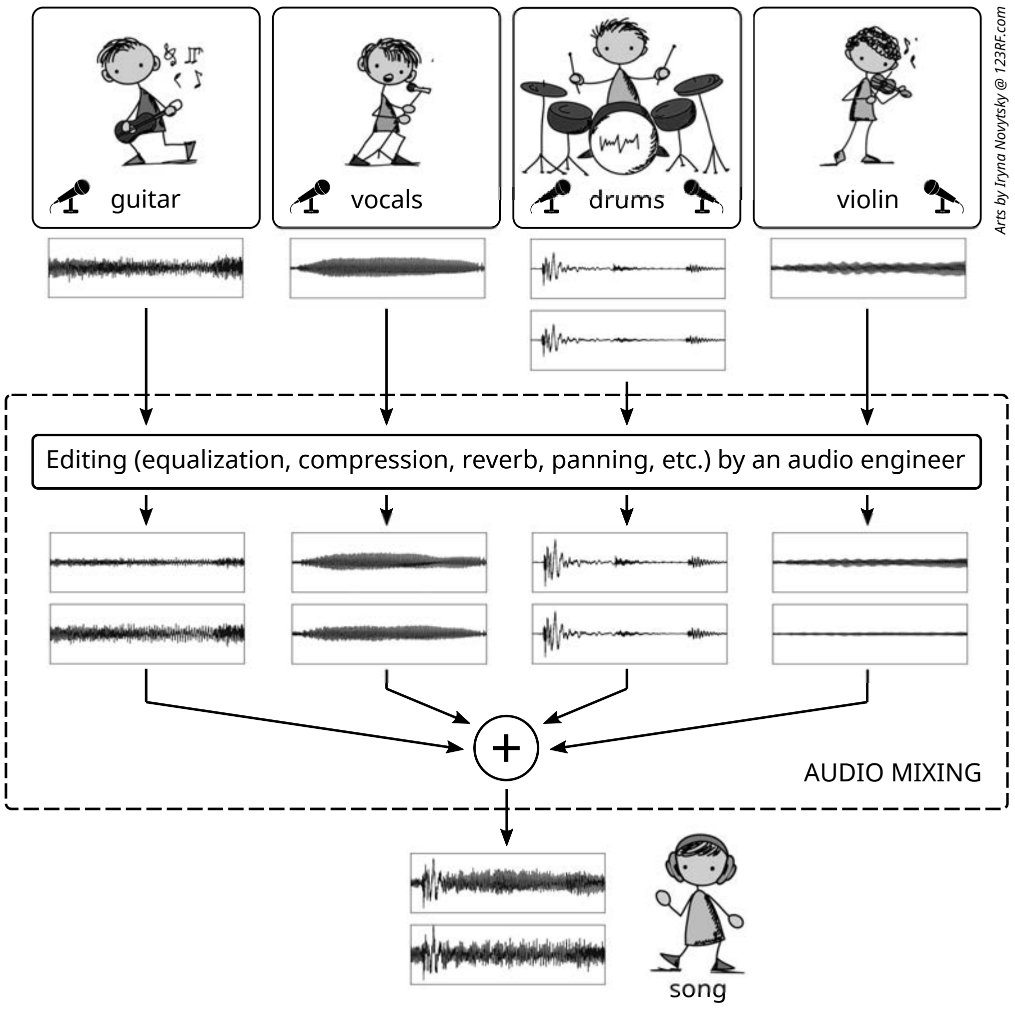

guitar
vocals
drums
violin
Arts by Iryna Novytsky @ 123RF.com
Editing (equalization, compression, reverb, panning, etc.) by an audio engineer
+
AUDIO MIXING
song

Figure 2.1: Illustration of a studio music production.

of a source spatial image is the signal captured by each microphone after propagating from the corresponding source.

It also worth mentioning that, in our study, a mixture is defined as a linear combination of the source images, as shown by Figure 2.1. In a music production, non-linear post-processing can be applied on the generated mixture. Conveniently, this kind of mixture can be viewed as a linear combination of source images, on which non-linear processing has been applied [Sturmel et al., 2012].

2.1.2 Source separation

Let us now formalize these concepts and introduce the notations we use in the following discussions. Let I and J denote the number of channels and sources, respectively, with their indexes $i \in \{1, 2, \ldots, I\}$ and $j \in \{1, 2, \ldots, J\}$ to indicate specific channel i and source j. Additionally, we may use $j \in \{\mathrm{S}, \mathrm{N}\}$ to indicate speech or noise, respectively. The observed I-channel mixture, denoted by $\mathbf{x}(t) \in \mathbb{R}^{I \times 1}$, is composed of two or more source spatial images, denoted by $\mathbf{c}_j(t) \in \mathbb{R}^{I \times 1}$:

$$\mathbf{x}(t) = \sum_{j=1}^{J} \mathbf{c}_j(t). \tag{2.1}$$

The mixture and source spatial images are time-domain digital signals indexed by $t \in \{0, 1, \ldots, T-1\}$, where T is the length of the signal. Multichannel audio source separation aims to recover the multichannel source spatial images $\mathbf{c}_j(t) = [c_{1j}(t),\, c_{2j}(t),\, \ldots,\, c_{Ij}(t)]^\top$ from the observed multichannel mixture signal $\mathbf{x}(t) = [x_1(t),\, x_2(t),\, \ldots,\, x_I(t)]^\top$, where $\cdot^\top$ denotes matrix or vector transposition. This definition applies to both point and diffuse sources in both naturally-mixed and artificially-mixed recordings.

The performance of audio source separation can be evaluated using subjective or objective assessment [Emiya et al., 2011; Cano et al., 2016]. Subjective assessment can be done via listening tests by following a methodology called MUSHRA [ITU, 2015] or one of its variants [Emiya et al., 2011; Cartwright et al., 2016]. Objective assessment is done by computing performance metrics, such as BSS Eval [Vincent et al., 2006], which is a set of quadratic error based metrics, or PEASS [Emiya et al., 2011], which is a set of perceptual model based metrics.

In our study, we use the metrics given by the variant of BSS Eval for source spatial images [Vincent et al., 2007]. This variant decomposes each channel of an estimated source image $\widehat{\mathbf{c}}_j(t) = [\widehat{c}_{1j}(t),\, \widehat{c}_{2j}(t),\, \ldots,\, \widehat{c}_{Ij}(t)]^\top$ as

$$\widehat{c}_{ij}(t) = c_{ij}(t) + e_{ij}^{\text{spat}}(t) + e_{ij}^{\text{interf}}(t) + e_{ij}^{\text{artif}}(t), \tag{2.2}$$

where $c_{ij}(t) \in \mathbb{R}$ is channel i of the true source image, while $e_{ij}^{\text{spat}}(t) \in \mathbb{R}$, $e_{ij}^{\text{interf}}(t) \in \mathbb{R}$, and $e_{ij}^{\text{artif}}(t) \in \mathbb{R}$ are different error components representing the spatial distortion, the interference, and the artifacts, respectively, for the corresponding channel of the estimated source image $\widehat{c}_{ij}(t) \in \mathbb{R}$.

Based on this decomposition, four different metrics, i.e., the signal-to-distortion ratio (SDR), the source-image-to-spatial-distortion ratio (ISR), the signal-to-interference ratio (SIR), and the signal-to-artifacts ratio (SAR), are computed as follows:

$$\mathrm{SDR}_j = 10\log_{10}\frac{\sum_{i=1}^{I}\sum_{t=1}^{T} c_{ij}(t)^2}{\sum_{i=1}^{I}\sum_{t=1}^{T}\left(\widehat{c}_{ij}(t) - c_{ij}(t)\right)^2}, \tag{2.3}$$

$$\mathrm{ISR}_j = 10\log_{10}\frac{\sum_{i=1}^{I}\sum_{t=1}^{T} c_{ij}(t)^2}{\sum_{i=1}^{I}\sum_{t=1}^{T} e_{ij}^{\text{spat}}(t)^2}, \tag{2.4}$$

$$\mathrm{SIR}_j = 10\log_{10}\frac{\sum_{i=1}^{I}\sum_{t=1}^{T}\left(c_{ij}(t) + e_{ij}^{\text{spat}}(t)\right)^2}{\sum_{i=1}^{I}\sum_{t=1}^{T} e_{ij}^{\text{interf}}(t)^2}, \tag{2.5}$$

$$\mathrm{SAR}_j = 10\log_{10}\frac{\sum_{i=1}^{I}\sum_{t=1}^{T}\left(c_{ij}(t) + e_{ij}^{\text{spat}}(t) + e_{ij}^{\text{interf}}(t)\right)^2}{\sum_{i=1}^{I}\sum_{t=1}^{T} e_{ij}^{\text{artif}}(t)^2}. \tag{2.6}$$

These four metrics are expressed in decibels (dB) and higher metric values indicate better performance. As indicated by (2.3), the SDR can be regarded as an overall metric.

This study considers the applications of audio source separation for music separation and speech enhancement. Speech enhancement is important to provide an automatic speech recognition (ASR) system some robustness, e.g., to noise or reverberation. As its name suggests, speech enhancement aims to enhance the target speech and attenuate the non-speech sounds. This can be viewed as a source separation problem where a noisy speech need to be separated into the target speech and the other sounds. The latter may then be discarded. The following section presents a brief description of ASR system and the evaluation metric we use in our study.

2.2 Automatic speech recognition (ASR)

Figure 2.2 shows the block diagram of a typical ASR system with optional enhancement blocks [Rabiner & Juang, 1993; Rabiner & Schafer, 2007; Gales & Young, 2008; Virtanen et al., 2012]. The basic flow involves the extraction of feature vectors from the input speech signal (in the feature extraction step) and the search for the most likely word sequence given these features (in the decoding step). The decoding step can be expressed as

$$\widehat{\mathbf{w}} = \arg\max_{\mathbf{w}} P\left(\mathbf{w}|\mathbf{v}\right), \tag{2.7}$$

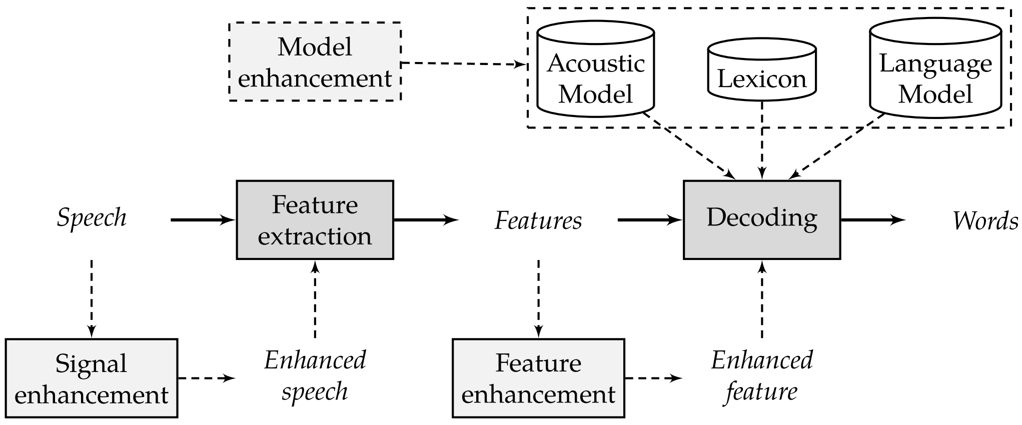


Figure 2.2: Block diagram of ASR. The solid arrows show the basic ASR flow, while the dashed arrows show an additional flow for robust ASR.

where $\widehat{\mathbf{w}}$ and $\mathbf{w}$ are the estimated and the hypothesized word sequences; $\mathbf{v}$ is the sequence of feature vectors; and $P(\mathbf{w}|\mathbf{v})$ is the probability of a word sequence $\mathbf{w}$ given the sequence of feature vectors $\mathbf{v}$. Following Bayes' rule, this formula leads to

$$\widehat{\mathbf{w}} = \arg\max_{\mathbf{w}} P(\mathbf{v}|\mathbf{w})\, P(\mathbf{w})\,, \tag{2.8}$$

where the likelihood $P(\mathbf{v}|\mathbf{w})$ is computed from an acoustic model and the prior $P(\mathbf{w})$ is computed from a language model. The language model models word sequences and the acoustic model typically models small speech sound units called phonemes. In order to link the acoustic model and the language model, a lexicon, also known as a pronunciation dictionary, is used. The lexicon contains a list of words and the variations of pronunciation, expressed in phonemes, of these words.

There exist various advanced techniques, including enhancement techniques, for different components of a robust ASR system [Virtanen et al., 2012; Li et al., 2014].

Signal related techniques works in the time or time-frequency domain of the single-channel or multichannel observed noisy speech [Benesty et al., 2005]. In general, these techniques are known as speech enhancement techniques, although they may involve noise reduction, dereverberation, or echo cancellation. The techniques include spectral subtraction [Boll, 1979], Wiener filtering [Lim & Oppenheim, 1979], non-negative matrix factorization (NMF) [Lee & Seung, 1999, 2000], and beamforming with post-filtering [Brandstein & Ward, 2001; Benesty et al., 2008]. Some of these methods are further discussed later in this chapter.

Feature related techniques include robust features, feature normalization, and feature enhancement. The most commonly used basic features are mel-frequency cepstral coefficients [Davis & Mermelstein, 1980] and perceptual linear predictive coefficients [Hermansky, 1990]. In order to compensate the noise, normalization techniques can be applied. These techniques may include cepstral mean normalization [Atal, 1974], cepstral mean and variance normalization [Viikki & Laurila, 1997], linear discriminant analysis [Izenman, 2008], maximum likelihood linear transform [Gopinath, 1998], and feature-space maximum likelihood linear regression [Gales, 1998]. Features can also be improved by enhancement techniques, including SPLICE [Deng et al., 2001], ALGONQUIN [Frey et al., 2001], and various neural network based methods [Ishii et al., 2013; Wöllmer et al., 2013; Nugraha et al., 2014; Himawan et al., 2015; Fujimoto & Nakatani, 2016]. Finally, there exist various robust features, including some which are motivated by human auditory properties [Stern & Morgan, 2012].

Model related techniques include advanced models, model training techniques, and model adaptation techniques. The state-of-the-art for acoustic modeling is the so-called DNN-HMM, in which deep neural network (DNN), as opposed to the more classical Gaussian mixture model, provides the posterior probabilities of the hidden Markov model states [Gales & Young, 2008; Hinton et al., 2012; Yu & Deng, 2015]. Both GMM-HMM and DNN-HMM models may be trained on either noisy or enhanced data from several different environments (known as multi-condition training) [Lippmann et al., 1987]. For further improvement, training can be performed by employing discriminative criteria, such as maximum mutual information and minimum Bayes' risk [Bahl et al., 1986; Goel & Byrne, 2000; Veselý et al., 2013]. Model adaptation techniques may then be utilized to compensate the difference between training and test conditions. These techniques include speaker adaptive training for GMM-HMM model [Povey et al., 2008] or DNN-HMM model [Miao et al., 2015] and other DNN adaptation techniques [Swietojanski et al., 2016; Samarakoon & Sim, 2016; Karanasou et al., 2017]. Besides, there exist other techniques based on uncertainty concept [Droppo et al., 2002; Liao & Gales, 2005] and missing data concept [Raj & Stern, 2005; Barker, 2012]. The state-of-the-art for language modeling is the so-called RNN-LM, in which recurrent neural network (RNN) is used as opposed to the more classical n-gram models [Rosenfeld, 2000; Mikolov et al., 2010]. For the n-gram models, various smoothing techniques have been studied [Chen & Goodman, 1996], including Kneser-Ney smoothing [Kneser & Ney, 1995].

Beyond the techniques above, there also exist system combination methods, such as recognizer output voting error reduction [Fiscus, 1997] and segmental conditional random field [Zweig & Nguyen, 2010].

In our study, ASR is solely used for the evaluation of the proposed audio source separation frameworks on a speech enhancement task. We employ the baseline system of the 4th CHiME Speech Separation and Recognition Challenge [Vincent et al., 2017b] which implements various state-of-the-art techniques. Concerning the features, the extraction of mel-frequency cepstral coefficients is followed by dimensionality reduction using linear discriminant analysis on concatenated frames, decorrelation using maximum likelihood linear transform, and speaker normalization using feature-space maximum likelihood linear regression. Both Gaussian mixture model and DNN based acoustic models are used. The Gaussian mixture model based acoustic model employs speaker adaptive training and the DNN based acoustic model is trained with the cross entropy criterion followed by state-level minimum Bayes' risk (sMBR) criterion [Yu & Deng, 2015, chap. 8]. The language model is a 3-gram model with 5-gram Kneser-Ney smoothing and rescoring of the lattice, which contains the word sequence hypotheses, using RNN-LM. For further details about this baseline system, see Hori et al. [2015]. The optimization of this system is beyond the scope of our study.

We use the word error rate (WER) as the speech recognition metric. The WER is based on the Levenshtein distance [Levenshtein, 1966], which counts the minimum number of operations (i.e., word insertions, deletions, and substitutions) required to convert the reference sentence into the transcribed one. The WER is computed as

$$\text{WER} = \frac{N_i + N_d + N_s}{N_w} \times 100\%, \tag{2.9}$$

where N_i, N_d, and N_s denote the number of word insertions, deletions, and substitutions, respectively, while N_w is the total number of words in the reference sentence. An insertion occurs when ASR introduces a new word which does not exist in the reference sentence, a deletion occurs when ASR completely fails to recognize a word, and a substitution occurs when ASR mistakenly recognizes a word as another. Lower WER indicates better performance.

As mentioned in the discussion of signal related techniques, we present some source separation techniques, including the ones for speech enhancement, later in this chapter. Before that, let us discuss sound representation in the time-frequency domain, on which most of these techniques work.

2.3 Time-frequency representation

Audio source separation methods typically operate in the time-frequency domain, in which the temporal and spectral characteristics of sound can be jointly represented. Sounds tend to be sparsely distributed in this domain.

The most commonly used time-frequency representation is the *short-time Fourier transform (STFT)* [Allen, 1977; Smith, 2011; Virtanen et al., 2017]. Other representations include the Mel scale [Stevens et al., 1937] and the equivalent rectangular bandwidth (ERB) scale [Glasberg & Moore, 1990] representations. These two representations use different perceptually-motivated nonlinear frequency scales. In this section, we only describe the STFT representation.

STFT analysis refers the computation of the time-frequency representation from the time-domain waveform. It is done by creating overlapping frames along the waveform and applying the *discrete Fourier transform* on each frame. Given channel i of the time domain mixture $x_i(t)$, the signal of frame index $n \in \{0, 1, \dots, N-1\}$ expressed as

$$x_i(t, n) = x_i(t + nH)h_a(t), \quad t \in \{0, 1, \dots, T-1\}, \tag{2.10}$$

where N is the number of frames, H the hop size between adjacent frames, T the frame length, and $h_a(t)$ the analysis window, such as a Hamming or Hanning function.

The application of the discrete Fourier transform on each frame results in the time-frequency representation

$$x_i(f, n) = \sum_{t=0}^{T_f - 1} x_i(t, n) e^{-\jmath 2\pi t f / F'}, \quad f \in \{0, 1, \dots, F = \lceil F'/2 \rceil, \dots, F' - 1\}, \tag{2.11}$$

where f is the discrete frequency bin index and F' is the number of frequency bins corresponding to the number of discrete Fourier transform points, which is typically $F' = T_f$. Since we commonly works on real signals, the spectrum for each frame is Hermitian. Therefore, the so-called negative frequency bins $f \in \{\lceil F'/2 \rceil + 1, \ldots, F' - 1\}$ may be ignored in the computation, since these bins can be constructed from the so-called positive frequency bins $f \in \{0, 1, \ldots, F = \lceil F'/2 \rceil\}$.

STFT synthesis refers to the transformation from the time-frequency representation to the time-domain waveform. It is done by applying the *inverse discrete Fourier transform* on each frame and performing *weighted overlap-add* to obtain the time-domain waveform. The application of the inverse discrete Fourier transform on channel i and source j of the estimated spatial images $\widehat{c}_{ij}(f, n)$ results in

$$\widehat{c}_{ij}(t, n) = \frac{1}{F'} \sum_{f=0}^{F'-1} \widehat{c}_{ij}(f, n) e^{\jmath 2\pi t f / F'}, \quad t \in \{0, 1, \ldots, T_f - 1\}. \tag{2.12}$$

The time-domain signal is then obtained as

$$\widehat{c}_{ij}(t) = \frac{\sum_{n=-\infty}^{\infty} \widehat{c}_{ij}(t - nH_f, n) h_s(t - nH_f)}{\sum_{n=-\infty}^{\infty} h_a(t - nH_f) h_s(t - nH_f)}, \tag{2.13}$$

where $h_s(t)$ is a synthesis window. This window is important to minimize artifacts at the frame boundaries. If the analysis and the synthesis windows are chosen subject to $\sum_{n=-\infty}^{\infty} h_a(t - nH_f) h_s(t - nH_f) = 1, \forall t \in \{0, 1, \ldots, T - 1\}$, (2.13) leads to the weighted overlap-add of Crochiere [1980]. If the analysis and the synthesis windows are set to be the same window, (2.13) leads to the weighted overlap-add of Griffin & Lim [1984]. In our study, we follow the latter by employing a Hamming window as the analysis and the synthesis windows.

The time domain formulation of the mixture as the sum of the source spatial images in (2.1) can be expressed in the time-frequency domain as

$$\mathbf{x}(f, n) = \sum_{j=1}^{J} \mathbf{c}_j(f, n), \tag{2.14}$$

where $\mathbf{x}(f, n) \in \mathbb{C}^{I \times 1}$ and $\mathbf{c}_j(f, n) \in \mathbb{C}^{I \times 1}$ denote the time-frequency representations computed from $\mathbf{x}(t)$ and $\mathbf{c}_j(t)$, respectively, for frequency bin $f \in \{0, 1, \ldots, F\}$ and time frame $n \in \{0, 1, \ldots, N - 1\}$. Thus, audio source separation aims to recover the multichannel source spatial images $\mathbf{c}_j(f,n) = [c_{1j}(f,n),\, c_{2j}(f,n),\, \ldots,\, c_{Ij}(f,n)]^\top$ from the observed multichannel mixture signal $\mathbf{x}(f,n) = [x_1(f,n),\, x_2(f,n),\, \ldots,\, x_I(f,n)]^\top$.

The following two sections present some state-of-the-art techniques for single-channel audio source separation, i.e., estimating $c_j(f,n)$ from $x(f,n)$, and for multichannel audio source separation, i.e., estimating $\mathbf{c}_j(f,n)$ from $\mathbf{x}(f,n)$.

2.4 State-of-the-art single-channel audio source separation

This section presents essential single-channel audio source separation methods, including time-frequency masking, NMF, and various DNN based approaches. Since we consider the single-channel case, i.e., $I = 1$, to be concise, the index i is not shown in the notations.

There are notable methods beyond the ones particularly discussed here, such as factorial hidden Markov model. It relies on a statistical model of the sources [Roweis, 2003; Ozerov et al., 2009]. Each source is modeled by a GMM-HMM trained on the corresponding source data. The models are then used for estimating a time-frequency mask (see Section 2.4.1).

2.4.1 Time-frequency masking

Time-frequency masking, as its name suggests, estimates the spatial images by filtering the time-frequency representation of the mixture using a mask. This can be expressed as

$$\widehat{c}_j(f,n) = \widehat{m}_j(f,n)x(f,n), \tag{2.15}$$

where $\widehat{m}_j(f,n)$ is the mask for frequency bin f and time frame n of source j. Typically, the mask is a real-valued scalar $\widehat{m}_j(f,n) \in \mathbb{R}$ [Ephraim & Malah, 1984], but a complex-valued mask $\widehat{m}_j(f,n) \in \mathbb{C}$ has been studied recently [Williamson et al., 2016]. For the real-valued case, the mask can be categorized as a *binary mask*, $\widehat{m}_j(f,n) \in \{0,1\}$, or a *soft mask* (also known as *ratio mask*), $\widehat{m}_j(f,n) \in [0,1]$.

The best possible binary or soft mask is called the *ideal binary mask* or the *ideal ratio mask*, respectively. Let us consider a speech enhancement scenario with a target speech source $c_{\mathrm{S}}(f,n)$ and a noise source $c_{\mathrm{N}}(f,n)$. The speech ideal ratio mask $m_{\mathrm{S}}^{\mathrm{rat}}(f,n)$ and ideal binary mask $m_{\mathrm{S}}^{\mathrm{bin}}(f,n)$ can be computed as

$$m_{\mathrm{S}}^{\mathrm{rat}}(f,n) = \frac{\|c_{\mathrm{S}}(f,n)\|}{\|x(f,n)\|} \tag{2.16}$$

$$m_{\mathrm{S}}^{\mathrm{bin}}(f,n) = \begin{cases} 1 & \text{if } m_{\mathrm{S}}^{\mathrm{rat}}(f,n) > \text{threshold} \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

where $\|\cdot\|$ is the Euclidean norm. The ideal ratio mask in (2.16) may be regarded as a single-channel Wiener filter, akin to the multichannel Wiener filter in (2.46) whose probabilistic interpretation is presented in Section 2.5.2.1.

Typically, the noise ideal ratio mask or ideal binary mask is simply computed as $m_{\mathrm{N}}^{\{\mathrm{rat,bin}\}}(f,n) = 1 - m_{\mathrm{S}}^{\{\mathrm{rat,bin}\}}(f,n)$. However, it is not necessarily so, e.g., by computing the noise mask $m_{\mathrm{N}}^{\mathrm{bin}}$ similarly to (2.17), but with a threshold that is not reciprocal to that for the speech mask $m_{\mathrm{S}}^{\mathrm{bin}}$. In general, different formulations can be used for different sources, subject to $0 \leq m_j(f,n) \leq 1$ [Gerkmann & Vincent, 2017].

Time-frequency masking is an integral part of *computational auditory scene analysis*. It is an adaptation of auditory scene analysis studies on human audition (see Section 2.1) to machine audition. Its goal is to achieve human audition performance in extracting auditory streams from single- or two-channel recordings [Wang & Brown, 2006, chap. 1]. Therefore, it is not limited to single-channel audio source separation. Following auditory scene analysis, segmentation and grouping mechanisms are adopted in computational auditory scene analysis [Brown & Wang, 2005; Weninger et al., 2017]. The segmentation mechanism covers time-frequency analysis using a perceptually-motivated representation, such as a gammatone filterbank, and feature extraction, which is done on the resulting time-frequency representation. Various features, such as fundamental frequency, harmonicity, and continuity, have been studied [Brown & Cooke, 1994; Wang & Brown, 1999]. Based on these features, the grouping mechanism then associates particular segments of the time-frequency representation to a specific sound source. Since this association map can be seen as a binary mask, the ideal binary mask was suggested to be the objective of computational auditory scene analysis [Wang, 2005, 2008].

It is worth mentioning that studies have shown that the ratio mask provides better perceptual quality than the binary mask [Jensen & Hendriks, 2012; Madhu et al., 2013; Koning et al., 2015]. This finding has also influenced researchers on computational auditory scene analysis to consider the ideal ratio mask as their objective [Hummersone et al., 2014].

Broadly speaking, time-frequency masking is the core of many source separation techniques. The mask may be estimated differently in different techniques. The DNN based approaches typically rely on DNNs to provide a mask directly or source spectra, which can be used to compute a mask. These approaches are further discussed in the later subsection. In the following subsection, we describe a technique called NMF. This technique relies on source spectral structures and estimates a mask via spectra decomposition.

2.4.2 Non-negative matrix factorization (NMF)

Let us consider a matrix $\mathbf{V} \in \mathbb{R}_+^{F \times N}$ consisting of a non-negative time-frequency representation $v(f,n) \in \mathbb{R}_+$, such as the magnitude $|x(f,n)|$ or the power $|x(f,n)|^2$.

Non-negative matrix factorization (NMF) [Lee & Seung, 1999] approximates this non-negative matrix $\mathbf{V}$ as

$$\widehat{\mathbf{V}} = \mathbf{B}\mathbf{A}, \tag{2.18}$$

where $\mathbf{B} \in \mathbb{R}_+^{F \times K}$ is a non-negative matrix consisting of *basis spectra* $\mathbf{b}_k \in \mathbb{R}_+^{F \times 1}$ representing spectral structures, $\mathbf{A} \in \mathbb{R}_+^{K \times N}$ is a non-negative matrix consisting of time-varying *activations* $\mathbf{a}_k \in \mathbb{R}_+^{1 \times N}$, and $K < \min(F, N)$ is the number of components indexed by $k \in \{1, 2, \ldots, K\}$. Assuming that V consists of J sources, the K components can be grouped into subsets $\mathcal{K}_j$, corresponding to each source:

$$\widehat{\mathbf{V}} = \sum_{k=1}^{K} \mathbf{b}_k \mathbf{a}_k = \sum_{j=1}^{J} \sum_{k \in \mathcal{K}_j} \mathbf{b}_k \mathbf{a}_k. \tag{2.19}$$

A time-frequency mask for source j can then be computed as

$$\widehat{\mathbf{M}}_j = \frac{\sum_{k \in \mathcal{K}_j} \mathbf{b}_k \mathbf{a}_k}{\sum_{k=1}^{K} \mathbf{b}_k \mathbf{a}_k}, \tag{2.20}$$

where the division operation is done element-wise and $\widehat{\mathbf{M}}_j \in [0,1]^{F \times N}$ consisting of $\widehat{m}_j(f,n)$. The estimation of source j can then be computed as in (2.15). For details on the theoretical aspects of source separation by

NMF, the interested reader may refer to Févotte & Idier [2011]; Smaragdis et al. [2014]; Şimşekli et al. [2015].

In the so-called *unsupervised NMF*, the basis spectra and activations are estimated from the observed mixture directly without prior knowledge about the sources [Smaragdis & Brown, 2003]. Its counterpart, *supervised NMF*, estimates the basis spectra from isolated signals of the sources, via *dictionary learning*, and uses these basis spectra to predict the activations given the mixture [Smaragdis, 2007]. In another approach called *semi-supervised NMF*, some basis spectra are learned beforehand as in supervised NMF and additional basis spectra are estimated from the mixture [Mysore & Smaragdis, 2011].

Various algorithms for estimating the parameters of NMF have been proposed [Badeau & Virtanen, 2017]. The most popular one is based on the *multiplicative update rules* [Lee & Seung, 2000], in which the basis spectra and the activations are estimated in an iterative manner. Following [Févotte & Idier, 2011], the multiplicative update rules for the β-divergence, which is a family of cost functions parametrized by $\beta \in \mathbb{R}$, are expressed as

$$B \leftarrow B \circ \frac{\left(V \circ (BA)^{\beta-2}\right) A^\top}{(BA)^{\beta-1} A^\top} \tag{2.21}$$

$$A \leftarrow A \circ \frac{B^\top \left(V \circ (BA)^{\beta-2}\right)}{B^\top (BA)^{\beta-1}} \tag{2.22}$$

where the division and exponentiation are done element-wise, while $\circ$ denotes element-wise multiplication. Various cost functions have been investigated, including the Euclidean distance ($\beta = 2$) [Lee & Seung, 1999], the Kullback-Leibler (KL) divergence ($\beta = 1$) [Lee & Seung, 2000], and the Itakura-Saito (IS) divergence ($\beta = 0$) [Févotte et al., 2009].

2.4.3 DNN based single-channel audio source separation

In this subsection, we present the basics of DNNs and review some single-channel source separation techniques employing DNNs. Most of the techniques use DNNs in the context of time-frequency masking (Section 2.4.1) by estimating a mask directly or estimating source spectra, from which a mask can be derived. A few others use DNNs in the context of NMF (Section 2.4.2).

2.4.3.1 Basics of DNNs

An artificial neuron is a computational model inspired by the biological neuron. Artificial neurons can be interconnected to form an artificial neural network. Hereafter, both the terms neuron and neural network refer to the artificial ones. There are three aspects in designing a neural network: the neuron, the architecture, and the learning [Rojas, 1996, chap. 1].

The neuron aspect describes how the inputs are processed. It typically follows the McCulloch-Pitts model [McCulloch & Pitts, 1943] as shown by Figure 2.3. Mathematically, it can be expressed as

$$h = \sigma\left(\sum_n w_n x_n + b\right), \tag{2.23}$$

which says that the output h is obtained by applying a non-linear activation function σ to the affine transformation of the inputs $x_n, n \in \{1,2,3\}$ given the neuron parameters, that are the *weights* $w_n, n \in \{1,2,3\}$ and possibly the *bias* b. The bias may be used to provide an activation threshold such that when the weighted sum of inputs is less than the bias, the neuron is not activated.

In the past, the *sigmoid*, $\mathrm{sigm}(z) = (1+e^{-z})^{-1}$, and the *hyperbolic tangent*, $\tanh(z) = (1-e^{-2z})(1+e^{-2z})^{-1}$, were the prominent non-linear activation functions σ. Recently, various non-linear functions have been studied, such as the *hard sigmoid* [Gulcehre et al., 2016], which is a piecewise linear approximation of the sigmoid in order to achieve a faster computation and implemented as $\mathrm{hsig}(z) = \max(0, \min(1, sz + 0.5))$, where s is a slope parameter; the *rectifier* [Nair & Hinton, 2010], $\mathrm{rect}(z) = \max(0, z)$; and the *softplus* [Dugas et al., 2000], $\mathrm{sofp}(z) = \ln(1+e^z)$, which can be seen as a smooth approximation of the rectifier. Neurons with rectifier function are also known as rectified linear units. Additionally, there also exist other types of neurons which do not have any parameters, such as the ones for computing the mean of the inputs (known as *average pooling* operation), the ones for taking the maximum value among the inputs (known as *max pooling*), and the ones for multiplying the inputs, as in sum-product networks [Poon & Domingos, 2011].

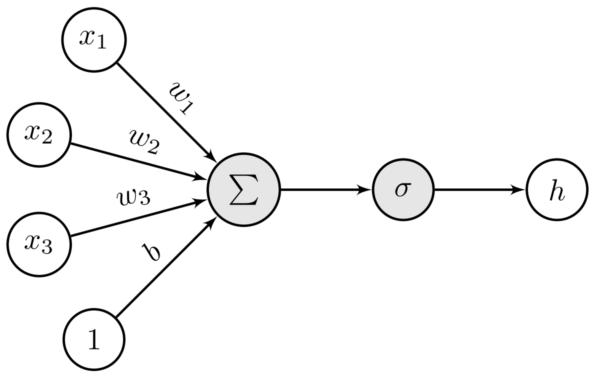


Figure 2.3: McCulloch-Pitts neuron model. The white nodes in the left-hand side are the inputs. The gray nodes represent some operations applied on the inputs. Typically, these nodes are depicted as a single node, called neuron. The white node in the right-hand side is the output, also known as the neuron activation.

The architecture aspect defines how the neurons are interconnected. The neurons are typically organized into layers ordered from the input side to the output side.

A *feedforward neural network (FNN)* passes the data through unidirectional interlayer connections from the input layer, the hidden layers, to the output layer. Most importantly, these connections do not form a cycle. A *multilayer perceptron* [Rosenblatt, 1958] is an FNN with fully-connected layers, that are layers whose neurons are connected to all neurons in the previous layer. In contrast, a *convolutional neural network* [Lecun et al., 1998], which resembles a *neocognitron* [Fukushima, 1980], uses locally-connected layers, where the neurons of a layer are connected to a subset of the neurons in the previous layer. This subset of neurons typically represents an area in an N-dimensional space. An *autoencoder* [Bourlard & Kamp, 1988] can be seen as a special case of multilayer perceptron which tries to reconstruct the inputs. The *denoising autoencoder* [Vincent et al., 2008] is an extension of the autoencoder concept which aims to estimate the clean version of the noisy inputs. Thus, its output layer has the same dimension as its input layer.

A *recurrent neural network (RNN)* [Elman, 1990] passes the data from the input layer to the output layer through the hidden neurons, whose activations are influenced by past and current inputs. Since this property allows RNN to capture a context, RNN is known to be good for modeling a sequence. Instead of hidden neurons, advanced RNNs consider hidden *cells*, in which multiple neurons (known as *gates*) are utilized, such as *long short-term memory (LSTM)* [Hochreiter & Schmidhuber, 1997] and *gated recurrent unit* [Chung et al., 2014]. Employing these two cell types, studies have been done to explore variations of architectures [Jozefowicz et al., 2015; Greff et al., 2017]. A *bidirectional RNN* [Schuster & Paliwal, 1997] is an architecture where a hidden layer consists of a pair of sub-layers. One sub-layer reads the sequence in forward direction and the other sub-layer reads the sequence in backward direction. The outputs of these two sub-layers are combined, e.g., by summing or concatenating, and then fed to the next layer.

Other notable architectures include the *Boltzmann machine* [Ackley et al., 1985], which is a 2-layer network with bidirectional interlayer and intralayer connections, and the *restricted Boltzmann machine* [Smolensky, 1986], where only the interlayer connections are considered. Several restricted Boltzmann machines can be stacked to form a deep belief network [Bengio et al., 2006]. The success of this approach triggered the trend of deep neural network research in the last decade.

A *deep neural network (DNN)* is typically defined as an FNN with more than one hidden layer [Hinton et al., 2012; Wang, 2017]. Besides, [Schmidhuber, 2015, sec. 3] proposes a concept of causal connection chain and a concept of problem depth, which is the number of connection sets with modifiable weights in the chain, then determines whether an architecture is deep or not based on the problem depth. If we apply this concept to the definition of DNN mentioned earlier, DNN has at least a problem depth of three, because the data passes through modifiable hidden weight sets of two different hidden layers and the output layer. RNN has a long causal connection chain because a set of outputs at a time instance is not only influenced by a set of inputs at the same time instance, but also other time instances in the past. This results in a high problem depth because the data flows though connections whose weights are modifiable. Consequently, both DNNs and RNNs require *deep learning*.

The learning aspect mainly specifies how the parameters of the neurons, that are weights and biases, are optimized.

Backpropagation [Rumelhart et al., 1986; Werbos, 1988] is the most popular optimization method used in this context. It is a gradient descent based algorithm, in which the data is first passed from the input side through the network to compute the error in the output side (forward pass). This error is then passed back to the input side while computing the gradients, which are the partial derivatives of the error function with respect to the parameters (backward pass). Finally, the gradients are used to update the parameters following a particular rule (parameter update). The weight update rules include the ones with fixed learning rate, such as the standard update and its variant with momentum [Rumelhart et al., 1986] or Nesterov's accelerated gradient [Nesterov, 1983]; and the ones with adaptive learning rates, such as AdaGrad [Duchi et al., 2011], AdaDelta [Zeiler, 2012] or Adam [Kingma & Ba, 2014]. The application of backpropagation for RNNs is known as *backpropagation through time* [Werbos, 1990]. Other gradient descent based algorithms used for training the networks include Newton, Gauss-Newton, and Levenberg-Marquardt algorithms [Wilamowski & Yu, 2010; Yu & Wilamowski, 2011]. Beside gradient descent based algorithms, evolutionary algorithms, such as genetic algorithm [Montana & Davis, 1989], can also be used.

This learning aspect also specifies how the parameters are initialized and how the learning is regularized. *Parameter initialization* schemes have been studied and shown to improve the learning, such as weight initialization for neurons with sigmoid and hyperbolic tangent activation functions [Glorot & Bengio, 2010], weight initialization for rectified linear units [He et al., 2015], and bias initialization for the forget gate of LSTM [Jozefowicz et al., 2015]. *Regularization* is a way to avoid *overfitting* and improve the generalization of the models. Various techniques have been studied, including early stopping mechanism based on validation errors [Prechelt, 2012], L_1 or L_2 weight regularization [Bengio, 2012], DropOut [Srivastava et al., 2014], DropConnect [Wan et al., 2013], and gradient clipping or normalization [Pascanu et al., 2013].

In our study, the designs of neural networks may be different from one experiment to another. In general, the architectures we used follow either the topologies of multilayer perceptron or bidirectional RNN with LSTM units. The networks are trained using the backpropagation algorithm with Nesterov's accelerated gradient or AdaDelta weight update rules. Some regularization techniques are employed, such as early stopping, L2 weight regularization, dropout, and gradient normalization. The networks are implemented using Theano[1] [Bergstra et al., 2010; Theano Dev Team, 2016] and Keras[2] [Chollet et al., 2015] libraries in Python. Backpropagation is done utilizing the automatic gradient computation provided by Theano. The details for each experiment are presented in the corresponding discussion.

The following Section 2.4.3.2 presents DNN usages for single-channel source separation. The usages for multichannel source separation are presented in Section 2.5.3.

2.4.3.2 DNN based separation techniques

Figure 2.4 illustrates of typical DNN usages for single-channel audio source separation, in the case of a mixture of two sources, such as speech and noise. The mixture time-frequency representation is pre-processed to extract relevant features. Given these features as inputs, a DNN is utilized either for estimating the source spectra whose ratio yields a time-frequency mask as in (2.16) [Huang et al., 2014a,b; Tu et al., 2014; Araki et al., 2015; Huang et al., 2015; Uhlich et al., 2015; Grais et al., 2017; Osako et al., 2017] or for directly estimating the time-frequency mask [Narayanan & Wang, 2013; Wang & Wang, 2013; Jiang et al., 2014; Weninger et al., 2014; Narayanan & Wang, 2015; Wang & Wang, 2015; Grais et al., 2016; Williamson et al., 2016; Zhang & Wang, 2016; Delfarah & Wang, 2017]. The estimated target time-frequency representation is then obtained as the product of the mixture time-frequency representation and the estimated time-frequency mask as in (2.15).

There also exist DNN based approaches which are motivated by the success of NMF in decomposing the time-frequency structure. DNNs have been used to estimate the source activations [Kang et al., 2015; Williamson et al., 2015; Li et al., 2016]. On the contrary, Tseng et al. [2015] use the activations estimated by NMF as the DNN inputs for estimating a time-frequency mask. Further, Le Roux et al. [2015] propose a DNN architecture representing the iterations of multiplicative updates (see (2.22)) for estimating the target source. Each hidden layer corresponds to a single update of activations and the output layer does the time-frequency masking. The network weights act as the basis spectra and these weights are not tied across layers, which is equivalent to using different basis spectra for different iterations. The study also propose a learning algorithm to ensure that the weights are non-negative.

[1]See `http://deeplearning.net/software/theano`.
[2]See `https://github.com/fchollet/keras`.

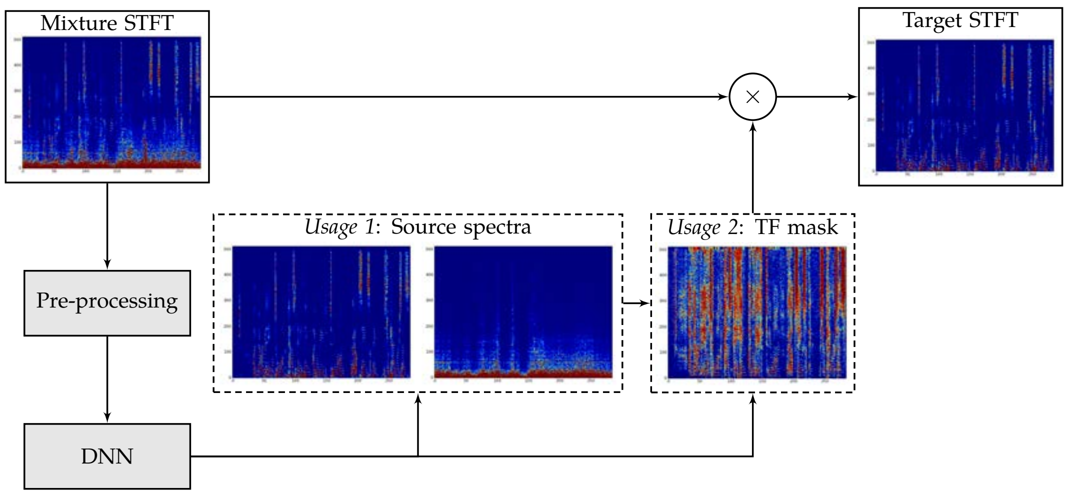


Figure 2.4: Typical usages of DNNs for single-channel audio source separation: estimating the source spectra and estimating the time-frequency mask.

Various DNN architectures and training criteria have been investigated and compared in Weninger et al. [2014]; Erdogan et al. [2015]; Weninger et al. [2015]. These studies experiment with different DNNs, LSTM based RNNs, and bidirectional LSTM based RNNs to provide a real-valued ratio mask estimate $\widehat{m}_{\mathrm{S}}^{\mathrm{rat}}(f,n)$. The networks are trained by minimizing the mask estimation error:

$$\mathcal{D}_{\mathrm{MA}} = \sum_{f,n} \left(m_{\mathrm{S}}^{\mathrm{rat}}(f,n) - \widehat{m}_{\mathrm{S}}^{\mathrm{rat}}(f,n)\right)^2, \tag{2.24}$$

by minimizing the error of spectra computed using the estimated mask:

$$\mathcal{D}_{\mathrm{SA}} = \sum_{f,n} \left(\widehat{m}_{\mathrm{S}}^{\mathrm{rat}}(f,n)\,|x(f,n)| - |s(f,n)|\right)^2, \tag{2.25}$$

where $s(f,n)$ is the target speech spectra, or by minimizing the error of signal in the complex-valued time-frequency domain computed using the estimated mask:

$$\mathcal{D}_{\mathrm{PSA}} = \sum_{f,n} \left|\widehat{m}_{\mathrm{S}}^{\mathrm{rat}}(f,n)x(f,n) - s(f,n)\right|^2. \tag{2.26}$$

In summary, these study shows that the performance of RNNs is better than that of DNNs and the performance of bidirectional LSTM based RNNs is better than that of LSTM based RNNs. They also show that $\mathcal{D}_{\mathrm{PSA}}$ outperforms the other two cost functions. This indicates that taking phase information into account in the DNN training is beneficial although the estimated mask is real-valued and thus, it does not affect the phase. In addition, Wang & Wang [2015] compute the error on the time-domain:

$$\mathcal{D}_{\text{IDFT}} = \frac{1}{TN} \sum_{t,n} \left(\text{IDFT} \left(\widehat{m}_{\text{S}}^{\text{rat}}(f,n) x(f,n) \right) - s(t,n) \right)^2 , \tag{2.27}$$

where IDFT$(\cdot)$ include the reconstruction of the negative frequency bins before the inverse discrete Fourier transform is applied (see Section 2.3). Essentially, this $\mathcal{D}_{\text{IDFT}}$ is equivalent to $\mathcal{D}_{\text{PSA}}$. The study shows that $\mathcal{D}_{\text{IDFT}}$ outperforms a cost function similar to $\mathcal{D}_{\text{MA}}$.

Most studies focus either on speech separation, such as Narayanan & Wang [2013]; Tu et al. [2014]; Weninger et al. [2014]; Erdogan et al. [2015]; Weninger et al. [2015]; Wang & Wang [2015]; Williamson et al. [2016]; Zhang & Wang [2016]; Delfarah & Wang [2017], or on music separation, such as Huang et al. [2014b]; Uhlich et al. [2015]; Grais et al. [2016, 2017]; Osako et al. [2017]. Only few studies investigate both applications, such as Huang et al. [2015].

It is worth mentioning that many of the study directions mentioned above had not yet been considered in the beginning of 2015 when our study was started. This is also indicated by the publication years of the cited studies.

2.5 State-of-the-art multichannel audio source separation

Following the discussion of single-channel audio source separation in the previous section, this section presents essential multichannel audio source separation methods. It covers beamforming, the classical expectation-maximization (EM) based multichannel source separation framework, and various DNN based approaches. The EM based separation framework is the basis for the DNN based frameworks proposed later in our study.

There are also notable methods beyond the ones particularly discussed here, such as independent component analysis and its variants. *Independent component analysis* relies on the assumption that the source signals are statistically independent and non-Gaussian [Comon, 1994; Hyvärinen & Oja, 2000]. It is formulated for a linear mixture and since audio mixtures are typically convolutive, it is applied frequency-wise in the time-frequency domain [Smaragdis, 1998; Parra & Spence, 2000]. It achieves blind source separation, i.e., the source signals and the mixing matrix, which defines the contribution of each source signal to each channel of the mixture, are estimated from the observed signal solely. Because of this, it has two inherent challenges, namely permutation indeterminacy, due to the unknown order of the sources, and scaling indeterminacy, due to the unknown scale of the mixing or the source signals. *Independent vector analysis* aims to solve the permutation indeterminacy by jointly processing all frequency bins and exploiting the dependencies across frequencies so that the order of the sources across frequencies is the same [Kim et al., 2007; Lee et al., 2007]. While independent component analysis is formulated for the determined mixture case, i.e., the number of channels is equal to the number of sources $I = J$, *sparse component analysis* works for the under-determined case, i.e., the number of channels is lower than the number of sources $I < J$, by exploiting the sparsity of the time-frequency representations [Jourjine et al., 2000; Bofill & Zibulevsky, 2001].

2.5.1 Beamforming

Microphone array processing focuses on exploiting the spatio-temporal information captured by different microphones in an array relying on the fact that different sounds coming from different sources, that are separated in space, travel on different paths and durations. It is typically seen as an extension of general sensor array processing for antenna, radar, and sonar. However, general sensor array processing methods cannot directly be applied to microphone arrays. This is because the characteristics of the sounds of interests, e.g., speech, and the environments, which determine the sound propagation paths, must be taken care of [Brandstein & Ward, 2001; Benesty et al., 2008].

Beamforming is the most popular microphone array processing approach. It is commonly used for speech enhancement in ASR with a multichannel input. It refers to any method that combines multichannel signals so that the resulting single-channel signal has desirable properties, such as reduced noise or less distorted speech [Markovich-Golan et al., 2017].

The most basic beamforming method is the delay-and-sum beamformer [Veen & Buckley, 1988; Kumatani et al., 2012]. This beamformer can be achieved in two steps: synchronization and weight-and-sum [Benesty et al., 2008, chap. 3]. For a stationary source and microphones that are separated in a space, different microphones capture the same sound coming from the source with different delays. The synchronization step is done by shifting the signal in each channel based on these delays so that all channels are aligned. Theoretically, the delays can be computed given the relative location of the source with respect to the microphones. In practice, the delays are estimated from the observed multichannel signal by time difference of arrival estimation methods, such as the generalized cross correlation method with phase transform weighting [Knapp & Carter, 1976]. This synchronization allows constructive superposition of the signals coming from the desired direction, i.e., the direction of the target source, and may cause destructive superposition of the signals coming from the other directions resulting in noise attenuation. The weight-and-sum step then performs weighted superposition on the aligned channels. For the delay-and-sum beamformer, all channels have the same importance. Thus, it is equivalent to a simple average over the aligned channels. Given the multichannel observed mixture $\mathbf{x}(f,n) = [x_1(f,n), x_2(f,n), \ldots, x_I(f,n)]^\top$, delay-and-sum beamforming can be expressed as

$$\widehat{s}(f,n) = \sum_{i=1}^{I} w_{\text{DS}} x_i(f,n) e^{-\jmath 2\pi f \tau_i}, \tag{2.28}$$

where $\widehat{s}(f,n)$ is the beamformer output, $w_{\text{DS}} = \frac{1}{I}$ the equal weight for all aligned channels, $x_i(f,n)$ the channel i of the beamformer input, and τ_i the delay for channel i in the time domain used for the synchronization. In matrix form, the same formulation can be expressed as

$$\widehat{s}(f,n) = \mathbf{w}_{\text{DS}}(f)^* \mathbf{x}(f,n), \tag{2.29}$$

$$\mathbf{w}_{\text{DS}}(f) = w_{\text{DS}} \mathbf{d}(f) = \frac{1}{I}\mathbf{d}(f), \tag{2.30}$$

$$\mathbf{d}(f) = \left[e^{-\jmath 2\pi f \tau_1}, e^{-\jmath 2\pi f \tau_2}, \ldots, e^{-\jmath 2\pi f \tau_I}\right]^\top, \tag{2.31}$$

where $\cdot^*$ denotes the Hermitian transposition, $\mathbf{w}_{\text{DS}}(f)$ is the delay-and-sum beamformer for frequency bin f, and $\mathbf{d}(f)$ the so-called steering vector for frequency bin f. The steering vector relies on the time difference of arrival estimation for the target source. The beamformer output correspond to a single-channel version of the multichannel target source spatial image.

A variant of delay-and-sum beamformer which is more robust is called weighted delay-and-sum beamformer, also known as filter-and-sum beamformer or BeamformIt [Anguera et al., 2007]. Instead of employing a time-invariant equal weight for all aligned channels as in the delay-and-sum beamformer w_{DS}, the weighted delay-and-sum beamformer employs time-varying channel weights $w_{\mathrm{WDS}}(n)$. It can be expressed as

$$\widehat{s}(f,n) = \mathbf{w}_{\mathrm{WDS}}(f,n)^{*}\mathbf{x}(f,n), \tag{2.32}$$

$$\mathbf{w}_{\mathrm{WDS}}(f,n) = w_{\mathrm{WDS}}(n)\mathbf{d}(f,n), \tag{2.33}$$

$$\mathbf{d}(f,n) = \left[e^{-\jmath 2\pi f\tau_1(n)}, e^{-\jmath 2\pi f\tau_2(n)}, \dots, e^{-\jmath 2\pi f\tau_I(n)}\right]^{\top}, \tag{2.34}$$

where $\mathbf{w}_{\mathrm{WDS}}(f,n)$ and $\mathbf{d}(f,n)$ are the weighted delay-and-sum beamformer and the steering vector, respectively, for frequency bin f and time frame n, while $\tau_i(n)$ is the delay for channel i and time frame n in the time domain.

Different beamformers can also be derived based on different desired output properties. This leads to beamformer variants, including the minimum variance distortionless response beamformer and the generalized eigenvalue (GEV) beamformer [Warsitz & Haeb-Umbach, 2007; Kumatani et al., 2012; Heymann et al., 2016].

Considering speech as the target source, the minimum variance distortionless response beamformer aims to minimize the residual noise, subject to a distortionless constraint in the direction of the target:

$$\mathbf{w}_{\mathrm{MVDR}}(f) = \underset{\mathbf{w}(f)}{\arg\min}\, \mathbf{w}(f)^{*}\mathbf{R}_{\mathbf{c}_{\mathrm{N}}}(f)\mathbf{w}(f), \quad \text{s.t. } \mathbf{w}(f)\mathbf{d}(f) = 1, \tag{2.35}$$

which leads to

$$\mathbf{w}_{\mathrm{MVDR}}(f) = \frac{\mathbf{d}(f)^{*}\mathbf{R}_{\mathbf{c}_{\mathrm{N}}}(f)^{-1}}{\mathbf{d}(f)^{*}\mathbf{R}_{\mathbf{c}_{\mathrm{N}}}(f)^{-1}\mathbf{d}(f)}, \tag{2.36}$$

where $\mathbf{R}_{\mathbf{c}_{\mathrm{N}}}(f)$ is the time-invariant noise covariance matrix. The estimation of this matrix typically relies on voice activity detection, which predicts noise-only segments in the mixture $\mathbf{x}(f,n)$ [Cornelis et al., 2011; Serizel et al., 2014]. Let subset $\mathcal{N}_{\mathrm{N}}$ consists of noise-only time frames, the noise covariance matrix can be estimated as

$$\mathbf{R}_{\mathbf{c}_{\mathrm{N}}}(f) = \frac{1}{|\mathcal{N}_{\mathrm{N}}|}\sum_{n\in\mathcal{N}_{\mathrm{N}}} \mathbf{x}(f,n)\mathbf{x}(f,n)^{*}. \tag{2.37}$$

Also considering speech as the target source, the GEV beamformer aims to maximize the signal-to-noise ratio as

$$\mathbf{w}_{\text{GEV}}(f) = \underset{\mathbf{w}(f)}{\arg\max} \frac{\mathbf{w}(f)^* \mathbf{R}_{\mathbf{c}_\text{S}}(f)\mathbf{w}(f)}{\mathbf{w}(f)^* \mathbf{R}_{\mathbf{c}_\text{N}}(f)\mathbf{w}(f)}, \tag{2.38}$$

which leads to

$$\mathbf{w}_{\text{GEV}}(f) = \mathcal{P}\left(\mathbf{R}_{\mathbf{c}_\text{N}}(f)^{-1}\mathbf{R}_{\mathbf{c}_\text{S}}(f)\right), \tag{2.39}$$

where $\mathcal{P}(\cdot)$ computes the principal component and $\mathbf{R}_{\mathbf{c}_\text{S}}(f)$ is the time-invariant speech covariance matrix. The estimation of this matrix typically also relies on voice activity detection to predict noisy speech segments. With these, the noisy speech covariance matrix can be computed similarly to (2.37). Assuming that the speech and the noise are uncorrelated, the speech covariance matrix $\mathbf{R}_{\mathbf{c}_\text{S}}(f)$ can be obtained by subtracting the noise covariance matrix $\mathbf{R}_{\mathbf{c}_\text{N}}(f)$ from the noisy speech covariance matrix.

Post-filtering can then be applied to the beamformer output in order to improve it. The post-filter that is typically used for the GEV beamformer is the blind analytic normalization (BAN) [Warsitz & Haeb-Umbach, 2007; Heymann et al., 2016]:

$$g_{\text{BAN}}(f) = \frac{\sqrt{\frac{1}{I}\mathbf{w}_{\text{GEV}}(f)^* \mathbf{R}_{\mathbf{c}_\text{N}}(f)\mathbf{R}_{\mathbf{c}_\text{N}}(f)\mathbf{w}_{\text{GEV}}(f)}}{\mathbf{w}_{\text{GEV}}(f)^* \mathbf{R}_{\mathbf{c}_\text{N}}(f)\mathbf{w}_{\text{GEV}}(f)}, \tag{2.40}$$

which results in

$$\mathbf{w}_{\text{GEV-BAN}}(f) = g_{\text{BAN}}(f)\mathbf{w}_{\text{GEV}}(f). \tag{2.41}$$

This post-filter aims to minimize the distortion in the direction of the target source. Thus, ideally, the GEV beamformer with BAN is equivalent to the minimum variance distortionless response beamformer.

2.5.2 Expectation-maximization (EM) based multichannel audio source separation framework

In this subsection, we present the classical EM iterative framework employing the multichannel Gaussian model [Duong et al., 2010a]. This framework is the basis for our proposed frameworks presented in the following chapters.

While beamforming we previously discussed aims to obtain a single-channel target source, here we aim to obtain multichannel target source spatial images.

2.5.2.1 Multichannel Gaussian model

The coefficients of $\mathbf{c}_j(f,n)$ are assumed to be independent for different j, f, and n. These coefficients are modeled as a multivariate complex-valued zero-mean isotropic Gaussian distribution with time-varying covariance matrix $\mathbf{R}_{\mathbf{c}_j}(f,n)$:

$$\mathbf{R}_{\mathbf{c}_j}(f,n) = v_j(f,n)\mathbf{R}_j(f), \tag{2.42}$$

$$\mathbf{c}_j(f,n) \sim \mathcal{N}_c\left(\mathbf{0}, \mathbf{R}_{\mathbf{c}_j}(f,n)\right), \tag{2.43}$$

where $v_j(f,n) \in \mathbb{R}_+$ denotes the *time-varying power spectral density (PSD)* of source j for frequency bin f and time frame n, and $\mathbf{R}_j(f)$ is the *time-invariant spatial covariance matrix* of source j for frequency bin f [Duong et al., 2010a; Vincent et al., 2011]. This $I \times I$ covariance matrix represents spatial information by encoding the spatial position and the spatial width of the corresponding source through the inter-channel correlations. Assuming that the sources are uncorrelated, the mixture follows a multivariate complex-valued zero-mean Gaussian distribution with covariance matrix $\mathbf{R}_{\mathbf{x}}(f,n)$:

$$\mathbf{R}_{\mathbf{x}}(f,n) = \sum_{j=1}^{J} \mathbf{R}_{\mathbf{c}_j}(f,n) = \sum_{j=1}^{J} v_j(f,n)\mathbf{R}_j(f), \tag{2.44}$$

$$\mathbf{x}(f,n) \sim \mathcal{N}_c\left(\mathbf{0}, \mathbf{R}_{\mathbf{x}}(f,n)\right). \tag{2.45}$$

The covariance matrices $\mathbf{R}_{\mathbf{c}_j}(f,n)$, $\mathbf{R}_{\mathbf{x}}(f,n)$, and $\mathbf{R}_j(f)$ are complex-valued Hermitian positive-definite matrices.

Given the PSDs $v_j(f,n)$ and the spatial covariance matrices $\mathbf{R}_j(f)$ of all sources, the source spatial images can be estimated in the minimum mean squared error sense using multichannel Wiener filtering [Duong et al., 2010a] as

$$\widehat{\mathbf{c}}_j(f,n) = \mathbf{W}_j(f,n)\mathbf{x}(f,n), \tag{2.46}$$

where the *time-varying multichannel Wiener filter* $\mathbf{W}_j(f,n) \in \mathbb{C}^{I \times I}$ is given by

$$\mathbf{W}_j(f,n) = \mathbf{R}_{\mathbf{c}_j}(f,n)\mathbf{R}_{\mathbf{x}}(f,n)^{-1}. \tag{2.47}$$

Finally, the time-domain source estimates $\widehat{\mathbf{c}}_j(t)$ are recovered from $\widehat{\mathbf{c}}_j(f,n)$ by inverse STFT.

Following this formulation, source separation translates into the problem of estimating the PSDs $v_j(f,n)$ and the spatial covariance matrices $\mathbf{R}_j(f)$ of all sources. The EM algorithm by Duong et al. [2010a] aims to solve this problem in maximum likelihood sense. There exist alternative algorithm based on minimization-maximization [Sawada et al., 2013].

2.5.2.2 General iterative EM framework

Algorithm 1 summarizes the general iterative EM framework for estimating the PSDs $v_j(f,n)$ and the spatial covariance matrices $\mathbf{R}_j(f)$ of all sources proposed by Duong et al. [2010a] and extended by Ozerov et al. [2012]. In particular, Duong et al. [2010a] re-estimate the spectral parameters without constrained. This means their algorithm does not have spectrogram fitting step and thus, $v_j(f,n) = z_j(f,n)$. For the following discussions, the term 'spectral parameters' refers to the PSDs and they are used interchangeably. Further, they are also used interchangeably with 'spectrograms', which loosely mean PSD estimates [Liutkus & Badeau, 2015]. Likewise, the term 'spatial parameters' refers to the spatial covariance matrices. The sources in the mixture $\mathbf{x}(f,n)$ are assumed to be stationary to satisfy the model in (2.43).

In the beginning, the estimated source PSDs $v_j(f,n)$ are initialized in the *spectrogram initialization* step. This can be done, for instance by computing the mixture spectrogram and then dividing it by the number of sources, which implies that each source contributes equally to the mixture. The spatial covariance matrices $\mathbf{R}_j(f)$ are also initialized, for instance as identity matrices $\mathbf{I} \in \mathbb{C}^{I \times I}$, which implies that the channels are uncorrelated and the energy of each source is evenly distributed in each channel. Since the EM algorithm does not guarantee finding the global optimum, the initialization of the parameters $v_j(f,n)$ and $\mathbf{R}_j(f)$ has a strong influence on the final performance. It should be mentioned that the example initializations above are poor ones since the parameters should be differently initialized for different sources. Various initialization schemes have been used and studied [Duong et al., 2010a; Araki & Nakatani, 2011; Togami, 2011; Ozerov et al., 2012]. In general, the spectral parameters are initialized by clustering or NMF

Algorithm 1 General iterative EM framework [Duong et al., 2010a; Ozerov et al., 2012]

```
Inputs:
    STFT of mixture $\mathbf{x}(f,n)$
    Number of sources $J$
    Number of EM iterations $L$
    Spectral models $\mathrm{M}_0^{\mathrm{spec}}, \mathrm{M}_1^{\mathrm{spec}}, \dots, \mathrm{M}_J^{\mathrm{spec}}$
1: for each source $j$ of $J$ do
2:     Initialize the spectrogram: $v_j(f,n) \leftarrow$ spectrogram initialization
3:     Initialize the spatial covariance matrix: $\mathbf{R}_j(f) \leftarrow$ identity matrix $\mathbf{I}$
4: end for
5: for each EM iteration $l$ of $L$ do
6:     Compute the mixture covariance matrix:
           $\mathbf{R}_{\mathbf{x}}(f,n) = \sum_{j=1}^{J} v_j(f,n)\mathbf{R}_j(f)$        (2.44)
7:     for each source $j$ of $J$ do
8:         Compute the Wiener filter gain:
               $\mathbf{W}_j(f,n) = v_j(f,n)\mathbf{R}_j(f)\mathbf{R}_{\mathbf{x}}(f,n)^{-1}$        (2.47)
9:         Compute the spatial image:
               $\widehat{\mathbf{c}}_j(f,n) = \mathbf{W}_j(f,n)\mathbf{x}(f,n)$        (2.46)
10:        Compute the posterior second-order raw moments of the spatial
               image:
               $\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n) = \widehat{\mathbf{c}}_j(f,n)\widehat{\mathbf{c}}_j(f,n)^*$
                   $+ (\mathbf{I} - \mathbf{W}_j(f,n))\, v_j(f,n)\mathbf{R}_j(f)$        (2.48)
11:        Update the spatial covariance matrix:
               $\mathbf{R}_j(f) = \frac{1}{N}\sum_{n=1}^{N} v_j(f,n)^{-1}\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)$        (2.49)
12:        Compute the unconstrained PSD:
               $z_j(f,n) = \frac{1}{I}\mathrm{tr}\left(\mathbf{R}_j^{-1}(f)\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)\right)$        (2.50)
13:        Update the spectrogram:
               $v_j(f,n) \leftarrow$ spectrogram fitting by $\mathrm{M}_j^{\mathrm{spec}}\left(z_j(f,n)\right)$
14:    end for
15: end for
```

(continued on the next page)

in the time-frequency domain and the spatial parameters are initialized by source localization via time difference of arrival estimation.

Algorithm 1 General iterative EM framework [Duong et al., 2010a; Ozerov et al., 2012] (continued)

```
16: for each source $j$ of $J$ do
17:     Compute the final spatial image:
            $\widehat{\mathbf{c}}_j(f,n) = \frac{v_j(f,n)\mathbf{R}_j(f)}{\sum_{j'=1}^{J} v_{j'}(f,n)\mathbf{R}_{j'}(f)}\mathbf{x}(f,n)$        (2.46)
18: end for
Outputs:
    All spatial source images $[\widehat{\mathbf{c}}_1(f,n),\dots,\widehat{\mathbf{c}}_J(f,n)]$
```

The following iterations can be divided into E-step and M-step. In the E-step, given the estimated parameters $v_j(f,n)$ and $\mathbf{R}_j(f)$ of each source, the source image estimates $\widehat{\mathbf{c}}_j(f,n)$ are obtained by multichannel Wiener filtering (2.46) and the posterior second-order raw moments of the spatial source images $\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)$ are computed as

$$\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n) = \widehat{\mathbf{c}}_j(f,n)\widehat{\mathbf{c}}_j(f,n)^* + (\mathbf{I} - \mathbf{W}_j(f,n))\,\mathbf{R}_{\mathbf{c}_j}(f,n). \tag{2.48}$$

In the M-step, the spatial covariance matrices $\mathbf{R}_j(f)$ are updated as

$$\mathbf{R}_j(f) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{v_j(f,n)}\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n). \tag{2.49}$$

The source PSDs $v_j(f,n)$ are first estimated without constraints as

$$z_j(f,n) = \frac{1}{I}\text{tr}\left(\mathbf{R}_j(f)^{-1}\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)\right), \tag{2.50}$$

where tr$(\cdot)$ denotes the trace of a matrix. Then, they are updated according to given spectral models by fitting $v_j(f,n)$ from $z_j(f,n)$ in the *spectrogram fitting* step. The spectrogram fitting and possibly the spectrogram initialization rely on the spectral models, which are denoted by $\mathrm{M}_0^{\mathrm{E}}$, $\mathrm{M}_1^{\mathrm{E}}$, …, $\mathrm{M}_J^{\mathrm{E}}$ in Algorithm 1. These models may include NMF [Ozerov et al., 2012], Kernel Additive Modelling [Liutkus et al., 2014, 2015b], and continuity models [Duong et al., 2011].

2.5.3 DNN based multichannel audio source separation techniques

After discussing classical approaches, let us now discuss recently studied DNN based approaches. There exist a few DNN based approaches exploiting multichannel data. These approaches can be divided into three categories as follows:

1. approaches employing DNN input features derived from multichannel data for estimating single-channel masks, either directly or via intermediate variables [Jiang et al., 2014; Araki et al., 2015],
2. approaches employing DNNs for estimating intermediate variables, such as masks or spectra, which then are used to derive multichannel filters [Erdogan et al., 2016; Heymann et al., 2017; Uhlich et al., 2017; Wang et al., 2017],
3. approaches employing DNNs for directly estimating a multichannel filter [Xiao et al., 2016; Sainath et al., 2017].

It should be noted that this categorization did not exist when our study started. We provide it in retrospect in order to position our study with respect to concurrent studies. It is also worth mentioning that most of the above studies were published after our initial publication. The rest of this subsection provides some details about these studies.

2.5.3.1 Utilizing multichannel features for estimating a single-channel mask

Jiang et al. [2014] consider speech segregation in a binaural (two-channel) setting, where frequency-dependent DNNs are used to predict a *single-channel binary mask* for each channel of a gammatone filterbank. The study explores the use of binaural features with or without monaural features as the inputs of the DNNs. The binaural features are the normalized cross-correlation function or the inter-channel time difference, which is the delay that maximizes the cross-correlation function, combined with the inter-channel level difference. The monaural features are gammatone frequency cepstral coefficients. The proposed system is evaluated on simulated data sets in which a target speaker is always positioned at a fixed azimuth, i.e., 0°, and one, two, or four interference sources producing babble noise positioned at various azimuths. The experimental results show that the system performs well for unseen interfering source positions and various noisy and/or reverberant environments in terms of mask estimation and output signal-to-noise ratio.

Araki et al. [2015] consider speech enhancement in a multichannel setting, where a denoising autoencoder is used to predict a clean log mel frequency coefficients for each time frame. The ratio between the clean coefficients and the corresponding noisy ones in the linear domain forms a *single-channel ratio mask* for the enhancement. The study explores the use of multichannel features and their combination with the monaural features, i.e., the noisy log mel frequency coefficients, as the inputs of a DNN. The multichannel features include inter-channel level differences, inter-channel phase differences, and pre-estimated speech or noise computed using masks estimated by a spatial clustering approach [Nakatani et al., 2013]. Overall, the experimental results show that the pre-estimated speech combined with the monaural features outperforms the other combinations in terms of source separation and speech recognition metrics.

2.5.3.2 Estimating intermediate variables for deriving a multichannel filter

Erdogan et al. [2016] and Heymann et al. [2017] consider speech enhancement by *time-invariant minimum variance distortionless response beamforming* and *time-invariant GEV beamforming*, respectively. Both studies use DNN variants for estimating soft masks where the same DNN is used for each channel separately. Erdogan et al. [2016] uses LSTM based RNNs, while Heymann et al. [2017] uses multilayer perceptrons, convolutional neural networks, and bidirectional LSTM based RNNs. In both studies, the masks are then used to derive the beamformer weights by computing the speech and noise covariance matrices. In Erdogan et al. [2016], the estimated speech mask of channel i, $m_{i\mathrm{S}}(f,n) \in \mathbb{R}$, is used to estimate channel i of the noise spatial image as

$$\widehat{c}_{i\mathrm{N}}(f,n) = (1 - \widehat{m}_{i\mathrm{S}}(f,n))x_i(f,n). \tag{2.51}$$

The multichannel noise spatial image is then constructed as $\widehat{\mathbf{c}}_{\mathrm{N}}(f,n) = [\widehat{c}_{1\mathrm{N}}(f,n),\, \widehat{c}_{2\mathrm{N}}(f,n),\, \ldots,\, \widehat{c}_{I\mathrm{N}}(f,n)]^\top$ and used for computing the noise covariance matrix as

$$\mathbf{R}_{\mathbf{c}_\mathrm{N}}(f) = \frac{1}{N}\sum_{n=1}^{N} \widehat{\mathbf{c}}_{\mathrm{N}}(f,n)\widehat{\mathbf{c}}_{\mathrm{N}}(f,n)^*. \tag{2.52}$$

In order to increase the robustness, Erdogan et al. [2016] also experiment with combining the masks by max pooling across channels to produce a single mask and then apply it on all channels. In Heymann et al. [2017], the estimated channel-wise speech masks $\widehat{m}_{i\mathrm{S}}(f, n) \in [0, 1]$ are combined by median pooling to produce a single speech mask $\widehat{m}_{\mathrm{S}}(f, n)$. The same operation is also applied on the estimated channel-wise noise masks to produce a single noise mask $\widehat{m}_{\mathrm{N}}(f, n)$. Both speech and noise masks are then applied as

$$\mathbf{R}_{\mathbf{c}_{\mathrm{S}}}(f) = \sum_{n=1}^{N} \widehat{m}_{\mathrm{S}}(f, n)\mathbf{x}(f, n)\mathbf{x}(f, n)^*, \tag{2.53}$$

$$\mathbf{R}_{\mathbf{c}_{\mathrm{N}}}(f) = \sum_{n=1}^{N} \widehat{m}_{\mathrm{N}}(f, n)\mathbf{x}(f, n)\mathbf{x}(f, n)^*. \tag{2.54}$$

In order to improve the outputs, both studies consider post-filtering on the beamformer outputs. Erdogan et al. [2016] use the same mask obtained by max pooling and Heymann et al. [2017] apply BAN [Warsitz & Haeb-Umbach, 2007]. Both studies show that the proposed approach performs well in terms of source separation and speech recognition metrics. Although a precursor study [Heymann et al., 2016] indicates that the GEV computation is numerically more stable than the minimum variance distortionless response one and thus it is preferable, Erdogan et al. [2016] point out that their minimum variance distortionless response formulation is different from that of Heymann et al. [2016]. In a more recent study, Wang et al. [2017] show that beamforming by rank-1 multichannel Wiener filter, subject to a constant residual noise power constraint over time and frequency, outperforms minimum variance distortionless response and GEV beamformers. In this study, the source covariance matrices are computed using masks estimated by a bidirectional LSTM based RNN following Heymann et al. [2017].

Uhlich et al. [2017] considers musical instrument separation, in which an FNN and a bidirectional LSTM based RNN are employed in a system combination fashion to estimate the source magnitude spectra $|\widehat{\mathbf{c}}_j(f, n)|$. These spectra are then used to estimate the source spatial images as:

$$\widehat{\mathbf{c}}_j(f, n) = |\widehat{\mathbf{c}}_j(f, n)| \odot \exp\left(\jmath\angle\mathbf{x}(f, n)\right), \tag{2.55}$$

where $\odot$ denotes the Hadamard product, $\exp(\cdot)$ exponentiation, and $\angle\cdot$ the phase. The multichannel Wiener filter as in (2.46) is then applied as post-filter, where the source PSDs and the spatial covariance matrices are computed as:

$$v_j(f,n) = \frac{1}{2}\left|\widehat{\mathbf{c}}_j(f,n)\right|^2, \tag{2.56}$$

$$\mathbf{R}_j(f) = \frac{\sum_{n=1}^{N} \widehat{\mathbf{c}}_j(f,n)\widehat{\mathbf{c}}_j(f,n)^*}{\sum_{n=1}^{N} v_j(f,n)}. \tag{2.57}$$

2.5.3.3 Directly estimating a multichannel filter

Xiao et al. [2016] consider joint training of DNN based multichannel speech enhancement and DNN based acoustic modeling for robust ASR, in which the DNN parameters of these two components are jointly optimized. The enhancement DNN is used to predict the optimal beamformer weights given the spatial features computed by generalized cross correlation method with phase transform weighting on the multichannel signals. Average pooling across time frames is then applied on these weights to obtain *time-invariant beamformer* weights. The study shows that the proposed approach outperforms traditional delay-and-sum beamformer.

Sainath et al. [2017] also consider joint training of DNN based multichannel speech enhancement and DNN based acoustic modeling for robust ASR. The enhancement is done by convolutional layers (as in convolutional neural network) consisting of filters with particular length which are applied to the multichannel inputs, followed by summing the outputs of convolutional layers across the channels and max pooling across the time frames. This sequence of operations is equivalent to a *time-invariant beamformer*. The study introduces unfactored and factored DNN models. While the unfactored model uses a single convolutional layer, the factored model uses two layers for the enhancement. By setting the lengths of filters in these two layers in a particular way, the first and second layers are constrained to perform time-varying spatial and spectral filtering, respectively. Since the outputs of spectral filtering are still fed to max pooling across the time frames, it also results in time-invariant beamformer. The study introduces the use of raw time-domain waveform and complex-valued time-frequency domain spectra as the inputs of DNNs. It shows that the factored model outperforms the unfactored one in terms of WER. The factored model also performs similarly well for both types of inputs, although processing in the time-frequency domain is preferable because it has a lower computational cost. In addition, the study also experiments with LSTM based RNN for the spatial filtering.

2.5.3.4 Summary

In summary, Jiang et al. [2014] and Araki et al. [2015] show that although the separation is done in single-channel manner, the use of features derived from multichannel data provides better separation performance than that of monaural features in terms of both source separation and speech recognition metrics. However, because of this single-channel filtering, the results are suboptimal compared to the other discussed approaches employing multichannel filtering.

Erdogan et al. [2016], Heymann et al. [2017], and Wang et al. [2017] show that DNNs work well for estimating the parameters of time-invariant minimum variance distortionless response beamforming and GEV beamforming, respectively. Both approaches combine channel masks into a single mask and employ a post-filtering method, which are helpful to improve the beamformer outputs. Xiao et al. [2016] and Sainath et al. [2017] show that DNNs may act as a time-invariant beamformer whose parameters are represented by the DNN parameters (weights and possibly biases).

Because these two approaches jointly train speech enhancement and acoustic model, We may expect that the approaches that jointly train speech enhancement and acoustic model can achieve very good speech recognition performance outperforming the other approaches. However, when a variant of Heymann et al. [2017] and a variant of Xiao et al. [2016] are evaluated on the same task in the context of the 4th CHiME Speech Separation and Recognition Challenge[3], the variant of Heymann et al. [2017] outperforms the variant of Xiao et al. [2016]. This may be caused by the fact that spatial filtering needs a sufficient context. Basically, the networks of Xiao et al. [2016] or Sainath et al. [2017] predict a time-varying beamformer based on the short context whose length depends on the filter size of convolutional neural network or the weights of LSTM based RNN. Although this time-varying beamformer is then pooled to obtain a time-invariant beamformer, this is much less robust compared to the approaches by Erdogan et al. [2016]; Heymann et al. [2017]; Wang et al. [2017] where a time-invariant beamformer is computed once based on the source covariance matrices computed from the whole utterance.

[3]See `http://spandh.dcs.shef.ac.uk/chime_challenge/chime2016/results.html`.

2.6 Positioning of our study

Finally, let us position our study with respect to the other studies discussed above. Referring to the classification presented in Section 2.5.3, the frameworks we propose in our study fall into the second category, namely approaches employing DNNs for estimating intermediate variables, which are used to derive multichannel filters. In our frameworks, the DNNs are used for initializing and updating the spectral and spatial parameters of a multichannel Gaussian model as part of an EM like iterative framework to compute a *time-varying multichannel Wiener filter*. These iterative frameworks can be regarded as extensions of the one presented in Section 2.5.2.2.

The second framework we propose in Chapter 5, in which different DNNs model spectral and spatial parameters, is conceptually similar to the factored model of Sainath et al. [2017], in which different layers perform time-varying spatial and spectral filtering. The obvious difference between these two approaches is that our framework results in a time-varying multichannel Wiener filter, whereas the latter approach results in a time-invariant beamformer. Another important difference is that our approach is motivated by a signal model, while the approach of Sainath et al. [2017] is not. While most studies discussed in Section 2.5.3 specifically focus on speech enhancement for robust ASR, our frameworks are intended for general-purpose source separation. Besides, our main objective is to obtain the multichannel separated target sources, which is essentially different from the other studies which aim to obtain the single-channel speech. This allows us to apply our frameworks to other tasks beyond speech separation, such as singing voice separation and music instrument separation.

CHAPTER 3

Estimation of spectral parameters with deep neural networks

This chapter presents the first deep neural network (DNN) based multichannel audio source separation framework we propose in our study. The spectral parameters are modeled by DNNs and the spatial parameters are estimated in an iterative fashion as in the classical expectation-maximization (EM) based framework presented in Algorithm 1. We address several research questions related to the use of DNNs in this framework and the benefit of multiple iterations. These questions are reflected in the design choices made in the experiments. These choices notably include multiple spatial parameter updates after spectral parameter initialization and the use of multiple DNNs for estimating the spectral parameters at different iterations.

3.1 Research questions

In this chapter, we address the following research questions.

1. **Can DNNs be used to model the spectral parameters within a multichannel Gaussian model based separation framework?** As discussed in Section 2.5.2.2, the spectral parameters of the multichannel Gaussian model are the source power spectral densities (PSDs), which can be represented by the source spectrograms. Various studies have shown that DNNs work well for estimating source spectra in the context of single-channel audio source separation (see Section 2.4.3). We want to confirm that the source spectrograms estimated by DNNs are suitable for source separation within a multichannel Gaussian model based separation framework. This confirmation is the starting point of our study.

2. **Are traditional EM iterations still optimal?** In a traditional EM iteration, the E step and the M step are done alternately. In our case, the E step

corresponds to source separation and the M step corresponds to parameter updates, in which each parameter is re-estimated once. Following this, Algorithm 1 employs a single spatial parameter update and a single spectral parameter update in the M step. We expect that the initial spectrograms provided by the DNNs should be close to the targets already. On the contrary, the initial spatial covariance matrices, which are the identity matrices, are far from the targets. Therefore, we hypothesize that the spatial parameter update has to be done multiple times first before the spectral parameter update.

3. **Does performance increase with multiple iterations?** We speculate that the improvement provided by the multiple spatial parameter updates is bounded by the fixed spectrograms. Therefore, we hypothesize that the spectrograms may be updated, in this case using DNNs, to achieve higher performance. The DNNs for the spectrogram initialization and the spectrogram updates are trained on the same training targets derived from the ground truth. The difference between these two types of DNNs lies in the inputs. Broadly speaking, the DNN for initialization takes the mixture spectrogram and the other DNNs take the intermediate source spectrograms computed in the preceding M step. We do not know whether these intermediate source spectrograms provide better information to the DNNs than the mixture spectrogram does for estimating better source spectrograms. Thus, it is not obvious whether the updated spectrograms can provide a performance improvement. It is also not obvious whether these spectrograms allow the following spatial parameter updates to improve the performance.

4. **Does performance increase with a single DNN used for multiple spectrogram updates?** This is a follow-up question to the one above. Ideally, the spectrogram update of different EM iterations should be done by different DNNs. This will be costly in terms of training time when the number of EM iterations is set to be high so that there are many DNNs to be trained. Although the testing time would not be affected, it would be more practical if a single DNN trained on the intermediate source spectrograms of an EM iteration worked for multiple spectrogram updates. This might not work because there is most likely an input mismatch problem due to the varying intermediate source spectrograms over iterations. However, this remains to be investigated because DNNs are often able to cope with variations of the inputs.

These research questions are investigated by employing the framework described in the following section.

3.2 Iterative framework with spectral DNNs

In general, the proposed framework described in Algorithm 2 modifies the framework in Algorithm 1 so that the source spectral parameters $v_j(f, n)$ are estimated by DNNs [Nugraha et al., 2016a]. We also modify how the iterations are done to allow us to address our research questions. For each EM iteration, we allow multiple spatial parameter updates in the M step. The number of EM iterations is denoted by L with l as the iteration index and the number spatial parameter updates is denoted by K with k as the iteration index. The number of spatial parameter updates K is relevant to the research question 2. If $K = 1$, the algorithm follows the M step of the traditional EM iteration. The number of EM iterations L is relevant to the research questions 3 and 4. Further discussion about pre-processing, initialization, and multichannel filtering steps is presented in Section 3.3.2.

The framework is designed to use a single DNN or multiple DNNs for modeling the source spectra. In the single DNN case, the DNN is used for spectrogram initialization (without any following spectrogram fitting). In the multiple DNN case, one DNN is used for spectrogram initialization and one or more DNNs are used for spectrogram fitting. Ideally, different DNNs are trained for spectrogram fitting at different iterations. Thus, the maximum number of DNNs for spectrogram fitting equals the number of iterations L.

Let $\text{DNN}_0^{\text{spec}}$ and $\text{DNN}_l^{\text{spec}}$ be the DNNs used for spectrogram initialization and spectrogram fitting, respectively. $\text{DNN}_0^{\text{spec}}$ estimates the source spectra from the observed mixture and $\text{DNN}_l^{\text{spec}}$ aims to improve the source spectra estimated at iteration l. $\text{DNN}_0^{\text{spec}}$ and $\text{DNN}_l^{\text{spec}}$ estimate the spectra of all sources simultaneously. This is similar to the DNNs used in the context of single-channel source separation by Tu et al. [2014]; Huang et al. [2014a, 2015]. Additionally, $\text{DNN}_l^{\text{spec}}$ might be regarded as similar to the DNNs used in the context of single-channel speech enhancement by Liu et al. [2014]; Xu et al. [2014], where DNNs are used to estimate clean spectra from the corresponding noisy spectra. However, $\text{DNN}_l^{\text{spec}}$ is indeed different from the DNNs in these studies because $\text{DNN}_l^{\text{spec}}$ handles 'noises' due to a prior separation step.

Algorithm 2 Iterative framework with spectral DNNs

```
Inputs:
    STFT of mixture $\mathbf{x}(f,n)$
    Number of sources $J$
    Number of spatial updates $K$ and number of EM iterations $L$
    DNN spectral models $\mathrm{DNN}_0^{\mathrm{spec}}, \mathrm{DNN}_1^{\mathrm{spec}}, \ldots, \mathrm{DNN}_L^{\mathrm{spec}}$
Pre-processing step:
 1: Align the observed mixture: $\overline{\mathbf{x}}(f,n) \leftarrow \mathrm{align}(\mathbf{x}(f,n))$
 2: Extract features for the DNN inputs: $\sqrt{z_x(f,n)} \leftarrow \mathrm{feat}(\overline{\mathbf{x}}(f,n))$
Initialization step:
 3: Initialize all source spectrograms:
        $[v_1(f,n), \ldots, v_J(f,n)] \leftarrow \mathrm{DNN}_0^{\mathrm{spec}}\left(\sqrt{z_x(f,n)}\right)^2$
 4: for each source $j$ of $J$ do
 5:     Initialize the spatial covariance matrix: $\mathbf{R}_j(f) \leftarrow$ identity matrix $\mathbf{I}$
 6: end for
Multichannel filtering step:
 7: for each EM iteration $l$ of $L$ do
 8:     for each spatial update $k$ of $K$ do
 9:         Compute the mixture covariance matrix:
                $\mathbf{R}_{\mathbf{x}}(f,n) = \sum_{j=1}^{J} v_j(f,n)\mathbf{R}_j(f)$    (2.44)
10:         for each source $j$ of $J$ do
11:             Compute the Wiener filter gain:
                    $\mathbf{W}_j(f,n) = v_j(f,n)\mathbf{R}_j(f)\mathbf{R}_{\mathbf{x}}(f,n)^{-1}$    (2.47)
12:             Compute the spatial image:
                    $\widehat{\mathbf{c}}_j(f,n) = \mathbf{W}_j(f,n)\overline{\mathbf{x}}(f,n)$    (2.46)
13:             Compute the posterior second-order raw moments of the
                    spatial image:
                    $\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n) = \widehat{\mathbf{c}}_j(f,n)\widehat{\mathbf{c}}_j(f,n)^*$
                                 $+ (\mathbf{I} - \mathbf{W}_j(f,n))\, v_j(f,n)\mathbf{R}_j(f)$    (2.48)
14:             Update the spatial covariance matrix:
                    $\mathbf{R}_j(f) = \frac{1}{N}\sum_{n=1}^{N} v_j(f,n)^{-1}\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)$    (2.49)
15:         end for
16:     end for
```

(continued on the next page)

Algorithm 2 Iterative framework with spectral DNNs (continued)

```
17:     for each source $j$ of $J$ do
18:         Compute the unconstrained source spectrogram:
                $z_j(f,n) = \frac{1}{I}\mathrm{tr}\left(\mathbf{R}_j^{-1}(f)\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)\right)$    (2.50)
19:     end for
20:     Update all source spectrograms:
            $[v_1(f,n),\dots,v_J(f,n)] \leftarrow \mathrm{DNN}_l^{\mathrm{spec}}\left(\left[\sqrt{z_1(f,n)},\dots,\sqrt{z_J(f,n)}\right]\right)^2$
21: end for
22: for each source $j$ of $J$ do
23:     Compute the final spatial image:
            $\widehat{\mathbf{c}}_j(f,n) = \frac{v_j(f,n)\mathbf{R}_j(f)}{\sum_{j'=1}^{J} v_{j'}(f,n)\mathbf{R}_{j'}(f)}\overline{\mathbf{x}}(f,n)$    (2.46)
24: end for
Outputs:
    All spatial source images $[\widehat{\mathbf{c}}_1(f,n),\dots,\widehat{\mathbf{c}}_J(f,n)]$
```

In this chapter, we consider features in the magnitude short-time Fourier transform (STFT) domain as the inputs and outputs of DNNs. The inputs of $\mathrm{DNN}_0^{\mathrm{spec}}$ and $\mathrm{DNN}_l^{\mathrm{spec}}$ are denoted by $\sqrt{z_x(f,n)}$ and $\sqrt{z_j(f,n)}$, respectively. The outputs of both types of DNNs are denoted by $\sqrt{v_j(f,n)}$ and the training targets are denoted by $\sqrt{\widetilde{v}_j(f,n)}$. $\mathrm{DNN}_0^{\mathrm{spec}}$ takes the magnitude spectrum $\sqrt{z_x(f,n)}$ and yields the initial magnitude spectra $\sqrt{v_j(f,n)}$ for all sources simultaneously. Then, $\mathrm{DNN}_l^{\mathrm{spec}}$ takes the estimated magnitude spectra $\sqrt{z_j(f,n)}$ of all sources and yields the improved magnitude spectra $\sqrt{v_j(f,n)}$ for all sources simultaneously. The use of the square-root of $v_j(f,n)$ and $z_j(f,n)$ to represent the magnitude spectra comes from the fact that these two parameters are PSDs. The source PSDs are simply the squared DNN outputs.

The following section describes an implementation of this framework for a speech enhancement task, on which the research questions in this chapter are investigated. Another implementation for a music separation task is presented in the following chapter.

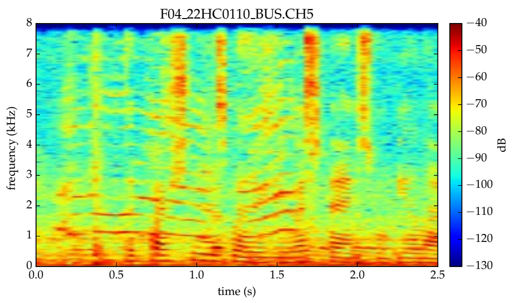
3.3 Experimental settings

3.3.1 Task and dataset

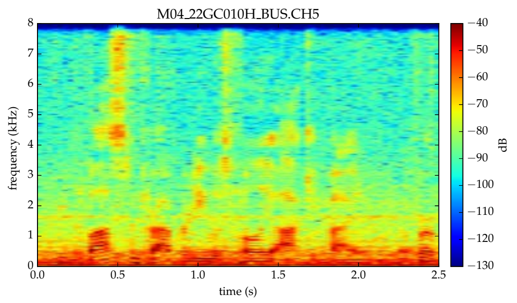
In order to address the research questions, we consider the application of the framework in Algorithm 2 to a multichannel speech enhancement problem. The dataset we used is that of the 3rd CHiME Speech Separation and Recognition Challenge, also known as the 'CHiME-3 challenge' [Barker et al., 2015]. The same dataset was also used in the follow-up 'CHiME-4 challenge' [Vincent et al., 2017b]. To be concise, let us simply call this dataset as the 'CHiME dataset' hereafter. Both CHiME-3 and CHiME-4 consider the use of an automatic speech recognition (ASR) system in real-world noisy environments on a multi-microphone tablet device. The dataset provides real and simulated 6-channel microphone array data in 4 varied environmental noise settings (bus, cafe, pedestrian area, and street junction) divided into training, development, and test sets. The training set consists of 1,600 real and 7,138 simulated utterances (`tr05_real` and `tr05_simu`), the development set consists of 1,640 real and 1,640 simulated utterances (`dt05_real` and `dt05_simu`), while the test set consists of 1,320 real and 1,320 simulated utterances (`et05_real` and `et05_simu`). The utterances are taken from the 5k vocabulary subset of the Wall Street Journal corpus [Garofalo et al., 2007]. All data are sampled at 16 kHz.

Figure 3.1 shows example spectrograms of channel 5 of two different noisy speech recordings for each environment type. Each pair of recordings are recorded in real-world environments on different days and in different places. Consequently, the noise in a recording may be significantly different from that in another recording although both recordings are from the same environment type. For example, Figure 3.1d shows a recording in a relatively quite cafe in which some glassware sounds can be heard and Figure 3.1c shows a recording in a much noisier cafe where babble noise caused by other people talking in the background is dominant. In general, we consider that the noise is non-stationary. For some cases, the noise tends to be stationary as in Figure 3.1b, in which the noise is dominated by the low-frequency constant humming sound from the bus engine. However, there might be other non-stationary noises inside the bus, such as a distinctive child shouting shown by Figure 3.1a.

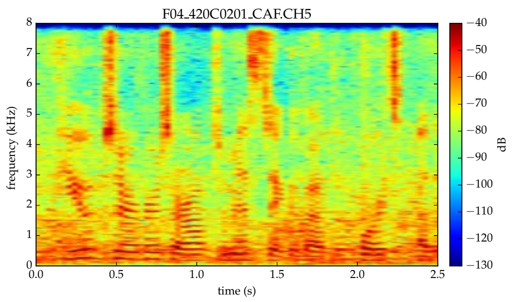
In this task, we deal with the separation of two sources ($J = 2$), namely speech and noise, from a 6-channel mixture ($I = 6$). The source separation



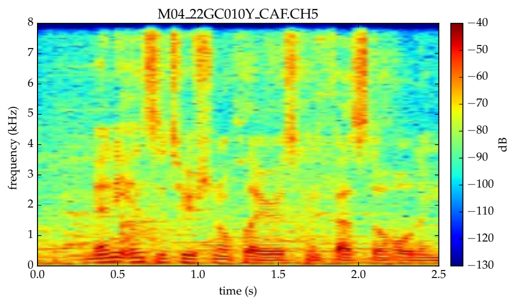
(a) Environment: bus



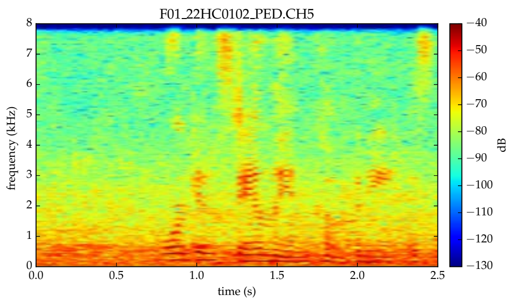
(b) Environment: bus



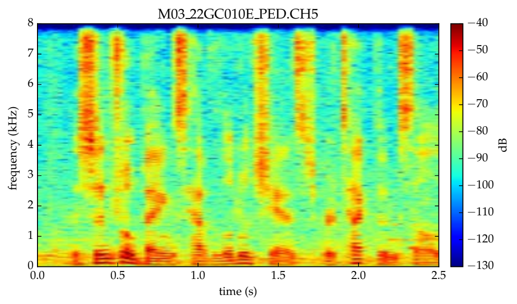
(c) Environment: cafe



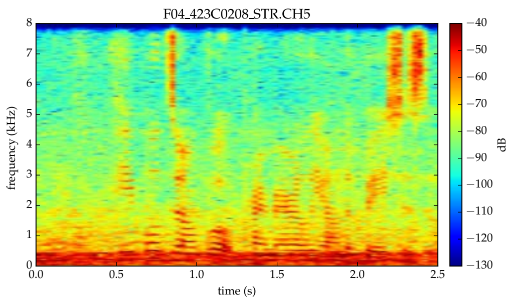
(d) Environment: cafe



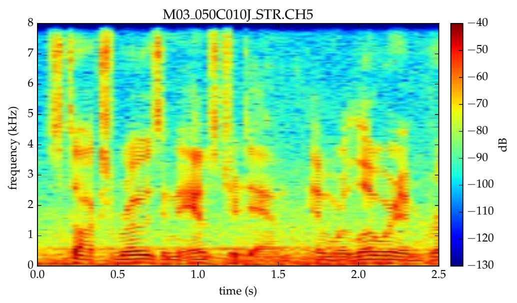
(e) Environment: pedestrian area



(f) Environment: pedestrian area

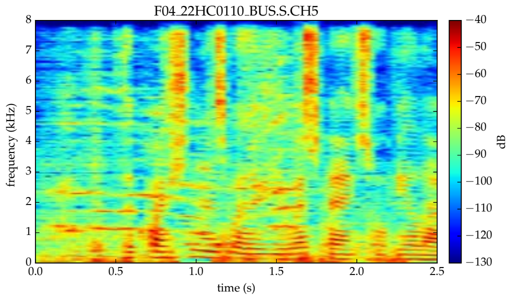


(g) Environment: street junction

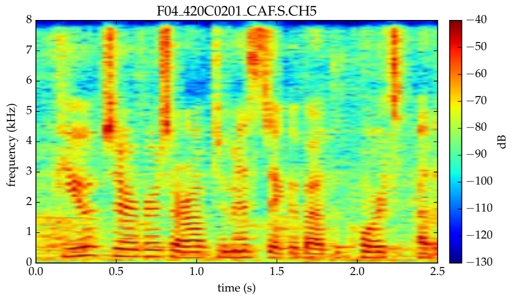


(h) Environment: street junction

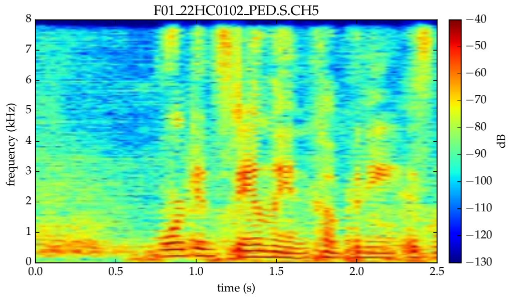
Figure 3.1: Example power spectrograms of channel 5 of two different noisy speech recordings for each environment type. The recordings are taken from the `dt05_real` set.



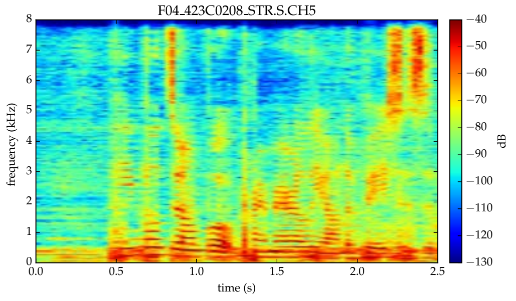
(a) Est. ground truth for Figure 3.1a



(b) Est. ground truth for Figure 3.1c



(c) Est. ground truth for Figure 3.1e



(d) Est. ground truth for Figure 3.1g

Figure 3.2: Example power spectrograms of channel 5 of the estimated *speech* ground truth for some recordings in Figure 3.1.

performance is evaluated using BSS Eval (see Section 2.1) on the multichannel separated speech and the speech recognition performance is evaluated using the word error rate (WER) (see Section 2.2) on the single-channel version of this multichannel separated speech. Instead of using the original BSS Eval written in Matlab, we use a re-implementation of it in Python which is more convenient for our experiment pipelines.

The multichannel ground truth speech and noise signals for the simulated set are available in the CHiME dataset, while those for the real set are extracted using the baseline simulation tool provided by the challenge organizers [Barker et al., 2015]. The extraction for the real set is possible because of the availability of close-talking microphone recordings. It is done based on the estimation of the impulse responses between the close-talking microphone and the microphones on the tablet device. The extraction results are good enough for DNN training (see Figure 3.2). However, they are not reliable enough to be used in any source separation performance evaluation. As an illustration, the child shout in Figure 3.1a is still present in the estimated ground truth in Figure 3.2a. Therefore, the source separation performance evaluation is only done on the simulated set. On the other hand, the speech recognition performance evaluation relies on the ground truth transcriptions and does not rely on the above ground truth signals. Since these transcriptions are reliable and the speech recognition performance for real recordings is more interesting indeed, we mainly discuss the performance on the real set. However, it should be mentioned that the performance on the simulated set may affect that on the real set because some ASR hyperparameters, e.g., the language model weight, are chosen based on the overall performance on the development set including the real and simulated sets [Hori et al., 2015].

The following subsection presents an implementation of the proposed source separation framework for a speech enhancement task.

3.3.2 An overview of the speech enhancement system

Any system implementing the framework follows three main steps, i.e., pre-processing, initialization, and multichannel filtering, with an optional post-processing step. Although these steps are common for all systems, the implementation details of the steps may be different from a system used in a set of experiments to another system used in another set of experiments. The following explanation provides a brief general description of each step and the details specific to the speech enhancement system depicted in Figure 3.3 used in all experiments presented in this chapter.

The pre-processing step aligns the observed mixture and extracts relevant features to be used as DNN inputs from, e.g., the complex-valued STFT coefficients of the multichannel mixture signal or another time-frequency representation. The alignment is required to satisfy the model in (2.43) that assumes that the sources do not move over time, which translates into the use of *time-invariant* spatial covariance matrices.

In this chapter, the STFT coefficients are extracted using a Hamming window of length 1024 and hopsize 512 resulting $F = 513$ frequency bins. The time differences of arrivals between the speaker's mouth and each of the microphones are first measured using the provided baseline speaker localization tool [Barker et al., 2015], which relies on a nonlinear variant of the steered response power with phase transform method [Loesch & Yang, 2010; Blandin et al., 2012]. All channels are then aligned with each other by shifting the phase of the input noisy signal $\mathbf{x}(f, n)$ in all time-frequency bins (f, n) by the opposite of the

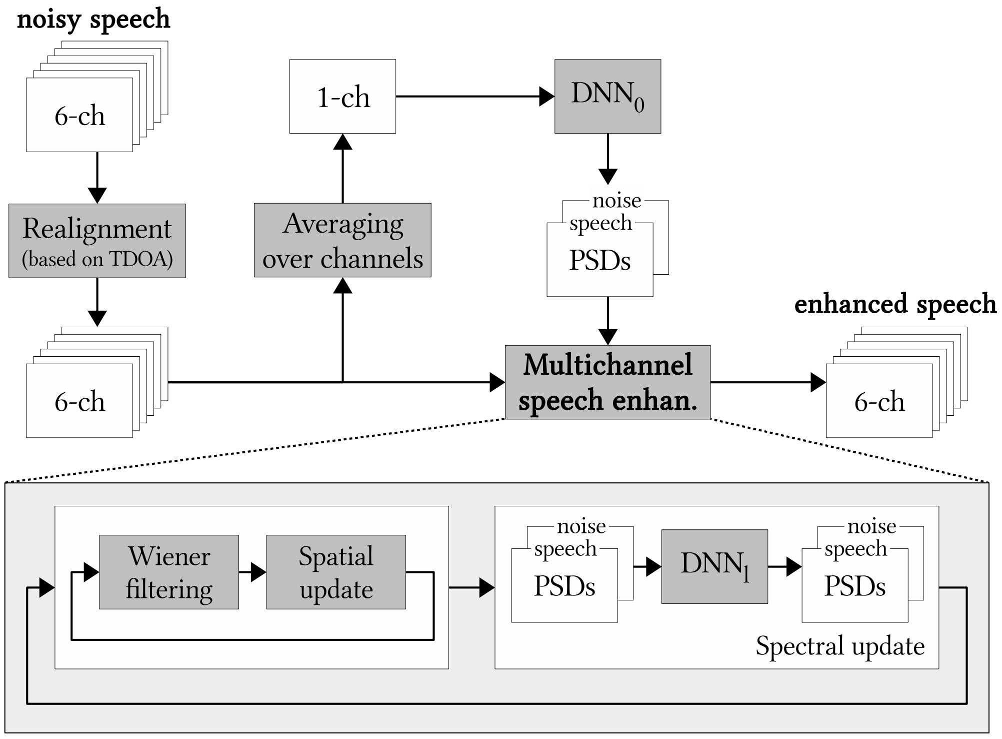


Figure 3.3: Proposed DNN based multichannel speech enhancement system.

measured time difference of arrival. Afterwards, we obtain a single-channel signal by averaging the realigned channels together. This combination of time alignment and channel averaging is equivalent to delay-and-sum beamforming (see Section 2.5.1). We describe in the following subsection how the resulting single-channel mixture, denoted by $\widetilde{x}(f,n)$, is used to compute the DNN inputs $\sqrt{z_x(f,n)}$.

The initialization step sets the initial spectral and spatial parameters.

In this chapter, the source spectrograms are initialized using $\text{DNN}_0^{\text{spec}}$. Besides, the source spatial covariance matrices are initialized as identity matrices $\mathbf{I} \in \mathbb{C}^{I\times I}$.

The multichannel filtering step iteratively performs the spectral and spatial parameter updates as defined by the framework.

In this chapter, the iterative parameter updates follow Algorithm 2, in which $\text{DNN}_l^{\text{spec}}$ is employed for spectrogram fitting at iteration l. In order to avoid numerical instabilities due to the use of single precision computing for DNNs, the spectrograms $v_j(f,n)$ are floored to 10^{-5} in the parameter update iterations.

The (optional) post-processing step applies some procedure on the estimated multichannel source spatial images required for the target application.

In this chapter, the estimated multichannel speech spatial image is averaged across channels to obtain a single-channel signal for the ASR evaluation. This is equivalent to delay-and-sum beamforming since the mixture $\mathbf{x}(f, n)$ has been realigned in the pre-processing step and thus, the separated multichannel spatial images are also aligned. This provided better ASR performance than the use of one of the channels.

3.3.3 DNN spectral models

This subsection presents the architecture, the inputs and outputs, the training criterion, the training algorithm, and the training data of the DNN spectral models $\text{DNN}_0^{\text{spec}}$ and $\text{DNN}_l^{\text{spec}}$. Recall that $\text{DNN}_0^{\text{spec}}$ is used in the initialization step and $\text{DNN}_l^{\text{spec}}$ is used in the multichannel filtering step presented in the previous subsection.

3.3.3.1 Architecture

The DNNs used in this chapter follow an multilayer perceptron architecture (see Section 2.4.3.1). In general, the number of hidden layers and the number of units in each input or hidden layer may vary, but the number of units in the output layer equals the dimension of spectra multiplied by the number of sources. $\text{DNN}_0^{\text{spec}}$ and $\text{DNN}_l^{\text{spec}}$ have an input layer, three hidden layers, and an output layer [Jaureguiberry et al., 2016]. Both types of DNNs have hidden and output layers sizes of $F \times J = 1026$. $\text{DNN}_0^{\text{spec}}$ has an input layer size of $F = 513$ and $\text{DNN}_l^{\text{spec}}$ of $F \times J = 1026$. The activation functions of the hidden and output layers are rectified linear units [Nair & Hinton, 2010; Glorot et al., 2011].

3.3.3.2 Inputs and outputs

In order to provide temporal context, the input frames are concatenated into *supervectors* consisting of a center frame, left context frames, and right context frames. In choosing the context frames, we use every second frame relative to the center frame in order to reduce the redundancies caused by the windowing of STFT. Although this causes some information loss, this enables the supervectors to represent a longer context [Nugraha et al., 2014; Uhlich

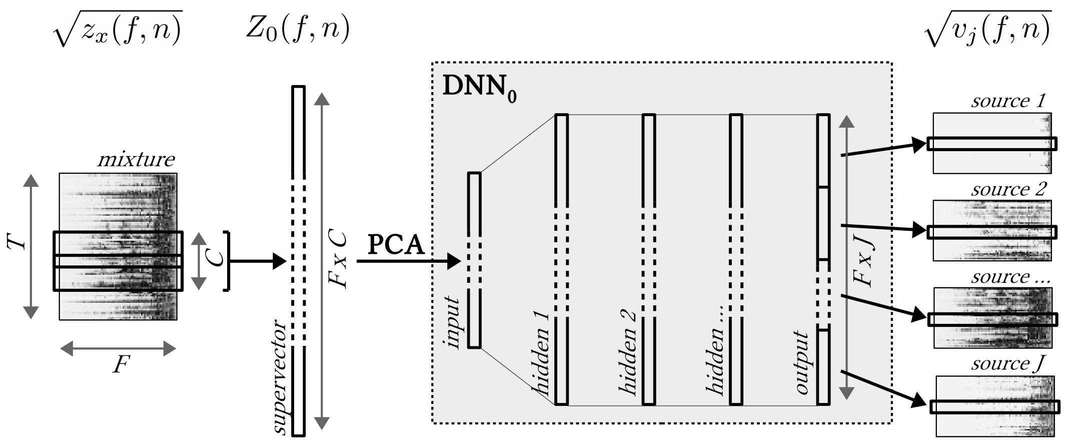


Figure 3.4: Illustration of the inputs and outputs of the DNN for spectrogram initialization. PCA denotes dimensionality reduction by principal component analysis. Inputs: magnitude spectrum of the mixture (left). Outputs: magnitude spectra of the sources (right). In this chapter, $J = 2$.

et al., 2015]. In addition, we do not use the magnitude spectra of the context frames directly, but the difference of magnitude between the context frames and the center frame. These differences act as complementary features similar to delta features. Following Weninger et al. [2014], let $\sqrt{z_x(f,n)} = |\widetilde{x}(f,n)|$ be the input frames of $\text{DNN}_0^{\text{spec}}$. The supervector can be expressed as

$$Z_0(f,n) = \begin{bmatrix} |\widetilde{x}(f,n-2c)| - |\widetilde{x}(f,n)| \\ \vdots \\ |\widetilde{x}(f,n-2)| - |\widetilde{x}(f,n)| \\ |\widetilde{x}(f,n)| \\ |\widetilde{x}(f,n+2)| - |\widetilde{x}(f,n)| \\ \vdots \\ |\widetilde{x}(f,n+2c)| - |\widetilde{x}(f,n)| \end{bmatrix} \tag{3.1}$$

where c is the one-sided context length in frames. In the following experiments, we considered $c = 2$, so that the supervectors for the input of the DNNs were composed by 5 time frames (2 left context, 1 center, and 2 right context frames). The supervector for $\text{DNN}_l^{\text{spec}}$, $Z_l(f,n)$, is constructed in a similar way where a stack of $\sqrt{z_j(f,n)}$ is used as input instead of $|\widetilde{x}(f,n)|$.

The dimension of the supervectors is reduced by principal component analysis to the dimension of the DNN input. Dimensionality reduction by principal component analysis significantly minimizes the computational cost of DNN training with a negligible effect on the performance of DNN provided

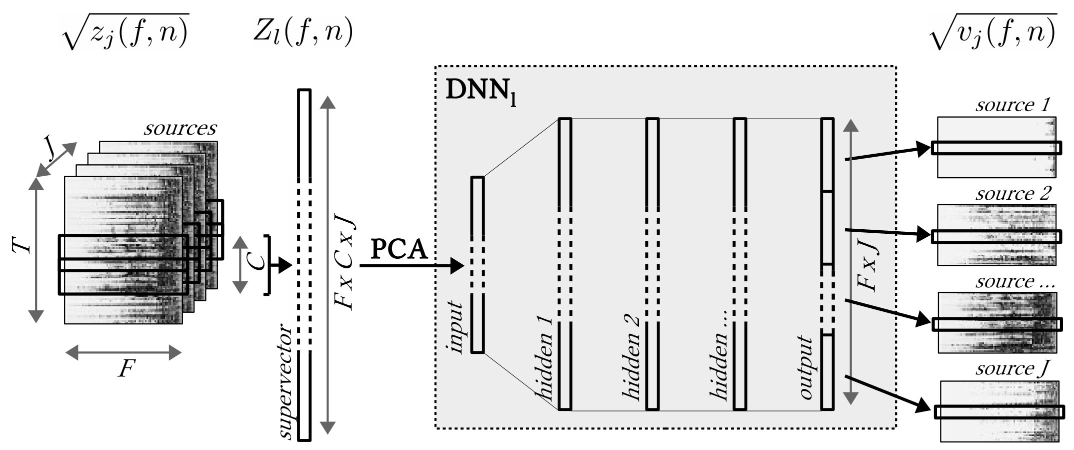


Figure 3.5: Illustration of the inputs and outputs of the DNNs for spectrogram fitting. PCA denotes dimensionality reduction by principal component analysis. Inputs: stack of magnitude spectra of all sources (left). Outputs: magnitude spectra of the sources (right). In this chapter, $J = 2$.

enough components are kept [Jaureguiberry et al., 2016]. Standardization (zero mean, unit variance) is done dimension-wise before and after principal component analysis. The standardization factors and the principal component analysis transformation matrix are computed on the whole training data and then kept for pre-processing of any input, i.e., the training, the validation, and the test data. Thus, strictly speaking, the inputs of DNNs are not the supervectors of magnitude spectra $Z_0(f, n)$ and $Z_l(f, n)$, but their transformation into reduced dimension vectors.

In this chapter, the supervector construction is done on each utterance without any padding. Let the utterance time frame $n \in \{0, 1, \ldots, N-1\}$ and the one-sided context length $c = 2$, the construction starts at $n = 2c = 4$ and ends at $n = N - 1 - 2c = N - 5$. By doing so, an utterance with length of N frames result in supervectors with length of $N - 4c$ frames. Using these supervectors as the DNN inputs, consequently, the DNN outputs have fewer frames than the input utterance. In order to obtain the same number of frames at testing time, we simply mirror the beginning and the end of DNN outputs, which means we may introduce energy that are not supposed to be in the spectrogram. This caused a minor flaw in the final spectrogram estimates. We show example spectrograms later in Section 3.4. Better edge handling method includes applying zero padding before the supervector construction, so that the number of supervectors is the same as the number of frames of the input utterance.

Figures 3.4 and 3.5 illustrates the inputs and outputs of the DNNs for spectrogram initialization and spectrogram fitting, respectively. F denotes the dimension of the spectra, $C = 2c+1$ the context length, and J the number of sources. In this chapter, $F = 513$, $C = 5$, and $J = 2$.

3.3.3.3 Training criterion

The DNNs are trained using the generalized Kullback-Leibler (KL) divergence [Lee & Seung, 2000] with an ℓ_2 weight regularization term as the cost function:

$$\mathcal{C} = \mathcal{D}_{\overline{\text{KL}}} + \mathcal{D}_{\ell_2}. \tag{3.2}$$

The generalized KL divergence $\mathcal{D}_{\overline{\text{KL}}}$ is implemented with a regularization parameter $\delta_{\text{cf}} = 10^{-3}$ as

$$\begin{aligned}\mathcal{D}_{\overline{\text{KL}}} = \frac{1}{JFN} \sum_{j,f,n} \Bigg(\left(\sqrt{\widetilde{v}_j(f,n)} + \delta_{\text{cf}} \right) \log \frac{\sqrt{\widetilde{v}_j(f,n)} + \delta_{\text{cf}}}{\sqrt{v_j(f,n)} + \delta_{\text{cf}}} \\ - \sqrt{\widetilde{v}_j(f,n)} + \sqrt{v_j(f,n)} \Bigg). \end{aligned} \tag{3.3}$$

The use of a regularization parameter δ_{cf} in the logarithm computation is a common practice to avoid numerical instabilities [Lefèvre et al., 2011; Sprechmann et al., 2015]. The ℓ_2 weight regularization term $\mathcal{D}_{\ell_2}$ is expressed as

$$\mathcal{D}_{\ell_2} = \frac{\lambda_{\ell_2}}{2} \sum w_{\text{DNN}}^2, \tag{3.4}$$

where w_{DNN} are the DNN weights and the regularization parameter is fixed to $\lambda_{\ell_2} = 10^{-5}$. This parameter was empirically determined so that the sum of the squared weights changes steadily along with epochs, and both the training cost and the validation error improve in a similar manner, especially in the beginning of training. This ℓ_2 weight regularization is commonly used for preventing overfitting [Bengio, 2012]. No regularization is applied to the biases.

It is worth mentioning that the use of the generalized KL divergence here is motivated by our study on various cost functions presented in Chapter 4. The study shows that this KL divergence is a favorable choice because it outperforms the other four considered cost functions in terms of source separation metrics. We assess the impact of this choice in Chapter 4.

3.3.3.4 Training algorithm

Following He et al. [2015], the DNN weights are initialized randomly from a zero-mean Gaussian distribution with standard deviation of $\sqrt{2/n_l}$, where n_l is the number of inputs to the neurons in layer l and, in this case, equals the size of the previous layer. The biases are initialized to zero.

The DNNs are trained by greedy layer-wise supervised training [Bengio et al., 2006] where the hidden layers are added incrementally. In the beginning, a network with one hidden layer is trained after random weight initialization. The output layer of this trained network is then substituted by a new hidden layer and a new output layer to form a new network with one more layer. The parameters of the existing hidden layer are kept. Thus, we can view this as a pre-training method for the training of a new deeper network. After random initialization for the parameters of the new layers, the new network is entirely trained. This procedure is done iteratively until the target number of hidden layers has been reached.

Training is done by backpropagation with a minibatch size of 100 and the AdaDelta parameter update algorithm whose hyperparameters are set to $\rho = 0.95$ and $\epsilon = 10^{-6}$, following Zeiler [2012]. The validation error is computed every epoch and the training is stopped after 10 consecutive epochs failed to obtain better validation error. The latest model which yields the best validation error is kept. Besides, the maximum number of training epochs is set to 100.

The experiments were started when high-level and user-friendly neural network libraries, such as Keras [Chollet et al., 2015] and Lasagne [Dieleman et al., 2015], were not released yet or still in their very early stage of development. Therefore, the DNNs and the related techniques for all experiments presented in this chapter were implemented using Theano [Bergstra et al., 2010; Theano Dev Team, 2016].

3.3.3.5 Training data

The DNNs used for the source separation evaluation were trained on both the real and simulated training sets (`tr05_real` and `tr05_simu`) with the real and simulated development sets (`dt05_real` and `dt05_simu`) as validation data. Conversely, we trained the DNNs used for the speech recognition evaluation on the real training set only (`tr05_real`) and validated them on the real development set only (`dt05_real`). These DNNs were also used for the performance comparison to the non-negative matrix factorization (NMF) based iterative EM framework. Experience shows that using only the real sets provides better WER for the real test set, which is the main point of interest, than using both the real and simulated sets. See Sivasankaran et al. [2015] and Section 4.4 for the perfomance comparison between these two different training settings.

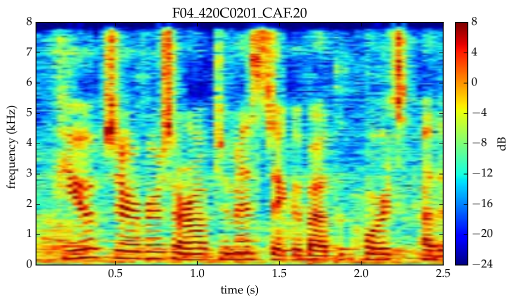
In order to obtain the DNN training targets, we apply the same delay-and-sum beamforming as used for the pre-processing step (see Section 3.3.2) to the multichannel target source spatial images. The resulting single-channel source signals are denoted by $\widetilde{c}_j(f,n)$ and they are related to the DNN training targets by $\sqrt{\widetilde{v}_j(f,n)} = |\widetilde{c}_j(f,n)|$. Recall that the ground truth signals for the real set are not perfect, thus the training targets for the real set are not as clean as they should be. This is illustrated in Figure 3.6. This figure shows the same utterance as the one shown in Figures 3.1c and 3.2b. It should be noted that Figure 3.2b shows the spectrogram of channel 5 of the estimated speech ground truth, while Figure 3.6a shows that of delay-and-sum beamforming output applied on the 6-channel estimated speech ground truth. Thus, it is understandable that the spectrogram shown in Figure 3.6a is better than that in Figure 3.2b.

3.4 Source spectra estimation

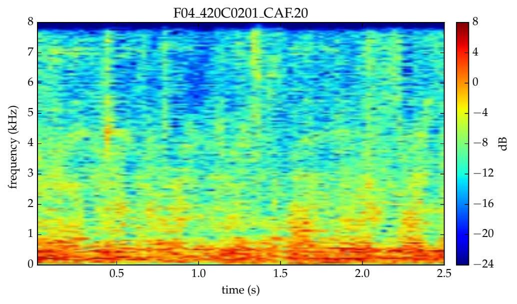
Figure 3.6 shows example magnitude spectrograms of the estimated ground truth and the $\text{DNN}_0^{\text{spec}}$ outputs for an utterance taken from the `dt05_real` set.

In general, the DNN is able to provide good estimation of the source spectra. Classically, and similarly to any other spectral modeling technique, the estimates are a smoothed version of the ground truth. However, the speech harmonics are well preserved.

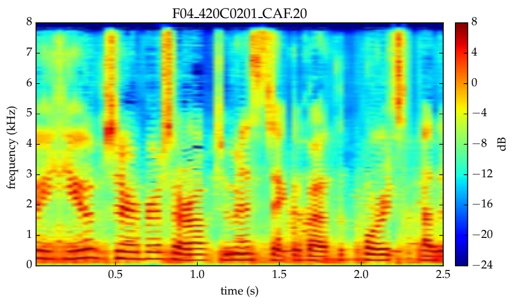
Using the same figure, we also want to show a minor flaw due to the edge handling method we discussed in Section 3.3.3.2. The flaw, i.e., the mirrored spectra, can be observed in the first 0.3 s of Figure 3.6c. We can only observe the one in the beginning here because we do not show the whole utterance. We did not investigate the impact of this flaw on the performance, but we believe that this does not have a detrimental effect because for each utterance the flaw only occurs in the first and the last $2c = 4$ frames. As comparison, the average utterance length of the `tr05_real` and the `dt05_real` sets are 205 and 189 frames, respectively.



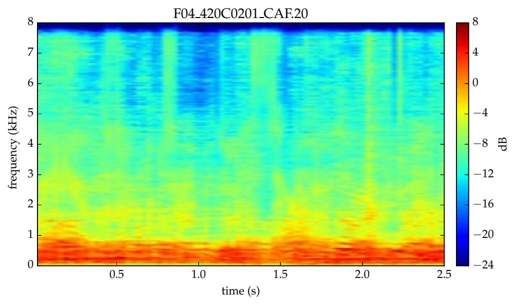
(a) Target speech spectra



(b) Target noise spectra



(c) Estimated speech spectra



(d) Estimated noise spectra

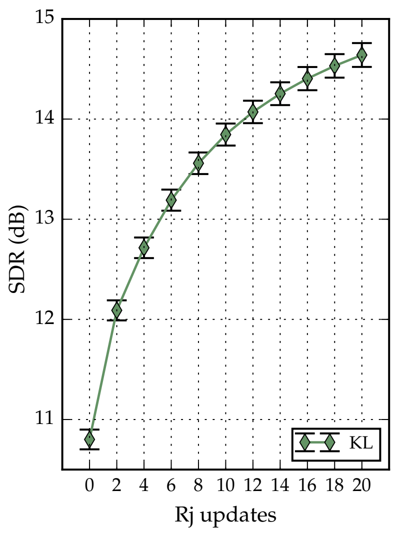
Figure 3.6: Example magnitude spectrograms of $\text{DNN}_0^{\text{spec}}$ training targets and outputs. These are for the same utterance as the one shown in Figures 3.1c and 3.2b.

In the following section, the source spectra estimated by $\text{DNN}_0^{\text{spec}}$ are used to estimate the source spatial covariance matrices.

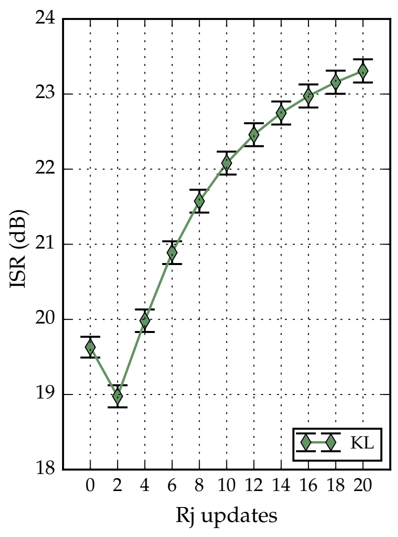
3.5 Impact of spatial parameter updates

Figure 3.7 shows the performance comparison for different numbers of spatial updates. In this case, the spectral parameters $v_j(f,n)$ are initialized by $\text{DNN}_0^{\text{spec}}$ and kept fixed during the iterative spatial parameter updates. In other words, the iteration only updates the spatial covariance matrices $\mathbf{R}_j(f)$. This is done by setting $L = 1$ and varying the number of spatial updates K, while ignoring the computation of $z_j(f,n)$ and the spectral parameters update (lines 17-20) in Algorithm 2. The performance metrics were computed on the resulting 6-channel estimated speech signals. The x-axis of each chart corresponds to the spatial update k. Thus, $k = 0$ is equivalent to single-channel source separation for each channel.

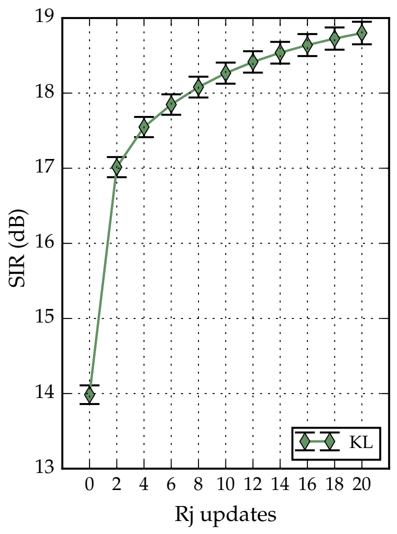
In general, the performance increases along with spatial updates. The decrease of source-image-to-spatial-distortion ratio (ISR) in the beginning



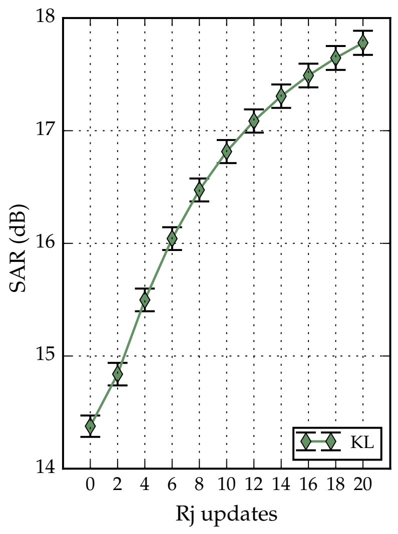
(a) Signal-to-distortion ratio (SDR)



(b) Source-image-to-spatial-distortion ratio (ISR)



(c) Signal-to-interference ratio (SIR)



(d) Signal-to-artifacts ratio (SAR)

Figure 3.7: Source separation performance for various numbers of spatial updates. The PSDs $v_j(f,n)$ are estimated by $\mathrm{DNN}_0^{\mathrm{spec}}$ and kept fixed during the iterative updates of the spatial covariance matrices $\mathbf{R}_j(f)$. The evaluation was done on the simulated test set (`et05_simu`). The figures show the mean value and the 95% confidence interval of each metric. Higher is better.

of spatial updates might be caused by the poor initialization of $\mathbf{R}_j(f) = \mathbf{I}$. However, it should be noted that the absolute value is still very good anyway.

By observing the confidence intervals, we can see that at some point signal-to-interference ratio (SIR) need more spatial updates to obtain a statistically significant improvement, compared to the other metrics. This implies that at some point it is getting harder to reduce the noise, compare to reduce the artifacts, the spatial error, or the overall distortion. This also indicates that the spatial updates are not improving all the metrics in the same degree. Therefore, beside the computational constraints, the setting of K should be determined based on whether it allows us to reach a target performance, e.g., signal-to-distortion ratio (SDR) is more than 14 dB.

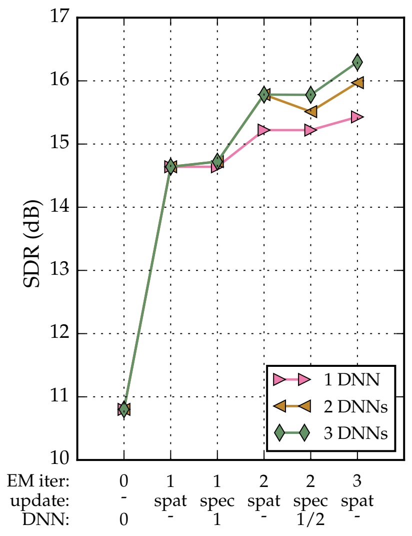
Although all metrics keep improving when k reaches 20, we may expect that these saturate not long after $k = 20$. The following section investigates whether the spectral parameters can be improved so that we can obtain further performance improvement by spatial updates.

3.6 Impact of spectral parameter updates

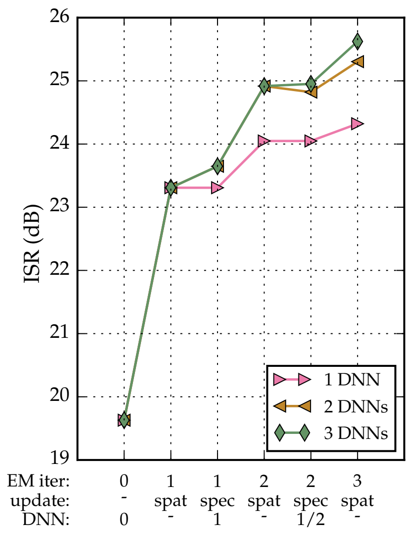
Figure 3.8 shows the performance comparison for different numbers of iterations after fixing the number of spatial updates to $K = 20$. The x-axis shows the index of EM iteration l, the update type (spatial or spectral), and the DNN index. Thus, $l = 0$ is equivalent to single-channel source separation, while $l = 1$ with spatial updates is equivalent to the final performance shown in Figure 3.7.

We trained two additional DNNs for spectrogram fitting, i.e., $\text{DNN}_1^{\text{spec}}$ and $\text{DNN}_2^{\text{spec}}$ for $l = 1$ and $l = 2$, respectively. This allowed us to try different settings for the iterative procedure:

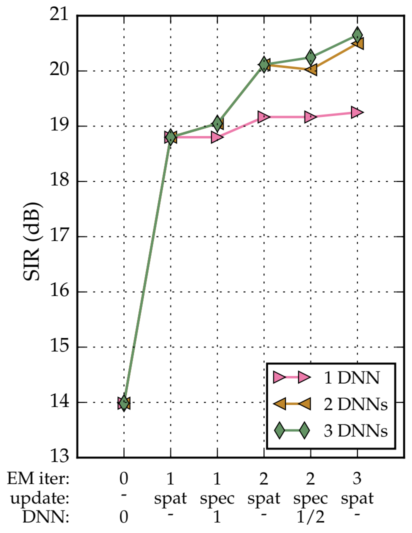
"1 DNN" uses only $\text{DNN}_0^{\text{spec}}$ for spectrogram initialization. The PSDs are kept fixed during three times, corresponding to the number of EM iterations, of spatial updates. Thus, this is equivalent to spatial updates with $K = 60$ after spectrogram initialization. Recall that $\text{DNN}_0^{\text{spec}}$ cannot be used for spectrogram fitting because its architecture only accommodates a spectrogram, i.e., the mixture spectrogram, as the input. Also, it should not be used for spectrogram fitting because it is trained for different purpose, i.e., for separating the mixture spectrogram into the source spectrograms.



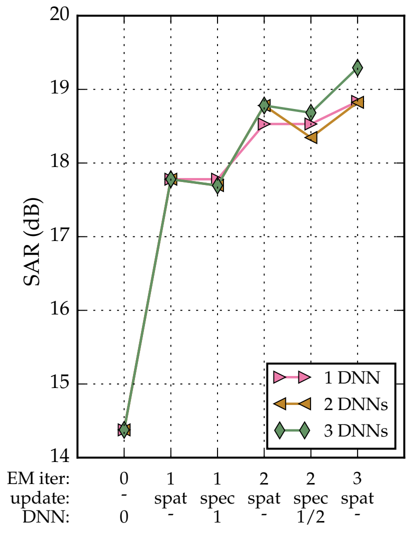
(a) Signal-to-distortion ratio (SDR)



(b) Source-image-to-spatial-distortion ratio (ISR)



(c) Signal-to-interference ratio (SIR)



(d) Signal-to-artifacts ratio (SAR)

Figure 3.8: Source separation performance for each update of the EM iterations with different numbers of DNNs. The number of spatial updates in an EM iteration is fixed to $K = 20$. In "1 DNN", there is no spectrogram fitting. Its final performance is equivalent to spatial updates with $K = 60$ after spectrogram initialization. In "2 DNNs", $\text{DNN}_1^{\text{spec}}$ is used for spectrogram fitting of both $l = 1$ and $l = 2$. In "3 DNNs", $\text{DNN}_1^{\text{spec}}$ and $\text{DNN}_2^{\text{spec}}$ are used for spectrogram fitting of $l = 1$ and $l = 2$, respectively. The evaluation was done on the simulated test set (`et05_simu`). The figures show the mean values. The 95% confidence intervals are similar to those in Figure 3.7. Higher is better.

"2 DNNs" uses $\text{DNN}_0^{\text{spec}}$ for spectrogram initialization and $\text{DNN}_1^{\text{spec}}$ for spectrogram fitting within spectral updates of $l = 1$ and $l = 2$.

"3 DNNs" uses $\text{DNN}_0^{\text{spec}}$ for spectrogram initialization, $\text{DNN}_1^{\text{spec}}$ for spectrogram fitting within spectral updates of $l = 1$, and $\text{DNN}_2^{\text{spec}}$ for spectrogram fitting within spectral updates of $l = 2$.

Figure 3.8 shows that the use of a specific DNN for a given iteration (here, $\text{DNN}_1^{\text{spec}}$ for $l = 1$ and $\text{DNN}_2^{\text{spec}}$ for $l = 2$) is beneficial. When a specific DNN is used, the spectral update provides a small improvement, which allows the following spatial update to yield significant improvement. This behavior can be observed by comparing the performance of the spectral updates of EM iteration l and the spatial updates of the following iteration $l+1$. Additionally, we can observe it by comparing the overall behavior of the "3 DNNs" curve to the "1 DNN" curve, in which no spectrogram fitting is done. It also shows that the use of the same DNN for several iterations (here, $\text{DNN}_1^{\text{spec}}$ for $l = 1$ and $l = 2$) is suboptimal. We can observe this by comparing the performance of "3 DNNs" to that of "2 DNNs" curve for $l = 2$ and $l = 3$. This is understandable because there is a mismatch between the input and the training data of the DNN.

In general, the iterative spectral and spatial updates improve the enhancement performance. This is clearly indicated by comparing the metrics, especially SIR, of "1 DNN", in which only spatial updates are done, and "3 DNNs", in which spectral and spatial updates are done. However, the performance saturates after a few EM iterations.

3.7 Comparison to NMF based iterative EM algorithm

In this section, we compare the performance of our system to that of multichannel NMF based iterative EM algorithm of Ozerov et al. [2012]. We use the implementation in the Flexible Audio Source Separation Toolbox[1] and follow the settings used by Salaün et al. [2014] which provided state-of-the-art performance for this dataset before the emergence of deep learning. The speech spectral and spatial models for this system are trained on the real training set (`tr05_real`), while the noise spectral and spatial models are initialized for each mixture using 5 seconds of background noise context

[1]See `http://bass-db.gforge.inria.fr/fasst`.

Table 3.1: Source separation performance metrics (in dB) of the multichannel NMF based and the multichannel DNN based systems. The evaluation was done on the simulated test set (`et05_simu`). The table shows the mean value of each metric. Higher is better.

Enhancement method	SDR	ISR	SIR	SAR
NMF based [Ozerov et al., 2012]	7.72	10.77	13.29	12.29
Proposed DNN based: KL (3 DNNs)	13.25	24.25	15.58	18.23

based on the available annotation. This setting is favourable to the NMF based system and, because of this setting, the comparison is not completely fair since our DNN based system does not exploit this contextual information. As described earlier, the DNNs used in this evaluation were also trained on the real training set only. The separation results from this evaluation are then used for the following speech recognition evaluation.

3.7.1 Source separation performance

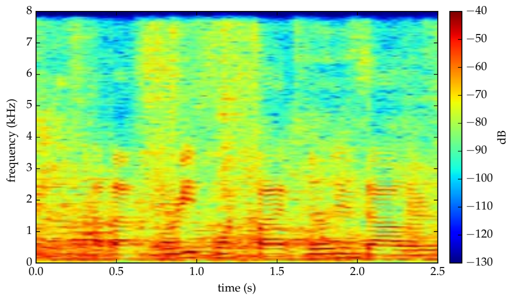
Table 3.1 compares the performance of the multichannel NMF based system after 50 EM iterations and the performance of our DNN based system after the spatial update of the EM iteration $l = 3$. Our DNN based system outperforms the NMF based one for all metrics. This confirms that DNNs are able to model spectral parameters much better than NMF does.

Figure 3.9 shows example spectrograms of the outputs of the NMF based system and our DNN based system. In general, both NMF and DNN based systems are able to enhance speech and attenuate noise. We can observe that the speech harmonics are much more attenuated in the NMF based system compared to the DNN based one. This is also perceptible in the informal listening test we did.

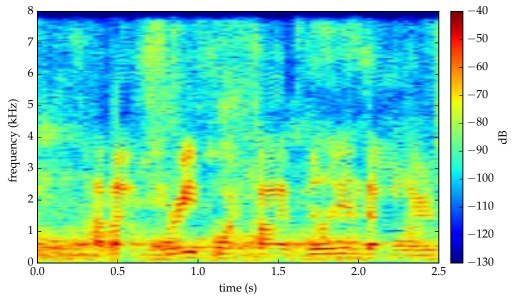
3.7.2 Speech recognition performance

Table 3.2 shows the speech recognition results in terms of WER. The evaluation uses the ASR system defined in the Kaldi recipe distributed by the CHiME-3 challenge organizers[2] [Barker et al., 2015; Hori et al., 2015]. See Section 2.2 for the summary of the ASR system and see Hori et al. [2015] for the details.

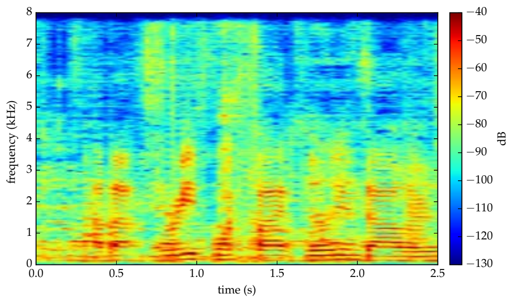
[2]See https://github.com/kaldi-asr/kaldi/tree/master/egs/chime3.



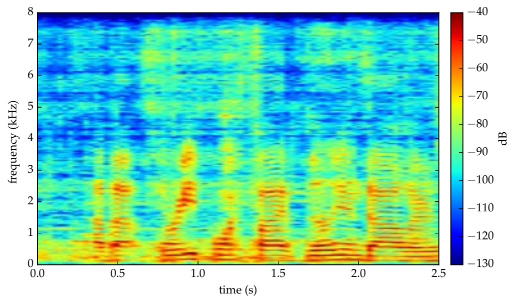
(a) Noisy



(b) NMF based



(c) Proposed DNN based (after spectrogram initialization)



(d) Proposed DNN based (after the spatial updates of the third EM iteration)

Figure 3.9: Example power spectrograms of channel 5 of (a) the noisy speech; (b) the output of the NMF based algorithm; (c) the output of the proposed DNN based algorithm after spectrogram initialization; and (d) the output of the proposed DNN based algorithm after the spatial updates of the third EM iteration. The utterance (`M05_440C0211_CAF`) is taken from the real test set (`et05_real`).

The evaluation results include the baseline performance (unprocessed noisy speech), BeamformIt, delay-and-sum beamforming, and the NMF based system we described above. The baseline performance was measured using only channel 5 of the observed 6-channel mixture. This channel is considered as the most useful channel because the corresponding microphone faces the user and is located at the bottom-center of the tablet device. BeamformIt and delay-and-sum beamforming were performed on the 6-channel mixture. BeamformIt is the baseline enhancement method in the CHiME-3. For both the NMF based and DNN based systems, we consider single-channel signals obtained by averaging over the channels of separated speech spatial images.

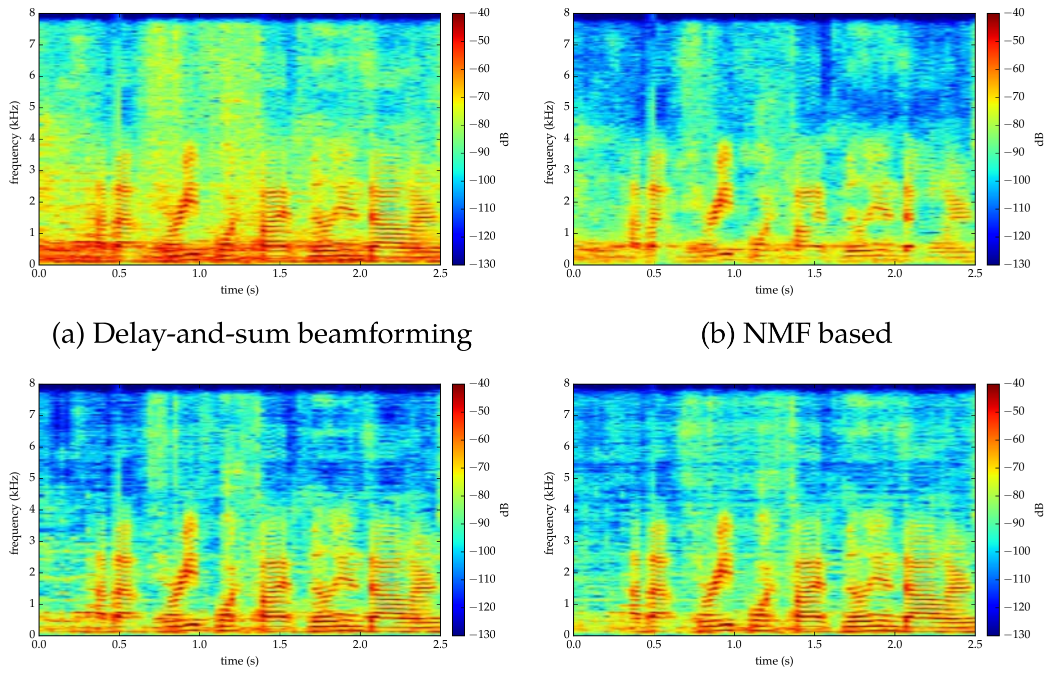
For the DNN based single-channel enhancement (see EM iteration $l = 0$), the WER on the real test set decreased by 21% relative w.r.t. the observed WER. This single-channel enhancement takes the output of delay-and-sum beamforming on the 6-channel mixture. However, this single-channel

Table 3.2: Speech recognition performance in terms of WER (%) using different enhancement methods. The ASR system uses the DNN+sMBR backend trained on multi-condition enhanced data followed by 5-gram Kneser-Ney smoothing and RNN-LM rescoring. The evaluation was done on the real sets. Boldface numbers show the best performance for each dataset. The 95% confidence intervals for the two best WERs are ± 0.26% for the development set and ± 0.40% for the test set. Lower is better.

Enhancement method	**EM iter.**	**Update type**	**Dev**	**Test**
Observed	-	-	9.65	19.28
BeamformIt	-	-	6.36	13.67
Delay-and-sum beamforming	-	-	6.35	13.70
NMF based [Ozerov et al., 2012]	50	-	6.10	13.41
Proposed DNN based: KL (3 DNNs)	0	-	6.64	15.18
	1	spatial	5.37	11.46
		spectral	5.19	11.46
	2	spatial	**4.87**	10.79
		spectral	4.99	11.12
	3	spatial	4.88	**10.14**

enhancement did not provide better performance compared to delay-and-sum beamforming alone. It indicates that proper exploitation of multichannel information is crucial. DNN based multichannel enhancement after the spatial updates of the third EM iteration then decreases the WER on the real test set by 33% relative w.r.t. the corresponding single-channel enhancement, 26% relative w.r.t. BeamformIt or delay-and-sum beamforming, and 24% relative w.r.t. the NMF based system.

Figure 3.10 shows example spectrograms of the outputs of the delay-and-sum beamformer, the NMF based system above, and our DNN based system. It compares the resulting single-channel enhanced speech signals used as ASR inputs. Figures 3.10c and 3.10d correspond to the DNN based system outputs at the zeroth EM iteration and at the third EM iteration after the spatial updates, respectively, in Table 3.2. The over-attenuation by the NMF based system seen in Figure 3.9b can also be observed Figure 3.10b. Both NMF and DNN based systems attenuate the noise better than delay-and-sum beamforming. The spectrogram improvement provided by the spatial



(a) Delay-and-sum beamforming

(b) NMF based

(c) Proposed DNN based (after spectrogram initialization)

(d) Proposed DNN based (after the spatial updates of the third EM iteration)

Figure 3.10: Example power spectrograms of the resulting single-channel enhanced speech by (a) Delay-and-sum beamforming on the 6-channel noisy speech; (b) averaging over channels of the 6-channel output of the NMF based algorithm; (c) averaging over channels of the 6-channel output of the proposed DNN based algorithm after spectrogram initialization; and (d) averaging over channels of the 6-channel output of the proposed DNN based algorithm after the spatial updates of the third EM iteration. The utterance (`M05_440C0211_CAF`) is taken from the real test set (`et05_real`).

updates for the proposed DNN based system is barely noticeable, yet it improves the WER.

3.8 Impact of environment mismatches

In addition to the above studies, we also studied the impact of training the speech enhancement system on certain noise environments of the CHiME dataset and testing it on the other environments. We considered three different training data settings as follows.

Single-environment setting includes four training data sets. Each set contains the training data of one environment.

Many-environment setting includes four training data sets. Each set contains the training data of three environments.

All-environment setting includes three training data sets. All sets contain the training data of all four environments. These sets differ in the total training data size. The first set amounts to one quarter of all training data and thus, the size is comparable to the sets in the single-environment setting. The second set amounts to three quarters of all training data and thus, the size is comparable to the sets in the many-environment setting. The utterances in both the first and the second sets are randomly selected and the number of utterances from different environments is the same. The third set contains all training data.

The experiments showed that for the same amount of data, the all-environment setting outperforms both the single-environment and the many-environment settings. In particular, it outperforms matched training, i.e., training and testing on the same environment. Surprisingly perhaps, mismatched training, i.e., training and testing on different environments, performs only marginally worse which indicates that the DNN generalizes rather well to other environments. The generalization improves as more environments are being included in the training set. The performance also increases along the training data size. See Vincent et al. [2017b] for further details.

3.9 Summary

This chapter presents a DNN based multichannel audio source separation framework where the spectral parameters are modeled by DNNs and the spatial parameters are estimated in an iterative fashion as in the classical EM based framework. We show that DNNs are able to provide good estimation of the spectral parameters, in this case the source magnitude spectrograms. Employing these spectral parameters, the spatial parameters can be well estimated using iterative spatial updates. The spatial updates provide evidence that multichannel separation yields significantly better performance than single-channel separation in terms of source separation and speech recognition metrics. We show that when the spatial parameters are initialized as identity matrices, multiple spatial updates should be done after spectrogram initialization. We generalize this into multiple spatial updates for each spectral update. We then show that the spectral updates by DNNs yield better source spectrograms which allow the following spatial updates to provide further performance improvement. We also show that the use of a specific DNN for spectral updates at a given iteration should be preferred, because a single DNN used for spectral updates at different iterations is suboptimal. Finally, we show that an implementation of the proposed DNN based framework significantly outperformed a multichannel NMF based system [Ozerov et al., 2012] in terms of source separation metrics. The DNN based system also provides a relative 24% decrease of the WER w.r.t. the NMF based system.

CHAPTER 4

On improving deep neural network spectral models

In the previous chapter, we showed deep neural networks (DNNs) work well for estimating the spectral parameters of a multichannel Gaussian model based separation framework. This chapter presents our exploration on improving these spectral DNNs. We mainly study the impact of various general-purpose and task-oriented cost functions. This notably includes the probabilistically-motivated Itakura-Saito (IS) divergence, the Kullback-Leibler (KL) divergence used in the previous chapter, and the task-oriented signal-to-distortion ratio (SDR) cost function. We then briefly study the use of perceptually-motivated equivalent rectangular bandwidth (ERB) time-frequency representations and the impact of DNN architectures, in which we consider a bidirectional long short-term memory (LSTM) based recurrent neural network (RNN) architecture. Finally, we consider data augmentation approaches to increase the DNN training data. This chapter presents the application to a speech enhancement problem as in the previous chapter and additionally, an application to a vocals-accompaniment separation problem in music.

4.1 Research questions

In this chapter, we address the following research questions.

1. **Can we represent our DNN based approach in a rigorous expectation-maximization (EM) framework?** Recall that the framework we proposed in Algorithm 2) follows the general iterative EM algorithm in Algorithm 1. In Duong et al. [2010a], this general algorithm does maximum likelihood estimation in which the M step finds new parameters, i.e., the unconstrained source power spectral densities (PSDs) $z_j(f,n)$ and the source spatial

covariance matrices $\mathbf{R}_j(f)$, that maximize the probability of the source spatial images estimated in the preceding E step given these parameters $p\left(\widehat{\mathbf{c}}_j(f,n)|z_j(f,n),\mathbf{R}_j(f)\right)$. The spectrogram fitting method introduced by Ozerov et al. [2012] takes $z_j(f,n)$ to obtain $v_j(f,n)$ by employing spectral models trained on some dataset. This fitting may not lead to a likelihood maximization if the models do not optimize the right criterion. In our case, our framework may not always maximize the likelihood in its parameter updates because of the DNNs used in spectrogram initialization and fitting. Févotte et al. [2009] showed that the IS divergence is equivalent to maximum likelihood estimation up to constant terms. This means that by using the IS divergence as the cost function for DNN training, every spectrogram initialization or spectrogram fitting in our framework performs maximum likelihood estimation. Consequently, the whole framework achieves maximum likelihood estimation as in a rigorous EM algorithm. However, it is worth mentioning that the use of the IS divergence for DNN training is uncommon. In this chapter, we investigate the impact of the DNN training cost function on the performance of our framework. We consider not only the above probabilistically-motivated IS divergence, but also other popular alternatives, i.e., the KL divergence, the Cauchy cost function, the phase-sensitive cost function, and the mean squared error (MSE). The details of these cost functions are given in Section 4.2.

2. **Is a perceptually-motivated time-frequency representation favorable for the proposed framework?** By taking the characteristics of human cochlea as a model, perceptually-motivated time-frequency representations have higher resolution at low frequencies and lower resolution at high frequencies compared to the short-time Fourier transform (STFT) representation. Additionally, perceptually-motivated representations typically have a lower dimensionality than the STFT representation. Time-frequency representations based on the perceptually-motivated ERB scale [Glasberg & Moore, 1990] have been successfully used in audio source separation [Roman et al., 2003; Vincent, 2006; Duong et al., 2010b; Ozerov et al., 2012]. In the context of modeling spectral representations by DNNs , the use of features with a low dimensionality allows us to use DNNs with fewer parameters and consequently, reduce the computational cost, especially the training time. In this respect, perceptually-motivated time-frequency representations are obviously more favorable than the

STFT representation. We now want to check whether the choice of time-frequency representation affects the system performance.

3. **Can RNN provide a better performance than feedforward neural network (FNN) in our multichannel Gaussian model based separation framework?** In this framework, spectral DNNs are use to predict a sequence of source spectra. As mentioned in Section 2.4.3.1, RNN is known to be good for modeling a sequence and many studies have proven it. RNN indeed provides a better performance than FNN in the context of single-channel source separation [Weninger et al., 2014; Uhlich et al., 2017]. We investigate whether this also applies for our multichannel Gaussian model based separation framework.

4. **Does task-oriented discriminative training improve the system performance?** Recall that for audio source separation, DNNs are typically used to estimate a mask or source spectra. The DNNs can be trained to minimize some cost function computed on the estimated mask or source spectra. As discussed in Section 2.4.3.2, studies in single-channel audio source separation have explored the use of more advanced cost functions for the training of DNNs estimating a mask. These include minimizing the error on the source magnitude or power spectra, the complex-valued time-frequency representations, and the time-domain signals computed using the estimated mask [Erdogan et al., 2015; Weninger et al., 2015; Wang & Wang, 2015]. The previously mentioned phase-sensitive cost function is an example of task-oriented cost function in the context of single-channel audio source separation. The use of task-oriented cost functions basically aims to reduce the mismatch between the DNN training objective and how the system performance is evaluated. In the case of source separation evaluation, maximizing the SDR of the estimated time-domain source signals is a natural choice. In our study, we employ an SDR-oriented cost function in the context of multichannel audio source separation.

5. **Is data augmentation effective to avoid overfitting?** As we will observe, we face an overfitting problem in the application of the proposed framework to a vocals-accompaniment separation task. This problem is typically addressed by regularizing the DNN training algorithm or increasing the variety of training data. For this task, the development set of the dataset we use consists of fifty songs with various music genres resulting in only two hours of data. This data amount is relatively low for DNN training. It

is significantly lower than the 15 hours of data from the CHiME dataset used in our speech enhancement task. We study the use of training data augmentation to address this overfitting problem.

These research questions are investigated by employing the speech enhancement system proposed in the previous chapter and a singing-voice, or vocals-accompaniment, separation system presented later in this chapter.

4.2 Cost functions for spectral DNNs

In this section, we present several general-purpose and task-oriented cost functions considered in this chapter.

4.2.1 General-purpose cost functions

The following cost functions measure the discrepancies between the target $\sqrt{\widetilde{v}_j(f,n)}$ and the estimate $\sqrt{v_j(f,n)}$.

The Itakura-Saito (IS) divergence [Itakura & Saito, 1968] is expressed as

$$\mathcal{D}_{\text{IS}} = \frac{1}{JFN} \sum_{j,f,n} \left(\frac{\widetilde{v}_j(f,n)}{v_j(f,n)} - \log \frac{\widetilde{v}_j(f,n)}{v_j(f,n)} - 1 \right) . \tag{4.1}$$

This metric is known to yield signals with good perceptual quality. Therefore, it has become a popular metric in the audio processing community, including for audio source separation based on non-negative matrix factorization (NMF) [Bertin et al., 2009; Févotte et al., 2009; Lefèvre et al., 2011].

From the theoretical point of view of our frameworks, this metric is attractive because it results in maximum likelihood estimation of the spectra [Févotte et al., 2009]. Thus, when DNNs are trained using the IS divergence, the whole Algorithm 2 achieves maximum likelihood estimation.

The generalized Kullback-Leibler (KL) divergence [Lee & Seung, 2000] is expressed as

$$\mathcal{D}_{\text{KL}} = \frac{1}{JFN} \sum_{j,f,n} \left(\sqrt{\widetilde{v}_j(f,n)} \log \frac{\sqrt{\widetilde{v}_j(f,n)}}{\sqrt{v_j(f,n)}} - \sqrt{\widetilde{v}_j(f,n)} + \sqrt{v_j(f,n)} \right), \tag{4.2}$$

which reduces to the KL divergence [Kullback & Leibler, 1951] when $\sum_{j,f,n} \sqrt{\widetilde{v}_j(f,n)} = \sum_{j,f,n} \sqrt{v_j(f,n)} = 1$. The generalized KL divergence is also a popular choice for NMF-based audio source separation [Févotte et al., 2009; Févotte & Ozerov, 2010] and has shown to be effective for DNN training [Huang et al., 2014b]. To be concise, hereafter, let us simply call this generalized KL divergence as the KL divergence.

The Cauchy cost function [Liutkus et al., 2015a] is expressed as

$$\mathcal{D}_{\text{Cau}} = \frac{1}{JFN} \sum_{j,f,n} \left(\frac{3}{2} \log \left(\widetilde{v}_j(f,n) + v_j(f,n) \right) - \log \sqrt{v_j(f,n)} \right). \tag{4.3}$$

This metric has been proposed recently for NMF-based-based audio source separation and advocated as performing better than the IS divergence in some cases.

The mean squared error (MSE) is expressed as

$$\mathcal{D}_{\text{MSE}} = \frac{1}{2JFN} \sum_{j,f,n} \left(\sqrt{\widetilde{v}_j(f,n)} - \sqrt{v_j(f,n)} \right)^2. \tag{4.4}$$

This metric is the most widely used cost function for various optimization processes, including the training of DNNs for regression tasks.

4.2.2 Task-oriented cost functions

The phase-sensitive cost function [Erdogan et al., 2015; Weninger et al., 2015] minimizes the error of single-channel source signals in the complex-valued STFT domain. This metric is defined as

$$\mathcal{D}_{\text{PS}} = \frac{1}{2JFN} \sum_{j,f,n} |m_j(f,n)\widetilde{x}(f,n) - \widetilde{c}_j(f,n)|^2, \tag{4.5}$$

where $m_j(f,n) = v_j(f,n) \left(\sum_{j'} v_{j'}(f,n)\right)^{-1} \in \mathbb{R}_+$ is the single-channel Wiener filter, while $\widetilde{x}(f,n)$ and $\widetilde{c}_j(f,n)$ are the single-channel versions of the multichannel mixture $\mathbf{x}(f,n)$ and the multichannel ground truth source spatial images $\mathbf{c}_j(f,n)$, respectively. These single-channel signals can be obtained, e.g., by delay-and-sum beamforming.

The signal-to-distortion ratio (SDR) cost function we proposed in this study aims to maximize the SDR of the multichannel source images by minimizing the distortion in the time domain. This can be done by minimizing the denominator of the power ratio in (2.3). The SDR cost function used in this section is defined as

$$\mathcal{D}_{\text{SDR}} = \frac{1}{J} \sum_j \log_{10} \left(\sum_{i,t} \left(\widehat{c}_{ij}(t) - c_{ij}(t) \right)^2 \right) \tag{4.6}$$

where $\widehat{c}_{ij}(t)$ are the source spatial images computed by multichannel Wiener filtering in (2.46) given the source spatial covariance matrices $\mathbf{R}_j(f)$ and the mixture $\mathbf{x}(f,n)$. This cost function includes the inverse discrete Fourier transform and the weighted overlap-add method. Compared to (2.27) [Wang & Wang, 2015], which is equivalent to the above phase-sensitive cost function, we consider multichannel separation instead of single-channel separation and account for weighted overlap-add, which is known to handle artifacts at the frame edges (see Section 2.3). Thus, the SDR cost function involves a computation of the real-valued time-domain signal from the real-valued magnitude spectra in the time-frequency domain, via the complex-valued STFT coefficients in the time-frequency domain.

4.3 Impact of the cost function

In this section, we study the impact of the different cost functions presented above, except the task-oriented SDR cost function, which is addressed in Section 4.5. For doing so, we consider the same speech enhancement task as in Chapter 3.

4.3.1 Experimental settings

The experiments in this section follow the experimental settings and use the speech enhancement system presented in Section 3.3.2. The only difference is in the training criterion. For this section, let us generalize the cost function formula from Section 3.3.3.3 into

$$\mathcal{C} = \mathcal{D} + \mathcal{D}_{\ell_2}. \tag{4.7}$$

All experiments in Chapter 3 use $\mathcal{D}_{\overline{\text{KL}}}$ in (3.3), which is a regularized version of the KL divergence in (4.2), as $\mathcal{D}$.

In this section, we experiment with different cost functions presented in Section 4.2.1 as $\mathcal{D}$. In order to avoid numerical instabilities as mentioned in Section 3.3.3.3, we consider a regularized version of the IS divergence as

$$\mathcal{D}_{\overline{\text{IS}}} = \frac{1}{JFN} \sum_{j,f,n} \left(\frac{\widetilde{v}_j(f,n) + \delta_{\text{cf}}}{v_j(f,n) + \delta_{\text{cf}}} - \log \frac{\widetilde{v}_j(f,n) + \delta_{\text{cf}}}{v_j(f,n) + \delta_{\text{cf}}} - 1 \right) \tag{4.8}$$

and a regularized version of the Cauchy cost function as

$$\mathcal{D}_{\overline{\text{Cau}}} = \frac{1}{JFN} \sum_{j,f,n} \left(\frac{3}{2} \log \left(\widetilde{v}_j(f,n) + v_j(f,n) + \delta_{\text{cf}} \right) - \log \left(\sqrt{v_j(f,n)} + \delta_{\text{cf}} \right) \right), \tag{4.9}$$

where the regularization parameter is fixed to $\delta_{\text{cf}} = 10^{-3}$. Additionally, a geometric analysis on the phase-sensitive cost function (4.5) leads to

$$\mathcal{D}_{\overline{\text{PS}}} = \frac{1}{2JFN} \sum_{j,f,n} \left(m_j(f,n) \left| \widetilde{x}(f,n) \right| - \left| \widetilde{c}_j(f,n) \right| \cos \left(\angle \widetilde{x}(f,n) - \angle \widetilde{c}_j(f,n) \right) \right)^2, \tag{4.10}$$

where $\angle \cdot$ denotes the angle of the complex-valued STFT coefficients. Employing (4.10), the computation of the phase-sensitive cost function does not involve complex number.

In addition, it is worth mentioning that the use of flooring function, such as in rectified linear unit, for the DNN outputs seems to be important for the DNN training with the IS divergence, the KL divergence, the Cauchy cost function, and the phase-sensitive cost function. We found in additional experiments (not shown here) that training failed when a linear activation function was used for the output layer with these cost functions.

The following subsections present the source separation and the speech recognition performance comparison for spectral DNNs trained with different cost functions, i.e., the IS divergence, the KL divergence, the Cauchy cost function, the phase-sensitive cost function, and the MSE.

4.3.2 Source separation performance

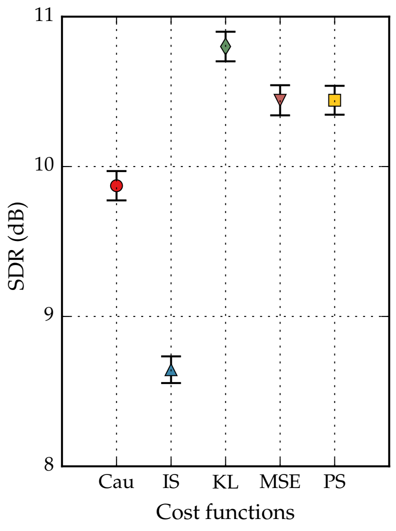
We first evaluate the impact of the cost function by using the PSDs $v_j(f,n)$ estimated by $\text{DNN}_0^{\text{spec}}$ and keeping the spatial covariance matrices $\mathbf{R}_j(f)$ as the identity matrix. This is achieved by setting $L=0$ in Algorithm 2. This is equivalent to single-channel source separation for each channel.

Figure 4.1 shows the performance comparison for the resulting 6-channel estimated speech signal on the simulated test set (`et05_simu`). In general, the KL divergence, the phase-sensitive cost function, and the MSE have good and comparable performance among all cost functions considered here. The IS divergence almost always has the worst performance and the Cauchy cost function achieves the best signal-to-interference ratio (SIR), but suffers from a poor signal-to-artifacts ratio (SAR). From this experimental study, we conclude that the IS divergence and the Cauchy cost function should be avoided for single-channel source separation with a DNN spectral model.

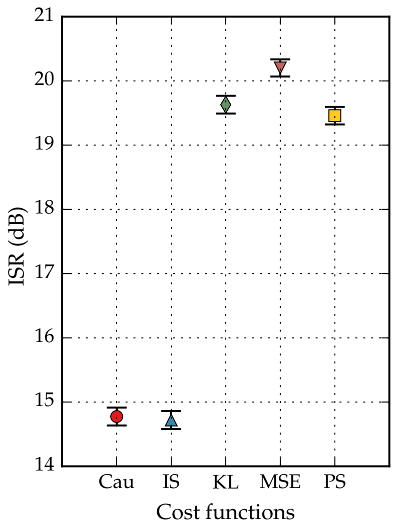
We then investigate the impact of spatial parameter updates for these different cost functions as we did in Section 3.5 for the KL divergence. Figure 4.2 shows the performance comparison for different numbers of spatial updates for different cost functions. For a complete comparison, it also includes the KL divergence that was already presented in Figure 3.7. Recall that for this evaluation the spectral parameters $v_j(f,n)$ are initialized by $\text{DNN}_0^{\text{spec}}$ and kept fixed during the iterative spatial parameter updates. Also, recall that $k=0$ is equivalent to single-channel source separation for each channel whose results are shown in Figure 4.1.

The most noticeable fact is that the performance of the phase-sensitive cost function is good for most metrics in the first few updates, but then it saturates quickly. By contrast, the performance of the other cost functions increases along with the spatial updates for most metrics. This indicates that although the phase-sensitive cost function performs well for single-channel source separation, it may not be suitable for multichannel source separation. In a more general respect, not all cost functions are appropriate for the training of spectral DNNs for multichannel source separation.

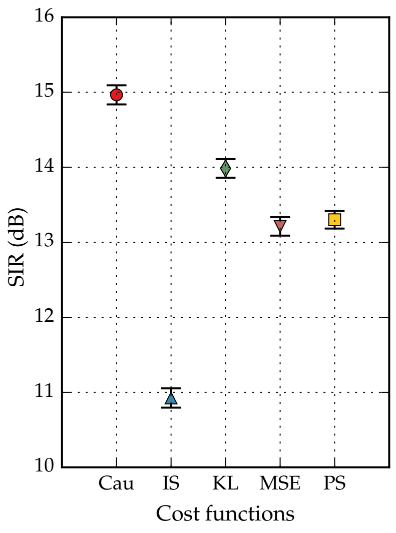
The figure also shows that different cost functions have a different improvement rate along the spatial updates. We may expect that different cost functions saturate at different k. Interestingly, although the IS divergence



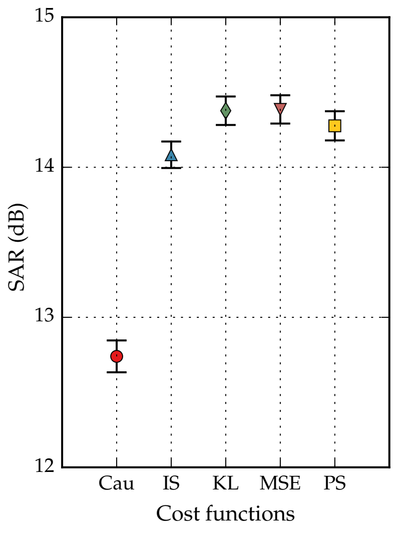
(a) Signal-to-distortion ratio (SDR)



(b) Source-image-to-spatial-distortion ratio (ISR)

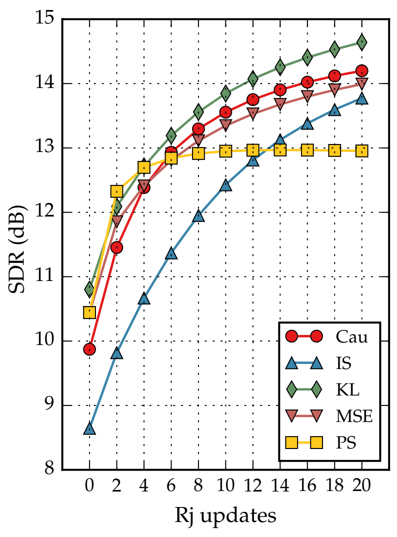


(c) Signal-to-interference ratio (SIR)

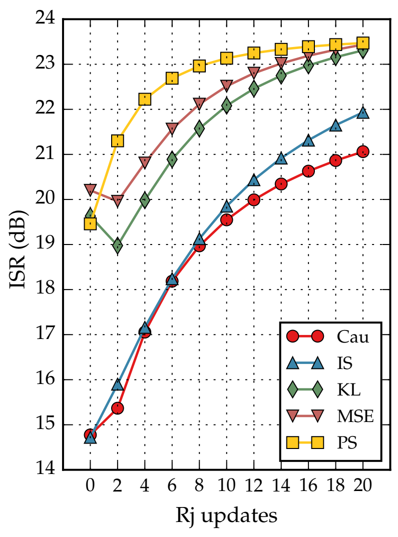


(d) Signal-to-artifacts ratio (SAR)

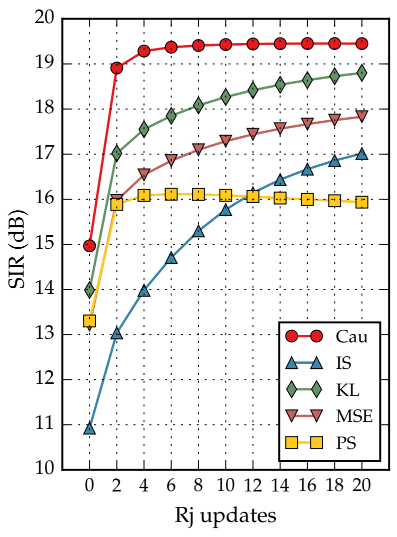
Figure 4.1: Single-channel source separation performance for the DNNs trained with different cost functions. The PSDs $v_j(f,n)$ are estimated by $\text{DNN}_0^{\text{spec}}$ and the spatial covariance matrices $\mathbf{R}_j(f)$ are the identity matrix. The evaluation was done on the simulated test set (`et05_simu`). The figures show the mean value and the 95% confidence interval of each metric. The mean SDR, ISR, SIR, and SAR with the 95% confidence intervals measured on the observed 6-channel mixture signal are 3.8 $\pm$ 0.1 dB, 18.7 $\pm$ 0.1 dB, 4.0 $\pm$ 0.1 dB, and 69.8 $\pm$ 0.5 dB, respectively. Higher is better.



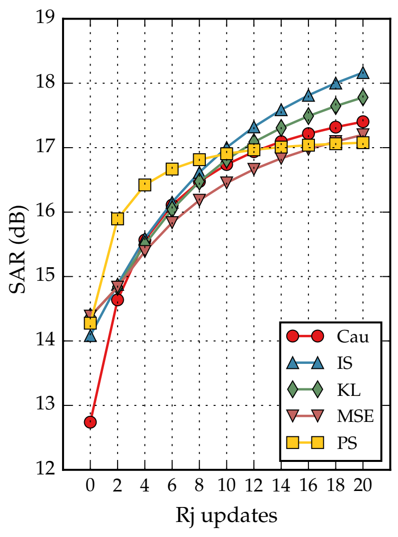
(a) Signal-to-distortion ratio (SDR)



(b) Source-image-to-spatial-distortion ratio (ISR)



(c) Signal-to-interference ratio (SIR)



(d) Signal-to-artifacts ratio (SAR)

Figure 4.2: Source separation performance for various numbers of spatial updates with the DNNs trained with different cost functions. The PSDs $v_j(f,n)$ are estimated by $\text{DNN}_0^{\text{spec}}$ and kept fixed during the iterative updates of the spatial covariance matrices $\mathbf{R}_j(f)$. The evaluation was done on the simulated test set (`et05_simu`). The figures show the mean values. The 95% confidence intervals are similar to those in Figure 4.1. Higher is better.

is almost always outperformed by the other cost functions, its improvement rate for each metric is generally the highest compared to the others and it has not saturated yet after 20 spatial updates. It is probable that it would outperform the other cost functions after many more spatial updates. After 20 spatial updates, we can observe that some cost functions are better than the others for some metrics. This suggests that the cost function selection should depend on the task, e.g., fewer artifacts are preferable to low interference, and on the computational constraints, e.g., only few updates can be done. Nonetheless, the KL divergence is the most reasonable choice for general-purpose training because it improves all of the metrics well on average. In addition, although the use of IS divergence is theoretically-motivated, there are better alternatives.

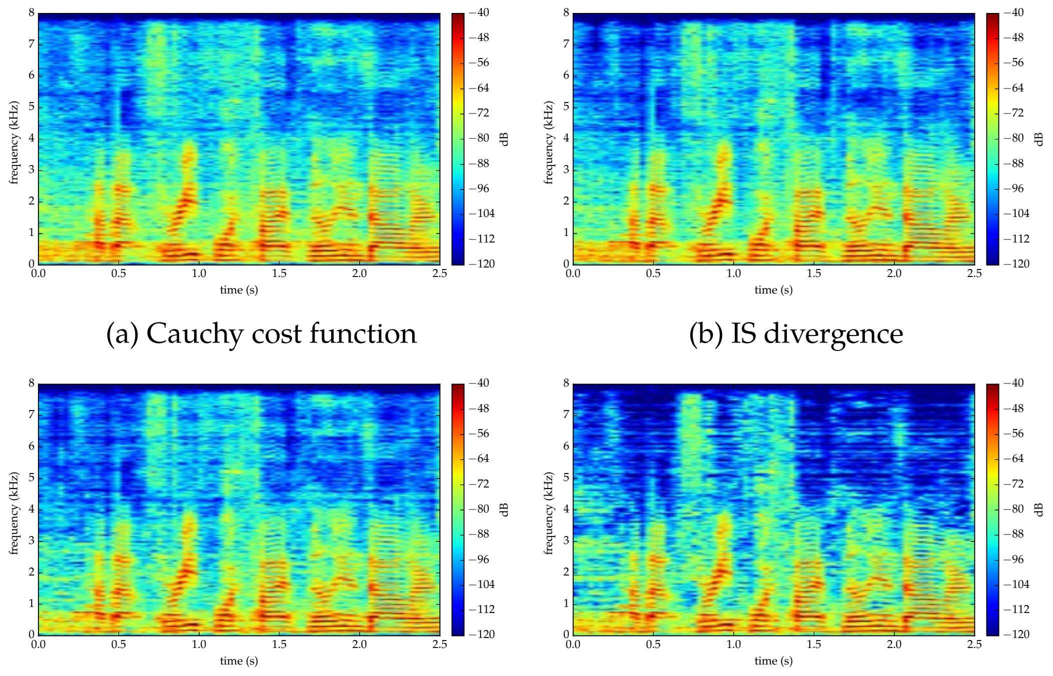
The following subsection uses the separation results from the experiments in this subsection to evaluate the speech recognition performance. In this evaluation, we exclude the phase-sensitive cost function because as shown above, its overall distortion, represented by SDR, is significantly lower than the others after 20 spatial updates.

4.3.3 Speech recognition performance

The speech recognition evaluation is done on the resulting single-channel enhanced speech. Recall that this speech recognition evaluation uses DNNs trained and validated on the real sets, while the above source separation evaluation uses DNNs trained and validated on the real and the simulated sets. This is described in Section 3.3.3.5.

Figure 4.3 shows example spectrograms for different cost functions. This figure is comparable to Figure 3.10. Figure 3.10c illustrates the use of KL divergence after spectrogram initialization, Figure 4.3c after the spatial updates of the first EM iteration, and Figure 3.10d after the spatial updates of the third EM iteration. The difference between these spectrograms is hardly visible and it also happens for the other cost functions. Therefore, we present only the spectrograms after the spatial updates of the first EM iteration in Figure 4.3.

In general, the difference between the spectrograms for the frequencies below 4 kHz is barely noticeable. The frequencies above 4 kHz indeed look the same for the Cauchy cost function, the IS divergence, and the KL divergence. By contrast, the frequencies above 4 kHz of the MSE have noticeably less noise. Although we aim for noise attenuation in general, here, the noise



(a) Cauchy cost function (b) IS divergence

(c) KL divergence (d) MSE

Figure 4.3: Example power spectrograms of the resulting single-channel enhanced speech after the spatial updates of the first EM iteration. The utterance (`M05_440C0211_CAF`) is taken from the real test set (`et05_real`).

is attenuated differently across the frequency bands. This tends to induce unpleasant artifacts.

Table 4.1 shows the performance comparison in terms of word error rate (WER). It shows the WERs after spectrogram initialization, corresponding to single-channel enhancement, and after the spatial updates with $K = 20$ of the first EM iteration of each system. The above spectrograms correspond to the latter.

After spectrogram initialization, the best WER on the test set is 14.85% with a 95% confidence interval of $\pm$ 0.48%. Almost all cost functions have statistically comparable performance.

After the spatial updates of the first EM iteration, the best WER on the test set reaches 10.83% $\pm$ 0.42%. This performance is achieved by the IS divergence and the difference to the performance of the Cauchy cost function is statistically insignificant. Interestingly, the performance of the IS divergence is better than that of the KL divergence, which achieves the best SDR in the above source separation evaluation. Also, surprisingly, the MSE that outperforms the other cost functions for single-channel enhancement does

Table 4.1: Speech recognition performance in terms of WER (%) using the different cost functions. The ASR system uses the DNN+sMBR back-end trained on multi-condition enhanced data followed by 5-gram Kneser-Ney smoothing and RNN-LM rescoring. The evaluation was done on the real sets. Boldface numbers show the best performance for each dataset. The 95% confidence intervals for the two best WERs are ± 0.26% for the development set and ± 0.42% for the test set. Lower is better.

Enhancement method	EM iter.	Update type	Dev	Test
Proposed DNN based: Cauchy (1 DNN)	0	-	6.86	15.61
	1	spatial	4.96	11.13
Proposed DNN based: IS (1 DNN)	0	-	6.47	15.21
	1	spatial	**4.88**	**10.83**
Proposed DNN based: KL (1 DNN)	0	-	6.64	15.18
	1	spatial	5.37	11.46
Proposed DNN based: MSE (1 DNN)	0	-	6.74	14.85
	1	spatial	5.80	13.01

not perform well after the spatial updates. However, it should be emphasized that the spatial updates themselves improve the WER. This low performance of the MSE may be due to the artifacts shown in Figure 4.3d as we discussed above. Additionally, Figure 4.2d shows that the IS divergence achieves the best SAR after the spatial updates. These findings suggest that the WER positively correlates with the SAR more than the other metrics.

Finally, Table 4.2 presents the performance of our proposed DNN based multichannel speech enhancement system, in which the multiple spectral DNNs are trained with the IS divergence. This table is comparable to Table 3.2. By comparing the best WERs and the corresponding confidence intervals from these two tables, we can observe that both KL and IS divergences perform similarly on the development set, but the IS divergence performs significantly better than the KL divergence on the test set.

Let us now conclude this section. The KL divergence, the MSE, or the phase-sensitive cost function are reasonable choices for DNN training in the context of single-channel source separation. By contrast, in the context of multichannel source separation, the Cauchy cost function, the KL divergence, or the IS divergence may be used. The IS divergence is a plausible choice because it achieves the best speech recognition performance. However, its

Table 4.2: Speech recognition performance in terms of WER (%) of the proposed system using the IS divergence. The ASR system uses the DNN+sMBR back-end trained on multi-condition enhanced data followed by 5-gram Kneser-Ney smoothing and RNN-LM rescoring. The evaluation was done on the real sets. Boldface numbers show the best performance for each dataset. The 95% confidence intervals for the two best WERs are ± 0.25% for the development set and ± 0.39% for the test set. Lower is better.

Enhancement method	EM iter.	Update type	Dev	Test
Proposed DNN based: IS (3 DNNs)	0	-	6.47	15.21
	1	spatial	4.88	10.83
		spectral	5.03	10.79
	2	spatial	4.77	9.72
		spectral	4.76	9.47
	3	spatial	**4.68**	**9.32**

source separation performance is unappealing because it falls behind the others in most spatial update indexes and most metrics although it exhibits a good improvement rate along with the spatial updates. Thus, the KL divergence remains the most reasonable choice because it performs well in both source separation and speech recognition evaluations.

4.4 Impact of time-frequency representations, DNN architectures, and DNN training data

In this section, we still consider the same multichannel speech enhancement task, but now we briefly study the impact of different DNN architectures and training data. Regarding the training data, we employ the available simulated data from the same CHiME dataset as an additional data. This can be seen as a data augmentation. In addition, we also briefly study the impact of different time-frequency representations.

For the experiments in this section, we use some simplified experimental settings that are detailed in the following subsection. Here, we use a single spectral DNN, i.e., $\text{DNN}_0^{\text{spec}}$, only and focus on the speech recognition evaluation.

4.4.1 Experimental settings

In general, the experimental settings in this section are similar to those described in Section 3.3.2. The following description details the few differences.

4.4.1.1 Time-frequency representations

We consider two time-frequency representations, i.e., STFT coefficients and perceptually-motivated quadratic ERB coefficients. The STFT coefficients are extracted as in Chapter 3 using a Hamming window of length 1024 and a hopsize of 512 samples, resulting $F = 513$ frequency bins. The quadratic ERB coefficients are computed using a time-domain filterbank [Duong et al., 2010b], resulting in $F = 128$ frequency bins.

4.4.1.2 DNN architectures and inputs

We use two FNNs and two RNNs as $\text{DNN}_0^{\text{spec}}$ for the experiments in Section 4.4. The FNNs follow an multilayer perceptron architecture, while the RNNs follow a bidirectional LSTM architecture. The FNNs consist of an input layer, four hidden layers, and an output layer. The RNNs consist of an input layer, two bidirectional LSTM hidden layers, and an output layer. For these FNNs and RNNs, the sizes of the input layer and the output layer are F and $F \times J$, respectively, where $J = 2$. For the FNNs, the sizes of the hidden layers are $F \times J$. For the RNNs, the sizes of the bidirectional LSTM hidden layers are set such that the total number of parameters is comparable to that of the FNNs. This is done to achieve a fair comparison of the different architectures. Rectified linear units are employed in all layers of FNNs and the output layer of RNNs. Each LSTM layer employs the hyperbolic tangent as the activation of the cell and the hard sigmoid as the activation of the gates [Hochreiter & Schmidhuber, 1997]. Table 4.3 summarizes the sizes of the different DNNs used for the experiments in this section.

The quadratic ERB coefficients represent locally-observed covariance matrices. Their extraction from the re-aligned mixture signal $\overline{\mathbf{x}}(t)$ results in $\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n)$, that is akin to $\overline{\mathbf{x}}(f,n)\overline{\mathbf{x}}(f,n)^*$ where $\overline{\mathbf{x}}(f,n)$ are the STFT coefficients extracted from $\overline{\mathbf{x}}(t)$. To be consistent with each other, the input frames of

Table 4.3: Comparison of the different DNNs used in Section 4.4. A bidirectional LSTM hidden layer consists of a forward LSTM layer and a backward LSTM layer. The left numbers in the bidirectional LSTM hidden layer sizes correspond to these two layers and the right numbers correspond to the size of each LSTM layer.

TF representation	STFT		QERB	
Architecture	FNN	RNN	FNN	RNN
Input layer size	513	513	128	128
Hidden layer number	4	2	4	2
Hidden layer size	1026	2×352	256	2×88
Output layer size	1026	1026	256	256
Total parameters	4,742,172	4,786,114	296,192	300,160

$\text{DNN}_0^{\text{spec}}$ are computed as

$$\sqrt{z_x(f,n)} = \begin{cases} \sqrt{\frac{1}{I}\text{tr}\left(\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n)\right)} & \text{for the QERB coefficients} \\ \sqrt{\frac{1}{I}\text{tr}\left(\overline{\mathbf{x}}(f,n)\overline{\mathbf{x}}(f,n)^*\right)} & \text{for the STFT coefficients} \end{cases} \tag{4.11}$$

which corresponds to the square-root of the power spectrum averaged across channels. For the FNNs, these input frames are used to construct 5-frame supervectors as in Section 3.3.3.2. However, here, the supervectors are constructed after applying zero padding beforehand and without delta features. For the RNNs, only standardization is applied to these input frames. As in Chapter 3, the standardization factors for the FNNs and the RNNs, also the principal component analysis transformation matrix for the FNNs, are computed on the whole training set.

4.4.1.3 DNN training criterion, algorithm, and data

The DNNs are trained using the KL divergence in (3.3) with $\delta_{\text{cf}} = 10^{-5}$. The ℓ_2 weight regularization in (3.4) with $\lambda_{\ell_2} = 10^{-6}$ is used for the FNNs.

The rectified linear units are initialized according to He et al. [2015]. The LSTM cells are initialized according to Glorot & Bengio [2010] and the gates are initialized according to Saxe et al. [2013]. The biases of the LSTM cells' forget gates are initialized to 1 as proposed by Jozefowicz et al. [2015]. The biases of the output layers are also initialized to 1. Different from what we did in Chapter 3, we do not train the DNNs in this section with greedy layer-wise supervised training. We simply use the above random weight initialization schemes for the whole network. Training is done by backpropagation with the AdaDelta parameter update algorithm, whose hyperparameters are set to $\rho = 0.95$ and $\epsilon = 10^{-6}$ [Zeiler, 2012]. The parameter update is done for each minibatch. For the FNNs, a minibatch consists of 128 random frames. For the RNNs, a minibatch consists of 8 random full utterances. Based on the longest utterance in the minibatch, zero padding is applied on the other utterances. To minimize the zero padding, we perform so-called bucketing to group the utterances based on their length. Each minibatch consists of utterances from the same bucket. Dropout [Srivastava et al., 2014] with rate 0.2 is applied on all FNNs and RNNs. For the FNNs, it is applied in between adjacent hidden layers, also in between the last hidden layer and the output layer. For the RNNs, it is applied on the output of LSTM layers before the summing of forward and backward layer activations. Gradient normalization [Pascanu et al., 2013] with threshold 1.0 is employed in the training of all FNNs and RNNs. The same early stopping as in Chapter 3 is used, but the maximum number of training epochs is set to 1000, which is never reached in practice. The actual numbers of epochs varies between 83 and 310.

The DNN training targets $\sqrt{\widetilde{v}_j(f,n)}$ are computed similarly to (4.11) from the re-aligned source spatial image signals $\bar{\mathbf{c}}_j(t)$. In this section, we consider the use of two different supersets for DNN training:

SS-R consists of the real training set (`tr05_real`) and the real development set (`dt05_real`),

SS-F consists of SS-R with the addition of the simulated training set (`tr05_simu`) and the simulated development set (`dt05_simu`).

The training sets in these supersets are used for the DNN training data, while the development sets are used for the validation data.

The DNNs and the related techniques for all experiments presented in this and the following sections were implemented using Keras [Chollet et al., 2015] with Theano [Bergstra et al., 2010; Theano Dev Team, 2016] as its back-end.

4.4.1.4 Multichannel filtering

When the quadratic ERB coefficients are used, instead of (2.48), the posterior second-order raw moments of the spatial source images are computed as

[Duong et al., 2010b]

$$\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n) = \mathbf{W}_j(f,n)\widetilde{\mathbf{R}}_{\widetilde{\mathbf{x}}}(f,n)\mathbf{W}_j(f,n)^* + (\mathbf{I} - \mathbf{W}_j(f,n))\,\mathbf{R}_{\mathbf{c}_j}(f,n). \quad (4.12)$$

Thus, the estimation of the spatial images $\widehat{\mathbf{c}}_j(f,n)$ is not required for parameter updates. In the end of multichannel filtering, the inverse quadratic ERB transformation involves the estimation of the *windowed* time-domain spatial images $\widehat{\mathbf{c}}_j(f,t)$ by Wiener filtering given the final filters $\mathbf{W}_j(f,n)$ and the filterbank inversion to recover the time-domain spatial images $\widehat{\mathbf{c}}_j(t)$.

4.4.2 Discussions

Table 4.4 summarizes the speech recognition evaluation. The evaluation uses the automatic speech recognition (ASR) system defined in the Kaldi recipe distributed by the CHiME-4 challenge organizers[1] [Vincent et al., 2017b]. In principle, this system is similar to the one we used earlier from the CHiME-3 challenge. The main difference is that the acoustic model of this new system is trained on multi-condition noisy data, instead of multi-condition enhanced data as in the previous system. This new system performs well and allows rapid evaluation of different enhancement methods. See Section 2.2 for the summary of the ASR system and see Hori et al. [2015] for the details.

Let us compare the performance of different time-frequency representations given the same DNN architecture and the same DNN training data. For the FNNs (see rows 1, 2, 5, 6), the quadratic ERB representation outperforms the STFT especially for single-channel separation, although the performance of SS-R on the test set is statistically similar. For multichannel separation, the quadratic ERB representation also outperforms the STFT, although the performance of SS-R on both the development and the test set is statistically similar. For the RNNs (see rows 3, 4, 7, 8), the STFT generally outperforms the quadratic ERB representation for single-channel separation and vice-versa for multichannel separation. However, the performance on the development set for single-channel separation is statistically similar. In short, the quadratic ERB representation provides better speech recognition performance than the STFT representation. Considering that the quadratic ERB allows us to use smaller DNNs, this time-frequency representation appears to be a favorable choice for a speech enhancement task, or other similar tasks, such as a singing-voice separation task.

[1]See https://github.com/kaldi-asr/kaldi/tree/master/egs/chime4.

Table 4.4: Speech recognition performance in terms of WER (%) using the different time-frequency representations, the different DNN architectures, and the different training datasets. The ASR system uses the DNN+sMBR back-end trained on multi-condition noisy data followed by 5-gram Kneser-Ney smoothing and RNN-LM rescoring. The evaluation was done on the real sets. Boldface numbers show the best performance for each dataset. The 95% confidence intervals for the two best WERs are ± 0.26% for the development set and ± 0.38% for the test set. Lower is better.

TF rep.	**Arch.**	**EM iter.**	**Update type**	**SS-R**		**SS-F**		**Row no.**
				Dev	**Test**	**Dev**	**Test**	
STFT	FNN	0	-	8.10	15.32	9.68	20.41	1
		1	spatial	5.45	9.13	6.29	13.17	2
	RNN	0	-	7.07	13.70	7.73	15.91	3
		1	spatial	6.50	9.68	5.68	10.16	4
QERB	FNN	0	-	7.38	15.08	8.39	18.16	5
		1	spatial	**5.21**	9.21	5.69	11.00	6
	RNN	0	-	7.21	14.29	7.74	15.12	7
		1	spatial	5.57	9.30	5.39	**9.02**	8

Let us now compare the performance of different DNN architectures given the same time-frequency representation and the same DNN training data. For both the STFT (see rows 1-4) and the quadratic ERB representation (see rows 5-8), the RNNs significantly outperforms the FNNs for single-channel separation. This is in line with the findings in other studies, such as Weninger et al. [2015] and Uhlich et al. [2017]. Interestingly, for multichannel separation, the FNNs significantly outperform the RNNs when SS-R is used, but otherwise when SS-F is used. In short, the RNNs may be preferable than the FNNs because the RNNs dominate the FNNs for single-channel separation and the RNNs are reasonably good compared to the FNNs for multichannel separation, especially when the training involves more data.

Finally, let us compare the performance of different DNN training datasets given the same time-frequency representation and the same DNN architecture by observing pairs of rows 1-2, 3-4, 5-6, and 7-8. In general, the use of SS-F results in worse speech recognition performance than that of SS-R. This is in line with our initial study [Sivasankaran et al., 2015]. This may indicate that there are mismatches between the simulated data and the real data, especially in the real test set. If so, we may argue that the quadratic ERB representation and the RNNs handle the mismatches better than the STFT and the FNNs, respectively. The quadratic ERB representation may be good here because it represents speech signals better than the STFT. The RNNs may be good here because they indeed handle context better that the FNNs, or our context handling for the FNNs is suboptimal. Roughly speaking, SS-F is dominated by simulated training data because there are 7,138 simulated utterances and 1,600 real utterances. The standardization and the dimensionality reduction steps for the FNN inputs rely on the statistics computed on this biased training data. Due to the mismatches, the real data is not standardized and reduced properly. It is one possible reason why the speech recognition performance of SS-F is worse than that of SS-R in this case. Another possible reason is related to the ground truth used during the training. Compared to the ground truth of the simulated data, that of the real data is noisier because of the imperfect estimation as discussed in Section 3.3.1. In Vincent et al. [2017b], we show that the performance gap between SS-R and SS-F is reduced when the ground truth of the simulated data is estimated as that of the real data. The enhancement system used by Vincent et al. [2017b] employ the STFT and FNNs similar to what we use here, but the ASR system uses the Gaussian mixture model back-end trained on multi-condition enhanced data.

4.5 Impact of a multichannel task-oriented cost function

In this section, we discuss another application of our proposed framework, i.e., a singing-voice separation system. This system is used for separating the singing voice, or the vocals, and the accompaniment from a mixture. Compared to the previous speech enhancement system, this separation system considers the same number of sources, i.e., $J = 2$, but fewer channels, i.e., $I = 2$. Using this system, we study the impact of the task-oriented SDR cost function and the impact of data augmentation. The previous experiments using a speech enhancement task to evaluate the performance on one of the sources only, i.e., speech. The experiments in this section evaluate the performance on both sources, i.e., vocals and accompaniment.

4.5.1 Experimental settings

4.5.1.1 Task and dataset

In general, we consider the problem of music separation in the context of the professionally-produced music recordings task, also known as the MUS task, of SiSEC 2016 [Liutkus et al., 2017]. The official dataset used in this task is called Demixing Secrets Dataset. This is derived from the accompanying materials of [Senior, 2011][2]. This dataset consists of 100 full-track songs of diverse music genres by various artists with their corresponding sources grouped into 'vocals', 'bass', 'drums', and 'other'. All mixtures and sources are stereo signals sampled at 44.1 kHz. The dataset is divided evenly into a development set and an evaluation set. Each set contains 50 songs. Recall that the development set of the CHiME dataset is used as the validation data only. By contrast, for the experiments in this section, we divide the development set of the Demixing Secrets Dataset into training data and validation data. This division is detailed in Section 4.5.1.3.

In this section, we focus on the problem of vocals-accompaniment separation. For doing so, we sum the 'bass', 'drums', and 'other' signals to obtain the 'accompaniment' signal. In short, we deal with the separation of two sources ($J = 2$), namely vocals and accompaniment, from a stereo mixture ($I = 2$). Following the official evaluation, the source separation performance is evaluated using BSS Eval (see Section 2.1) on the multichannel separated vocals and accompaniment. The separated sources are divided into segments whose length and shift are 30 s and 15 s, respectively. BSS Eval is computed on each segment. The first, the second, and the third quartiles, are then computed on the evaluation results of all segments.

The following subsection presents an implementation of the proposed source separation framework for this vocals-accompaniment separation task.

4.5.1.2 An overview of the singing-voice separation system

The explanation below provides the details of pre-processing, initialization, and multichannel filtering steps specific to the system used in all experiments presented in this section. There is no post-processing step applied in this system. See Section 3.3.2 for the general description of each step.

The pre-processing step extracts the STFT coefficients using a Hamming window of length 2048 and hopsize 1024 resulting in $F = 1025$

[2]See `http://www.cambridge-mt.com/MixingSecrets.htm`.

frequency bins. For this music separation task, the sources are assumed to not move over time. Thus, no re-alignment needs to be applied before the feature extraction.

The initialization step sets the initial source spectrogram using $\text{DNN}_0^{\text{spec}}$. This DNN is trained with a general-purpose cost function, i.e., the MSE. Besides, the source spatial covariance matrices are initialized as identity matrices $\mathbf{I} \in \mathbb{C}^{I \times I}$.

The multichannel filtering step generally follows Algorithm 2. However, instead of using (2.49), the spatial parameter update is computed using the so-called weighted spatial parameter update in (5.1), with the number of spatial updates fixed to $K = 4$. We assess the impact of this weighted spatial parameter update in Chapter 5. The number of EM iterations is set to $L = 1$. For spectrogram fitting, we train two additional spectral DNNs, i.e., $\text{DNN}_1^{\text{spec-MSE}}$ and $\text{DNN}_1^{\text{spec-SDR}}$. The first DNN is trained with a general-purpose cost function, i.e., the MSE. The second DNN is trained with a task-oriented cost function, i.e., the SDR-oriented cost function in (4.6). The MSE, instead of the KL divergence, is used here because these experiments were done concurrently with our investigation on the impact of the cost function. We did not have a chance to perform the same experiments with the KL divergence. In order to avoid numerical instabilities due to the use of single precision computing for DNNs, the spectrograms $v_j(f,n)$ are floored to 10^{-5} in the parameter update iterations.

4.5.1.3 DNN spectral models

Similarly to Section 3.3.3, this subsection presents the inputs and outputs, the architecture, the training criteria, the training algorithm, and the training data of the DNN spectral models mentioned above.

Inputs and outputs

We consider two settings for the inputs and outputs of the DNNs. In the first setting, the inputs of $\text{DNN}_0^{\text{spec}}$ $\sqrt{z_x(f,n)}$ are computed as in (4.11) for the STFT coefficient case. The training targets of all of these DNNs $\sqrt{\widetilde{v}_j(f,n)}$ are computed similarly to $\sqrt{z_x(f,n)}$. The inputs of $\text{DNN}_1^{\text{spec-MSE}}$ and $\text{DNN}_1^{\text{spec-SDR}}$ are a stack of $\sqrt{z_j(f,n)}$, which corresponds to the outputs of $\text{DNN}_0^{\text{spec}}$. See Figures 3.4 and 3.4 for better illustrations. In the second setting, a filterbank based on the ERB scale [Vincent, 2006] is applied on the power spectra $z_x(f,n)$, $\widetilde{v}_j(f,n)$, and $z_j(f,n)$ from the first setting to reduce the dimensionality from $F = 1025$ to $F = 256$. This is done mainly to reduce the training time of the DNNs since smaller DNNs may be utilized (see the comparison of parameter numbers in Table 4.5). It is also worth noting that this is only applied on the DNN inputs and outputs. Dimensionality reduction is performed by applying the filterbank matrix on the STFT to obtain the low-dimensionality ERB representation used as the DNN inputs. Dimensionality expansion is then performed by applying the transposition of the filterbank matrix on the DNN outputs to obtain the spectra in the STFT domain. This expansion is required because the multichannel model parameters and the multichannel filtering still use the full STFT coefficients with $F = 1025$. It should be mentioned that this reduction-expansion process is only equivalent to the power spectra in the STFT domain up to a constant factor. However, this factor does not affect the resulting Wiener filter.

Architectures

All DNNs used in this section follow a bidirectional LSTM based RNN architecture as in the previous Section 4.4.1.2. These RNNs consist of an input layer, two bidirectional LSTM hidden layers, and an output layer. The size of the input layer of $\text{DNN}_0^{\text{spec}}$ is F and those of $\text{DNN}_1^{\text{spec-MSE}}$ and $\text{DNN}_1^{\text{spec-SDR}}$ are $F \times J$. The size of the output layers of all DNNs are $F \times J$. The sizes of the bidirectional LSTM hidden layers for the STFT coefficients and the ERB coefficients are 1024 and 256, respectively. The activation functions are the same as in Section 4.4.1.2. Table 4.5 summarizes the sizes of the different RNNs used for the experiments in this section.

Training criteria

As mentioned in Section 4.5.1.2, $\text{DNN}_0^{\text{spec}}$ and $\text{DNN}_1^{\text{spec-MSE}}$ are trained with the MSE as the cost function, while $\text{DNN}_1^{\text{spec-SDR}}$ are trained with the SDR cost function. The SDR cost function could also be used in the training of DNN for spectrogram initialization. In this case, without any prior knowledge regarding the source location, we could set the source spatial covariance matrices as identity matrices, which would result in single-channel source separation. The SDR cost function has a higher computational complexity than the MSE due to the inclusion of STFT synthesis (see related discussion in Section 2.3). This high complexity leads to a long DNN training time. Since

Table 4.5: Comparison of the different RNNs used in Section 4.5. A bidirectional LSTM hidden layer consists of a forward LSTM layer and a backward LSTM layer. The left numbers in the bidirectional LSTM hidden layer sizes correspond to these two layers and the right numbers correspond to the size of each LSTM layer.

TF representation	STFT		ERB	
DNN type	$\mathrm{DNN}_0^{\text{spec}}$	$\mathrm{DNN}_1^{\text{spec-M/S}}$	$\mathrm{DNN}_0^{\text{spec}}$	$\mathrm{DNN}_1^{\text{spec-M/S}}$
Input layer size	1025	2050	256	512
Hidden layer number	2	2	2	2
Hidden layer size	2×1024	2×1024	2×256	2×256
Output layer size	2050	2050	512	512
Total parameters	35,680,258	44,077,058	2,232,832	2,757,120

our study emphasizes multichannel source separation anyway, we decided to use the SDR cost function in the training of DNN for spectrogram fitting only. In doing so, we use the source spatial covariance matrices estimated by the spatial parameter updates of the first EM iteration given the source spectra estimated by $\mathrm{DNN}_0^{\text{spec}}$.

Training algorithm

The parameter initialization mainly follows Section 4.4.1.3. $\mathrm{DNN}_0^{\text{spec}}$ and $\mathrm{DNN}_1^{\text{spec-MSE}}$ are initialized randomly. The resulting $\mathrm{DNN}_1^{\text{spec-MSE}}$ is then used for initializing $\mathrm{DNN}_1^{\text{spec-SDR}}$. The parameter update algorithm, the dropout, and the gradient normalization are the same as in Section 4.4.1.3. The RNNs are unrolled for 128 time steps. The parameter update is done for each minibatch consisting of 8 random 128-frame segments. For the training of $\mathrm{DNN}_0^{\text{spec}}$ and $\mathrm{DNN}_1^{\text{spec-MSE}}$, patience in early stopping is set to 10. For the training of $\mathrm{DNN}_1^{\text{spec-SDR}}$, it is set to 5. The maximum number of training epochs is set to 100 for all cases.

Training data

In general, the DNNs are trained and validated on the available development set. Each song is divided into overlapped segments. The segment length and shift are 2560 frames (corresponding to 59.47 s in the time domain) and 1024 frames (23.78 s), respectively. Each segment is divided into 768 left context frames, 1024 core frames, and 768 right context frames. The DNN training only uses the core frames. Each segment is regarded as a standalone song in the training time and time-invariant source spatial covariance matrices, which are needed for the following DNN training, are estimated for each segment. The left and right context frames are useful to make this estimation more reliable.

We discard segments whose corresponding vocals or accompaniment is completely silent. We then compute the vocals-to-accompaniment ratio for each segment and compute the quartiles, i.e., the first quartile Q_1 and the third quartile Q_3. We then further discard segments whose ratio is considered as an outlier, i.e., ratio $< 2Q_1 - Q_3$ and $2Q_3 - Q_1 <$ ratio. This processing results in 392 development segments. These development segments are then randomly divided into training segments and validation segments with a ratio of 5:1. This results in the so-called 'base' setting consisting of 326 training segments and 66 validation segments. It is worth mentioning that this segmentation is required because we want to consider data augmentation, in which the augmented data is derived from the base setting. We also define a couple of smaller base settings, i.e., '$\frac{1}{2}$base' and '$\frac{1}{5}$base', in which the training and the validation segments are subsets of those of the 'base' setting. Recall that the 'base' setting consists of segments from the 50 development songs. The '$\frac{1}{2}$base' and the '$\frac{1}{5}$base' settings consist of segments from 25 and 10 randomly selected development songs, respectively. We then derive several other settings in which the training and the validation segments consist of those of the base or the smaller base setting augmented by mixtures of randomly selected segments of different sources from the base or the smaller base setting. The amount of data augmentation is indicated by 'v'. For example, in the 'base+2v' setting, each vocals segment results in three mixture segments by mixing the vocals segment with its corresponding original accompaniment segment and two other randomly selected accompaniment segments from the same song or from different songs. The randomly selected segments are different from each other. It is worth mentioning that, because of this random mixing, the augmented data generally does not have a good musicality. Table 4.6 summarizes these different training data settings.

4.5.2 Discussions

The evaluation results in this section are presented in boxplots [McGill et al., 1978]. For each pair of boxes, the light blue and the light green ones show

Table 4.6: Comparison of the different DNN training data settings used in Section 4.5. The total development data shows the approximate total length without the overlapped parts, i.e., only the core frames.

Setting	**Training (in segments)**	**Validation (in segments)**	**Total dev. data (in hours)**
$\frac{1}{2}$base+1v	326	66	2.6
$\frac{1}{5}$base+4v	360	75	2.9
base	326	66	2.6
base+1v	652	132	5.2
base+2v	978	198	7.8
base+4v	1630	330	12.9
base+8v	2934	594	23.3

the performance on the development set and the evaluation set, respectively. The bottom and the top of each box indicate Q_1 and Q_3, respectively. The red line in the middle of a box indicates the median Q_2, while the notch width shows the 95% confidence interval around the median. The median value is also written on the top of each box. The symbol $\times$ shows the mean value. The whisker reaches 1.5 times the interquartile range, i.e., $Q_3 - Q_1$, beyond Q_1 and Q_3.

4.5.2.1 Task-oriented cost function

Figure 4.4 shows the BSS Eval results for the multichannel separated sources at different processing steps, where the STFT representation is used for the DNNs. It consists of eight boxplots arranged in two columns and four rows. The left and right columns show the performance on the vocals and the accompaniment, respectively. From top to bottom, the rows show the SDR, the source-image-to-spatial-distortion ratio (ISR), the SIR, and the SAR. Each boxplot consists of four pairs of boxes. From left to right, these boxes represent the performance after spectrogram initialization using $\text{DNN}_0^{\text{spec}}$, after the iterative spatial updates, after spectrogram fitting using $\text{DNN}_1^{\text{spec-MSE}}$, and after spectogram fitting using $\text{DNN}_1^{\text{spec-SDR}}$. It should be noted that both spectrogram fitting operations are applied after the iterative spatial updates.

For the discussion, let us mainly observe the median values and their confidence intervals. The iterative spatial updates provide minor improvements on both separated sources for all metrics. Both spectrogram fittings generally also provide minor improvements. If we compare the resulting performance to the performance after spectrogram initialization, there are

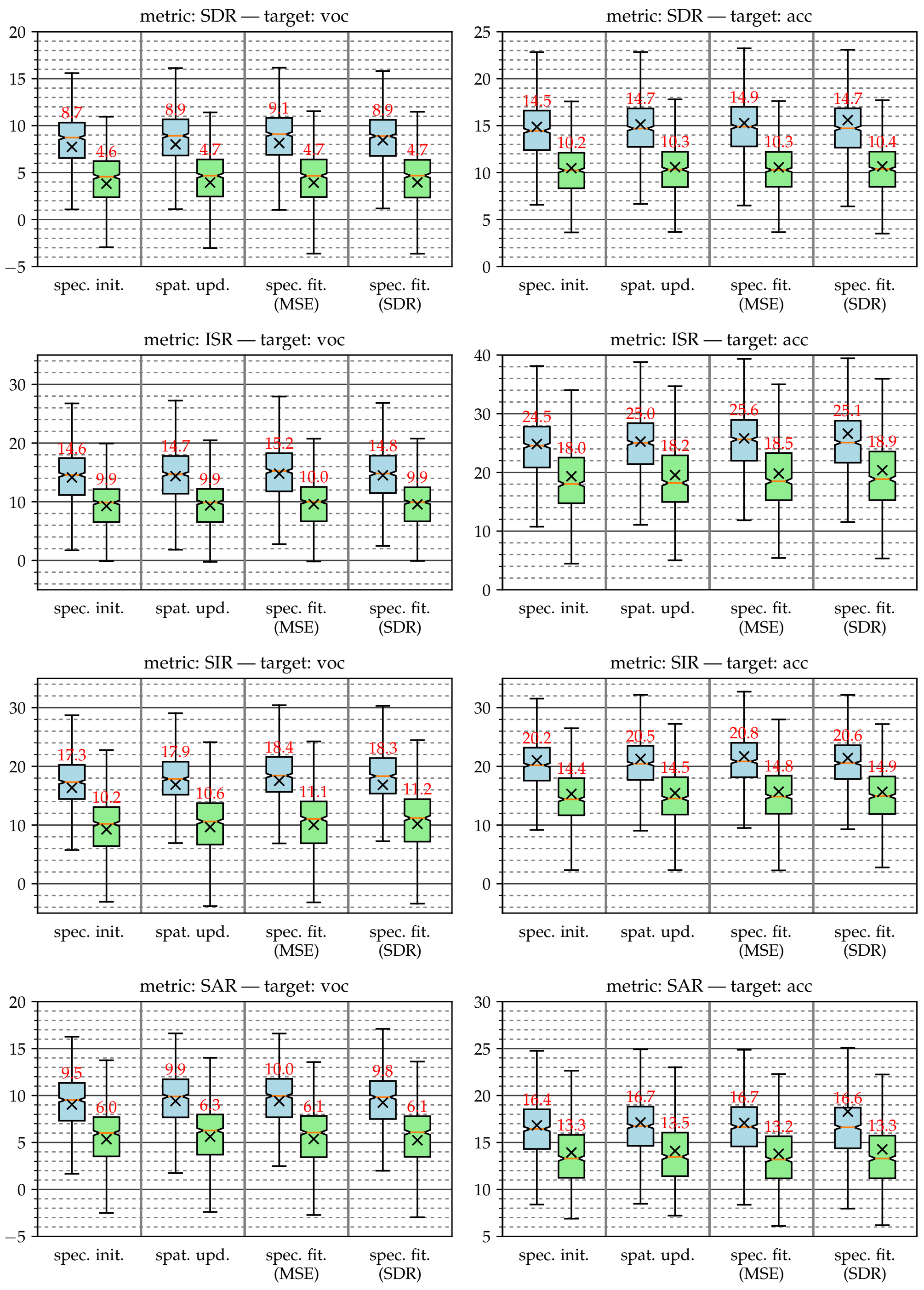
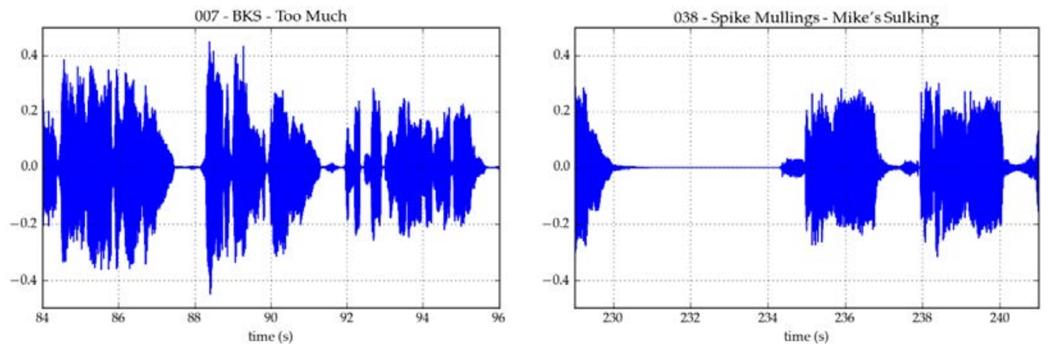
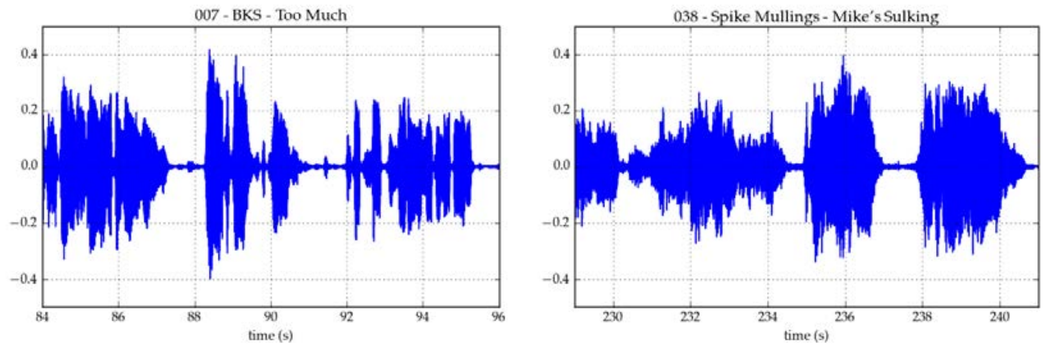


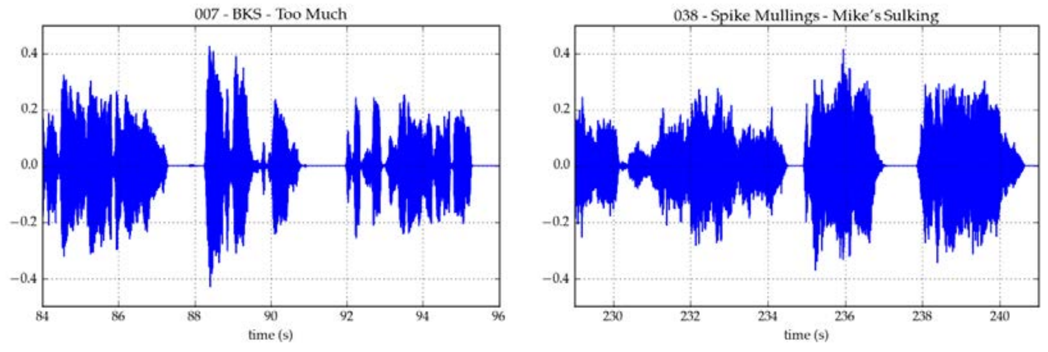
Figure 4.4: Source separation performance for the multichannel separated vocals (left) and accompaniment (right) at different processing steps (presented by the x-axis): the performance after spectrogram initialization using $\text{DNN}_0^{\text{spec}}$, after the iterative spatial updates, and after spectogram fitting using either $\text{DNN}_1^{\text{spec-MSE}}$ or $\text{DNN}_1^{\text{spec-SDR}}$. The DNNs operate on the STFT and are trained on the base data setting. The y-axis is in dB. Higher is better.



(a) Reference



(b) Spectrogram fitting using $\text{DNN}_1^{\text{spec-MSE}}$



(c) Spectrogram fitting using $\text{DNN}_1^{\text{spec-SDR}}$

Figure 4.5: Example left channels of the reference vocals, the separated vocals after spectrogram fitting using $\text{DNN}_1^{\text{spec-MSE}}$, and the separated vocals after spectrogram fitting using $\text{DNN}_1^{\text{spec-SDR}}$. Both segments are taken from two different songs in the evaluation set.

some cases where the improvement is statistically significant. However, they occur on the development set. The exceptional cases for the evaluation set are the statistically significant improvements of the SIR on the vocals after spectrogram fitting using either $\text{DNN}_1^{\text{spec-MSE}}$ or $\text{DNN}_1^{\text{spec-SDR}}$. The performance difference either spectrogram fittings is generally insignificant, although the DNN trained with the SDR cost function shows a slightly lower median performance. Interestingly, if we observe the mean values instead of the medians, this DNN provides noticeable improvements of the SDR, ISR, and SAR on the accompaniment of both development and evaluation sets. This indicates that the DNN trained with the SDR cost function indeed improves the separation of a few songs better than the other DNN trained with the MSE.

Figure 4.5 shows the separated vocals of two different segments from two different evaluation songs and their corresponding references. Spectrogram fitting using $\text{DNN}_1^{\text{spec-SDR}}$ seems to remove the low energy residual distortion compared to spectrogram fitting using $\text{DNN}_1^{\text{spec-MSE}}$. This is noticeable, e.g., between 91-92 s in the left segment and between 237-238 s in the right segment. Except for these time intervals, the waveforms seem to be almost the same. The residual interference between 230-235 s in the right segment after spectrogram fitting using $\text{DNN}_1^{\text{spec-MSE}}$ does not seem to be reduced when $\text{DNN}_1^{\text{spec-SDR}}$ is used. This might be due to the use of $\text{DNN}_1^{\text{spec-MSE}}$ parameters as the initial parameters for the training of $\text{DNN}_1^{\text{spec-SDR}}$ and this training does not change the parameters significantly.

In addition, we compare the computation time of parameter estimation at different processing steps (see Figure 4.4) by running a single-threaded process for each song on Intel® Xeon® CPU E5-2630 v3 @ 2.40GHz. The measured computation time excludes the STFT analysis, the final multichannel Wiener filtering, and the STFT synthesis. The average song length on the whole dataset is 4.2 min. The average elapsed time after spectral initialization, spatial update, spectral update with $\text{DNN}_1^{\text{spec-MSE}}$, and spectral update with $\text{DNN}_1^{\text{spec-SDR}}$ are 11.8 min, 13.3 min, 31.7 min, and 31.4 min, respectively. Roughly, the average computation time for spectral initialization, spatial update, and spectral update are 11.8 min, 1.5 min, and 18.3 min, respectively. This high computational cost is because the DNNs used here have a huge number of parameters. In order to address this, in Section 4.5.2.3, we consider another time-frequency representation, i.e., the ERB representation, with a lower dimensionality that allows us to use DNNs with a smaller number of parameters. Before that, let us compare the performance of our systems to that of the state-of-the-art system.

4.5.2.2 Comparison with the state of the art

Table 4.7 compares the performance we achieve to that of the ideal binary mask, which is regarded as the oracle setting in the MUS task of SiSEC 2016 [Liutkus et al., 2017], and the system of Uhlich et al. [2017], which achieved state-of-the-art performance on this dataset. The ideal binary mask is computed for the left and right channels independently. Uhlich's system is also discussed in Section 2.5.3. In the summary of the SiSEC 2016 by Liutkus et al. [2017], this system is denoted as the 'UHL3' system.

The table shows that our methods perform remarkably well for the development set. Our methods are significantly better than Uhlich's method for all metrics and all sources. They even provide a statistically significant improvement compared to the ideal binary mask for the ISR and the SIR on the vocals and for the SIR on the accompaniment. Unfortunately, this superior performance on the development set is not reflected on the evaluation set. Our methods are outperformed by Uhlich's method, especially for the SDR and the SAR on all sources. The performance difference of these three methods for the ISR on the vocals and the SIR on the accompaniment is statistically insignificant. On the contrary, both of our methods provide a statistically significant improvement for the SIR on the vocals and the ISR on the accompaniment compared to Uhlich's method.

From the above comparison, we conclude that our methods work well but achieve a different interference versus artifacts trade-off than Uhlich's. Specifically, our methods generate more artifacts, but they are more effective in reducing interference, which is arguably more difficult. In the case a higher SAR is desirable, smoothing methods can be applied to increase it at the cost of reducing the SIR [Vincent, 2010]. However, there exist no such method for increasing the SIR if it is low in the first place. Additionally, our methods and Uhlich's method have a comparable performance in handling spatial error on the vocals, which are mixed to the center in most cases. However, our methods statistically outperform Uhlich's method in handling spatial error on the accompaniment, which is possibly panned to the left or right channels. Note that all of these methods, including Uhlich's, employ the weighted spatial parameter updates introduced in Nugraha et al. [2016b] and presented in Chapter 5.

Table 4.7: Source separation performance comparison to the oracle setting (the ideal binary mask; IBM) and the state-of-the-art for the MUS task of SiSEC 2016. The table shows the median values. The 95% confidence intervals vary between ± 0.2 dB and ± 0.5 dB. Boldface numbers show the best performance for each triplet of a metric, a source, and a dataset disregarding the oracle performance (IBM). By taking the confidence intervals into account, italic numbers show performance that is not statistically different from the best performance. Higher is better.

Method	**Vocals**				**Accompaniment**			
	SDR	**ISR**	**SIR**	**SAR**	**SDR**	**ISR**	**SIR**	**SAR**
Development set								
IBM	9.9	13.8	17.5	11.7	15.7	25.4	19.5	18.7
Uhlich et al. [2017]	6.9	11.2	12.0	9.0	12.7	21.0	17.0	16.1
spec. fit. (MSE)	**9.1**	**15.2**	**18.4**	**10.0**	**14.9**	**25.6**	**20.8**	**16.7**
spec. fit. (SDR)	*8.9*	*14.8*	*18.4*	*9.9*	*14.7*	*25.1*	*20.6*	*16.6*
Evaluation set								
IBM	9.5	13.4	16.6	11.2	15.2	24.2	18.8	18.2
Uhlich et al. [2017]	**5.4**	**10.4**	9.2	**8.3**	**11.0**	17.6	**15.1**	**14.7**
spec. fit. (MSE)	4.7	*10.0*	*11.1*	6.1	10.3	*18.5*	*14.8*	13.2
spec. fit. (SDR)	4.7	*9.9*	**11.2**	6.1	10.4	**18.9**	*14.9*	13.3

The most noticeable issue is that there is a big gap between our performance on the development set and that on the evaluation set. This may indicate that we have an overfitting problem in our DNN training. This may be due to the regularization for the DNN training, which includes dropout, gradient normalization, and early stopping, being sub-optimal. Instead of optimizing these regularization techniques, we decided to investigate data augmentation in order to increase the training data variety, which may result in better regularization and better performance on the evaluation set.

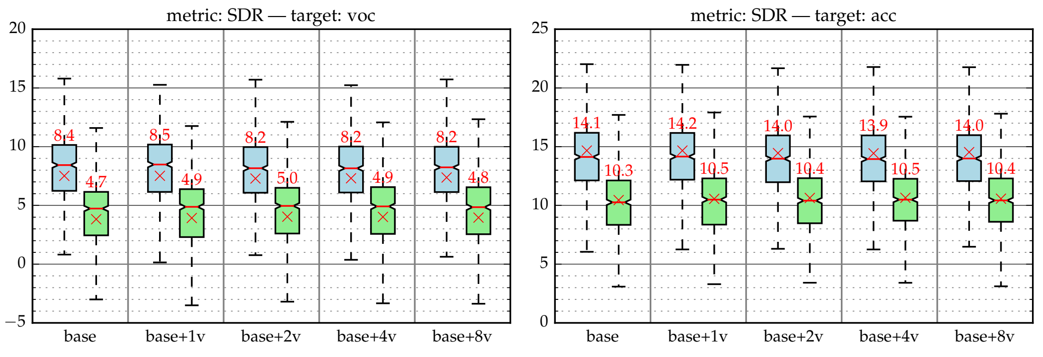
4.5.2.3 Data augmentation

Figure 4.6 shows the SDR evaluated on the multichannel separated sources after spectrogram fitting using different DNNs trained on different training data settings with different data amounts (see Table 4.6). In this case, the ERB representation is used for the DNNs.

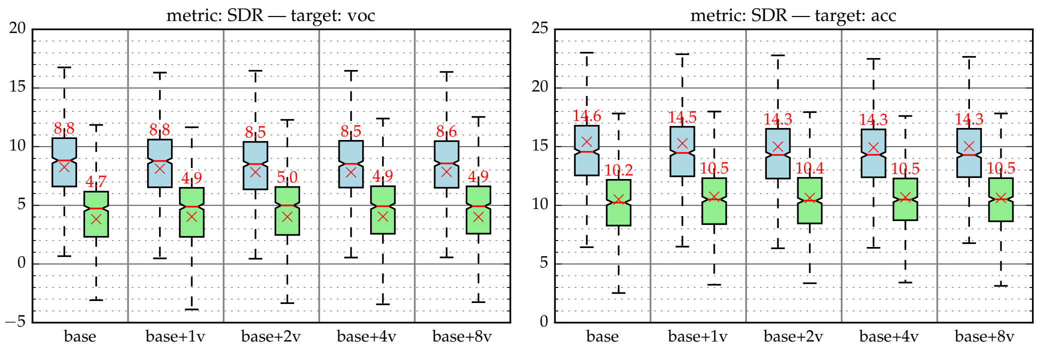
The first thing we need to observe is the performance of the ERB representation compared to that of the STFT representation on the base data setting. For spectrogram fitting using $\text{DNN}_1^{\text{spec-MSE}}$, the STFT provides a significantly better performance than the ERB representation on the development set. However, the performance of both representations on the evaluation set is similar. This shows that the ERB representation is favorable for our experiments because we do not lose performance on the evaluation set, which is our point of interest, and we can minimize the DNN training and testing time due to the smaller number of DNN parameters (see Table 4.5).

As in Section 4.5.2.1, we also compare the computation time of parameter estimation at different processing steps (see Figure 4.4) by running a single-threaded process for each song on Intel® Xeon® CPU E5-2630 v3 @ 2.40GHz. The measured computation time excludes the quadratic ERB extraction and the inverse quadratic ERB transformation, including the final multichannel Wiener filtering. Recall that the average song length on the whole dataset is 4.2 min. The average elapsed time after spectral initialization, spatial update, spectral update with $\text{DNN}_1^{\text{spec-MSE}}$, and spectral update with $\text{DNN}_1^{\text{spec-SDR}}$ are 0.5 min, 2.3 min, 3.4 min, and 3.4 min, respectively. Roughly, the average computation time for spectral initialization, spatial update, and spectral update are 0.5 min, 1.8 min, and 1.1 min, respectively. It means the DNN for initialization and the DNNs for update in this section provide $23.6\times$ and $16.6\times$ speedup, respectively, over those in Section 4.5.2.1.

Surprisingly, the different training data settings with significantly different data amounts do not have any significant impact on the performance. This also means that the increase of training data variety due to the mixing of random segments does not effectively reduce the performance gap occurring between the development and the evaluation sets. The maximum absolute improvements for the evaluation set are only 0.3 dB both on the vocals and the accompaniment. Conveniently, this is in line with a study published recently by Uhlich et al. [2017]. Compared to our data augmentation approach, this study considers a more elaborated approach, i.e., random channel swapping, random amplitude scaling, random segmentation, and random mixing. Since



(a) Spectrogram fitting using $\text{DNN}_1^{\text{spec-MSE}}$



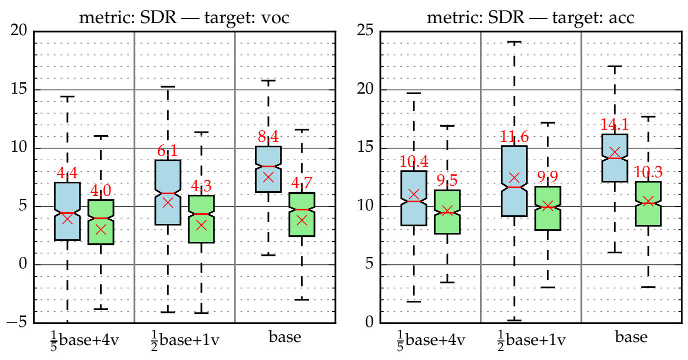
(b) Spectrogram fitting using $\text{DNN}_1^{\text{spec-SDR}}$

Figure 4.6: SDR for the multichannel separated vocals (left) and accompaniment (right) after spectrogram fitting using different DNNs trained on different training data settings (presented by the x-axis). The sizes of the training data settings are increasing from left to right. The DNNs operate on the ERB representation. The y-axis is in dB. Higher is better.

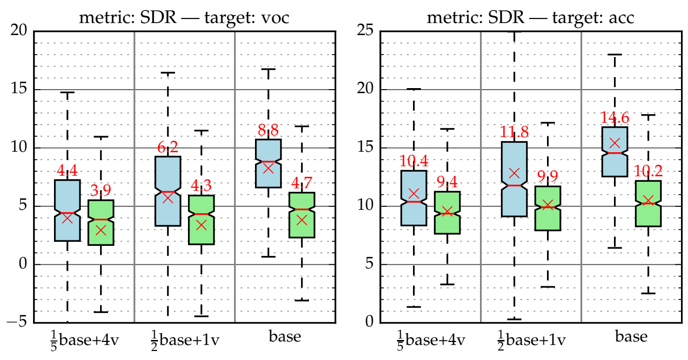
these randomizations are done on-the-fly in the creation of minibatches during the DNN training, we cannot compare the total DNN training data amount. With that approach, the study achieves 0.2 dB absolute improvement for both vocals and accompaniment on the evaluation set.

The above results motivate us to further investigate the impact of the DNN training data size. Figure 4.7 shows the SDR evaluated on the multichannel separated sources after spectrogram fitting using different DNNs trained on different training data settings (see Table 4.6).

All three data settings, i.e., the '$\frac{1}{5}$base+4v', the '$\frac{1}{2}$base+1v', and the 'base' settings, have a comparable total training data amount. The figure shows that although the data amounts are similar, the SDRs are clearly different. This indicates that the performance is not affected by the total training data



(a) Spectogram fitting using $\text{DNN}_1^{\text{spec-MSE}}$



(b) Spectogram fitting using $\text{DNN}_1^{\text{spec-SDR}}$

Figure 4.7: SDR for the multichannel separated vocals (left) and accompaniment (right) after spectrogram fitting using different DNNs trained on different training data settings (presented by the x-axis). All training data settings have a similar training data size. The DNNs operate on the ERB representation. The y-axis is in dB. Higher is better.

amount. It is more likely affected by the initial song variety in the '$\frac{1}{5}$base', the '$\frac{1}{2}$base', or the 'base' setting. We can observe that the performance increases along with the number of songs from which the DNN training segments are derived. We also can observe that the gap between the performance of the development set and that of the evaluation set widens along with the number of songs. This leads to the conclusion that, in this case, the source separation performance is bounded by the dataset and deriving additional training data from this dataset is not effective to improve the performance.

4.6 Summary

This chapter presents our studies on the impact of different design choices on system performance. These design choices include different cost functions, different time-frequency representations, different DNN architectures, and different DNN training data sizes. These design choices are employed in a speech enhancement system, as in Chapter 3, and a vocals-accompaniment separation system.

Among the general-purpose cost functions we consider, the KL divergence is the most reasonable choice because it provides the best source separation performance and works well in terms of speech recognition performance. The probabilistically-motivated IS divergence provides the best speech recognition performance, but its source separation performance falls behind the others. Unfortunately, the newly proposed task-oriented SDR cost function fails to improve the median vocals-accompaniment separation performance compared to the simple MSE, but it does improve the average SDR to some extent.

We show that time-frequency representations based on the perceptually-motivated ERB scale are favorable. These low-dimensional representations allow us to use smaller DNNs and reduce the DNN training time. The quadratic ERB representation derived using a time-domain filterbank provides a speech recognition performance improvement compared to the previously used STFT representation. Another ERB representation derived using a frequency-domain filterbank provides a similar source separation performance to that of STFT representation.

We also show that RNNs indeed handle the context better than the frame concatenation based approach we used for FNNs (see Chapter 3). For multichannel separation, the performance gap provided by RNNs and FNNs is negligible when the training involves a relatively small amount of data. Interestingly, the performance of FNNs falls behind that of RNNs when more training data is used.

Finally, data augmentation is not trivial and it is an active separate research topic indeed [Žmolíková et al., 2016; Sivasankaran et al., 2017; Zhang et al., 2017].

Additionally, our vocals-accompaniment separation systems performs remarkably close to the state-of-the-art performance in the context of the professionally-produced music recordings task of SiSEC 2016. Our systems exhibit artifacts which is detrimental to the overall distortion. However, our systems are significantly better in removing the accompaniment from the vocals, which may be favorable for a singing-voice separation task. It is also worth mentioning that in a precursor study [Nugraha et al., 2016b], we proposed vocals-instrument separation systems that extract four sources, namely vocals, bass, drums, and other. See Liutkus et al. [2017] for the performance comparison between our systems and the other systems submitted to the MUS task of SiSEC 2016.

CHAPTER 5

Estimation of spatial parameters with deep neural networks

In the two previous chapters, we presented our study on the estimation of the spectral parameters with deep neural networks (DNNs). By contrast, this chapter deals with the estimation of the spatial parameters. We introduce our second DNN based multichannel audio source separation framework. In this framework, both the spectral and the spatial parameters are initialized and updated by DNNs with possibly additional iterative expectation-maximization (EM) spatial parameter updates. Furthermore, we present a weighted spatial parameter estimation formula, which is a generalization of the exact EM formulation in (2.49).

5.1 Research questions

In this chapter, we address the following research questions.

1. **Is the exact EM spatial parameter update optimal?** The framework we proposed in Algorithm 2) uses the exact EM spatial parameter update in (2.49) following the general iterative EM algorithm in Algorithm 1. Another formulation of spatial parameter estimation was used by Liutkus et al. [2015b] and shown to be effective. In Section 5.2, we present a general formula for weighted spatial parameter updates and detail how different choices of the weights may lead to the exact EM spatial parameter estimation or to the estimation used by Liutkus et al. [2015b]. We then study the impact of this weighted spatial parameter update on the system performance.

2. **Can DNNs be used to model the spatial parameters within a multichannel Gaussian model based separation framework?** We have shown in the two previous chapters that DNNs satisfactorily estimate the spectral

parameters of the multichannel Gaussian model, i.e., the time-varying source power spectral densities (PSDs). We now want to explore the use of DNNs for estimating the spatial parameters, i.e., the time-invariant source spatial covariance matrices. To the best of our knowledge, this is a novel use of DNNs. In our study, we propose a suitable representation of these matrices and consider various design choices for the DNN training cost function and the DNN architecture.

3. **Can EM parameter updates be completely replaced by DNNs?** This is a follow-up question to the above. If DNNs also satisfactorily estimate the spatial parameters of the multichannel Gaussian model, we may expect that all parameter updates can be done with spectral and spatial DNNs only. We investigate whether additional updates following the EM formulation are still required.

These research questions are investigated by applying the newly proposed framework to the same speech enhancement problem considered in the two previous chapters.

5.2 Weighted spatial parameter updates

Intuitively, we should ignore possibly noisy source spatial information, if any, when a source is inactive or silent. This can be seen in (2.49) as $v_j(f,n)^{-1}\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)$ becomes inaccurate when $v_j(f,n)$ tends towards 0.

We propose the following general formula for weighted spatial parameter estimation:

$$\mathbf{R}_j(f) = \left(\sum_{n=1}^{N}\omega_j(f,n)\right)^{-1}\sum_{n=1}^{N}\frac{\omega_j(f,n)}{v_j(f,n)}\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n), \tag{5.1}$$

where $\omega_j(f,n)$ denotes the weight of source j for frequency bin f and frame n. This formula is a generalization of the spatial parameter estimation in (2.49) and that of Liutkus et al. [2015b].

When $\omega_j(f,n)$ is set to be equal for all time-frequency bins (f,n), e.g., $\omega_j(f,n) = 1$, (5.1) reduces to the exact EM formulation in (2.49). When $\omega_j(f,n) = v_j(f,n)$, (5.1) reduces to [Liutkus et al., 2015b]

$$\mathbf{R}_j(f) = \left(\sum_{n=1}^{N}v_j(f,n)\right)^{-1}\sum_{n=1}^{N}\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n). \tag{5.2}$$

Table 5.1: Speech recognition performance in terms of word error rate (WER) (%) using the different spatial parameter updates. The automatic speech recognition (ASR) system uses the DNN+sMBR back-end trained on multi-condition noisy data followed by 5-gram Kneser-Ney smoothing and RNN-LM rescoring. The evaluation was done on the real sets. Boldface numbers show the best performance for each dataset. The 95% confidence intervals for the two best WERs are ± 0.25% for the development set and ± 0.36% for the test set. Lower is better.

TF rep.	**Arch.**	**EM iter.**	**Update type**	**Spatial update**	**SS-R**		**SS-F**		**Row no.**
					Dev	**Test**	**Dev**	**Test**	
STFT	FNN	1	spatial	exact	5.45	9.13	6.29	13.17	1
		1	spatial	weighted	**4.71**	**7.77**	5.58	11.34	2
	RNN	1	spatial	exact	6.50	9.68	5.68	10.16	3
		1	spatial	weighted	4.89	8.27	4.73	8.75	4
QERB	FNN	1	spatial	exact	5.21	9.21	5.69	11.00	5
		1	spatial	weighted	4.73	8.06	5.13	9.44	6
	RNN	1	spatial	exact	5.57	9.30	5.39	9.02	7
		1	spatial	weighted	4.93	8.14	4.78	7.86	8

Experience shows that this weighting trick mitigates inaccurate estimates in certain time-frequency bins and increases the importance of those bins for which $v_j(f,n)$ is high.

In the following discussion, we refer the spatial parameter update in (2.49) as the *exact* spatial parameter update and that in (5.1) as the *weighted* spatial parameter update. We do not address the optimization of the weights $\omega_j(f,n)$ in this study. We stick to $\omega_j(f,n) = v_j(f,n)$ as in (5.2).

We evaluate the impact of the weighted spatial parameter update on the speech recognition performance. Except for the spatial parameter update, all experimental settings used here are the same as in Section 4.4.

Table 5.1 compares the performance of the exact and the weighted spatial parameter updates. It basically extends the experimental results shown in Table 4.4. The performance of the exact update, which is shown in the odd rows of Table 5.1, is taken from Table 4.4. Recall from Section 4.4.1.3 that SS-R considers only the real data set for the DNN training, while SS-F considers both the real and the simulated data sets.

Table 5.1 demonstrates that the weighted spatial parameter update provides a significant performance improvement in all cases. The improvement ranges from 1.15% to 1.82% absolute, or from 12% to 15% relative, w.r.t. the exact spatial update on the test set. On average, the weighted spatial update provides 14% relative improvement w.r.t. the exact spatial update on the test set.

In addition, see Nugraha et al. [2016b] for the impact of weighted spatial parameter updates on the performance of vocals-instruments separation.

In the following section, we present a new framework in which spatial DNNs are used to perform the whole spatial parameter update rather than to estimate the weights $\omega_j(f, n)$. Yet, we will see that the DNN training implicitly encompasses the weight learning.

5.3 Iterative framework with spectral and spatial DNNs

The framework described in Algorithm 3 extends our framework in Algorithm 2 by employing DNNs for initializing and updating both spectral and spatial parameters. The computation of the source posteriors within this algorithm relies on Algorithm 4. Algorithm 3 also includes additional iterative weighted EM spatial parameter updates (see Section 5.2) with a convergence criterion described in Section 5.5. Similarly to Algorithm 2, Algorithm 3 also follows pre-processing, initialization, and multichannel filtering steps whose general descriptions are presented in Section 3.3.2.

Recall that k and l denote the spatial update iteration and the EM iteration, respectively. In Algorithm 2, we use one spectral DNN for spectrogram initialization, $\text{DNN}_0^{\text{spec}}$, and possibly one or more spectral DNNs for spectrogram fitting, $\text{DNN}_l^{\text{spec}}$. In addition to those DNNs, in Algorithm 3, we use one spatial DNN for initializing the spatial parameters, $\text{DNN}_0^{\text{spat}}$, and possibly one or more spatial DNNs for updating the spatial parameters, $\text{DNN}_{l,k}^{\text{spat}}$.

Spectral DNNs are used to estimate the source magnitude spectra $\sqrt{\widetilde{v}_j(f,n)}$ given the mixture magnitude spectrum $\sqrt{z_x(f,n)}$ (for $\text{DNN}_0^{\text{spec}}$) or a stack of source magnitude spectra estimated in the preceding iteration (for $\text{DNN}_l^{\text{spec}}$). The spectral parameters are then easily computed by squaring these source magnitude spectra.

Algorithm 3 Iterative framework with spectral and spatial DNNs.

```
Inputs:
    Time-domain mixture $\mathbf{x}(t)$
    Number of sources $J$
    Number of spatial updates $K$ and number of EM iterations $L$
    DNN spectral models $\mathrm{DNN}_0^{\mathrm{spec}}, \mathrm{DNN}_1^{\mathrm{spec}}, \ldots, \mathrm{DNN}_L^{\mathrm{spec}}$
    DNN spatial models $\mathrm{DNN}_0^{\mathrm{spat}}, \mathrm{DNN}_{1,1}^{\mathrm{spat}}, \ldots, \mathrm{DNN}_{L,K}^{\mathrm{spat}}$
Pre-processing step:
 1: Align the observed mixture: $\overline{\mathbf{x}}(t) \leftarrow \mathrm{align}(\mathbf{x}(t))$
 2: Extract QERB representation: $\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n) \leftarrow \mathrm{QERBT}(\overline{\mathbf{x}}(t))$
 3: Compute features for the DNN inputs:
        $\sqrt{z_x(f,n)} = \sqrt{\frac{1}{I}\mathrm{tr}\left(\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n)\right)}$    (4.11)
        $\mathbf{L}_{\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}}(f,n) \leftarrow \mathrm{Cholesky}\left(\frac{1}{z_x(f,n)}\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n)\right)$
Initialization step:
 4: Initialize all source spectrograms:
        $[v_1(f,n),\ldots,v_J(f,n)] \leftarrow \mathrm{DNN}_0^{\mathrm{spec}}\left(\sqrt{z_x(f,n)}\right)^2_{\forall f\in[1,F],\forall n\in[1,N]}$
 5: Estimate the Cholesky decomposition of all source spatial covariance
        matrices:
        $[\mathbf{L}_{\mathbf{R}_1}(f),\ldots,\mathbf{L}_{\mathbf{R}_J}(f)] \leftarrow$
            $\mathrm{DNN}_0^{\mathrm{spat}}\left(\mathbf{L}_{\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}}(f,n), \sqrt{z_x(f,n)}, \left[\sqrt{v_1(f,n)},\ldots,\sqrt{v_J(f,n)}\right]\right)_{\forall n\in[1,N]}$
 6: for each source $j$ of $J$ do
 7:     Initialize the spatial covariance matrix: $\mathbf{R}_j(f) = \mathbf{L}_{\mathbf{R}_j}(f)\mathbf{L}_{\mathbf{R}_j}(f)^*$
 8: end for
```

(continued on the next page)

Recall from our discussion about the multichannel Gaussian model in Section 2.5.2.1 that the covariance matrices $\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n)$, $\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)$, and $\mathbf{R}_j(f)$ are complex-valued Hermitian positive-definite matrices. In this newly proposed framework, we exploit the fact that the Cholesky decomposition of a Hermitian positive-definite matrix yields a unique lower triangular matrix with real and positive diagonal entries, e.g., $\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n) = \mathbf{L}_{\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}}(f,n)\mathbf{L}_{\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}}(f,n)^*$.

Instead of the time-invariant source spatial covariance matrices $\mathbf{R}_j(f)$, spatial DNNs are used to estimate their Cholesky decompositions $\mathbf{L}_{\mathbf{R}_j}(f)$ as unconstrained lower triangular complex-valued matrices. The essential benefit of estimating the Cholesky decomposition is that, although the DNN outputs $\mathbf{L}_{\mathbf{R}_j}(f)$ are unconstrained, the reconstructed source spatial

Algorithm 3 Iterative framework with spectral and spatial DNNs (continued).

```
Multichannel filtering step:
 9: for each EM iteration $l$ of $L$ do
10:     Compute the source posteriors:                                    ▷ (see Algorithm 4)
            $\left[\widehat{\mathbf{R}}_{\mathbf{c}_1}(f,n),\dots,\widehat{\mathbf{R}}_{\mathbf{c}_J}(f,n)\right] \leftarrow$
                $\text{post}\left(\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n),\left[v_1(f,n),\dots,v_J(f,n)\right],\left[\mathbf{R}_1(f),\dots,\mathbf{R}_J(f)\right]\right)$
11:     for each source $j$ of $J$ do
12:         Compute the unconstrained source spectrogram:
                $z_j(f,n)=\frac{1}{I}\text{tr}\left(\mathbf{R}_j^{-1}(f)\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)\right)$    (2.50)
13:     end for
14:     Update all source spectrograms:
            $[v_1(f,n),\dots,v_J(f,n)] \leftarrow$
                $\text{DNN}_l^{\text{spec}}\left(\left[\sqrt{z_1(f,n)},\dots,\sqrt{z_J(f,n)}\right]\right)^2_{\forall f\in[1,F],\forall n\in[1,N]}$
15:     for each spatial update $k$ of $K$ do
16:         Compute the source posteriors:                                ▷ (see Algorithm 4)
                $\left[\widehat{\mathbf{R}}_{\mathbf{c}_1}(f,n),\dots,\widehat{\mathbf{R}}_{\mathbf{c}_J}(f,n)\right] \leftarrow$
                    $\text{post}\left(\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n),\left[v_1(f,n),\dots,v_J(f,n)\right],\left[\mathbf{R}_1(f),\dots,\mathbf{R}_J(f)\right]\right)$
17:         for each source $j$ of $J$ do
18:             Compute features for the DNN inputs:
                    $\sqrt{z_j(f,n)}=\sqrt{\frac{1}{I}\text{tr}\left(\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)\right)}$
                    $\mathbf{L}_{\widehat{\mathbf{R}}_{\mathbf{c}_j}}(f,n) \leftarrow \text{Cholesky}\left(\frac{1}{z_j(f,n)}\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)\right)$
19:         end for
20:         Re-estimate the Cholesky decomposition of all source spatial
                covariance matrices:
                $[\mathbf{L}_{\mathbf{R}_1}(f),\dots,\mathbf{L}_{\mathbf{R}_J}(f)] \leftarrow$
                    $\text{DNN}_{l,k}^{\text{spat}}\left(\begin{bmatrix}\mathbf{L}_{\widehat{\mathbf{R}}_{\mathbf{c}_1}}(f,n) & \dots & \mathbf{L}_{\widehat{\mathbf{R}}_{\mathbf{c}_J}}(f,n)\\ \sqrt{z_1(f,n)} & \dots & \sqrt{z_J(f,n)}\\ \sqrt{v_1(f,n)} & \dots & \sqrt{v_J(f,n)}\end{bmatrix}\right)_{\forall n\in[1,N]}$
21:         for each source $j$ of $J$ do
22:             Update the spatial covariance matrix:
                    $\mathbf{R}_j(f)=\mathbf{L}_{\mathbf{R}_j}(f)\mathbf{L}_{\mathbf{R}_j}(f)^*$
23:         end for
24:     end for
25: end for
```

(continued on the next page)

Algorithm 3 Iterative framework with spectral and spatial DNNs (continued).

```
26: while not converged do        ▷ optional EM spatial updates (see Section 5.5)
27:     Compute the source posteriors:                    ▷ (see Algorithm 4)
            $\left[\widehat{\mathbf{R}}_{\mathbf{c}_1}(f,n),\dots,\widehat{\mathbf{R}}_{\mathbf{c}_J}(f,n)\right] \leftarrow$
                $\mathrm{post}\left(\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n),[v_1(f,n),\dots,v_J(f,n)],[\mathbf{R}_1(f),\dots,\mathbf{R}_J(f)]\right)$
28:     for each source $j$ of $J$ do
29:         Update the spatial covariance matrix:
                $\mathbf{R}_j(f)=\left(\sum_{n=1}^{N} v_j(f,n)\right)^{-1}\sum_{n=1}^{N}\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)$        (5.2)
30:     end for
31: end while

32: for each source $j$ of $J$ do
33:     Compute the final spatial image:
            $\widehat{\mathbf{c}}_j(t)=\mathrm{IQERBT}\left(\frac{v_j(f,n)\mathbf{R}_j(f)}{\sum_{j'=1}^{J} v_{j'}(f,n)\mathbf{R}_{j'}(f)},\ \overline{\mathbf{x}}(t)\right)$
34: end for
```

Outputs:
All spatial source images $[\widehat{\mathbf{c}}_1(t),\dots,\widehat{\mathbf{c}}_J(t)]$

Algorithm 4 Source posterior computation

Inputs:
QERB representation of mixture $\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n)$
Source spectrograms $[v_1(f,n),\dots,v_J(f,n)]$
Source spatial covariance matrices $[\mathbf{R}_1(f),\dots,\mathbf{R}_J(f)]$

```
1: Compute the mixture covariance matrix:
       $\mathbf{R}_{\mathbf{x}}(f,n)=\sum_{j=1}^{J} v_j(f,n)\mathbf{R}_j(f)$        (2.44)
2: for each source $j$ of $J$ do
3:     Compute the Wiener filter gain:
           $\mathbf{W}_j(f,n)=v_j(f,n)\mathbf{R}_j(f)\mathbf{R}_{\mathbf{x}}(f,n)^{-1}$        (2.47)
4:     Compute the posterior second-order raw moments of the spatial
           image:
           $\widehat{\mathbf{R}}_{\mathbf{c}_j}(f,n)=\mathbf{W}_j(f,n)\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n)\mathbf{W}_j(f,n)^*$
                   $+\left(\mathbf{I}-\mathbf{W}_j(f,n)\right)v_j(f,n)\mathbf{R}_j(f)$        (4.12)
5: end for
```

Outputs:
Source posteriors $\left[\widehat{\mathbf{R}}_{\mathbf{c}_1}(f,n),\dots,\widehat{\mathbf{R}}_{\mathbf{c}_J}(f,n)\right]$

covariance matrices $\mathbf{R}_j(f) = \mathbf{L}_{\mathbf{R}_j}(f)\mathbf{L}_{\mathbf{R}_j}(f)^*$ are always positive-semidefinite. By contrast, imposing a positive semidefiniteness constraint on the DNN outputs when using spatial DNNs to directly estimate the source spatial covariance matrices $\mathbf{R}_j(f)$ would be more difficult.

Inspired by the spatial parameter update in (2.49) or (5.1), we generally want to provide spatial DNNs with source spectrograms $v_j(f, n)$ and source covariance matrices $\widehat{\mathbf{R}}_{\mathbf{c}_j}(f, n)$ or the mixture covariance matrix $\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}$. For $\text{DNN}_0^{\text{spat}}$, we provide the source magnitude spectra estimated by $\text{DNN}_0^{\text{spec}}$ $\sqrt{v_j(f, n)}$, the Cholesky decomposition of the normalized mixture covariance matrix $\mathbf{L}_{\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}}(f, n)$, and the square-root of the corresponding normalization factor (which is the mixture magnitude spectrum) $\sqrt{z_x(f, n)}$. For $\text{DNN}_{l,k}^{\text{spat}}$, we provide the source magnitude spectra estimated in the preceding iteration $\sqrt{v_j(f, n)}$, the Cholesky decomposition of the normalized source covariance matrices $\mathbf{L}_{\widehat{\mathbf{R}}_{\mathbf{c}_j}}(f, n)$, and the square-root of the corresponding normalization factor $\sqrt{z_j(f, n)}$. The normalization of the mixture or source covariance matrices is required to avoid numerical issues during the Cholesky decomposition. The normalization factor is also provided so that the scale of the covariance matrix can be exploited by the spatial DNNs.

The spectral and the spatial DNNs used in the experiments presented in this chapter are trained separately by minimizing different cost functions specific to the DNN purpose. This makes it possible to observe whether each spectral or spatial DNN provides a performance improvement. The spectral and the spatial DNNs could then be jointly fine-tuned. This fine-tuning would resemble the deep unfolding approach [Le Roux et al., 2015; Wisdom et al., 2016] and the whole system consisting of multiple DNNs would resemble end-to-end systems, e.g., the systems of Xiao et al. [2016] and Sainath et al. [2017]. We consider this joint training as one of possible future directions.

In Algorithm 3, we employ the quadratic equivalent rectangular bandwidth (ERB) representation [Duong et al., 2010b]. QERBT denotes the quadratic ERB extraction and IQERBT denotes the inverse quadratic ERB transformation, including the Wiener filtering and the filterbank inversion to recover the time-domain spatial images $\widehat{\mathbf{c}}_j(t)$. See the related discussion in Section 4.4.1.

5.4 Experimental settings

This section presents the settings for all experiments described in the following sections.

5.4.1 Task and dataset

We consider the application of the framework in Algorithm 3 to several multichannel speech enhancement problems using the same CHiME dataset as the one described in Section 3.3.1. We deal with the separation of two sources ($J = 2$), namely speech and noise, from a 6-channel mixture ($I = 6$), a 4-channel mixture ($I = 4$), and a 2-channel mixture ($I = 2$). As shown by Vincent et al. [2017b], the fewer microphones are used, the more challenging it is to enhance the speech. In contrast to the evaluation in Chapter 3, here, we focus on the speech recognition performance measured by the WER.

For the experiments on 2-channel mixtures, we follow the guidelines of the 2-channel task of the CHiME-4 challenge. For each utterance in the development and the test sets of the CHiME dataset, the organizers randomly selected one pair of microphones from the five front-facing microphones. In doing so, they tried to avoid selecting failed microphones based on the cross-correlation coefficients. For the experiments on 4-channel mixtures, we performed a similar channel selection, i.e., randomly selected one 4-tuple of microphones from the five front-facing microphones and took the cross-correlation coefficients into consideration for each utterance in the development and the test sets. In addition, we considered all possible combinations of microphones in the real training set. Thus, for each training utterance, we obtained five different 4-channel signals for the 4-channel task and ten different 2-channel signals for the 2-channel task.

The following subsection presents the implementation details of the proposed source separation framework.

5.4.2 An overview of the speech enhancement system

The description below provides the details of pre-processing, initialization, multichannel filtering, and post-processing steps specific to the system used in all experiments presented in this section. See Section 3.3.2 for the general description of each step. Following this overview, the details of spectral and spatial DNNs are presented.

The pre-processing step aligns the observed mixture as described in Section 3.3.2 and then extracts the quadratic ERB coefficients using a time-domain filterbank as Duong et al. [2010b], resulting in $F = 128$ frequency bins.

The initialization step sets the initial source spectrograms using $\text{DNN}_0^{\text{spec}}$ and the initial source covariance matrices using $\text{DNN}_0^{\text{spat}}$.

The multichannel filtering step iteratively performs the spectral and spatial parameter updates as defined by Algorithm 3. Our experiments consider one additional spectral DNN, i.e., $\text{DNN}_1^{\text{spec}}$, and one additional spatial DNN, i.e., $\text{DNN}_{1,1}^{\text{spat}}$. In order to avoid numerical instabilities due to the use of single precision computing for DNNs, the spectrograms $v_j(f,n)$ are floored to 10^{-5}.

The post-processing step averages the estimated multichannel speech spatial image across channels to obtain a single-channel signal for the ASR evaluation as in Section 3.3.2.

5.4.3 DNN spectral models

We now present the architecture, the inputs and outputs, the training criterion, the training algorithm, and the training data of the DNN spectral models $\text{DNN}_0^{\text{spec}}$ and $\text{DNN}_1^{\text{spec}}$.

5.4.3.1 Architecture, inputs, and outputs

Both $\text{DNN}_0^{\text{spec}}$ and $\text{DNN}_1^{\text{spec}}$ follow a bidirectional long short-term memory (LSTM) architecture as in Section 4.4.1.2. Similarly to the DNN sizes in that section, both DNNs consist of an input layer, two bidirectional LSTM hidden layers, and an output layer. $\text{DNN}_0^{\text{spec}}$ has an input layer size of $F = 128$ and $\text{DNN}_1^{\text{spec}}$ of $F \times J = 256$. The size of each bidirectional LSTM hidden layer in both DNNs is $2 \times F = 256$. The sizes of the output layers are $F \times J = 256$.

The inputs of $\text{DNN}_0^{\text{spec}}$ are computed as $\sqrt{z_x(f,n)} = \sqrt{\text{tr}\left(\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n)\right)/I}$ as in (4.11). The inputs of $\text{DNN}_1^{\text{spec}}$ are a stack of $\sqrt{z_j(f,n)}$. As in Section 4.4.1.2, the DNN inputs are standardized frequency bin-wise and the standardization factors are computed on the whole training set. The spectral DNN outputs correspond to the square-roots of the source spectrograms $v_j(f,n)$. Figure 5.1a depicts the architecture of the spectral DNNs we used in this section.

5.4.3.2 Training criterion, algorithm, and data

Both spectral DNNs are trained using the Kullback-Leibler (KL) divergence in (3.3) with $\delta_{\text{cf}} = 10^{-5}$. The DNN training algorithm mainly follows the description in Section 4.4.1.3, although there are a few differences. A minibatch consists of 16, instead of 8, randomly selected full utterances. Although this does not ensure a performance improvement, this reduces the time spent by one training epoch. Dropout with rate 0.2 is now applied on the bidirectional LSTM layer inputs [Zaremba et al., 2015] and the fully-connected output layer input.

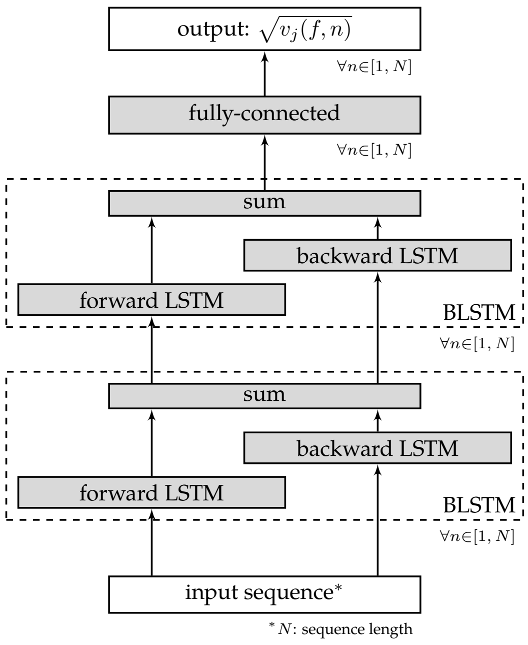
As also described in Section 4.4.1.3, the spectral DNN training targets $\sqrt{\widetilde{v}_j(f,n)}$ are computed similarly to (4.11) from the re-aligned source spatial image signals $\overline{\mathbf{c}}_j(t)$. In this section, we use the real training set (`tr05_real`) and the real development set (`dt05_real`). Our previous experiments have shown that using the real sets only is sufficient to achieve a good performance. This setting corresponds to SS-R in Section 4.4.1.3.

5.4.4 DNN spatial models

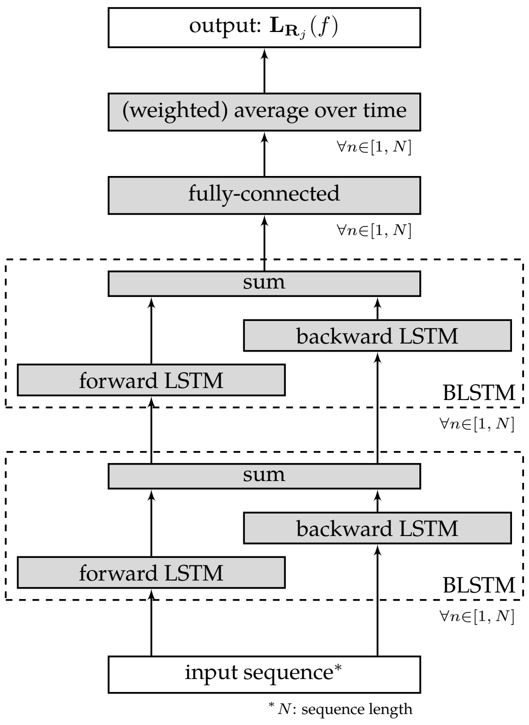
We now present the architecture, the inputs and outputs, the training algorithm, and the training data of the DNN spectral models $\text{DNN}_0^{\text{spat}}$ and $\text{DNN}_{1,1}^{\text{spat}}$. Similarly to our spectral DNNs, these spatial DNNs also estimate the Cholesky decompositions of all source spatial covariance matrices simultaneously. However, spatial DNNs work frequency bin-wise. This means that the DNNs perform the estimation for *one* frequency bin at a time. Motivated by the spatial parameter estimation (5.1) that is frequency-independent, both spatial DNNs are also designed to be frequency-independent. This means that the same DNNs are applied to each frequency bin. It is worth mentioning that the estimation for all frequency bins can be done in parallel so that this approach does not have any impact on the testing time. By contrast, this approach significantly increases the training data by putting all the frequency bins together.

5.4.4.1 Architecture, input, and outputs

Both $\text{DNN}_0^{\text{spat}}$ and $\text{DNN}_{1,1}^{\text{spat}}$ also follow a bidirectional LSTM architecture consisting of an input layer, two bidirectional LSTM hidden layers, and an output layer. $\text{DNN}_0^{\text{spat}}$ has an input layer size of $I^2 + J + 1$ and $\text{DNN}_{1,1}^{\text{spat}}$ of $(I^2 + 2)J$. The size of each bidirectional LSTM hidden layer in all DNNs is



(a) All-frequency spectral DNN



(b) Single-frequency spatial DNN

Figure 5.1: Spectral and spatial DNN architectures used in in Chapter 5. For the input sequences of spectral DNNs, see lines 4 and 14 of Algorithm 3 and Section 5.4.3.1. For the input sequences of spatial DNNs, see lines 5 and 20 of Algorithm 3 and Section 5.4.4.1. The white boxes represent the visible data and the gray boxes represent the hidden layers or the hidden processes.

$2 \times 2I^2$. The sizes of the output layers are $2I^2$. Table 5.2 summarizes the sizes of the different recurrent neural networks (RNNs) used for the experiments in this section.

Section 5.3 and Algorithm 3 have provided some details on how the DNN inputs are computed. The inputs of $\text{DNN}_0^{\text{spat}}$ are supervectors constructed from the Cholesky decomposition of the normalized mixture covariance matrix, the square-root of the corresponding normalization factor, and the estimated source magnitude spectra. The Cholesky decomposition of a Hermitian positive-definite matrix results in a unique complex-valued lower triangular matrix with positive real-valued diagonal entries. For constructing the supervectors, we decompose the complex-valued entries into their real and imaginary parts, then concatenate them with the real-valued diagonal entries, the square-root of the normalization factor, and the estimated source magnitude spectra. A similar approach is applied to the inputs of $\text{DNN}_{1,1}^{\text{spat}}$. The DNN inputs are standardized entry-wise and the standardization factors

Table 5.2: Comparison of the different spatial DNNs used in the different speech enhancement tasks in Chapter 5. A bidirectional LSTM hidden layer consists of a forward LSTM layer and a backward LSTM layer. The left numbers in the bidirectional LSTM hidden layer sizes correspond to these two layers and the right numbers correspond to the size of each LSTM layer.

	Formula	**Task**		
		6-channel	**4-channel**	**2-channel**
DNN$_0^{\text{spat}}$ input layer size	I^2+J+1	39	19	7
DNN$_{1,1}^{\text{spat}}$ input layer size	$(I^2+2)J$	76	53	12
Hidden layer number	2	2	2	2
Hidden layer size	$2 \times I^2 J$	2×72	2×32	2×8
Output layer size	$I^2 J$	72	32	8

are computed on the whole training set. The spatial DNN outputs correspond to the Cholesky decompositions $\mathbf{L}_{\mathbf{R}_j}(f)$ of the time-invariant source spatial covariance matrices. In order to obtain these time-invariant matrices, the final layer of spatial DNNs computes the average over all time frames on the inputs, which are akin to the Cholesky decompositions $\mathbf{L}_{\mathbf{R}_j}(f,n)$ of the time-varying source spatial covariance matrices. This averaging layer is used during the DNN training and thus, the DNN training targets are time-invariant. In our experiments, we consider two different averaging layers as described later in Section 5.4.5.2. Figure 5.1b depicts the general architecture of the spatial DNNs we used in this section.

5.4.4.2 Training algorithm and data

The DNN training algorithm mainly follows the description in Section 4.4.1.3, although there are a few differences. A minibatch consists of 128 randomly selected frequency bins from 16 randomly selected full utterances. Dropout with rate 0.2 is applied as in Section 5.4.3.2.

Let $\widetilde{\mathbf{R}}'_j(f)$ denote the unnormalized source spatial covariance matrix ground truth. How this ground truth is computed is explored in Section 5.5. In order to avoid numerical issues during the DNN training due to extreme (very small or very big) values, we normalize the source spatial covariance matrices before we compute the Cholesky decomposition of each matrix. The normalization is expressed as

$$\widetilde{\mathbf{R}}_j(f) = \frac{\sqrt{I}}{\frac{1}{J}\sum_j \left\| \widetilde{\mathbf{R}}'_j(f) \right\|_F} \widetilde{\mathbf{R}}'_j(f), \tag{5.3}$$

where $\sqrt{I}$ corresponds to the Frobenius norm of the identity matrix. By normalizing all source spatial covariance matrices with the same factor, the resulting Wiener filter is preserved. The Cholesky decomposition is then computed for each source spatial covariance matrix resulting in a lower triangular matrix. In order to obtain the DNN training targets, the diagonal and strictly lower triangular entries of this matrix are then concatenated as in the supervector construction for DNN inputs described in Section 5.4.4.1.

Similarly to spectral DNNs, spatial DNNs are trained on the real training set (`tr05_real`) and validated on the real development set (`dt05_real`). The simulated data in the CHiME dataset is barely reverberated and thus, the speech covariance matrices are close to rank-1 matrices. On the other hand, the real sets, which are our main point of interest in the evaluation, are more reverberated and thus, the speech covariance matrices are full-rank. Because of this mismatch, the simulated sets are avoided in the spatial DNN training.

All DNNs and the related techniques for all experiments presented in this section were implemented using Keras [Chollet et al., 2015] with Theano [Bergstra et al., 2010; Theano Dev Team, 2016] as its back-end.

5.4.5 Design choices for the DNN spatial models

We now describe the design choices for spatial DNNs we study in the experiments. We consider several choices for the DNN training cost function and the DNN architectures.

5.4.5.1 Cost functions

Let $\widetilde{\mathbf{R}}_j(f) = \mathbf{L}_{\widetilde{\mathbf{R}}_j}(f)\mathbf{L}_{\widetilde{\mathbf{R}}_j}(f)^*$ be the normalized source spatial covariance matrix ground truth from (5.3), where $\mathbf{L}_{\widetilde{\mathbf{R}}_j}(f)$ is its Cholesky decomposition from which the DNN training target is constructed. Also, let $\mathbf{R}_j(f) = \mathbf{L}_{\mathbf{R}_j}(f)\mathbf{L}_{\mathbf{R}_j}(f)^*$ be the estimated source spatial covariance matrix constructed from the DNN outputs $\mathbf{L}_{\mathbf{R}_j}(f)$.

For training the spatial DNNs, we consider three cost functions measuring the difference between $\widetilde{\mathbf{R}}_j(f)$ and $\mathbf{R}_j(f)$, as opposed to the difference between

$\mathbf{L}_{\widetilde{\mathbf{R}}_j}(f)$ and $\mathbf{L}_{\mathbf{R}_j}(f)$. Thus, these cost functions can be regarded as task-oriented cost functions.

The mean squared error (MSE) is expressed as

$$\mathcal{D}_{\text{RJ}} = \frac{1}{F}\sum_f \left\|\widetilde{\mathbf{R}}_j(f) - \mathbf{R}_j(f)\right\|_F^2, \tag{5.4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

The cosine distance [Herdin et al., 2005] is expressed as

$$\mathcal{D}_{\text{CD}} = 1 - \frac{1}{F}\sum_f \frac{\text{tr}\left(\widetilde{\mathbf{R}}_j(f)\mathbf{R}_j(f)^*\right)}{\left\|\widetilde{\mathbf{R}}_j(f)\right\|_F \left\|\mathbf{R}_j(f)\right\|_F}. \tag{5.5}$$

We also propose a **combined cost function** which fuses the two above cost functions as

$$\mathcal{D}_{\text{RC}} = \mathcal{D}_{\text{RJ}} + \kappa\mathcal{D}_{\text{CD}}, \tag{5.6}$$

where the fusion parameter is set to $\kappa = 10$. In order to decide the value of this parameter, we observed the validation errors after the first epoch when the MSE and the cosine distance are used alone for the training of $\text{DNN}_0^{\text{spat}}$ on the 6-channel task. The parameter κ is then simply determined so that the two validation errors have a roughly similar scale, which implies that both cost functions have a similar importance. We did not perform any optimization of this parameter κ.

5.4.5.2 Architectures and input variants

We consider architectures using one of two different averaging layers, i.e., a simple averaging layer or a weighted averaging layer. The latter is motivated by the weighted spatial parameter update in (5.2). Let $\mathbf{L}_{\mathbf{R}_j}(f,n)$ denote the inputs of the averaging layer. The architecture with a simple averaging layer ('SAvg') computes the outputs as

$$\mathbf{L}_{\mathbf{R}_j}(f) = \frac{1}{N}\sum_{n=1}^{N} \mathbf{L}_{\mathbf{R}_j}(f,n). \tag{5.7}$$

On the other hand, the architecture with a weighted averaging layer ('WAvg') computes the outputs as

$$\mathbf{L}_{\mathbf{R}_j}(f) = \frac{\sum_{n=1}^{N} \sqrt{v_j(f,n)}\mathbf{L}_{\mathbf{R}_j}(f,n)}{\sum_{n=1}^{N} \sqrt{v_j(f,n)}}. \tag{5.8}$$

Since these two averaging layers are parameterless, the total number of DNN parameters for both architectures 'SAvg' and 'WAvg' is the same.

Additionally, we consider two different input settings for $\text{DNN}_{1,1}^{\text{spat}}$, namely, a default setting and an extended setting. The default setting shown in Table 5.2 consists of $\mathbf{L}_{\widetilde{\mathbf{R}}_{\mathbf{c}_j}}(f,n)$, $\sqrt{\overline{z}_j(f,n)}$, and $\sqrt{v_j(f,n)}$ of all sources. The extended setting, denoted as '+Rx', includes $\mathbf{L}_{\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}}(f,n)$ and $\sqrt{z_x(f,n)}$ in addition to the default setting. We use this extended setting in the experiments on the 6-channel task. In this case, the size of the $\text{DNN}_{1,1}^{\text{spat}}$ input layer is $(I^2+2)J + (I^2+1) = 113$, instead of 76 as shown in Table 5.2.

5.5 Estimation of the oracle source spatial covariance matrices

In this section, we investigate how to compute the source spatial covariance matrices used for the spatial DNN training. Let $\widetilde{\mathbf{R}}_{\overline{\mathbf{c}}_j}(f,n)$ denote the source covariance matrix computed from the re-aligned target source spatial image signal $\overline{\mathbf{c}}_j(t)$. We assume that the true source spectrograms can be computed as

$$v_j^{\text{true}}(f,n) = \frac{1}{I}\text{tr}\left(\widetilde{\mathbf{R}}_{\overline{\mathbf{c}}_j}(f,n)\right). \tag{5.9}$$

This $v_j^{\text{true}}(f,n)$ is what we use as the training target of spectral DNNs indeed. Following the exact parameter estimation in (2.49), we may then assume that the true source spatial covariance matrices can be computed as

$$\mathbf{R}_j^{\text{true}}(f) = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{v_j^{\text{true}}(f,n)}\widetilde{\mathbf{R}}_{\overline{\mathbf{c}}_j}(f,n)\right). \tag{5.10}$$

Alternatively, given the true source spectrograms $v_j^{\text{true}}(f,n)$ and the mixture covariance matrix $\widetilde{\mathbf{R}}_{\overline{\mathbf{x}}}(f,n)$, we also can perform iterative EM spatial parameter updates which have shown to improve the performance well. In doing so, we initialize the source spatial covariance matrices as identity

Table 5.3: Speech recognition performance in terms of WER (%) using the different oracle settings for the 6-channel task. The ASR system uses the DNN+sMBR back-end trained on multi-condition enhanced data followed by 5-gram Kneser-Ney smoothing and RNN-LM rescoring. The evaluation was done on the real sets. Boldface numbers show the best performance for each dataset. The 95% confidence intervals for the two best WERs are $\pm$ 0.23% for the development set and $\pm$ 0.32% for the test set. Lower is better.

ID	Description	Dev	Test
(1)	$v_j(f,n) \leftarrow v_j^{\text{true}}(f,n),\ \mathbf{R}_j(f) \leftarrow \mathbf{R}_j^{\text{true}}(f)$	4.63	8.59
(2)	$v_j(f,n) \leftarrow v_j^{\text{true}}(f,n),\ \mathbf{R}_j(f) \leftarrow \mathbf{R}_j^{\text{conv}}(f)$	**3.87**	**6.21**

matrices and iteratively apply the weighted EM spatial parameter update in (5.2) until convergence. Instead of a fixed number of updates K, we consider the convergence criterion:

$$1 - \frac{1}{JF} \sum_{j,f} \frac{\operatorname{tr}\left(\mathbf{R}_j^k(f)\left(\mathbf{R}_j^{k-1}(f)\right)^*\right)}{\left\|\mathbf{R}_j^k(f)\right\|_F \left\|\mathbf{R}_j^{k-1}(f)\right\|_F} < 10^{-6}. \tag{5.11}$$

The update is stopped as soon as this criterion is achieved. Let $\mathbf{R}_j^{\text{conv}}(f)$ denote the resulting source spatial covariance matrices.

Table 5.3 compares the oracle speech recognition performance obtained using $v_j^{\text{true}}(f,n)$ and either $\mathbf{R}_j^{\text{true}}(f)$ or $\mathbf{R}_j^{\text{conv}}(f)$ for the 6-channel task. It shows that $\mathbf{R}_j^{\text{conv}}(f)$ provides statistically significant improvement compared to $\mathbf{R}_j^{\text{true}}(f)$. We attribute this to the fact that the source spatial image signals in the real sets are not accurate especially from a spatial point of view, more than a spectral point of view (see discussion in Section 3.3.1).

Based on the results shown in Table 5.3, we use $\mathbf{R}_j^{\text{conv}}(f)$ as the spatial covariance matrix ground truth for the following experiments. To be specific, it is used as the unnormalized spatial covariance matrix ground truth $\widetilde{\mathbf{R}}'_j(f)$ discussed in Section 5.4.4.2.

The following subsections present our study on the impact of spatial parameter estimation with DNNs, the impact of different spatial DNN architectures, the impact of different cost functions for the spatial DNN training, and the comparison to the generalized eigenvalue (GEV) beamformer with the blind analytic normalization (BAN) post-filter (see Section 2.5.1), which achieved state-of-the-art performance on the CHiME dataset [Heymann et al., 2017].

5.6 Spatial parameter estimation with DNNs

Table 5.4 shows the speech recognition performance using different framework settings for the 6-channel task. In systems **(1)** and **(2)**, we initialize the source spectrograms using $\text{DNN}_0^{\text{spec}}$. While in **(1)**, the source spatial covariance matrices are set to be identity matrices, they are initialized using $\text{DNN}_0^{\text{spat}}$ in **(2)**. Following these spectral and spatial parameter initializations using DNNs, we update the source spectrograms using $\text{DNN}_1^{\text{spec}}$ in **(3)** and then, update the source spatial covariance matrices using $\text{DNN}_{1,1}^{\text{spat}}$ in **(4)**. Both spatial DNNs are based on the weighted architecture 'WAvg' and trained with the combined cost function. We assess the choice of the architecture in Section 5.7 and the choice of the cost function in Section 5.8. We also consider additional iterative weighted EM spatial parameter updates until convergence in systems **(5)**, **(6)**, **(7)**, and **(8)**. The convergence criterion is the same as in (5.11). For the following discussions, let us focus on the performance on the real set.

The systems **(1)** and **(5)** are similar to those presented in the two previous chapters. Method **(1)** is equivalent to single-channel source separation. Method **(5)** is akin to the iterative EM spatial parameter updates in Sections 3.5 and 4.3. The difference is that in **(5)**, the number of spatial updates varies from one utterance to another. In this case, the average numbers of spatial updates on the development and the test sets are 44.8 and 40.8, respectively. These are more than double those used in Sections 3.5 and 4.3.

The spatial parameter initialization with DNN in **(2)** is shown to be effective. It achieves a WER of 8.45% on the test set which is worse than what **(5)** achieves, i.e., 8.07%. Based on the confidence intervals, this difference might appear as not statistically significant. However, a paired difference test by Matched Pairs Sentence-Segment Word Error[1] [Gillick & Cox, 1989] indicates that **(5)** is significantly better than **(2)** with $p < 0.001$. This paired test is more reliable than simple confidence intervals since it accounts for correlations between the outputs of different systems. In terms of computational cost, **(2)** is more favorable than **(5)** since to achieve a similar level of performance, **(5)** needs many iterations of spatial updates as mentioned above.

The spectral parameter update with $\text{DNN}_1^{\text{spec}}$ in **(3)** slightly improves the performance on the test set. However, the following spatial parameter update

[1]We used the implementation in the NIST Scoring Toolkit (SCTK). See `http://www1.icsi.berkeley.edu/Speech/docs/sctk-1.2/sctk.htm`.

Table 5.4: Speech recognition performance in terms of WER (%) using different settings on Algorithm 3 for the 6-channel task. The spatial DNNs are based on the weighted architecture 'WAvg' and trained with the combined cost function on $\mathbf{R}_j^{\text{conv}}(f)$. The ASR system uses the DNN+sMBR back-end trained on multi-condition noisy data followed by 5-gram Kneser-Ney smoothing and RNN-LM rescoring. The evaluation was done on the real sets. Boldface numbers show the best performance for each dataset. The 95% confidence intervals for the two best WERs are ± 0.25% for the development set and ± 0.34% for the test set. Lower is better.

Enhancement system		**Dev**	**Test**
ID	**Description**		
(1)	spectral initialization with $\text{DNN}_0^{\text{spec}}$	7.17	14.03
(2)	**(1)** + spatial initialization with $\text{DNN}_0^{\text{spat}}$	4.64	8.45
(3)	**(2)** + spectral update with $\text{DNN}_1^{\text{spec}}$	4.68	8.11
(4)	**(3)** + spatial update with $\text{DNN}_{1,1}^{\text{spat}}$	4.81	8.33
(5)	**(1)** + weighted EM spatial updates until convergence	4.79	8.07
(6)	**(2)** + weighted EM spatial updates until convergence	4.67	7.50
(7)	**(3)** + weighted EM spatial updates until convergence	**4.56**	**6.96**
(8)	**(4)** + weighted EM spatial updates until convergence	4.58	6.97

with $\text{DNN}_{1,1}^{\text{spat}}$ in **(4)** does not provide any performance improvement. Taking the confidence intervals into account, the performance differences between **(2)**, **(3)**, and **(4)** are not significant. According to the paired difference test, the difference between **(2)** and **(3)** is significantly different with $p < 0.001$.

The application of the iterative weighted EM spatial parameter updates on top of **(2)**, **(3)**, and **(4)** improves the performance as shown in **(6)**, **(7)**, and **(8)**. Based on the confidence intervals, **(6)**–**(8)** are statistically better than **(2)**–**(5)**, but the performance differences between **(6)**, **(7)**, and **(8)** are not significant. According to the paired difference test, **(7)** and **(8)** significantly differ from **(6)** with $p < 0.001$ and $p = 0.002$, respectively. Systems **(7)** and **(8)** achieve virtually the same performance. The average numbers of spatial updates in **(8)** on the development and the test sets are 50.3 and 49.1, respectively.

In addition, we compare the computation time of parameter estimation for several systems by running a single-threaded process for each utterance on Intel® Xeon® CPU E5-2630 v3 @ 2.40GHz. The measured computation time excludes the quadratic ERB extraction and the inverse quadratic ERB transformation (see Section 5.3). The average utterance length on the development and the test sets are 6.0 and 5.9 s, respectively. The average computation time on the development set are 2.2 s for **(2)**, 6.3 s for **(4)**, and 42.1 s for **(8)**. The average computation time on the test set are 2.1 s for **(2)**, 6.2 s for **(4)**, and 40.4 s for **(8)**.

Overall, the best performance is achieved by **(7)** with a WER of 6.96%. This is a relative 50% improvement w.r.t. **(1)** and a relative 14% improvement w.r.t. **(5)**.

In summary, the proposed spatial DNN is effective to initialize the source spatial covariance matrices. By employing spectral and spatial parameter initializations by DNNs, we achieve a similar performance to that of spectral parameter initialization by DNN followed by many iterations of spatial parameter updates. The following spectral and spatial parameter updates by DNNs do not significantly affect the performance. However, when the iterative weighted EM spatial parameter updates are applied afterwards, we can achieve a statistically significant performance improvement. Thus, both spectral and spatial parameter updates by DNNs are beneficial eventually.

5.7 Impact of different spatial DNN architectures

Table 5.5 shows the speech recognition performance using the different spatial DNN architectures described in Section 5.4.4.1 for the 6-channel task. Methods **(1)**, **(4)**, and **(8)** in Table 5.4 are shown again as systems **(1)**, **(2)**, and **(6)** in Table 5.5 to ease the observation. The systems mentioned in the following discussion refer to the ones in Table 5.5.

Let us first observe systems **(2)**, **(3)**, **(4)**, and **(5)** whose parameters are only estimated by DNNs, without the iterative weighted EM spatial parameter updates. The spatial DNNs in systems **(2)** and **(4)** follow the weighted averaging architecture 'WAvg', whereas those in **(3)** and **(5)** follow the simple averaging architecture 'SAvg'. While $\mathrm{DNN}_{1,1}^{\mathrm{spat}}$ in **(2)** uses the default input setting, that in **(4)** uses the extended setting '+Rx'. Likewise, $\mathrm{DNN}_{1,1}^{\mathrm{spat}}$ in **(3)** uses the default input setting and that in **(5)** uses the extended setting '+Rx'. Among these four systems, the best WER on the test set is 7.99%. Taking the confidence intervals into account, the performance differences between these four systems might appear as not statistically significant. Yet, according to the paired difference test, **(4)** is significantly better than **(2)** ($p = 0.013$), **(3)** ($p < 0.001$), and **(5)** ($p = 0.002$).

Table 5.5: Speech recognition performance in terms of WER (%) using the different spatial DNN architectures for the 6-channel task. The spatial DNNs are trained with the combined cost function on $\mathbf{R}_j^{\text{conv}}(f)$. The ASR system uses the DNN+sMBR back-end trained on multi-condition noisy data followed by 5-gram Kneser-Ney smoothing and RNN-LM rescoring. The evaluation was done on the real sets. Boldface numbers show the best performance for each dataset. The 95% confidence intervals for the two best WERs are $\pm$ 0.25% for the development set and $\pm$ 0.34% for the test set. Lower is better.

Enhancement system				
ID	**Description**	**DNN$^{\text{spat}}$ arch.**	**Dev**	**Test**
(1)	spectral initialization with DNN$_0^{\text{spec}}$	-	7.17	14.03
(2)	**(1)** + spatial initialization with DNN$_0^{\text{spat}}$ + spectral update with DNN$_1^{\text{spec}}$ + spatial update with DNN$_{1,1}^{\text{spat}}$	WAvg	4.81	8.33
(3)	**(1)** + spatial initialization with DNN$_0^{\text{spat}}$ + spectral update with DNN$_1^{\text{spec}}$ + spatial update with DNN$_{1,1}^{\text{spat}}$	SAvg	4.67	8.54
(4)	**(1)** + spatial initialization with DNN$_0^{\text{spat}}$ + spectral update with DNN$_1^{\text{spec}}$ + spatial update with DNN$_{1,1}^{\text{spat}}$	WAvg +Rx	4.63	7.99
(5)	**(1)** + spatial initialization with DNN$_0^{\text{spat}}$ + spectral update with DNN$_1^{\text{spec}}$ + spatial update with DNN$_{1,1}^{\text{spat}}$	SAvg +Rx	4.63	8.41
(6)	**(2)** + weighted EM spatial updates until convergence	WAvg	4.58	**6.97**
(7)	**(3)** + weighted EM spatial updates until convergence	SAvg	**4.47**	7.18
(8)	**(4)** + weighted EM spatial updates until convergence	WAvg +Rx	4.60	7.26
(9)	**(5)** + weighted EM spatial updates until convergence	SAvg +Rx	4.67	7.62

Let us now observe systems **(6)**, **(7)**, **(8)**, and **(9)** with the iterative weighted EM spatial parameter updates. Among these four systems, the best WER on the test set is 6.97%. Based on the confidence intervals, the performance differences between these four systems might again appear as not statistically significant. Additionally, **(6)**–**(8)** provide a statistically significant improvement compared to **(2)**–**(5)**. According to the paired difference test, there is no performance difference between **(6)** and **(7)**, but **(6)** is significantly better than **(8)** ($p = 0.036$) and **(9)** ($p < 0.001$).

Let us compare the simple averaging architecture ('SAvg' and 'SAvg+Rx') with the weighted averaging architecture ('WAvg' and 'WAvg+Rx') by observing the differences between **(2)** and **(3)**, between **(4)** and **(5)**, between **(6)** and **(7)**, also between **(8)** and **(9)**. In general, as already indicated in the discussions above, the weighted averaging architecture provides a slightly better performance than the simple averaging architecture. The performance differences are not statistically significant according to the confidence intervals. However, according to the paired difference test, **(4)** is significantly better than **(5)** ($p = 0.002$) and **(8)** is significantly better than **(9)** ($p = 0.007$). This indicates that explicit weighting is useful, but not crucial in this case. The parameters of the simple averaging architecture might adjust during the DNN training so as to provide a similar effect as explicit weighting. In other words, the DNNs may implicitly learn the weighting from the training data.

Let us now compare the two different input settings for $\text{DNN}_{1,1}^{\text{spat}}$ by observing the differences between **(2)** and **(4)**, between **(3)** and **(5)**, between **(6)** and **(8)**, also between **(7)** and **(9)**. The extended input setting provides a slightly better performance than the default input setting when the iterative EM spatial parameter updates are not applied. The performance differences are not statistically significant according to the confidence intervals, but according to the paired difference test, **(4)** is significantly better than **(2)** ($p = 0.013$). Surprisingly, the iterative EM spatial parameter updates perform worse when the extended input setting is used. The performance differences are again not statistically significant according to the confidence intervals, but according to the paired difference test, **(6)** is significantly better than **(8)** ($p = 0.036$) and **(7)** is significantly better than **(9)** ($p < 0.001$).

In summary, the statistical test based on the confidence intervals shows that the different spatial DNN architectures we considered here do not have a significant impact on the system performance. The paired difference test indicates that the weighted averaging architecture is better than the simple averaging architecture. It also suggests that the default input setting is better than the extended input setting when the iterative weighted EM spatial updates are applied and vice versa when these EM spatial updates are not applied. Disregarding the statistical test, the best performance is achieved by system **(6)** which employs spatial DNNs following the weighted averaging architecture 'WAvg' and using the default input setting for $\mathrm{DNN}^{\mathrm{spat}}_{1,1}$.

5.8 Impact of different spatial DNN cost functions

Table 5.6 shows the speech recognition performance using DNNs trained with the cost functions described in Section 5.4.5.1 for the 6-channel task. Methods **(1)**, **(4)**, and **(8)** in Table 5.4 are shown again as systems **(1)**, **(2)**, and **(5)** in Table 5.6 to ease the observation. The systems mentioned in the following discussion refer to the ones in Table 5.6.

Let us first observe systems **(2)**, **(3)**, and **(4)** whose parameters are only estimated by DNNs. The spatial DNNs in these systems are trained with the combined cost function, the MSE, and the cosine distance, respectively. Based on the confidence intervals, the performance of the cosine distance on the test set is not statistically different from that of the combined cost function. Also, the performance of the MSE is not statistically different from that of the cosine distance, but significantly worse than the performance of the combined cost function. According to the paired difference test, **(3)** is significantly worse than both **(2)** ($p < 0.001$) and **(4)** ($p = 0.003$). Let us consider two vectors as simpler analogies of the two matrices we consider in our cost computation. The cosine distance deals with the angle between two vectors. Thus, minimizing the cosine distance means finding the direction of the target vector. The cosine distance is scale-invariant and therefore, it does not care about the magnitude difference. On the other hand, the MSE deals with the difference of each vector component. Ideally, minimizing the MSE means finding the magnitude and the direction of the target vector. This also applies to our source spatial covariance matrices. Based on the results of these three systems, we observe that the combination of the MSE and the cosine distance performs better than the MSE or the cosine distance alone for finding the azimuth (direction) and the distance (magnitude) of the target source w.r.t. the microphones.

As in Section 5.6 and 5.7, based on the confidence intervals, the iterative EM spatial parameter updates in systems **(5)**–**(7)** provide a statistically

Table 5.6: Speech recognition performance in terms of WER (%) using DNNs trained with the different cost functions for the 6-channel task. The spatial DNNs are based on the weighted architecture 'WAvg' and trained on $\mathbf{R}_j^{\mathrm{conv}}(f)$. The ASR system uses the DNN+sMBR back-end trained on multi-condition noisy data followed by 5-gram Kneser-Ney smoothing and RNN-LM rescoring. The evaluation was done on the real sets. Boldface numbers show the best performance for each dataset. The 95% confidence intervals for the two best WERs are ± 0.25% for the development set and ± 0.34% for the test set. Lower is better.

Enhancement system			**Dev**	**Test**
ID	**Description**	**DNN$^{\mathrm{spat}}$ cost**		
(1)	spectral initialization with DNN$_0^{\mathrm{spec}}$	-	7.17	14.03
(2)	**(1)** + spatial initialization with DNN$_0^{\mathrm{spat}}$ + spectral update with DNN$_1^{\mathrm{spec}}$ + spatial update with DNN$_{1,1}^{\mathrm{spat}}$	$\mathcal{D}_{\mathrm{RC}}$	4.81	8.33
(3)	**(1)** + spatial initialization with DNN$_0^{\mathrm{spat}}$ + spectral update with DNN$_1^{\mathrm{spec}}$ + spatial update with DNN$_{1,1}^{\mathrm{spat}}$	$\mathcal{D}_{\mathrm{RJ}}$	4.75	9.23
(4)	**(1)** + spatial initialization with DNN$_0^{\mathrm{spat}}$ + spectral update with DNN$_1^{\mathrm{spec}}$ + spatial update with DNN$_{1,1}^{\mathrm{spat}}$	$\mathcal{D}_{\mathrm{CD}}$	4.70	8.65
(5)	**(2)** + weighted EM spatial updates until convergence	$\mathcal{D}_{\mathrm{RC}}$	4.58	**6.97**
(6)	**(3)** + weighted EM spatial updates until convergence	$\mathcal{D}_{\mathrm{RJ}}$	4.57	7.59
(7)	**(4)** + weighted EM spatial updates until convergence	$\mathcal{D}_{\mathrm{CD}}$	**4.46**	7.45

significant performance improvement on the test set compared to **(2)**–**(4)**. Also, the performance differences between **(5)**, **(6)**, and **(7)** are not statistically significant. However, according to the paired difference test, **(5)** is significantly better than both **(6)** ($p < 0.001$) and **(7)** ($p < 0.001$).

In summary, the combined cost function which integrates the MSE and the cosine distance provides the best performance compared to the MSE or the cosine distance alone. We do not investigate the impact of the factor κ in the combined cost function (see (5.6)). The performance may be improved further when κ is tuned.

5.9 Comparison with GEV-BAN beamforming

In this subsection, we compare our approach with the GEV beamformer with the BAN post-filter of Heymann et al. [2017], which achieved state-of-the-art performance on the CHiME dataset. Recall from Section 2.5.1 that the speech covariance matrix $\mathbf{R}_{\mathbf{c}_\mathrm{S}}(f)$ and the noise covariance matrix $\mathbf{R}_{\mathbf{c}_\mathrm{N}}(f)$ are required to derive the GEV beamformer. Then, recall from Section 2.5.3 that Heymann et al. [2017] compute these source covariance matrices using the speech mask $\widehat{m}_\mathrm{S}(f,n)$ and the noise mask $\widehat{m}_\mathrm{S}(f,n)$ estimated by DNNs. The DNNs are trained on binary masks, computed as the thresholded ratio between the speech and the noise magnitude spectra, as targets. Nonetheless, the DNNs estimate ratio masks instead of binary masks.

Here, we employ the source magnitude spectra estimated by $\mathrm{DNN}_0^{\mathrm{spec}}$ to derive the speech mask as

$$\widehat{m}_\mathrm{S}(f,n) = \frac{\sqrt{v_S(f,n)}}{\sqrt{v_S(f,n)} + \sqrt{v_N(f,n)}}, \tag{5.12}$$

and the noise mask as $\widehat{m}_\mathrm{N}(f,n) = 1 - \widehat{m}_\mathrm{S}(f,n)$. The resulting masks are arguably akin to the ratio masks estimated by DNNs by Heymann et al. [2017]. The speech and the noise masks are then used to compute the speech and the noise covariance matrices as in (2.53) and (2.54). Employing these source covariance matrices, the GEV beamformer is computed as (2.39) and the BAN post-filter in (2.40) is applied afterwards. Let us simply call this beamformer as the GEV-BAN beamformer.

Our implementation of the GEV-BAN beamformer differs from the one of Heymann et al. [2017] in two respects. First, instead of directly estimating the source masks by DNNs, we derive the source masks from the source magnitude spectra estimated by a DNN. Second, our speech and noise masks sum to 1, while this constraint was not applied by Heymann et al. [2017].

Table 5.7 shows the speech recognition performance for the different tasks compared to the GEV-BAN beamformer. Methods **(1)**, **(4)**, and **(8)** in Table 5.4 are shown again as systems **(1)**, **(3)**, and **(4)** in Table 5.7 to ease the observation. The systems mentioned in the following discussion refer to the ones in Table 5.7.

Table 5.7: Speech recognition performance in terms of WER (%) for the different tasks compared to the GEV-BAN beamformer. The spatial DNNs are based on the weighted architecture 'WAvg' and trained with the combined cost function on $\mathbf{R}_j^{\text{conv}}(f)$. The ASR system uses the DNN+sMBR back-end trained on multi-condition noisy data followed by 5-gram Kneser-Ney smoothing and RNN-LM rescoring. The evaluation was done on the real sets. Boldface numbers show the best performance for each pair of a task and a dataset. For the 6-channel task, the 95% confidence intervals for the two best WERs are ± 0.25% for the development set and ± 0.34% for the test set. For the 4-channel task, those are ± 0.26% and ± 0.37%. For the 2-channel task, those are ± 0.30% and ± 0.46%. Lower is better.

Enhancement system		**Dev**	**Test**
ID	**Description**		
	6-channel task		
(1)	spectral initialization with $\text{DNN}_0^{\text{spec}}$	7.17	14.03
(2)	**(1)** + GEV-BAN beamformer	5.37	8.15
(3)	**(1)** + spatial initialization with $\text{DNN}_0^{\text{spat}}$ + spectral update with $\text{DNN}_1^{\text{spec}}$ + spatial update with $\text{DNN}_{1,1}^{\text{spat}}$	4.81	8.33
(4)	**(3)** + weighted EM spatial updates until convergence	**4.58**	**6.97**
	4-channel task		
(5)	spectral initialization with $\text{DNN}_0^{\text{spec}}$	7.30	13.91
(6)	**(5)** + GEV-BAN beamformer	5.51	9.15
(7)	**(5)** + spatial initialization with $\text{DNN}_0^{\text{spat}}$ + spectral update with $\text{DNN}_1^{\text{spec}}$ + spatial update with $\text{DNN}_{1,1}^{\text{spat}}$	5.07	9.12
(8)	**(7)** + weighted EM spatial updates until convergence	**4.87**	**8.47**
	2-channel task		
(9)	spectral initialization with $\text{DNN}_0^{\text{spec}}$	8.55	17.39
(10)	**(9)** + GEV-BAN beamformer	6.94	13.62
(11)	**(9)** + spatial initialization with $\text{DNN}_0^{\text{spat}}$ + spectral update with $\text{DNN}_1^{\text{spec}}$ + spatial update with $\text{DNN}_{1,1}^{\text{spat}}$	7.19	14.78
(12)	**(11)** + weighted EM spatial updates until convergence	**6.63**	**13.48**

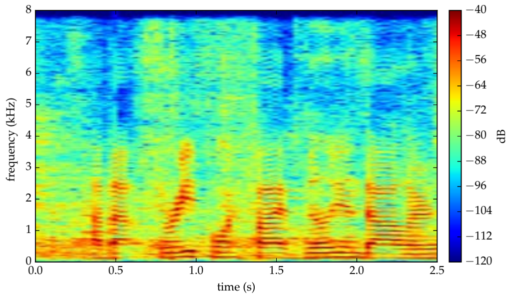
Systems **(1)**–**(4)** address the 6-channel task, **(5)**–**(8)** the 4-channel task, and **(9)**–**(12)** the 2-channel task. Systems **(1)**, **(5)**, and **(9)** correspond to single-channel speech enhancement. Systems **(2)**, **(6)**, and **(10)** employ the time-invariant GEV-BAN beamformer as described above. Systems **(3)**, **(4)**, **(7)**, **(8)**, **(11)**, and **(12)** are our proposed systems. The source parameters for the time-varying multichannel Wiener filtering in systems **(3)**, **(7)**, and **(11)** are estimated by DNNs only. In addition to these parameter estimations by DNNs, the spatial parameters in systems **(4)**, **(8)**, and **(12)** are re-estimated by the iterative EM spatial updates.

As we have observed several times, the single-channel speech enhancement systems **(1)**, **(5)**, and **(9)** are significantly outperformed by multichannel speech enhancement. This can be observed in all three different tasks.

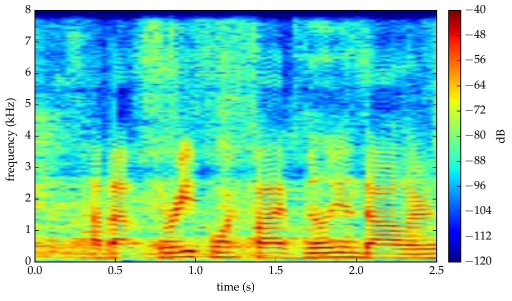
Let us first compare the performance of the GEV-BAN beamformer to that of our systems *without* the iterative EM spatial updates. Generally speaking, both approaches perform similarly for all tasks. The performance differences are mostly statistically insignificant, except on the development set for the 6-channel task, in which our system is significantly better, and on the test set for the 2-channel task, in which the GEV-BAN beamformer is significantly better.

Let us then compare the performance of the GEV-BAN beamformer to that of our systems *with* the iterative EM spatial updates. For the 6-channel task, our system significantly outperforms the GEV-BAN beamformer on both the development and the test sets. For the 4-channel task, our system also outperforms the GEV-BAN beamformer on the development set. According to the paired difference test, it is also significantly better ($p = 0.001$) on the test set. Finally, on the 2-channel task, our system performs better than the GEV-BAN beamformer on both the development and the test sets, but the performance differences are not statistically significant.

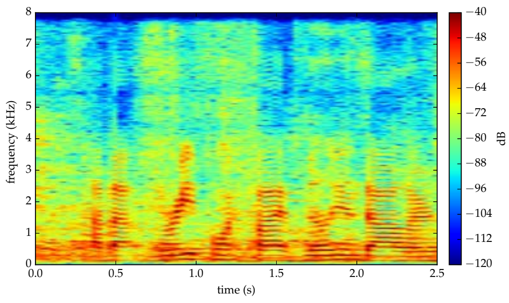
Figure 5.2 shows example spectrograms of the outputs of the GEV-BAN beamformer and our proposed system with the iterative EM spatial updates for each of the three different tasks. The utterance is the same as the one in Figures 3.10 and 4.3. Our system noticeably attenuates noise better than the GEV-BAN beamformer, especially for the 2-channel task. Additionally, in the particular example shown in Figure 5.2e, the GEV-BAN beamformer results in an unusually high energy at around 0.5 s for almost all frequencies, due to some discontinuity in the time-domain signal.



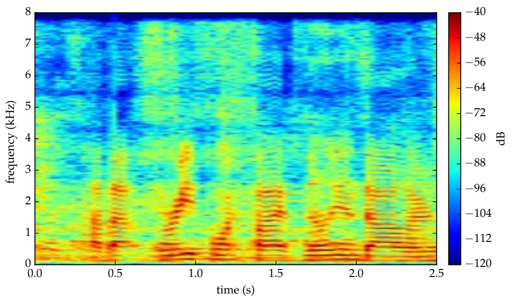
(a) 6-channel: GEV-BAN



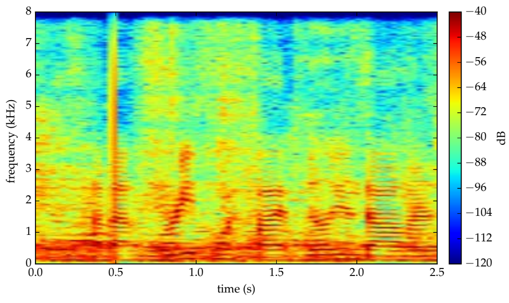
(b) 6-channel: proposed



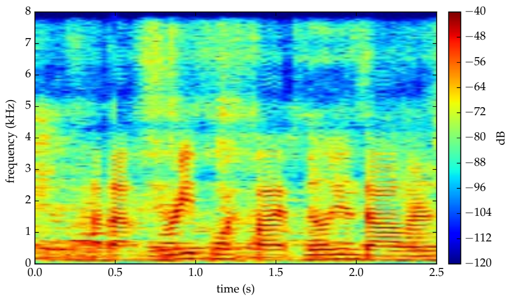
(c) 4-channel: GEV-BAN



(d) 4-channel: proposed



(e) 2-channel: GEV-BAN



(f) 2-channel: proposed

Figure 5.2: Example power spectrograms of the resulting single-channel enhanced speech for all three different tasks by the GEV-BAN beamformer and the proposed method with the iterative EM spatial updates. The utterance (`M05_440C0211_CAF`) is taken from the real test set (`et05_real`).

5.10 Summary

This chapter presents a weighted spatial parameter estimation formula, which is shown to outperform the exact EM formula, and a DNN based multichannel audio source separation framework. This framework employs DNNs for estimating both spectral and spatial parameters with possibly additional iterative EM spatial parameter updates. We show that DNNs are able to provide a good estimation of the spatial parameters, i.e., the source spatial covariance matrices. This is done by exploiting the Cholesky decomposition of the input and output covariance matrices. We show that a spatial DNN is effective for initializing the spatial parameters. We also consider an additional DNN for updating the spatial parameters. For these spatial DNNs, we experimented with different cost functions and found that the combination of the MSE and the cosine distance outperforms the MSE or the cosine distance alone. We also experimented with different architectures for spatial DNNs and found that the architecture with a weighted averaging layer is better than that with a simple averaging layer. In addition, the iterative EM spatial parameter updates are still useful to improve the source spatial covariance matrices and ultimately, the system performance.

This chapter also presents a performance comparison between our proposed systems and the GEV-BAN beamformer of Heymann et al. [2017], which achieved state-of-the-art performance for the task and the dataset we consider. On the test set, the performance of our proposed system *without* additional iterative EM spatial parameter is statistically similar to that of the GEV-BAN beamformer on the 6-channel and the 4-channel tasks, but significantly worse on the 2-channel task. The performance of our proposed system *with* the iterative EM spatial parameter updates is always better than that of the GEV-BAN beamformer on all three tasks and the performance differences are statistically significant on the 6-channel and the 4-channel tasks. Using these systems, we can provide relative 14%, 7%, and 1% decreases of the WERs w.r.t. the GEV-BAN beamformer on the 6-channel, the 4-channel, and the 2-channel tasks, respectively.

CHAPTER 6

Conclusions and perspectives

In this chapter, we summarize our study and discuss the potential extensions.

6.1 Conclusions

This thesis presented our study in addressing the problem of multichannel audio source separation by exploiting deep neural networks (DNNs). In doing so, we used the classical expectation-maximization (EM) based multichannel source separation framework [Duong et al., 2010a] as the basis. This framework employs a multichannel Gaussian model, in which the sources are characterized by their spectral parameters, i.e., the source power spectral densities (PSDs), and their spatial parameters, i.e., the source spatial covariance matrices. To put it simply, our study explores and optimizes the use of DNNs for modeling these source PSDs and source spatial covariance matrices. Employing these source spectral and spatial parameters, we derive a time-varying multichannel Wiener filter for the separation of each source. To the best of our knowledge, this is the first DNN based multichannel audio source separation framework ever published and the only one to date that employs a time-varying filter in the context of speech enhancement.

In Chapter 3, we described the first DNN based multichannel audio source separation framework we proposed in our study. In this framework, the spectral parameters are modeled by DNNs and the spatial parameters are estimated iteratively as in the classical EM based framework. We then assessed the impact of different design choices on the performance. These design choices notably include multiple spatial parameter updates after spectral parameter initialization and the use of multiple DNNs for estimating the spectral parameters at different iterations. Experiments showed that doing multiple spatial parameter updates before a spectral parameter update provides better performance than alternating one spatial and one spectral parameter updates, as in the classical EM framework. Experiments also showed that different DNNs could be used for the spectral parameter update of different iterations to improve the overall performance. Finally, we showed that the proposed DNN based framework outperforms the multichannel non-negative matrix factorization (NMF) based framework of Ozerov et al. [2012], which achieved state-of-the-art performance for the considered CHiME dataset [Barker et al., 2015; Vincent et al., 2017b] before the emergence of deep learning. In terms of signal-to-distortion ratio (SDR), the DNN based system provides an absolute 5.5 dB increase w.r.t. the NMF based system. In terms of word error rate (WER), the DNN based system provides a relative 24% decrease.

In Chapter 4, we presented the impact of different spectral DNN design choices on system performance. These design choices include different cost functions, different time-frequency representations, different DNN architectures, and different DNN training data sizes. The Kullback-Leibler (KL) divergence has shown to be the most reasonable choice because it performs well in terms of both source separation and speech recognition metrics. The probabilistically-motivated Itakura-Saito (IS) divergence performs well in terms of speech recognition, but falls behind the other considered cost functions in terms of source separation. Unfortunately, the newly proposed task-oriented SDR cost function fails to significantly improve the source separation performance compared to the simple mean squared error (MSE). Nonetheless, it should be mentioned that this is one of the first uses of task-oriented discriminative training in a multichannel scenario. We showed that time-frequency representations based on the perceptually-motivated equivalent rectangular bandwidth (ERB) scale are favorable. We also showed that bidirectional long short-term memory (LSTM) based recurrent neural networks (RNNs) perform better than feedforward neural networks (FNNs), especially when training involves a large amount of data. We then showed that data augmentation fails to overcome overfitting issues in vocals-accompaniment separation. Nevertheless, our vocals-accompaniment separation systems perform remarkably close to the DNN based system of Uhlich et al. [2017], which achieved state-of-the-art performance on the professionally-produced music recordings task of SiSEC 2016 [Liutkus et al., 2017]. Our systems exhibit artifacts which are detrimental to the overall distortion, but they are significantly better in removing the accompaniment from the vocals, which may be favorable for a singing-voice estimation task. In terms of signal-to-interference ratio (SIR) on the vocals, our systems provide up to 2 dB increase w.r.t. Uhlich's system.

In Chapter 5, we presented a weighted spatial parameter estimation formula, which is a generalization of the exact EM formula, and described the second DNN based multichannel audio source separation framework we proposed in our study. In this framework, both the spectral and the spatial parameters are initialized and updated by DNNs with possibly additional iterative EM spatial parameter updates. We showed that DNNs could provide a good estimation of the spatial parameters, i.e., the source spatial covariance matrices, by exploiting the Cholesky decomposition of the Hermitian positive-definitive covariance matrices. We experimented with different cost functions, different architectures, and different input settings for these spatial DNNs. We showed that the newly proposed cost function combining the MSE and the cosine distance performs well. We also showed that additional iterative EM spatial parameter updates are still useful to improve the source spatial covariance matrices. Finally, this chapter presents a performance comparison between our proposed systems and generalized eigenvalue (GEV) beamformer with blind analytic normalization (BAN) beamforming of Heymann et al. [2017], which achieved state-of-the-art performance on the CHiME dataset [Barker et al., 2015; Vincent et al., 2017b]. On the real test set, our system provides relative 14%, 7%, and 1% decreases of the WERs w.r.t. the GEV-BAN beamformer for the 6-channel, the 4-channel, and the 2-channel tasks, respectively. This difference is statistically significant for the 6-channel and the 4-channel tasks.

To sum up, we have proposed two DNN based multichannel audio source separation frameworks in this thesis. The classical iterative framework based on the multichannel Gaussian model is used as the basis of the two proposed frameworks. The first proposed framework exploits DNNs for modeling the spectral parameters of the multichannel Gaussian model, i.e., the source PSDs. We then extend this first framework by also exploiting DNNs for modeling the spatial parameters of the model, i.e., the source spatial covariance matrices, resulting in the second proposed framework. Despite some differences in the details, in general, the second framework accommodates the first framework. Thus, we may claim that the main contribution of this thesis is a unified framework for DNN based multichannel audio source separation based on the multichannel Gaussian model.

Parts of this thesis have been published in the following journal articles and conference papers.

- Journal articles

- **Nugraha, A. A.**, Liutkus, A., & Vincent, E. (2016). Multichannel audio source separation with deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(9), 1652–1664.
- Vincent, E., Watanabe, S., **Nugraha, A. A.**, Barker, J., & Marxer, R. (2016). An analysis of environment, microphone and data simulation mismatches in robust speech recognition. *Computer Speech & Language*, 46, 535-557.

- Conference papers

  - **Nugraha, A. A.**, Liutkus, A., & Vincent, E. (2016). Multichannel music separation with deep neural networks. In *Proceedings of European Signal Processing Conference (EUSIPCO)* (pp. 1748–1752). Budapest, Hungary.
  - Sivasankaran, S., **Nugraha, A. A.**, Vincent, E., Morales-Cordovilla, J. A., Dalmia, S., Illina, I., & Liutkus, A. (2015). Robust ASR using neural network based speech enhancement and feature simulation. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (pp. 482–489). Scottsdale, USA.

6.2 Perspectives

Building upon the study presented in this thesis, the following directions might be considered in the future.

Integrated training of spectral and spatial DNNs

For the experiments presented in Chapter 5, we trained spectral and spatial DNNs separately by minimizing different cost functions. We could then fine-tune both spectral and spatial DNNs in an integrated manner. The objective of fine-tuning could be minimizing the error of the multichannel Wiener filter or even further, maximizing the SDR as in the multichannel task-oriented cost function proposed in Section 4.2.2. This integrated training would resemble the deep unfolding approach [Le Roux et al., 2015; Wisdom et al., 2016].

Different datasets and/or tasks

It would be interesting to evaluate the frameworks presented in this thesis on other datasets than the CHiME dataset and the Demixing Secrets Dataset. Moreover, from our data augmentation experiments in Section 4.5.2.3, we concluded that the source separation performance is bounded by the dataset and deriving additional training data from this dataset is not effective to improve the performance. It would be valuable if we can verify this statement by doing the same experiments on a bigger dataset.

It would also be interesting if we can evaluate the proposed frameworks for some tasks other than speech enhancement for automatic speech recognition (ASR), vocals-accompaniment separation, and vocals-instruments separation, such as sound scene identification [Richard et al., 2017] and automatic melody transcription [Benetos et al., 2013]. Considering other tasks also opens the opportunity to explore and assess novel task-oriented cost functions for DNN training.

Separation of moving audio sources

Recall that the whole study presented in this thesis considers the multichannel Gaussian model in (2.5.2.1). In this model, each source is characterized by a time-varying PSD and *time-invariant* spatial covariance matrices, which implies that each source does not move during a given recording (but it can be at different positions in different recordings). When we consider the separation of moving audio sources, *time-varying* spatial covariance matrices have to be considered. Several studies, such as Duong et al. [2011] and Kounades-Bastian et al. [2016], have proposed iterative EM algorithms to estimate these time-varying spatial parameters. We could build upon these studies to explore the use of DNNs in the context of multichannel audio source separation for moving sources. From our experiments in Section 5.6, we may claim that we can substitute the iterative EM spatial parameter updates by a single $\mathrm{DNN}_0^{\mathrm{spat}}$ to obtain a similar performance. It would be important to investigate whether this also applies when the estimated spatial parameters are time-varying. If so, the DNN based systems would be more appealing than the iterative algorithm based system, e.g., in the development of commercial products where reducing the testing time is essential.

One of the important questions is how to determine the DNN training targets. To be reliable, the spatial parameters should be estimated from several input frames. The time-invariant spatial parameters in Section 5.5 are estimated from the whole utterance. For the time-varying spatial parameters, we might want to consider a sliding window or a growing window, that starts from the beginning to the current frame. When the latter is used, some weighting mechanism might be considered, e.g., to give more importance to the recent frames.

Real-time multichannel audio source separation

Following the above discussion about commercial product development, online processing [Togami, 2011; Simon & Vincent, 2012] is also an essential matter. For this purpose, time-varying spatial parameters should be used, even when the sources are not moving. Thus, we have to overcome the same above challenge in determining the DNN training targets. Additionally, we have to use part of unidirectional networks instead of the bidirectional networks as in this thesis.

Robust speech enhancement integrating noise reduction, dereverberation, and echo cancellation

Smart voice-controlled speakers, such as Amazon Echo[1], Apple Homepod[2], and Google Home[3], are getting popular recently. Since these devices might be placed in any room or environment, they should include a speech enhancement system that is robust to various environmental noises and room reverberations. As the name implies, these devices are equipped with one or more microphones and loudspeakers. The microphones will surely capture the sound emitted by the loudspeakers. Thus, these devices should also include a good echo canceler. Following this demand, research towards integrating noise reduction, dereverberation, and echo cancellation should be done. Studies toward this direction include Togami & Kawaguchi [2014] and Doire et al. [2017].

Separation of multiple audio sources with the same type

Throughout this study, we addressed the problem of separating audio sources with different types, i.e., speech and noise in a speech enhancement task; singing-voice and music accompaniment in a vocals-accompaniment separation task. A more challenging problem is the separation of audio

[1]See `https://www.amazon.com/oc/echo`.
[2]See `https://www.apple.com/homepod`.
[3]See `https://madeby.google.com/home`.

sources with the same type, e.g., separating multiple overlapping speakers. The main challenge comes from the fact that the spectral properties of different sources with the same type, e.g., human speech, are similar. This challenge might be alleviated by exploiting spatial properties since different sources might be spatially separated. This suggests that multichannel features would be useful for spectral DNNs and thus, they are should also be explored. Studies toward this direction include Heittola et al. [2013] and Hershey et al. [2016].

Audio source separation with distributed microphone arrays

Another challenging problem would be multichannel audio source separation with distributed microphone arrays. The main challenge is to combine the information from different arrays, since the synchronization between arrays could not be ensured and information exchange between arrays would be limited. Studies toward this direction include Markovich-Golan et al. [2015] and Tavakoli et al. [2016].

Bibliography

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, 9(1), 147–169.

Allen, J. (1977). Short term spectral analysis, synthesis, and modification by discrete Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 25(3), 235–238.

Anguera, X., Wooters, C., & Hernando, J. (2007). Acoustic beamforming for speaker diarization of meetings. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7), 2011–2022.

Araki, S., Hayashi, T., Delcroix, M., Fujimoto, M., Takeda, K., & Nakatani, T. (2015). Exploring multi-channel features for denoising-autoencoder-based speech enhancement. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 116–120). Brisbane, Australia.

Araki, S. & Nakatani, T. (2011). Hybrid approach for multichannel source separation combining time-frequency mask with multi-channel wiener filter. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 225–228). Prague, Czech Republic.

Atal, B. S. (1974). Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. *The Journal of the Acoustical Society of America*, 55(6), 1304–1312.

Badeau, R. & Virtanen, T. (2017). Nonnegative matrix factorization. In E. Vincent, T. Virtanen, & S. Gannot (Eds.), *Audio Source Separation and Speech Enhancement* chapter 8. Wiley.

Bahl, L., Brown, P., de Souza, P., & Mercer, R. (1986). Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 11 (pp. 49–52). Tokyo, Japan.

Barker, J. (2012). Missing-data techniques: Recognition with incomplete spectrograms. In T. Virtanen, R. Singh, & B. Raj (Eds.), *Techniques for Noise Robustness in Automatic Speech Recognition* chapter 14. Wiley.

Barker, J., Marxer, R., Vincent, E., & Watanabe, S. (2015). The third 'CHiME' speech separation and recognition challenge: Dataset, task and baselines. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (pp. 504–511). Scottsdale, USA.

Benesty, J., Chen, J., & Huang, Y. (2008). *Microphone Array Signal Processing*. Springer.

Benesty, J., Makino, S., & Chen, J. (2005). *Speech Enhancement*. Springer.

Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H., & Klapuri, A. (2013). Automatic music transcription: challenges and future directions. *Journal of Intelligent Information Systems*, 41(3), 407–434.

Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science* chapter 19, (pp. 437–478). Springer.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2006). Greedy layer-wise training of deep networks. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)* (pp. 153–160). Vancouver, Canada.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., & Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)* Austin, USA. Oral presentation.

Bertin, N., Févotte, C., & Badeau, R. (2009). A tempering approach for Itakura-Saito non-negative matrix factorization. with application to music transcription. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 1545–1548). Taipei, Taiwan.

Blandin, C., Ozerov, A., & Vincent, E. (2012). Multi-source TDOA estimation in reverberant audio using angular spectra and clustering. *Signal Processing*, 92(8), 1950–1960.

Bofill, P. & Zibulevsky, M. (2001). Underdetermined blind source separation using sparse representations. *Signal Processing*, 81(11), 2353–2362.

Boll, S. (1979). Suppression of acoustic noise in speech using spectral subtraction. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27(2), 113–120.

Bourlard, H. & Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4), 291–294.

Brandstein, M. & Ward, D., Eds. (2001). *Microphone Arrays: Signal Processing Techniques and Applications*. Springer.

Bregman, A. S. (1990). *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press.

Brown, G. J. & Cooke, M. (1994). Computational auditory scene analysis. *Computer Speech & Language*, 8(4), 297–336.

Brown, G. J. & Wang, D. (2005). Separation of speech by computational auditory scene analysis. In *Speech Enhancement* chapter 16, (pp. 371–402). Springer.

Cano, E., FitzGerald, D., & Brandenburg, K. (2016). Evaluation of quality of sound source separation algorithms: Human perception vs quantitative metrics. In *Proceedings of European Signal Processing Conference (EUSIPCO)* (pp. 1758–1762). Budapest, Hungary.

Cartwright, M., Pardo, B., Mysore, G. J., & Hoffman, M. (2016). Fast and easy crowdsourced perceptual audio evaluation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 619–623). Shanghai, China.

Chen, S. F. & Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (pp. 310–318). Santa Cruz, USA.

Chollet, F. et al. (2015). Keras. `https://github.com/fchollet/keras`.

Chung, J., Gülçehre, Ç., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv e-prints*. Presented in NIPS 2014 Deep Learning and Representation Learning Workshop.

Comon, P. (1994). Independent component analysis, a new concept? *Signal Processing*, 36(3), 287–314.

Cornelis, B., Moonen, M., & Wouters, J. (2011). Performance analysis of multichannel wiener filter-based noise reduction in hearing aids under second order statistics estimation errors. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(5), 1368–1381.

Crochiere, R. (1980). A weighted overlap-add method of short-time Fourier analysis/synthesis. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(1), 99–102.

Davis, S. & Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4), 357–366.

Delfarah, M. & Wang, D. (2017). Features for masking-based monaural speech separation in reverberant conditions. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(5), 1085–1094.

Deng, L. (2014). Deep learning: Methods and applications. *Foundations and Trends® in Signal Processing*, 7(3-4), 197–387.

Deng, L., Acero, A., Jiang, L., Droppo, J., & Huang, X. (2001). High-performance robust speech recognition using stereo training data. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1 (pp. 301–304). Salt Lake City, USA.

Deng, L. & O'Shaughnessy, D. (2003). *Speech Processing: A Dynamic and Optimization-Oriented Approach*. CRC Press.

Dieleman, S., Schlüter, J., Raffel, C., Olson, E., Sønderby, S. K., Nouri, D., et al. (2015). Lasagne: First release. `http://dx.doi.org/10.5281/zenodo.27878`.

Doclo, S., Kellermann, W., Makino, S., & Nordholm, S. E. (2015). Multichannel signal enhancement algorithms for assisted listening devices: Exploiting spatial diversity using multiple microphones. *IEEE Signal Processing Magazine*, 32(2), 18–30.

Doire, C. S. J., Brookes, M., Naylor, P. A., Hicks, C. M., Betts, D., Dmour, M. A., & Jensen, S. H. (2017). Single-channel online enhancement of speech corrupted by reverberation and noise. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(3), 572–587.

Droppo, J., Acero, A., & Deng, L. (2002). Uncertainty decoding with SPLICE for noise robust speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1 (pp. I–57–I–60). Orlando, USA.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 2121–2159.

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., & Garcia, R. (2000). Incorporating second-order functional knowledge for better option pricing. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)* (pp. 472–478). Denver, USA.

Duong, N. Q. K., Tachibana, H., Vincent, E., Ono, N., Gribonval, R., & Sagayama, S. (2011). Multichannel harmonic and percussive component separation by joint modeling of spatial and spectral continuity. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 205–208). Prague, Czech Republic.

Duong, N. Q. K., Vincent, E., & Gribonval, R. (2010a). Under-determined reverberant audio source separation using a full-rank spatial covariance model. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(7), 1830–1840.

Duong, N. Q. K., Vincent, E., & Gribonval, R. (2010b). Under-determined reverberant audio source separation using local observed covariance and auditory-motivated time-frequency representation. In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation* (pp. 73–80). Saint-Malo, France.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2), 179–211.

Emiya, V., Vincent, E., Harlander, N., & Hohmann, V. (2011). Subjective and objective quality assessment of audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(7), 2046–2057.

Ephraim, Y. & Malah, D. (1984). Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(6), 1109–1121.

Erdogan, H., Hershey, J. R., Watanabe, S., & Le Roux, J. (2015). Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 708–712). Brisbane, Australia.

Erdogan, H., Hershey, J. R., Watanabe, S., Mandel, M. I., & Roux, J. L. (2016). Improved MVDR beamforming using single-channel mask prediction networks. In *Proceedings of INTERSPEECH* (pp. 1981–1985). San Francisco, USA.

Ewert, S., Pardo, B., Mueller, M., & Plumbley, M. D. (2014). Score-informed source separation for musical audio recordings: An overview. *IEEE Signal Processing Magazine*, 31(3), 116–124.

Févotte, C., Bertin, N., & Durrieu, J.-L. (2009). Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis. *Neural Computation*, 21(3), 793–830.

Févotte, C. & Idier, J. (2011). Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation*, 23(9), 2421–2456.

Févotte, C. & Ozerov, A. (2010). Notes on nonnegative tensor factorization of the spectrogram for audio source separation: statistical insights and towards self-clustering of the spatial cues. In *Proceedings of International Symposium on Computer Music Modeling and Retrieval* (pp. 102–115). Málaga, Spain.

Fiscus, J. G. (1997). A post-processing system to yield reduced word error rates: Recognizer Output Voting Error Reduction (ROVER). In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (pp. 347–354). Santa Barbara, USA.

Frey, B. J., Kristjansson, T. T., Deng, L., & Acero, A. (2001). Algonquin - learning dynamic noise models from noisy speech for robust speech recognition. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)* (pp. 1165–1171). Vancouver, Canada.

Fujimoto, M. & Nakatani, T. (2016). Multi-pass feature enhancement based on generative-discriminative hybrid approach for noise robust speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 5750–5754). Shanghai, China.

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4), 193–202.

Gales, M. & Young, S. (2008). The application of hidden markov models in speech recognition. *Foundations and Trends® in Signal Processing*, 1(3), 195–304.

Gales, M. J. F. (1998). Maximum likelihood linear transformations for hmm-based speech recognition. *Computer Speech & Language*, 12(2), 75–98.

Gannot, S., Vincent, E., Markovich-Golan, S., & Ozerov, A. (2017). A consolidated perspective on multimicrophone speech enhancement and source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(4), 692–730.

Garofalo, J., Graff, D., Paul, D., & Pallett, D. (2007). CSR-I (WSJ0) Complete LDC93S6A. DVD. Philadelphia: Linguistic Data Consortium.

Gerkmann, T. & Vincent, E. (2017). Spectral masking and filtering. In E. Vincent, T. Virtanen, & S. Gannot (Eds.), *Audio Source Separation and Speech Enhancement* chapter 5. Wiley.

Gillick, L. & Cox, S. J. (1989). Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1 (pp. 532–535). Glasgow, UK.

Glasberg, B. R. & Moore, B. C. J. (1990). Derivation of auditory filter shapes from notched-noise data. *Hearing Research*, 47(1), 103–138.

Glorot, X. & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)* (pp. 249–256). Sardinia, Italy.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier networks. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 (pp. 315–323). Fort Lauderdale, FL, USA.

Goel, V. & Byrne, W. J. (2000). Minimum bayes-risk automatic speech recognition. *Computer Speech & Language*, 14(2), 115–135.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Gopinath, R. A. (1998). Maximum likelihood modeling with gaussian distributions for classification. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2 (pp. 661–664). Seattle, USA.

Grais, E. M., Roma, G., Simpson, A. J., & Plumbley, M. D. (2016). Combining mask estimates for single channel audio source separation using deep neural networks. In *Proceedings of INTERSPEECH* San Francisco, USA.

Grais, E. M., Roma, G., Simpson, A. J. R., & Plumbley, M. D. (2017). Two-stage single-channel audio source separation using deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(9), 1469–1479.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99), 1–11.

Griffin, D. & Lim, J. (1984). Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2), 236–243.

Gulcehre, C., Moczulski, M., Denil, M., & Bengio, Y. (2016). Noisy activation functions. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 3059–3068). New York, USA.

Gur, M. B. & Niezrecki, C. (2009). A source separation approach to enhancing marine mammal vocalizations. *The Journal of the Acoustical Society of America*, 126(6), 3062–3070.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv e-prints*. `http://arxiv.org/abs/1502.01852`.

Heittola, T., Mesaros, A., Virtanen, T., & Gabbouj, M. (2013). Supervised model training for overlapping sound events based on unsupervised source separation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 8677–8681). Vancouver, Canada.

Herdin, M., Czink, N., Ozcelik, H., & Bonek, E. (2005). Correlation matrix distance, a meaningful measure for evaluation of non-stationary MIMO channels. In *Proceedings of IEEE Vehicular Technology Conference*, volume 1 (pp. 136–140). Stockholm, Sweden.

Hermansky, H. (1990). Perceptual linear predictive (PLP) analysis of speech. *The Journal of the Acoustical Society of America*, 87(4), 1738–1752.

Hershey, J., Chen, Z., Le Roux, J., & Watanabe, S. (2016). Deep clustering: Discriminative embeddings for segmentation and separation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 31–35). Shanghai, China.

Heymann, J., Drude, L., & Haeb-Umbach, R. (2016). Neural network based spectral mask estimation for acoustic beamforming. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 196–200). Shanghai, China.

Heymann, J., Drude, L., & Haeb-Umbach, R. (2017). A generic neural acoustic beamforming architecture for robust multi-channel speech processing. *Computer Speech & Language*, 46, 374–385.

Himawan, I., Motlicek, P., Imseng, D., Potard, B., Kim, N., & Lee, J. (2015). Learning feature mapping using deep neural network bottleneck features for distant large vocabulary speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 4540–4544). Brisbane, Australia.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6), 82–97.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.

Hori, T., Chen, Z., Erdogan, H., Hershey, J. R., Roux, J. L., Mitra, V., & Watanabe, S. (2015). The MERL/SRI system for the 3rd CHiME challenge using beamforming, robust feature extraction, and advanced speech recognition. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (pp. 475–481). Scottsdale, USA.

Huang, P.-S., Kim, M., Hasegawa-Johnson, M., & Smaragdis, P. (2014a). Deep learning for monaural speech separation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 1562–1566). Florence, Italy.

Huang, P.-S., Kim, M., Hasegawa-Johnson, M., & Smaragdis, P. (2014b). Singing-voice separation from monaural recordings using deep recurrent neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference (ISMIR)* (pp. 477–482). Taipei, Taiwan.

Huang, P.-S., Kim, M., Hasegawa-Johnson, M., & Smaragdis, P. (2015). Joint optimization of masks and deep recurrent neural networks for monaural source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(12), 2136–2147.

Hummersone, C., Stokes, T., & Brookes, T. (2014). On the ideal ratio mask as the goal of computational auditory scene analysis. In G. R. Naik & W. Wang (Eds.), *Blind Source Separation: Advances in Theory, Algorithms and Applications* (pp. 349–368). Springer.

Hyvärinen, A. & Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural Networks*, 13(4), 411–430.

Ishii, T., Komiyama, H., Shinozaki, T., Horiuchi, Y., & Kuroiwa, S. (2013). Reverberant speech recognition based on denoising autoencoder. In *Proceedings of INTERSPEECH* (pp. 3512–3516). Lyon, France.

Itakura, F. & Saito, S. (1968). Analysis synthesis telephony based on the maximum likelihood method. In *Proceedings of International Congress on Acoustics* (pp. C–17 – C–20). Tokyo, Japan.

ITU (2015). Recommendation ITU-R BS.1534-3: Method for the subjective assessment of intermediate quality level of audio systems.

Izenman, A. J. (2008). Linear discriminant analysis. In *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning* chapter 8, (pp. 237–280). Springer.

Jaureguiberry, X., Vincent, E., & Richard, G. (2016). Fusion methods for speech enhancement and audio source separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(7), 1266–1279.

Jensen, J. & Hendriks, R. C. (2012). Spectral magnitude minimum mean-square error estimation using binary and continuous gain functions. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1), 92–102.

Jiang, Y., Wang, D., Liu, R., & Feng, Z. (2014). Binaural classification for reverberant speech segregation using deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(12), 2112–2121.

Jourjine, A., Rickard, S., & Yilmaz, O. (2000). Blind separation of disjoint orthogonal signals: demixing N sources from 2 mixtures. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5 (pp. 2985–2988). Istanbul, Turkey.

Jozefowicz, R., Zaremba, W., & Sutskever, I. (2015). An empirical exploration of recurrent network architectures. In *Proceedings of International Conference on Machine Learning (ICML)* (pp. 2342–2350). Lille, France.

Juang, B. H. (2016). Deep neural networks – a developmental perspective. *APSIPA Transactions on Signal and Information Processing*, 5.

Kang, T. G., Kwon, K., Shin, J. W., & Kim, N. S. (2015). NMF-based target source separation using deep neural network. *IEEE Signal Processing Letters*, 22(2), 229–233.

Karanasou, P., Wu, C., Gales, M., & Woodland, P. C. (2017). I-vectors and structured neural networks for rapid adaptation of acoustic models. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(4), 818–828.

Kim, T., Attias, H. T., Lee, S. Y., & Lee, T. W. (2007). Blind source separation exploiting higher-order frequency dependencies. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1), 70–79.

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv e-prints*. `http://arxiv.org/abs/1412.6980`.

Knapp, C. & Carter, G. (1976). The generalized correlation method for estimation of time delay. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 24(4), 320–327.

Kneser, R. & Ney, H. (1995). Improved backing-off for M-gram language modeling. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1 (pp. 181–184). Detroit, USA.

Koning, R., Madhu, N., & Wouters, J. (2015). Ideal time-frequency masking algorithms lead to different speech intelligibility and quality in normal-hearing and cochlear implant listeners. *IEEE Transactions on Biomedical Engineering*, 62(1), 331–341.

Kounades-Bastian, D., Girin, L., Alameda-Pineda, X., Gannot, S., & Horaud, R. (2016). A variational EM algorithm for the separation of time-varying convolutive audio mixtures. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(8), 1408–1423.

Kullback, S. & Leibler, R. A. (1951). On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1), 79–86.

Kumatani, K., McDonough, J., & Raj, B. (2012). Microphone array processing for distant speech recognition: From close-talking microphones to far-field sensors. *IEEE Signal Processing Magazine*, 29(6), 127–140.

Kuttruff, H. (2014). *Room Acoustics*. CRC Press, 5th edition.

Le Roux, J., Hershey, J. R., & Weninger, F. (2015). Deep NMF for speech separation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 66–70). Brisbane, Australia.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.

Lee, D. D. & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755), 788–791.

Lee, D. D. & Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *Proceedings of the Conference on Neural Information Processing Systems* (pp. 556–562). Denver, USA.

Lee, I., Kim, T., & Lee, T.-W. (2007). Fast fixed-point independent vector analysis algorithms for convolutive blind source separation. *Signal Processing*, 87(8), 1859–1871.

Lefèvre, A., Bach, F., & Févotte, C. (2011). Online algorithms for nonnegative matrix factorization with the Itakura-Saito divergence. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (pp. 313–316). New Paltz, NY, USA.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8).

Li, H., Nie, S., Zhang, X., & Zhang, H. (2016). Jointly optimizing activation coefficients of convolutive NMF using DNN for speech separation. In *Proceedings of INTERSPEECH* (pp. 550–554). San Francisco, USA.

Li, J., Deng, L., Gong, Y., & Haeb-Umbach, R. (2014). An overview of noise-robust automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4), 745–777.

Liao, H. & Gales, M. J. F. (2005). Joint uncertainty decoding for noise robust speech recognition. In *Proceedings of INTERSPEECH* (pp. 3129–3132). Lisbon, Portugal.

Lim, J. S. & Oppenheim, A. V. (1979). Enhancement and bandwidth compression of noisy speech. *Proceedings of the IEEE*, 67(12), 1586–1604.

Lippmann, R., Martin, E., & Paul, D. (1987). Multi-style training for robust isolated-word speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 12 (pp. 705–708). Dallas, USA.

Liu, D., Smaragdis, P., & Kim, M. (2014). Experiments on deep learning for speech denoising. In *Proceedings of INTERSPEECH* (pp. 2685–2688). Singapore.

Liutkus, A. & Badeau, R. (2015). Generalized wiener filtering with fractional power spectrograms. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 266–270). Brisbane, Australia.

Liutkus, A., Fitzgerald, D., & Badeau, R. (2015a). Cauchy nonnegative matrix factorization. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (pp. 1–5). New Paltz, USA.

Liutkus, A., Fitzgerald, D., & Rafii, Z. (2015b). Scalable audio separation with light kernel additive modelling. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 76–80). Brisbane, Australia.

Liutkus, A., Fitzgerald, D., Rafii, Z., Pardo, B., & Daudet, L. (2014). Kernel additive models for source separation. *IEEE Transactions on Signal Processing*, 62(16), 4298–4310.

Liutkus, A. & Leveau, P. (2010). Separation of music+effects sound track from several international versions of the same movie. In *Proceedings of Audio Engineering Society (AES) Convention* (pp. 1–15). San Francisco, USA.

Liutkus, A., Stöter, F.-R., Rafii, Z., Kitamura, D., Rivet, B., Ito, N., Ono, N., & Fontecave, J. (2017). The 2016 signal separation evaluation campaign. In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation* (pp. 323–332). Grenoble, France.

Loesch, B. & Yang, B. (2010). Adaptive segmentation and separation of determined convolutive mixtures under dynamic conditions. In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation* (pp. 41–48). Saint-Malo, France.

Loizou, P. C. (2007). *Speech Enhancement: Theory and Practice*. CRC Press.

Madhu, N., Spriet, A., Jansen, S., Koning, R., & Wouters, J. (2013). The potential for speech intelligibility improvement using the ideal binary mask and the ideal wiener filter in single channel noise reduction systems: Application to auditory prostheses. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(1), 63–72.

Makino, S., Sawada, H., & Lee, T.-W., Eds. (2007). *Blind Speech Separation*. Springer.

Markovich-Golan, S., Bertrand, A., Moonen, M., & Gannot, S. (2015). Optimal distributed minimum-variance beamforming approaches for speech enhancement in wireless acoustic sensor networks. *Signal Processing*, 107, 4–20.

Markovich-Golan, S., Kellermann, W., & Gannot, S. (2017). Spatial filtering. In E. Vincent, T. Virtanen, & S. Gannot (Eds.), *Audio Source Separation and Speech Enhancement* chapter 10. Wiley.

McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4), 115–133.

McGill, R., Tukey, J. W., & Larsen, W. A. (1978). Variations of box plots. *The American Statistician*, 32(1), 12–16.

Miao, Y., Zhang, H., & Metze, F. (2015). Speaker adaptive training of deep neural network acoustic models using i-vectors. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(11), 1938–1949.

Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. In *Proceedings of INTERSPEECH* (pp. 1045–1048). Makuhari, Japan.

Montana, D. J. & Davis, L. (1989). Training feedforward neural networks using genetic algorithms. In *Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI)*, volume 89 (pp. 762–767). Detroit, USA.

Mysore, G. J. & Smaragdis, P. (2011). A non-negative approach to semi-supervised separation of speech from noise with the use of temporal dynamics. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 17–20). Prague, Czech Republic.

Naik, G. R. & Wang, W., Eds. (2014). *Blind Source Separation: Advances in Theory, Algorithms and Applications*. Springer.

Nair, V. & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 807–814). Haifa, Israel.

Nakatani, T., Araki, S., Yoshioka, T., Delcroix, M., & Fujimoto, M. (2013). Dominance based integration of spatial and spectral features for speech enhancement. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(12), 2516–2531.

Narayanan, A. & Wang, D. (2013). Ideal ratio mask estimation using deep neural networks for robust speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 7092–7096). Vancouver, Canada.

Narayanan, A. & Wang, D. (2015). Improving robustness of deep neural network acoustic models via speech separation and joint adaptive training. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(1), 92–101.

Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate O (1/k2). *Soviet Mathematics Doklady*, 27(2), 372–376.

Nugraha, A. A., Liutkus, A., & Vincent, E. (2016a). Multichannel audio source separation with deep neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(9), 1652–1664.

Nugraha, A. A., Liutkus, A., & Vincent, E. (2016b). Multichannel music separation with deep neural networks. In *Proceedings of European Signal Processing Conference (EUSIPCO)* (pp. 1748–1752). Budapest, Hungary.

Nugraha, A. A., Yamamoto, K., & Nakagawa, S. (2014). Single-channel dereverberation by feature mapping using cascade neural networks for robust distant speaker identification and speech recognition. *EURASIP Journal on Audio, Speech, and Music Processing*, 2014(13), 1–31.

Osako, K., Mitsufuji, Y., Singh, R., & Raj, B. (2017). Supervised monaural source separation based on autoencoders. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 11–15). New Orleans, USA.

Ozerov, A., Févotte, C., & Charbit, M. (2009). Factorial scaled hidden markov model for polyphonic audio representation and source separation. In *Proceedings of IEEE*

Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA) (pp. 121–124). New Paltz, USA.

Ozerov, A., Vincent, E., & Bimbot, F. (2012). A general flexible framework for the handling of prior information in audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(4), 1118–1133.

Parra, L. & Spence, C. (2000). Convolutive blind separation of non-stationary sources. *IEEE Transactions on Speech and Audio Processing*, 8(3), 320–327.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 1310–1318). Atlanta, USA.

Poon, H. & Domingos, P. (2011). Sum-product networks: A new deep architecture. In *Proceedings of IEEE International Conference on Computer Vision (ICCV) Workshops* (pp. 689–690). Barcelona, Spain.

Potamitis, I. (2008). One-channel separation and recognition of mixtures of environmental sounds: The case of bird-song classification in composite soundscenes. In G. A. Tsihrintzis, M. Virvou, R. J. Howlett, & L. C. Jain (Eds.), *New Directions in Intelligent Interactive Multimedia* (pp. 595–604). Springer.

Povey, D., Kuo, H.-K. J., & Soltau, H. (2008). Fast speaker adaptive training for speech recognition. In *Proceedings of INTERSPEECH* (pp. 1245–1248). Brisbane, Australia.

Prechelt, L. (2012). Early stopping – but when? In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade* (pp. 53–67). Springer, 2nd edition.

Rabiner, L. & Juang, B.-H. (1993). *Fundamentals of Speech Recognition*. Prentice-Hall.

Rabiner, L. R. & Schafer, R. W. (2007). Introduction to digital speech processing. *Foundations and Trends® in Signal Processing*, 1(1–2), 1–194.

Raj, B. & Stern, R. M. (2005). Missing-feature approaches in speech recognition. *IEEE Signal Processing Magazine*, 22(5), 101–116.

Richard, G., Virtanen, T., Bello, J. P., Ono, N., & Glotin, H. (2017). Introduction to the special section on sound scene and event analysis. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(6), 1169–1171.

Rivet, B., Wang, W., Naqvi, S. M., & Chambers, J. A. (2014). Audiovisual speech source separation: An overview of key methodologies. *IEEE Signal Processing Magazine*, 31(3), 125–134.

Rojas, R. (1996). *Neural Networks: A Systematic Introduction*. Springer.

Roman, N., Wang, D., & Brown, G. J. (2003). Speech segregation based on sound localization. *The Journal of the Acoustical Society of America*, 114(4), 2236–2252.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65, 386–408.

Rosenfeld, R. (2000). Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE*, 88(8), 1270–1278.

Roweis, S. T. (2003). Factorial models and refiltering for speech separation and denoising. In *Proceedings of EUROSPEECH* (pp. 1009–1012). Geneva, Switzerland.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088), 533–536.

Sainath, T. N., Weiss, R. J., Wilson, K. W., Li, B., Narayanan, A., Variani, E., Bacchiani, M., Shafran, I., Senior, A., Chin, K., Misra, A., & Kim, C. (2017). Multichannel signal processing with deep neural networks for automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(5), 965–979.

Salaün, Y., Vincent, E., Bertin, N., Souviraà-Labastie, N., Jaureguiberry, X., Tran, D. T., & Bimbot, F. (2014). The Flexible Audio Source Separation Toolbox Version 2.0. Show & Tell of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP). `https://hal.inria.fr/hal-00957412`.

Samarakoon, L. & Sim, K. C. (2016). Factorized hidden layer adaptation for deep neural network based acoustic modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(12), 2241–2250.

Sawada, H., Kameoka, H., Araki, S., & Ueda, N. (2013). Multichannel extensions of non-negative matrix factorization with complex-valued data. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(5), 971–982.

Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv e-prints*. `http://arxiv.org/abs/1312.6120`.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, 85–117.

Schuster, M. & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11), 2673–2681.

Senior, M. (2011). *Mixing secrets for the small studio*. Focal Press.

Serizel, R., Moonen, M., Dijk, B. V., & Wouters, J. (2014). Low-rank approximation based multichannel wiener filter algorithms for noise reduction with application in cochlear implants. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(4), 785–799.

Simon, L. S. & Vincent, E. (2012). A general framework for online audio source separation. In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation* (pp. 397–404). Tel-Aviv, Israel.

Şimşekli, U., Liutkus, A., & Cemgil, A. T. (2015). Alpha-stable matrix factorization. *IEEE Signal Processing Letters*, 22(12), 2289–2293.

Sivasankaran, S., Nugraha, A. A., Vincent, E., Morales-Cordovilla, J. A., Dalmia, S., Illina, I., & Liutkus, A. (2015). Robust ASR using neural network based speech enhancement and feature simulation. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (pp. 482–489). Scottsdale, USA.

Sivasankaran, S., Vincent, E., & Illina, I. (2017). Discriminative importance weighting of augmented training data for acoustic model training. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 4885–4889). New Orleans, USA.

Smaragdis, P. (1998). Blind separation of convolved mixtures in the frequency domain. *Neurocomputing*, 22(1), 21–34.

Smaragdis, P. (2007). Convolutive speech bases and their application to supervised speech separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(1), 1–12.

Smaragdis, P. & Brown, J. C. (2003). Non-negative matrix factorization for polyphonic music transcription. In *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (pp. 177–180). New Paltz, USA.

Smaragdis, P., Fevotte, C., Mysore, G. J., Mohammadiha, N., & Hoffman, M. (2014). Static and dynamic source separation using nonnegative factorizations: A unified view. *IEEE Signal Processing Magazine*, 31(3), 66–75.

Smith, J. O. (2011). *Spectral Audio Signal Processing*. W3K Publishing.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1 (pp. 194–281). MIT Press.

Sprechmann, P., Bronstein, A. M., & Sapiro, G. (2015). Supervised non-negative matrix factorization for audio source separation. In R. Balan, M. Begué, J. J. Benedetto, W. Czaja, & K. A. Okoudjou (Eds.), *Excursions in Harmonic Analysis, Volume 4*, Applied and Numerical Harmonic Analysis (pp. 407–420). Springer.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15, 1929–1958.

Stern, R. M. & Morgan, N. (2012). Features based on auditory physiology and perception. In T. Virtanen, R. Singh, & B. Raj (Eds.), *Techniques for Noise Robustness in Automatic Speech Recognition* chapter 8. Wiley.

Stevens, S. S., Volkmann, J., & Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3), 185–190.

Sturmel, N., Liutkus, A., Pinel, J., Girin, L., Marchand, S., Richard, G., Badeau, R., & Daudet, L. (2012). Linear mixing models for active listening of music productions in realistic studio conditions. In *Proceedings of Audio Engineering Society (AES) Convention* (pp. 1–10). Budapest, Hungary.

Swietojanski, P., Li, J., & Renals, S. (2016). Learning hidden unit contributions for unsupervised acoustic model adaptation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(8), 1450–1463.

Tavakoli, V., Jensen, J., Christensen, M., & Benesty, J. (2016). A framework for speech enhancement with ad hoc microphone arrays. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(6), 1038–1051.

Theano Dev Team (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*. `http://arxiv.org/abs/1605.02688`.

Togami, M. (2011). Online speech source separation based on maximum likelihood of local gaussian modeling. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 213–216). Prague, Czech Republic.

Togami, M. & Kawaguchi, Y. (2014). Simultaneous optimization of acoustic echo reduction, speech dereverberation, and noise reduction against mutual interference. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(11), 1612–1623.

Tseng, H. W., Hong, M., & Luo, Z. Q. (2015). Combining sparse NMF with deep neural network: A new classification-based approach for speech enhancement. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 2145–2149). Brisbane, Australia.

Tu, Y., Du, J., Xu, Y., Dai, L., & Lee, C.-H. (2014). Speech separation based on improved deep neural networks with dual outputs of speech features for both target and interfering speakers. In *Proceedings of the International Symposium on Chinese Spoken Language Processing (ISCSLP)* (pp. 250–254). Singapore.

Uhlich, S., Giron, F., & Mitsufuji, Y. (2015). Deep neural network based instrument extraction from music. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 2135–2139). Brisbane, Australia.

Uhlich, S., Porcu, M., Giron, F., Enenkl, M., Kemp, T., Takahashi, N., & Mitsufuji, Y. (2017). Improving music source separation based on deep neural networks through data augmentation and network blending. In *Proceedings of IEEE International*

Conference on Acoustics, Speech, and Signal Processing (ICASSP) (pp. 261–265). New Orleans, USA.

Veen, B. D. V. & Buckley, K. M. (1988). Beamforming: a versatile approach to spatial filtering. *IEEE ASSP Magazine*, 5(2), 4–24.

Veselý, K., Ghoshal, A., Burget, L., & Povey, D. (2013). Sequence-discriminative training of deep neural networks. In *Proceedings of INTERSPEECH* (pp. 2345–2349). Lyon, France.

Viikki, O. & Laurila, K. (1997). Noise robust HMM-based speech recognition using segmental cepstral feature vector normalization. In *Proceedings of the Tutorial and Research Workshop on Robust Speech Recognisiton for Unknown Communication Channels* (pp. 107–110). Pont-à-Mousson, France.

Vincent, E. (2006). Musical source separation using time-frequency source priors. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(1), 91–98.

Vincent, E. (2010). An experimental evaluation of Wiener filter smoothing techniques applied to under-determined audio source separation. In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation* (pp. 157–164). Saint-Malo, France.

Vincent, E., Bertin, N., Gribonval, R., & Bimbot, F. (2014). From blind to guided audio source separation: How models and side information can improve the separation of sound. *IEEE Signal Processing Magazine*, 31(3), 107–115.

Vincent, E., Gribonval, R., & Févotte, C. (2006). Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(4), 1462–1469.

Vincent, E., Jafari, M. G., Abdallah, S. A., Plumbley, M. D., & Davies, M. E. (2011). Probabilistic modeling paradigms for audio source separation. In W. Wang (Ed.), *Machine Audition: Principles, Algorithms and Systems* chapter 7, (pp. 162–185). IGI Global.

Vincent, E., Sawada, H., Bofill, P., Makino, S., & Rosca, J. P. (2007). First stereo audio source separation evaluation campaign: Data, algorithms and results. In *Proceedings of the International Conference on Independent Component Analysis and Signal Separation* (pp. 552–559). London, UK.

Vincent, E., Virtanen, T., & Gannot, S., Eds. (2017a). *Audio Source Separation and Speech Enhancement*. Wiley.

Vincent, E., Watanabe, S., Nugraha, A. A., Barker, J., & Marxer, R. (2017b). An analysis of environment, microphone and data simulation mismatches in robust speech recognition. *Computer Speech & Language*, 46, 535–557.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 1096–1103). Helsinki, Finland.

Virtanen, T., Singh, R., & Raj, B., Eds. (2012). *Techniques for Noise Robustness in Automatic Speech Recognition*. Wiley.

Virtanen, T., Vincent, E., & Gannot, S. (2017). Time-frequency processing – spectral properties. In E. Vincent, T. Virtanen, & S. Gannot (Eds.), *Audio Source Separation and Speech Enhancement* chapter 2. Wiley.

Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., & Fergus, R. (2013). Regularization of neural networks using dropconnect. In *Proceedings of the International Conference on Machine Learning (ICML)* (pp. 1058–1066). Atlanta, USA.

Wang, D. (2005). On ideal binary mask as the computational goal of auditory scene analysis. In P. Divenyi (Ed.), *Speech Separation by Humans and Machines* (pp. 181–197). Springer.

Wang, D. (2008). Time-frequency masking for speech separation and its potential for hearing aid design. *Trends in Amplification*, 12(4), 332–353.

Wang, D. (2017). Deep learning reinvents the hearing aid. *IEEE Spectrum*, 54(3), 32–37.

Wang, D. & Brown, G. J., Eds. (2006). *Computational Auditory Scene Analysis: Principles, Algorithms, and Applications*. Wiley.

Wang, D. L. & Brown, G. J. (1999). Separation of speech from interfering sounds based on oscillatory correlation. *IEEE Transactions on Neural Networks*, 10(3), 684–697.

Wang, Y. & Wang, D. (2013). Towards scaling up classification-based speech separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 21(7), 1381–1390.

Wang, Y. & Wang, D. (2015). A deep neural network for time-domain signal reconstruction. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 4390–4394). Brisbane, Australia.

Wang, Z., Vincent, E., Serizel, R., & Yan, Y. (2017). Rank-1 constrained multichannel Wiener filter for speech recognition in noisy environments. *arXiv e-prints*. `http://arxiv.org/abs/1707.00201`.

Warsitz, E. & Haeb-Umbach, R. (2007). Blind acoustic beamforming based on generalized eigenvalue decomposition. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(5), 1529–1539.

Weninger, F., Du, J., Marchi, E., & Gao, T. (2017). Single-channel classification and clustering approaches. In E. Vincent, T. Virtanen, & S. Gannot (Eds.), *Audio Source Separation and Speech Enhancement* chapter 7. Wiley.

Weninger, F., Erdogan, H., Watanabe, S., Vincent, E., Le Roux, J., Hershey, J. R., & Schuller, B. (2015). Speech enhancement with LSTM recurrent neural networks and its application to noise-robust ASR. In *Proceedings of the International Conference on Latent Variable Analysis and Signal Separation* (pp. 91–99). Liberec, Czech Republic.

Weninger, F., Le Roux, J., Hershey, J. R., & Schuller, B. (2014). Discriminatively trained recurrent neural networks for single-channel speech separation. In *Proceedings of IEEE Global Conference on Signal and Information Processing (GlobalSIP)* (pp. 577–581). Atlanta, USA.

Werbos, P. J. (1988). Backpropagation: past and future. In *Proceedings of IEEE International Conference on Neural Networks (ICNN)* (pp. 343–353 vol.1). San Diego, USA.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10), 1550–1560.

Wilamowski, B. M. & Yu, H. (2010). Neural network learning without backpropagation. *IEEE Transactions on Neural Networks*, 21(11), 1793–1803.

Williamson, D. S., Wang, Y., & Wang, D. (2015). Estimating nonnegative matrix model activations with deep neural networks to increase perceptual speech quality. *The Journal of the Acoustical Society of America*, 138(3), 1399–1407.

Williamson, D. S., Wang, Y., & Wang, D. (2016). Complex ratio masking for monaural speech separation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(3), 483–492.

Wisdom, S., Hershey, J. R., Le Roux, J., & Watanabe, S. (2016). Deep unfolding for multichannel source separation. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 121–125). Shanghai, China.

Wöllmer, M., Zhang, Z., Weninger, F., Schuller, B., & Rigoll, G. (2013). Feature enhancement by bidirectional LSTM networks for conversational speech recognition in highly non-stationary noise. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 6822–6826). Vancouver, Canada.

Xiao, X., Watanabe, S., Erdogan, H., Lu, L., Hershey, J., Seltzer, M. L., Chen, G., Zhang, Y., Mandel, M., & Yu, D. (2016). Deep beamforming networks for multi-channel speech recognition. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 5745–5749). Shanghai, China.

Xu, Y., Du, J., Dai, L.-R., & Lee, C.-H. (2014). An experimental study on speech enhancement based on deep neural networks. *IEEE Signal Processing Letters*, 21(1), 65–68.

Yu, D. & Deng, L. (2011). Deep learning and its applications to signal and information processing. *IEEE Signal Processing Magazine*, 28(1), 145–154.

Yu, D. & Deng, L. (2015). *Automatic Speech Recognition: A Deep Learning Approach*. Springer.

Yu, H. & Wilamowski, B. M. (2011). Levenberg-Marquardt training. In B. M. Wilamowski & J. D. Irwin (Eds.), *Intelligent Systems* chapter 12. CRC Press.

Zaremba, W., Sutskever, I., & Vinyals, O. (2015). Recurrent neural network regularization. *arXiv e-prints*. `http://arxiv.org/abs/1409.2329`.

Zeiler, M. D. (2012). ADADELTA: An adaptive learning rate method. *arXiv e-prints*. `http://arxiv.org/abs/1212.5701`.

Zhang, X. L. & Wang, D. (2016). A deep ensemble learning method for monaural speech separation. *IEEE Transactions on Audio, Speech, and Language Processing*, 24(5), 967–977.

Zhang, Z., Cummins, N., & Schuller, B. (2017). Advanced data exploitation in speech analysis: An overview. *IEEE Signal Processing Magazine*, 34(4), 107–129.

Žmolíková, K., Karafiát, M., Veselý, K., Delcroix, M., Watanabe, S., Burget, L., & Černocký, J. (2016). Data selection by sequence summarizing neural network in mismatch condition training. In *Proceedings of INTERSPEECH* (pp. 2354–2358). San Francisco, USA.

Zweig, G. & Nguyen, P. (2010). SCARF: A segmental conditional random field toolkit for speech recognition. In *Proceedings of INTERSPEECH* (pp. 2858–2861). Makuhari, Japan.

Résumé: Dans cette thèse, nous traitons le problème de la séparation de sources audio multicanale par réseaux de neurones profonds (*deep neural networks*, DNNs). Notre approche se base sur le cadre classique de séparation par algorithme espérance-maximisation (EM) basé sur un modèle gaussien multicanal, dans lequel les sources sont caractérisées par leurs spectres de puissance à court terme et leurs matrices de covariance spatiales. Nous explorons et optimisons l'usage des DNNs pour estimer ces paramètres spectraux et spatiaux. À partir des paramètres estimés, nous calculons un filtre de Wiener multicanal variant dans le temps pour séparer chaque source. Nous étudions en détail l'impact de plusieurs choix de conception pour les DNNs spectraux et spatiaux. Nous considérons plusieurs fonctions de coût, représentations temps-fréquence, architectures, et tailles d'ensembles d'apprentissage. Ces fonctions de coût incluent en particulier une nouvelle fonction liée à la tâche pour les DNNs spectraux: le rapport signal-à-distorsion. Nous présentons aussi une formule d'estimation pondérée des paramètres spatiaux, qui généralise la formulation EM exacte. Sur une tâche de séparation de voix chantée, nos systèmes sont remarquablement proches de la méthode de l'état de l'art actuel et améliorent le rapport source-interférence de 2 dB. Sur une tâche de rehaussement de la parole, nos systèmes surpassent la formation de voies GEV-BAN de l'état de l'art de 14%, 7% et 1% relatifs en terme d'amélioration du taux d'erreur sur les mots sur des données à 6, 4 et 2 canaux respectivement.
Mot-clés: séparation de sources audio multicanale, modèle gaussien multicanal, réseaux de neurones profonds

---

Abstract: This thesis addresses the problem of multichannel audio source separation by exploiting deep neural networks (DNNs). We build upon the classical expectation-maximization (EM) based source separation framework employing a multichannel Gaussian model, in which the sources are characterized by their power spectral densities and their source spatial covariance matrices. We explore and optimize the use of DNNs for estimating these spectral and spatial parameters. Employing the estimated source parameters, we then derive a time-varying multichannel Wiener filter for the separation of each source. We extensively study the impact of various design choices for the spectral and spatial DNNs. We consider different cost functions, time-frequency representations, architectures, and training data sizes. Those cost functions notably include a newly proposed task-oriented signal-to-distortion ratio cost function for spectral DNNs. Furthermore, we present a weighted spatial parameter estimation formula, which generalizes the corresponding exact EM formulation. On a singing-voice separation task, our systems perform remarkably close to the current state-of-the-art method and provide up to 2 dB improvement of the source-to-interference ratio. On a speech enhancement task, our systems outperforms the state-of-the-art GEV-BAN beamformer by 14%, 7%, and 1% relative word error rate improvement on 6-channel, 4-channel, and 2-channel data, respectively.
Keywords: multichannel audio source separation, multichannel Gaussian model, deep neural networks