



TECHNISCHE UNIVERSITÄT ILMENAU

FAKULTÄT FÜR ELEKTROTECHNIK UND INFORMATIONSTECHNIK  
DER TECHNISCHE UNIVERSITÄT ILMENAU

---

# DEEP LEARNING-BASED MUSIC SOURCE SEPARATION

**Stylios Ioannis Mimitakis**

Dissertation zur Erlangung des akademischen Grades Doktor-Ingenieur  
(Dr.-Ing)

1. Gutachter: **Prof. Dr.-Ing. Gerald Schuller**
2. Gutachter: **Prof. Bryan Pardo**
3. Gutachter: **Dr. Antoine Liutkus**

Tag der Einreichung: 04.01.2021

Tag der wissenschaftlichen Aussprache: 10.12.2021

**DOI:** 10.22032/dbt.50702

**URN:** urn:nbn:de:gbv:ilm1-2021000383



# Erklärung

Ich versichere, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Bei der Auswahl und Auswertung folgenden Materials haben mir die nachstehend aufgeführten Personen in der jeweils beschriebenen Weise unentgeltlich geholfen:


1. Dr. Yorgos Zikos: Korrekturlesen und Korrigieren der (übersetzt) Zusammenfassung

Weitere Personen waren an der inhaltlich-materiellen Erstellung der vorliegenden Arbeit nicht beteiligt. Insbesondere habe ich hierfür nicht die entgeltliche Hilfe von Vermittlungs- bzw. Beratungsdiensten (Promotionsberater oder anderer Personen) in Anspruch genommen. Niemand hat von mir unmittelbar oder mittelbar geldwerte Leistungen für Arbeiten erhalten, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form einer Prüfungsbehörde vorgelegt.

Ich bin darauf hingewiesen worden, dass die Unrichtigkeit der vorstehenden Erklärung als Täuschungsversuch bewertet wird und gemäß § 7 Abs. 10 der Promotionsordnung den Abbruch des Promotionsverfahrens zur Folge hat.

Erfurt, 28.12.2020  
(Ort, Datum)



(Unterschrift)





# Acknowledgements

I would like to thank Dr. Jakob Abeßer, Hanna Lukashevich, and my advisor Prof. Gerald Schuller for giving me the chance to be a part of two research projects, namely the Machine Sensing Training Network (MacSeNet) and the Informed Sound Activity Detection in Music Recordings (ISAD) project. I also thank my lab-mates at the Fraunhofer Institute for Digital Media Technology and the Technical University of Ilmenau, Luca Cuccovillo, Dr. David S. Johnson, Sascha Grollmisch, Patrick Kramer, Dr. Anna Kruspe, Christon Nadar, Renato Profeta, and Michael Taenzer. Additionally, I would like to thank Dr. Helen Cooper, Prof. Mark Plumbley, and Prof. WenWu Wang for a refreshing stay at the Centre for Vision and Signal processing, University of Surrey. My sincere thanks to collaborators who I had the pleasure working with: Dr. Derry FitzGerald, Dr. Christof Weiß, Prof. Meinard Müller, Prof. Scott H. Hawley, Prof. Tuomas Virtanen, Prof. Yoshua Bengio, Dmitriy Serdyuk, Dr. João F. Santos, Dr. Zafar Rafii, Dr. Antoine Liutkus, and to friends: Dr. Scott Beveridge, Dr. Estefanía Cano, Christos Chatzichristos, Cristóvão Cruz, Harris Christopoulos, Heracles Chrysanthos, Constantinos Demetriou, Sifis Garifallakis, Paschalis Gesios, Arnab Ghosh, Jody Hatton, Dr. Anastasia Lavrenko, Mpampis Livieratos, Dr. Paul Magron, Volodymyr Miz, Max Morrison, Konstantinos Pitas, Dr. Cagdas Ulas, Dr. Zhongwei Xu, Dr. Alfredo Zermini.

I am incredibly thankful to Adobe Research and especially my mentors Dr. Nicholas J. Bryan and Prof. Paris Smaragdis for an invaluable experience. Furthermore, I am very grateful to the MacSeNet clique of Dr. Rodrigo Pena, Dr. Lucas Rencker, and Dr. Iwona Sobieraj for endless discussions, arguments, and their support. I would also like to express my deepest gratitude towards Dr. Konstantinos Drossos that has been a collaborator and an amazing and withstanding friend. The latter, cannot be stressed out enough. Last but not least, I would like to thank and express my respect towards my family: Panagiotis, Persephone, Theda, Yorgos, and Kriton, for their continuous support and encouragement.



# Abstract

This thesis addresses the problem of music source separation using deep learning methods. The deep learning-based separation of music sources is examined from three angles. These angles are: the signal processing, the neural architecture, and the signal representation. From the first angle, it is aimed to understand what deep learning models, using deep neural networks (DNNs), learn for the task of music source separation, and if there is an analogous signal processing operator that characterizes the functionality of these models. To do so, a novel algorithm is presented. The algorithm, referred to as the neural couplings algorithm (NCA), distills an optimized separation model consisting of non-linear operators into a single linear operator that is easy to interpret. Using the NCA, it is shown that DNNs learn data-driven filters for singing voice separation, that can be assessed using signal processing. Moreover, by enabling DNNs to learn how to predict filters for source separation, DNNs capture the structure of the target source and learn robust filters.

From the second angle, it is aimed to propose a neural network architecture that incorporates the aforementioned concept of filter prediction and optimization. For this purpose, the neural network architecture referred to as the Masker-and-Denoiser (MaD) is presented. The proposed architecture realizes the filtering operation using skip-filtering connections. Additionally, a few inference strategies and optimization objectives are proposed and discussed. The performance of MaD in music source separation is assessed by conducting a series of experiments that include both objective and subjective evaluation processes. Experimental results suggest that the MaD architecture, with some of the studied strategies, is applicable to realistic music recordings, and the MaD architecture has been considered one of the state-of-the-art approaches in the Signal Separation and Evaluation Campaign (SiSEC) 2018.

Finally, the focus of the third angle is to employ DNNs for learning signal representations that are helpful for separating music sources. To that end, a new method is proposed using a novel re-parameterization scheme and a combination of optimization objectives. The re-parameterization is based on sinusoidal functions that promote interpretable DNN representations. Results from the conducted experimental procedure suggest that the proposed method can be efficiently employed in learning interpretable representations, where the filtering process can still be applied to separate music sources. Furthermore, the usage of optimal transport (OT) distances as optimization objectives is useful for computing additive and distinctly structured signal representations for various types of music sources.



# Zusammenfassung

Diese Dissertation befasst sich mit dem Problem der Trennung von Musikquellen durch den Einsatz von deep learning Methoden. Die auf deep learning basierende Trennung von Musikquellen wird unter drei Gesichtspunkten untersucht. Diese Perspektiven sind: die Signalverarbeitung, die neuronale Architektur und die Signaldarstellung. Aus der ersten Perspektive, soll verstanden werden, welche deep learning Modelle, die auf DNNs basieren, für die Aufgabe der Musikquellentrennung lernen, und ob es einen analogen Signalverarbeitungsoperator gibt, der die Funktionalität dieser Modelle charakterisiert. Zu diesem Zweck wird ein neuartiger Algorithmus vorgestellt. Der Algorithmus wird als NCA bezeichnet und destilliert ein optimiertes Trennungsmodell, das aus nicht-linearen Operatoren besteht, in einen einzigen linearen Operator, der leicht zu interpretieren ist.

Aus der zweiten Perspektive, soll eine neuronale Netzarchitektur vorgeschlagen werden, die das zuvor erwähnte Konzept der Filterberechnung und -optimierung beinhaltet. Zu diesem Zweck wird die als Masker and Denoiser (MaD) bezeichnete neuronale Netzarchitektur vorgestellt. Die vorgeschlagene Architektur realisiert die Filteroperation unter Verwendung skip-filtering connections Verbindungen. Zusätzlich werden einige Inferenzstrategien und Optimierungsziele vorgeschlagen und diskutiert. Die Leistungsfähigkeit von MaD bei der Musikquellentrennung wird durch eine Reihe von Experimenten bewertet, die sowohl objektive als auch subjektive Bewertungsverfahren umfassen.

Abschließend, der Schwerpunkt der dritten Perspektive liegt auf dem Einsatz von DNNs zum Erlernen von solchen Signaldarstellungen, für die Trennung von Musikquellen hilfreich sind. Zu diesem Zweck wird eine neue Methode vorgeschlagen. Die vorgeschlagene Methode verwendet ein neuartiges Umparametrisierungsschema und eine Kombination von Optimierungszielen. Die Umparametrisierung basiert sich auf sinusförmigen Funktionen, die interpretierbare DNN-Darstellungen fördern. Der durchgeführten Experimente deuten an, daß die vorgeschlagene Methode beim Erlernen interpretierbarer Darstellungen effizient eingesetzt werden kann, wobei der Filterprozess noch auf separate Musikquellen angewendet werden kann. Die Ergebnisse der durchgeführten Experimente deuten an, daß die vorgeschlagene Methode beim Erlernen interpretierbarer Darstellungen effizient eingesetzt werden kann, wobei der Filterprozess noch auf separate Musikquellen angewendet werden kann. Darüber hinaus der Einsatz von optimal transport (OT) Entfernungen als Optimierungsziele sind für die Berechnung additiver und klar strukturierter Signaldarstellungen.



# Acronyms

**ADC** analog-to-digital converter.

**ANN** artificial neural network.

**BM** binary mask.

**BPTT** back-propagation through time.

**CNN** convolutional neural network.

**DAC** digital-to-analog converter.

**DAE** denoising autoencoder.

**DCT** discrete cosine transform.

**DFT** discrete Fourier transform.

**DNN** deep neural network.

**EDJM** extended data Jacobian matrix.

**FNN** feed-forward neural network.

**GPU** graphics processing unit.

**GRU** gated recurrent unit.

**GUI** graphical user interface.

**HPSS** harmonic-percussive source separation.

**IAM** ideal amplitude mask.

**IS** Itakura-Saito.

**ISTFT** inverse short-time Fourier transform.

**ITU** International Telecommunication Union.

**KL** Kullback-Leibler.

**LSTM** long short-term memory.

**MaD** Masker-and-Denoiser.

**MDCT** modified discrete cosine transform.

**MSE** mean squared error.

**MSS-DAE** music source separation denoising autoencoder.

**MUSHRA** multiple stimulus with hidden reference and anchors.

**NCA** neural couplings algorithm.

**OT** optimal transport.

**PCA** principal component analysis.

**PQMF** pseudo-quadrature mirror filter-banks.

**PSD** power spectral density.

**ReLU** rectified linear unit.

**RNN** recurrent neural network.

**SAR** signal-to-artifacts ratio.

**SDR** signal-to-distortion ratio.

**SIR** signal-to-interference ratio.

**SI-SDR** scale-invariant signal-to-distortion ratio.

**SiSEC** signal separation and evaluation campaign.

**SM** soft-masking.

**SNR** signal-to-noise ratio.

**STFT** short-time Fourier transform.



**SVD** singular value decomposition.

**TOD-R** trace-to-off-diagonal-ratio.

**TwinNet** Twin Network.

**W-DO** (windowed) disjoint-orthogonality.

**WSS** wide-sense stationary.



# Glossary

**artificial neural network** is an information processing system that is inspired by biological neurons.

**back-propagation** is a learning algorithm that is very popular in deep learning research, and is based on gradient descent and the chain rule of differentiation.

**deep learning** is one approach to artificial intelligence, comprising hierarchically multiple (commonly greater than two) information processing systems, that gather knowledge and learn complicated concepts related to a particular numerical computation problem [1].

**eigenvector** or the characteristic vector of a linear transformation, is the non-zero vector that changes only by a scalar factor when that linear transformation is applied to that vector.

**intrinsic dimension (of data)** is the minimum dimensionality needed to accurately represent the data.

**learning algorithm** is a computer program that is designed to increase the defined performance measure, over a class of defined tasks, as the observations from data (experience) are also increased [2].

**linear program** is a method to achieve the best plausible outcome, given an objective that is mathematically expressed, with the method's requirements being described by linear relationships.

**model** is the description of an observed system, that processes information and yields an outcome, using mathematical concepts.

**NP-hard** refers to a class of computational problems, such as decision problems, that are at least as hard as the non-deterministic polynomial (time) (NP) problems that can be verified, not solved, in polynomial time.

**optimal transport** is the problem of allocating (transporting) resources or information from one (geometrical) domain to another.

**simplex** is the generalization of a geometric object, such as a polytope, to arbitrary dimensions.

**skip connections** is a simple and effective operation for deep learning approaches, that allows the information from previous layers to be leaked (skipped) to the proceeding ones.

**skip-filtering connections** is a type of skip connections that allows the information from previous layers to be masked by the output of proceeding layers. The filtering term is based on the analogy of time-frequency masking and filtering in audio, music, and speech signal processing.

**supervised learning** is a type of learning algorithm that requires output targets or labels for each sample in a data-set.

**Toeplitz-matrix** is an algebraic matrix that contains main and secondary diagonals, whose entries are constants.

**unsupervised learning** is a type of learning algorithm that learns useful features, structures, and the structure of a given data-set, without explicit usage of output targets or labels.

**up-mixing (audio/music)** is the process of computing multiple auditory signal channels, that have been previously mixed together by the process of audio/music down-mixing.

# Contents

<b>Acknowledgements</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>Zusammenfassung</b>	<b>v</b>
<b>Acronyms</b>	<b>vii</b>
<b>Glossary</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Thesis Structure . . . . .	2
1.2 Scope of the Thesis . . . . .	2
1.3 Challenges & Thesis Contributions . . . . .	5
1.4 Associated Publications . . . . .	6
1.5 Notation . . . . .	7
<b>2 Fundamentals of Deep Learning-Based Music Source Separation</b>	<b>11</b>
2.1 Introduction . . . . .	11
2.2 From Signal Models to Neural Networks . . . . .	12
2.2.1 Signal Modelling Basics . . . . .	12
2.2.2 Signal Representations . . . . .	15
2.2.3 Audio Signal Transforms . . . . .	18
2.2.4 Artificial Neural Networks . . . . .	22
2.3 Mixing Models & Separation by Masking . . . . .	30
2.3.1 Instantaneous Mixing Model . . . . .	30
2.3.2 Convolutional Mixing Model . . . . .	31
2.3.3 Relaxed Mixing Model . . . . .	32
2.3.4 Music Source Separation via Masking . . . . .	33
2.4 Source Separation Performance Evaluation . . . . .	35
2.4.1 Objective Assessment . . . . .	36
2.4.2 Subjective Assessment . . . . .	37

<b>3</b>	<b>Neural Network-Based Music Source Separation</b>	<b>39</b>
3.1	Introduction . . . . .	39
3.2	Denoising Autoencoders . . . . .	42
3.2.1	Background . . . . .	42
3.2.2	Connection to Music Source Separation & Related Work . . . . .	44
3.2.3	Implementation of the Models . . . . .	45
3.3	Computing Mapping Functions . . . . .	46
3.3.1	Motivation & Related Work . . . . .	46
3.3.2	The Neural Couplings Algorithm . . . . .	47
3.4	Experimental Procedure . . . . .	53
3.4.1	Training & Assessing Separation Models . . . . .	53
3.4.2	Computing & Assessing the Neural Couplings . . . . .	54
3.4.3	Results & Discussion . . . . .	55
3.5	Summary . . . . .	62
<b>4</b>	<b>Masker-and-Denoiser Architecture</b>	<b>65</b>
4.1	Introduction . . . . .	65
4.2	Related Work . . . . .	66
4.2.1	Spectral Approximation Approaches . . . . .	66
4.2.2	Mask Prediction Approaches . . . . .	68
4.2.3	Skip-filtering connection Approaches . . . . .	68
4.3	Proposed Architecture . . . . .	70
4.3.1	Overview of the Architecture . . . . .	70
4.3.2	Signal Analysis & Input Pre-processing . . . . .	71
4.3.3	The Masker . . . . .	72
4.3.4	The Recurrent Inference Algorithm . . . . .	75
4.3.5	The Denoiser . . . . .	76
4.3.6	Output Processing & Synthesis . . . . .	76
4.4	Training Objectives . . . . .	77
4.4.1	Twin Network . . . . .	79
4.5	Experimental Procedure . . . . .	81
4.5.1	Training . . . . .	81
4.5.2	Evaluation . . . . .	82
4.6	Results & Discussion . . . . .	84
4.6.1	Singing Voice & Accompaniment Source Separation . . . . .	84
4.6.2	Harmonic & Percussive Separation . . . . .	90
4.7	Summary . . . . .	91
<b>5</b>	<b>Learning Representations for Separation</b>	<b>93</b>
5.1	Introduction . . . . .	93
5.2	Related Work . . . . .	95
5.3	Proposed Method . . . . .	96
5.3.1	The Encoder . . . . .	97
5.3.2	The Decoder . . . . .	100
5.4	Experimental Procedure . . . . .	102
5.4.1	Data-set . . . . .	102

5.4.2	Initialization & Hyper-parameter Selection . . . . .	102
5.4.3	Training . . . . .	103
5.4.4	Evaluation . . . . .	104
5.4.5	Assessing Design Choices . . . . .	105
5.5	Results & Discussion . . . . .	106
5.5.1	Results from Design Choices Evaluation . . . . .	106
5.5.2	Representation Learning Results . . . . .	107
5.5.3	Sinkhorn Distances Results . . . . .	113
5.6	Summary . . . . .	115
<b>6</b>	<b>Conclusions</b>	<b>117</b>
6.1	Thesis Summary . . . . .	117
6.2	Outlook . . . . .	119
	<b>Appendix</b>	<b>121</b>
<b>A</b>	<b>List of Used Sound Examples</b>	<b>121</b>
<b>B</b>	<b>Adam Algorithm</b>	<b>122</b>
<b>C</b>	<b>W-Disjointness Orthogonality</b>	<b>123</b>
<b>D</b>	<b>Supplementary Results: Masker-and-Denoiser Architecture</b>	<b>124</b>
<b>E</b>	<b>Supplementary Results: Learning Representations for Separation</b>	<b>125</b>
	<b>List of Figures</b>	<b>127</b>
	<b>List of Tables</b>	<b>130</b>
	<b>Bibliography</b>	<b>131</b>





# Chapter 1

## Introduction

### Preface

This thesis consists of research work from the author as an early stage researcher and a Ph.D. student at the Fraunhofer Institute for Digital Media Technology and at the Technical University of Ilmenau, respectively. The time period of the conducted research spans from August 2015 to August 2020. The research works presented in this thesis have been supported by the two following projects:

1. Machine Sensing Training Network (MacSeNet) (2015-2018): An Innovative Training Network (ITN) with the aim of training a new generation of creative, entrepreneurial, and innovative early stage researchers (ESRs) in the research area of measurement and estimation of signals using knowledge or data about the underlying structure.
2. Informed Sound Activity Detection in Music Recordings (ISAD) (2018-2020): A project supported by the German Research Foundation, with the goal to explore fundamental techniques and computational tools for detecting sound sources or characteristic sound events that are present in a given music recording.

Throughout this thesis the followed citation style includes in-text citations that are numbered in square brackets, for example [1]. Each bracket refers to the full citation that is listed in the Bibliography. The Bibliography is sorted numerically, and not alphabetically. The author's scientific publications are denoted by square brackets followed by an alphanumerical indication, for example [A22], where "A" stands for authorship. The above mentioned projects have enabled the author to work on research areas beyond the content described in the following chapters and the corresponding publications are also enlisted in the Bibliography.

## 1.1 Thesis Structure

This thesis is organized into six chapters:

The rest of Chapter 1 contains information about the structure of the thesis followed by a general introduction to the problem of music source separation. Furthermore, this chapter highlights the motivation, scope, challenges, and the contributions of this thesis. The associated publications and the used mathematical notations are also given in this chapter.

Chapter 2 provides technical background information on fundamental concepts that are commonly used in deep learning-based music source separation. More specifically, this chapter introduces the reader to the basics of digital signal processing, signal models, representations, artificial neural networks, masking-based separation of music sources, and the evaluation of source separation methods.

Chapter 3 focuses on how artificial neural networks are used to separate music sources. In particular, the focus is on spectral-based separation and a particular neural network model that has been used extensively for signal recovery and music source separation. In addition to this, an algorithm is introduced for examining what neural networks learn from spectral data. This chapter includes details regarding previously conducted research in music source separation, based on the previously mentioned neural network model, and the relation of the proposed algorithm to other research works.

Chapter 4 provides technical details for the proposed neural network architecture for music source separation. Target applications are singing voice separation and harmonic-percussive source separation. Additionally, this chapter contains information regarding extended inference strategies and optimization objectives, followed by the conducted experimental procedure for assessing the separation performance of the proposed architecture. The relationship of the proposed architecture to previous research is provided in the subsequent sections.

Chapter 5 focuses on learning representations from music signals. A particular focus is given on the unsupervised learning of representations that are helpful for the downstream task of singing voice separation. This chapter also provides details regarding prior research that focuses on the learning of representations for audio, speech, and music signals.

Chapter 6 concludes this thesis by summarizing the research work presented herein and highlights future research directions.

## 1.2 Scope of the Thesis

Music plays an important role in arts and cultural activities. Although digitalization of music has enabled many developments in archiving, sharing, analyzing, and processing of music content, the interaction between humans and music content is still limited. That is because the manipulation of individual auditory concepts, such as audio objects, is

an open problem. The ability to manipulate individual audio objects, e.g., change the acoustic characteristics of a music instrument in a recording, enables the re-purposing of music content. Examples of music re-purposing include music remixing, up-mixing for immersive applications, acoustic condition matching, and karaoke entertainment systems, among others.

Audio objects that are of high interest in this thesis are *music sources*. Although the definition of a music source might refer strictly to a specific music instrument, such as a violin in a string quartet, the term commonly refers to a group of music instruments that have perceptually similar audible characteristics and/or convey subjectively similar music information [3]. The task of estimating music sources from observed mixture signals is referred to as *music source separation*. Music source separation is a special case of source separation research [4], with the corresponding sources conveying music information. The music sources to be separated are referred to as the *target sources* and the rest of the sources are often regarded as the *interfering source(s)*.

Based on the grouping of the target music sources two important cases of music source separation emerge. These cases are singing voice separation and harmonic-percussive source separation (HPSS). Singing voice separation considers the singing voice signal as the target source. That includes any style of singing, independent from the music genre, and in many recent studies [A9] it refers also to any additional backing vocals or synthesized samples of singing voice. Music sources that do not fall in the category of singing voice, are commonly denoted as the accompaniment source [A9] that interferes with the target source in the observed mixture. On the other hand, HPSS aims at separating music mixture signals into harmonic and percussive sources. Harmonic sources refer to music instruments whose signals can be described as a combination of (sustained) oscillatory signals that evolve in time<sup>1</sup>, and percussive sources refer to music instruments whose signals are described mostly by transients<sup>2</sup> [3]. This broad categorization assumed in HPSS falls short for many music sources. An example is signing voice that can be characterized by a combination of both transients (vocal pulses) and oscillatory signals. Due to the limitations of this broad categorization singing voice separation is commonly distinguished from HPSS [A9]. An illustration of the above mentioned music signals is given in Figure 1.1.

Independent from the cases of separating music sources with particular signal characteristics, music source separation methods can be categorized based on the required *a priori* information [9]. On one side of the spectrum, *blind* methods do not require any information either about the target sources or about the process that yields the observed mixture signal, i.e., the *mixing process* [4], [9]. Conversely, *informed* separation methods require any available information regarding the target sources [10] and the mixing process [9]. In realistic scenarios, information about the individual target sources and the mixing process is scarce. This in turn, makes the informed separation methods less practical in real-world applications. On the other hand, the disregard of *a priori* information by blind separation methods necessitates the use of assumptions regarding the target

---

<sup>1</sup>In ideal cases, the frequencies of oscillatory signals that are used to characterize harmonic sources can be also harmonically related.

<sup>2</sup>In the context of computational musicology and music signal processing, defining the discriminative characteristics of harmonic and percussive sources is still an open problem [5], [6]. In this thesis, the widely-adopted definition of harmonic and percussive sources [3], [7], [8] is considered.

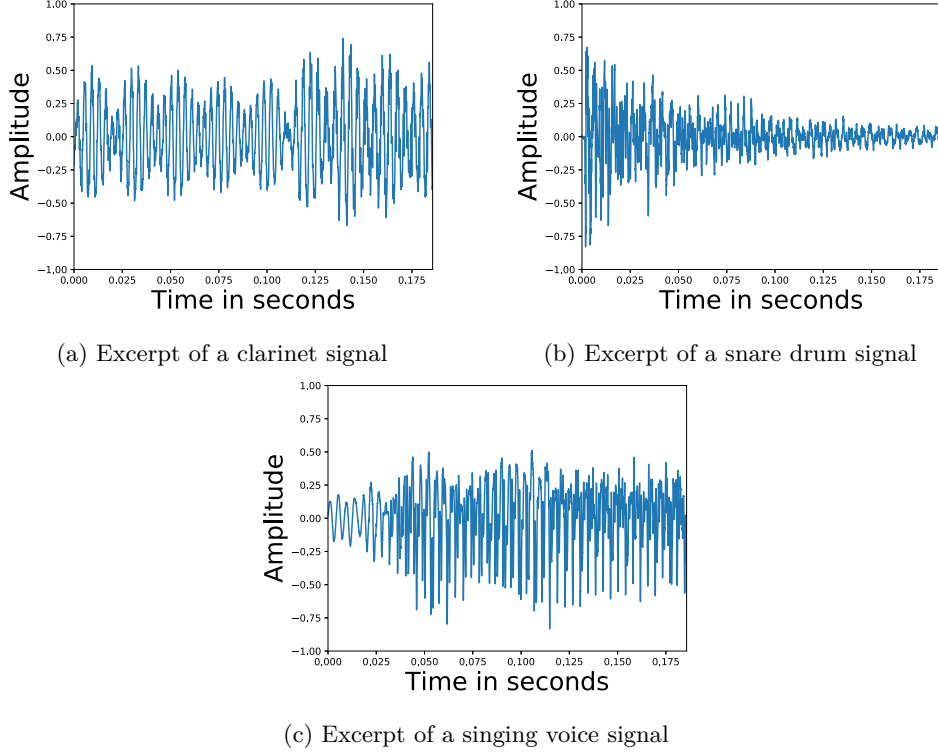


Figure 1.1: Three short-time excerpts of music signals, illustrating the structure of signals that exhibit (a) harmonic, (b) percussive, and (c) both signal characteristics.

sources and the mixing process. Most commonly used assumptions are the statistical independence or the (low) mutual information between the target sources [4].

Most of the time, these assumptions do not hold in music source separation, due to the structure of the target and interfering sources and the exhibited correlations between them [A9], [3], [9]. Consequently, blind separation methods fail at robustly separating music sources [A9], [9]. To alleviate this severe limitation, research has underlined the importance of introducing empirical information about the target sources or the mixing process [9], or the development of methods for computing side information from the mixture signal that is relevant to the target sources [11]. Although it is possible to further categorize those methods based on the exploited empirical or side information [A9], this thesis focuses on data-driven approaches to music source separation. Readers interested in more specific taxonomies of audio and music source separation methods are kindly referred to [9] and [A9].

Data-driven approaches acquire *a priori* information, regarding the target sources and the mixing process, by means of a learning algorithm and a collection of music signals, i.e., a *data-set*. A data-driven approach that has received a lot of attention, due to its remarkable performance in music source separation, is deep learning. Deep learning replaces human engineered signal processing operations, that are commonly used to compute side-

information regarding the mixing process and the target sources, by learnable operators (functions). These functions are learned with respect to the objective of estimating the target source(s) from the mixture, i.e., directly separating the target source(s).

### 1.3 Challenges & Thesis Contributions

In this thesis, deep learning approaches to music source separation are seen as a *domain informed* method for performing source separation. The domain refers to the set of information that the deep learning approaches employ for learning. This set of information contains the underlying signal structure of the target source(s), the mixture signal(s), and the inter-dependencies between the source(s) and the mixture. The learning from the domain can be either supervised or unsupervised.

Given the two previously mentioned learning paradigms, this thesis pursues and identifies the following goals and challenges. In supervised learning, the goal is to efficiently learn the un-mixing process. In this case, one challenge that arises is the applicability of the learned un-mixing process to music signals outside the observed domain, i.e., generalizing to unseen music mixtures. Another challenge is the learning of the un-mixing process when the available computational resources or the size of the domain, used for learning, are limited. Last but not least, another challenge is the interpretation and understanding of what deep learning approaches to music source separation learn using the provided domain information. The latter challenge is emerging because it offers an additional angle of evaluating deep learning approaches, other than the usage of typical metrics, that assesses the learning capabilities of the corresponding approaches.

In the case of unsupervised learning, the goal is to learn the underlying signal structure of the source(s) and of the mixture so that it can be used for un-mixing the target source(s). In this case, two challenges are identified. The first challenge is to learn music signal representations that are convenient for separating the target source(s). The second challenge is the interpretation of the learned signal representations. Having in mind the above described goals and the associated challenges, the work presented in the next chapters of this thesis makes the following contributions:

- An algorithm for distilling non-linear deep neural networks (DNNs), enabling the understanding of how DNNs separate music sources in the frequency domain.
- Experimental evidence that DNNs learn data-driven filter operators that can be further enhanced by using skip-filtering connections; a revisit of a simple technique that is utilized in state-of-the-art music source separation approaches.
- An efficient neural architecture that uses skip-filtering connections and is applicable to singing voice separation, and HPSS, leading to competitive results.
- A re-parameterization scheme for the decoding functions of denoising autoencoders (DAEs), enabling the computation of interpretable signal representations that are convenient for music source separation.

## 1.4 Associated Publications

The thesis content is based on the selected publications listed below. For publications that contain a significant overlap of contributions between co-authors, a clarification is given beneath each corresponding list of publication regarding the author's.

### Journal Articles

- [A9] Z. Rafii, A. Liutkus, F. R. Stöter, **S. I. Mimitakis**, D. FitzGerald, and B. Pardo, “An Overview of Lead and Accompaniment Separation in Music”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1307–1335, 2018
- [A22] **S. I. Mimitakis**, K. Drossos, E. Cano, and G. Schuller, “Examining the Mapping Functions of Denoising Autoencoders in Singing Voice Separation”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 266–278, 2020

The author's contributions in the overview article presented in [A9], are the literature overview, method descriptions, and illustration of some figures for the data-driven and optimization-based approaches.

### Peer-reviewed Conference Papers

- [A6] **S. I. Mimitakis**, K. Drossos, T. Virtanen, and G. Schuller, “A Recurrent Encoder-Decoder Approach with Skip-filtering Connections for Monaural Singing Voice Separation”, in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017, pp. 1–6
- [A10] **S. I. Mimitakis**, K. Drossos, J.-F. Santos, G. Schuller, T. Virtanen, and Y. Bengio, “Monaural Singing Voice Separation with Skip-Filtering Connections and Recurrent Inference of Time-Frequency Mask”, in *Proceedings of the 43rd International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018)*, 2018
- [A11] K. Drossos, **S. I. Mimitakis**, D. Serdyuk, G. Schuller, T. Virtanen, and Y. Bengio, “MaD TwinNet: Masker-Denoiser Architecture with Twin Networks for Monaural Sound Source Separation”, in *Proceedings of the 2018 IEEE International Joint Conference on Neural Networks (IJCNN)*, 2018
- [A12] K. Drossos, P. Magron, **S. I. Mimitakis**, and T. Virtanen, “Harmonic-Percussive Source Separation with Deep Neural Networks and Phase Recovery”, in *Proceedings of the 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2018, pp. 421–425
- [A14] **S. I. Mimitakis**, E. Cano, D. FitzGerald, K. Drossos, and G. Schuller, “Examining the Perceptual Effect of Alternative Objective Functions for Deep Learning Based Music Source Separation”, in *Proceedings of the 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 679–683

- [A21] **S. I. Mimitakis**, K. Drossos, and G. Schuller, “Unsupervised Interpretable Representation Learning for Singing Voice Separation”, in *Proceedings of the 27th European Signal Processing Conference (EUSIPCO 2020)*, 2020

The work presented in [A10] semantically builds on [A6] and contains two main ideas: the usage and model improvements for skip-filtering connections and the recurrent inference algorithm. The author’s contributions concern the first idea that has been extensively used ever since in music source separation based on deep learning. The work presented in [A11] builds on [A10], with the author’s contributions being the experimental procedure and equal contribution to the development stages. In [A12], the work of [A11] is applied to the problem of HPSS combined with a phase retrieval algorithm. The author has equal contributions with K. Drossos and P. Magron with respect to the development of the experimental procedure and the writing process. In [A14], the author has developed the experimental procedure and has equally contributed to E. Cano for conducting listening tests.

### Scientific Archives

- [A5] **S. I. Mimitakis** and G. Schuller, *Investigating the Potential of Pseudo Quadrature Mirror Filter-Banks in Music Source Separation Tasks*, 2017. arXiv: 1706.04924 [cs.SD]
- [A24] **S. I. Mimitakis**, K. Drossos, and G. Schuller, *Revisiting Representation Learning for Singing Voice Separation with Sinkhorn Distances*, 2020. arXiv: 2007.02780 [cs.SD]

In [A5], the author’s contribution lies in the idea of investigating the applicability of a particular signal representation to the problem of music source separation. The filter-bank design method is accredited to G. Schuller.

### Data-sets

- [A8] Z. Rafii, A. Liutkus, F. Stöter, **S. I. Mimitakis**, and R. Bittner, *The MUSDB18 Corpus for Music Separation*, 2017. [Online]. Available: <https://doi.org/10.5281/zenodo.1117372>

The author’s contribution in the MUSDB18 data-set [A8] is the production and mixing of the raw audio multi-tracks, originally used in a previous version of the MUSDB18 data-set known as the Demixing Secrets Dataset<sup>3</sup> (DSD100).

## 1.5 Notation

In order to preserve consistency of the mathematical notation throughout this thesis, a list of symbols is given below. The list’s primary use is for the convenience of the reader.

- Vectors are denoted by **boldface** letters, such as **x**.

---

<sup>3</sup><http://www.sisec17.audiolabs-erlangen.de>

- Vectors are assumed to be column-vectors, whose elements can be accessed via indexing yielding scalars. More formally,

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_{[1]} \\ x_{[2]} \\ \vdots \\ x_{[n]} \end{bmatrix},$$

where the  $n$ -th element is represented using a numerical subscript, i.e.,  $x_n$ , but also using brackets, i.e.,  $x_{[n]}$ . The latter notation is used as an intermediate point between the notation used in digital signal processing and the algebraic operations commonly used in deep learning research.

- The (column-)vector containing  $N$  values that are equal to one is defined as

$$\mathbf{1}_N = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$

- Matrices are denoted by capital **boldface** letters, such as  $\mathbf{X}$ . Similarly to the vector notation, the matrix elements are denoted by

$$\mathbf{X} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{NM} \end{bmatrix} = \begin{bmatrix} X_{[1,1]} & X_{[1,2]} & \cdots & X_{[1,M]} \\ \vdots & \vdots & \ddots & \vdots \\ X_{[n,1]} & X_{[n,2]} & \cdots & X_{[n,M]} \end{bmatrix}.$$

- The  $N \times M$  all-zero matrix (null matrix) is denoted as  $\mathbf{0}^{N \times M}$  and is defined as

$$\mathbf{0}^{N \times M} = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}.$$

- Diagonal matrices are defined as matrices whose entries outside the main diagonal are all zero. All diagonal matrices are assumed to be symmetric diagonal matrices, i.e., they are square.
- A diagonal matrix can be constructed using the  $\text{diag}(\cdot)$  operator and given a vector argument. For example, given a vector argument  $\mathbf{v} \in \mathbb{R}^3$  the corresponding diagonal matrix  $\mathbf{V} \in \mathbb{R}^{3 \times 3}$  can be computed as  $\mathbf{V} = \text{diag}(\mathbf{v})$ , where

$$\text{diag}(\mathbf{v}) = \begin{bmatrix} v_{[1]} & 0 & 0 \\ 0 & v_{[2]} & 0 \\ 0 & 0 & v_{[3]} \end{bmatrix}.$$

- The trace of a matrix  $\mathbf{X} \in \mathbb{R}^{N \times N}$  is defined as

$$\text{tr}(\mathbf{X}) = \sum_{n=1}^N X_{[n,n]}.$$



- All element-wise operations on vectors and matrices are characterized by a circular symbol. For example, multiplication is denoted by “ $\odot$ ”, exponentiation using an exponent  $\alpha$  by “ $\cdot^{\odot\alpha}$ ”, and division by “ $\oslash$ ”. Exception to the circular symbol that can still refer to element-wise operations, are defined functions (presented in advance), the square root function “ $\sqrt{\cdot}$ ”, and the absolute value function “ $|\cdot|$ ” that, when applied to vectors and matrices, an element-wise operation is assumed.
- The application of the absolute value function “ $|\cdot|$ ” to complex-valued numbers, vectors, and matrices refers to the modulus of complex numbers, that is defined as

$$|x + iy| = \sqrt{x^2 + y^2},$$

where  $i = \sqrt{-1}$ .

- Transposition of a vector, or a matrix is denoted by  $\mathbf{x}^\top$  and by  $\mathbf{X}^\top$ , respectively. Hermitian transposition is denoted by  $\mathbf{X}^H$ .
- The set of real, complex, integer, positive integer, and non-negative real numbers is denoted by  $\mathbb{R}$ ,  $\mathbb{C}$ ,  $\mathbb{Z}$ ,  $\mathbb{Z}_+$ , and  $\mathbb{R}_{\geq 0}$ , respectively. Less general sets, such as data-sets, are denoted by letters and are defined using curly brackets. For example, a data-set containing  $K$   $\mathbf{x}$  and  $\mathbf{y}$  vectors is denoted as  $\mathcal{D} \in \{\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^K$ .
- Intervals are indicated using brackets, e.g.  $(\alpha, \beta)$ ,  $[\alpha, \beta]$ . Intervals are also used as subscripts. In that case, a sub-set is assumed. For example, the sub-set of integers in the closed interval of  $[-5, 5]$  is denoted by  $\mathbb{Z}_{[-5, 5]}$ .
- Functions are denoted by lower case letters with the corresponding arguments being inside the parentheses, e.g.,  $f(\cdot)$ .
- Functions of particular interest, that are defined and explained in this thesis, are denoted by capital calligraphic letters. Examples include the discrete Fourier transform  $\mathcal{F}_{\text{DFT}}(\cdot)$  and the mean squared error loss function  $\mathcal{L}_{\text{MSE}}(\cdot)$ , among others.
- The ceiling and floor functions are denoted as  $\lceil \cdot \rceil$  and  $\lfloor \cdot \rfloor$ , respectively.
- To describe a method that has multiple levels of dependencies, this thesis also uses random variable notation. In this case, signals of interest are considered as realizations of random variables. Random variables are denoted in italic, i.e. “ $x$ ” in contrast to the scalar “ $x$ ”.
- Random variables, vectors, matrices, functions, and sets, can also have an alphabetical subscript or superscript, e.g.,  $\mathbf{e}^{(\text{distortion})}$  or  $f_{\text{dec}}(\cdot)$ . The subscripts and superscripts are used to denote the link of the variable, vector, matrix, function, or set, to a specific operation or computational outcome. When multiple operations or computational outcomes are addressed, the asterisk “ $*$ ” is used for brevity to replace the alphabetical subscripts or superscripts.



## Chapter 2

# Fundamentals of Deep Learning-Based Music Source Separation

### 2.1 Introduction

The task of separating music sources is approached using digital signal processing techniques [3]. These techniques operate on discrete-time signals defined on a finite set. To obtain such signals, let us briefly consider a very common signal acquisition pipeline. Emitted acoustic signals are first captured by an electroacoustic device, such as a microphone. A microphone converts acoustic pressure fluctuations over time into a continuous electrical signal [12]. By means of a signal transformation, such as the (short-time) Fourier transform, this electrical signal can be described by frequency dependent amplitude and phase information.

Since music signals are meant for human listening and the human auditory system is band-limited from 20 Hz to 20 kHz, the continuous electrical signal can be accordingly band-limited, discretized, and encoded as a numerical sequence. To that aim, the Nyquist-Shannon sampling theorem informs us that a sampling frequency greater or equal to 40 kHz is necessary for the discretization of a band-limited audio signal. After the discretization, the amplitude values of the signal are mapped to the nearest amplitude values of a pre-defined and finite numerical set. This process of mapping is commonly known as quantization. Both processes, discretization and quantization, are standard in digital audio consumer devices and are usually implemented by the analog-to-digital converter (ADC). The inverse operations are performed by the digital-to-analog converter (DAC). Proper quantization of audio signals leads to inaudible errors due to the masking effects in the human auditory system [13]. Typical audio converters use 16 bits per sample, and a sampling frequency of 44.1 kHz.

Although it could be argued that the above processes are sub-optimal for a wide range of signal processing applications [14] and these processes could be further optimized for the task of separating signals, nearly all digital music content is acquired using these

conventional processes. And so this thesis aims at addressing the problem of separating music sources from mixture signals that are acquired from a conventional acquisition process, disregarding more advanced signal acquisition techniques [14]. The rest of this chapter is organized as follows: Section 2.2 introduces the basics of signal modelling and establishes the connection between signal models and artificial neural networks (ANNs). Section 2.3 describes mixing models that are relevant in deep learning-based music source separation, and focuses on the separation of music sources from realistic mixtures. To this end, Section 2.4 provides information regarding the evaluation procedures used for assessing the performance of music source separation approaches.

## 2.2 From Signal Models to Neural Networks

The modelling of signals and the investigation of signal representations are open research topics that have received attention from various research areas, such as sparse-aware signal processing [15], compressed sensing [14], and deep learning [1]. Consequently, there is a vast amount of information regarding concepts, algorithms, and approaches for modelling and representing signals. This section aims at conveying the important information relevant to audio and music signal processing and the connection between signal modelling and artificial neural networks.

### 2.2.1 Signal Modelling Basics

Almost every digital signal processing technique, including source separation, can be broken down and understood using the fact that any discrete-time signal can be composed as a weighted sum of elementary functions [16]. These elementary functions are commonly referred to as *basis functions*. Assuming that a music signal has been acquired according to the acquisition process described in Section 2.1, let  $\mathbf{x} \in \mathbb{R}_{[-1,1]}^T$  denote the discrete-time music signal, where  $T$  is an integer denoting the number of time-domain samples and determines the dimensionality of  $\mathbf{x}$ . This signal is henceforth denoted as the time-domain signal. Composing  $\mathbf{x}$  using a weighted sum of  $N$  basis functions can be expressed as

$$\mathbf{x} = \mathbf{W}\mathbf{z}. \quad (2.1)$$

In Eq. (2.1),  $\mathbf{W}$  is a  $T \times N$  matrix that consists of  $N$  column vectors that are the basis functions. Each basis function is denoted as  $\mathbf{w}_n$  and is a vector of size  $T$ . In order to compose the signal  $\mathbf{x}$  each  $\mathbf{w}_n$  is weighted by the vector element  $z_{[n]}$ . In the context of composing the time-domain audio signal  $\mathbf{x}$ , Eq. (2.1) is also known as *synthesis*. Using the synthesis operation, the task of source separation can now be seen as the selection of the basis functions or the elements contained in  $\mathbf{z}$  that correspond to the target source(s) and not to the interfering source(s)<sup>1</sup> [4], [7], or as the new estimation of  $\mathbf{W}$  and  $\mathbf{z}$  with respect to the target source(s)<sup>2</sup>.

The vector elements contained in  $\mathbf{z}$  form the *representation* of the time-domain signal  $\mathbf{x}$ . Generally, the representation is said to be *complete* when the number of basis functions

---

<sup>1</sup>The selection process follows the assumption that the basis functions in  $\mathbf{W}$  lead to disjoint and orthogonal sources [17], a property that is later discussed in Section 2.3.4.

<sup>2</sup>The estimation of the corresponding variables with respect to an objective can be performed as in the training of artificial neural networks discussed in Section 2.2.4.

is equal to the dimensionality of  $\mathbf{x}$ , i.e.,  $N = T$ . For the cases of  $N < T$  and  $N > T$  the representation is said to be *under-complete* and *over-complete*, respectively [16]. It should be stated that a complete representation can be considered as over-complete even when  $N = T$ . That is usually the case when  $\mathbf{W}$  contains redundant or linear-dependent basis functions. As a consequence of the redundancy, the representation is called over-complete and redundant, or just redundant for short.

In many realistic scenarios, audio and music signals can last from a few seconds to several hours. The variability of the signals' length imposes also a change in the size of each basis function  $\mathbf{w}_n$  when Eq. (2.1) is used to obtain the corresponding signals. Even in the case of processing a signal of only a few seconds long, the number of samples  $T$  can be very large, e.g.,  $T = 10^5$  for a duration of 5 seconds only at the sampling frequency of 20 kHz. That is because  $T$  is the product between the sampling frequency, that is usually at least 32 kHz for music signals, and the duration of the signal in seconds. Consequently, the operation described in Eq. (2.1) becomes very demanding, with respect to the computational resources, and in many practical scenarios inefficient. Additionally, the dependency of Eq. (2.1) on the number of time-domain samples, makes the operation of Eq. (2.1) not applicable for processing audio and music signals of arbitrary length.

To alleviate the above problems and be able to develop digital signal processing techniques that efficiently process audio and music signals of arbitrary lengths, the local structure of the time-domain signals is exploited. For very short-time excerpts (segments) of a time-domain signal, it can be observed that the statistical properties, such as the mean and the variance, remain approximately constant. Additionally, periodic structures can be observed by correlating any of the previously mentioned segments with time-shifted versions of the same segment. In other words, the signals are weakly *stationary* and are denoted as wide-sense stationary (WSS). An illustration of the weak stationarity is given in Figure 2.1.

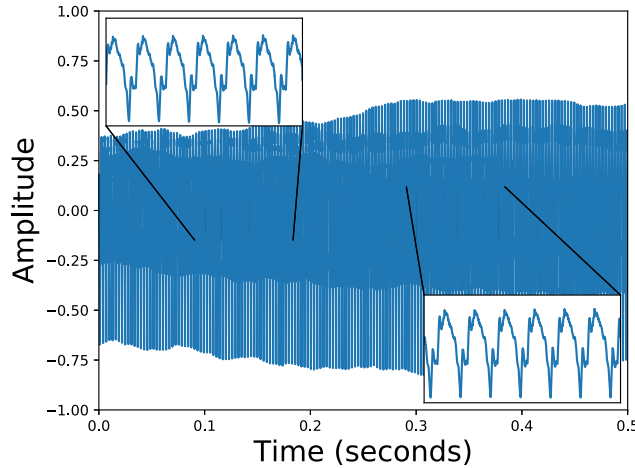


Figure 2.1: A half-second signal recording of a cello music instrument. Zooming into a segment of 100 milliseconds (ms), a periodic and WSS signal is observed.

In practice, audio and music signals are subdivided into small segments, denoted as time-frames, and it is assumed that the (segmented) signal in each time-frame is WSS. Subdividing a long signal yields a matrix  $\mathbf{X} \in \mathbb{R}_{[-1,1]}^{T' \times M}$  that contains  $M$  time-frames that are available in the audio signal  $\mathbf{x} \in \mathbb{R}_{[-1,1]}^T$ . Each time-frame is a vector that contains  $T'$  samples and  $T'$  is smaller than the number of samples contained in the original signal, i.e.,  $T' \ll T$ . The segmentation and re-arrangement of the signal  $\mathbf{x}$  into the matrix  $\mathbf{X}$  is formally expressed as

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{[0]} & \mathbf{x}_{[s]} & \mathbf{x}_{[2s]} & \dots \\ \vdots & \vdots & \vdots & \ddots \\ \mathbf{x}_{[T'-1]} & \mathbf{x}_{[s+T'-1]} & \mathbf{x}_{[2s+T'-1]} & \dots \end{bmatrix},$$

where  $s$  is a positive integer, such that  $T' \geq s > 0$ , and denotes the step-size of the time-domain signal segmentation. The step-size of the segmentation determines the overlap between the adjacent time-frames. The matrix  $\mathbf{X}$  can now be used to process the audio signal or to compute the signal representation, using a predefined and fixed number of time-domain samples. The operation for obtaining the corresponding vector  $\mathbf{x}$  from the matrix form  $\mathbf{X}$  is a vectorization algorithm that is commonly referred to as the overlap-add, computed as

$$x_{[t]} = \sum_m X_{[t-m, s, m]} \quad \forall t \in [0, 1, \dots, T-1], \quad (2.2)$$

where  $t$  and  $m \in [0, 1, \dots, M-1]$  are indices for the time-domain samples of  $\mathbf{x}$  and time-frames of  $\mathbf{X}$ , respectively. For the equality expressed in Eq. (2.2) to hold, it is assumed that

$$X_{[t', m]} = 0 \quad \text{if } t' \notin [0, 1, \dots, T'-1].$$

In order for the overlap-add algorithm to perfectly yield  $\mathbf{x}$  when  $s < T'$  using Eq. (2.2), the application of a windowing function is necessary<sup>3</sup>. The windowing function is applied element-wise to each column vector of  $\mathbf{X}$ , formally expressed as

$$\mathbf{X} = \text{diag}(\mathbf{v}) \begin{bmatrix} \mathbf{x}_{[0]} & \mathbf{x}_{[s]} & \mathbf{x}_{[2s]} & \dots \\ \vdots & \vdots & \vdots & \ddots \\ \mathbf{x}_{[T'-1]} & \mathbf{x}_{[s+T'-1]} & \mathbf{x}_{[2s+T'-1]} & \dots \end{bmatrix}, \quad (2.3)$$

where  $\mathbf{v} \in \mathbb{R}^{T'}$  is the windowing function and  $\text{diag}(\cdot)$  is the diagonal operator. It should be noted that windowing functions do not satisfy the equality expressed in Eq. (2.2) for all  $s$  values, i.e., the overlap-add procedure does not yield the original audio signal. For a comprehensive review of various step-sizes for various windowing functions, interested readers are referred to [18], [19]. An illustration of the Hamming windowing function, that is a typical windowing function in music source separation, is given in Figure 2.2.

---

<sup>3</sup>Windowing functions are useful also for reducing cross-talk between frequency components in relevant frequency analyses [18]. However, in the context of music source separation experimental evidence [A5] suggests that the cross-talk reduction of windowing functions plays a negligible role in the estimation or separation of sources. In relevant literature [A9], nearly all of the approaches for music source separation select the appropriate windowing function based on the empirically chosen time-resolution of the short-time transform; a mindset adopted by this thesis.

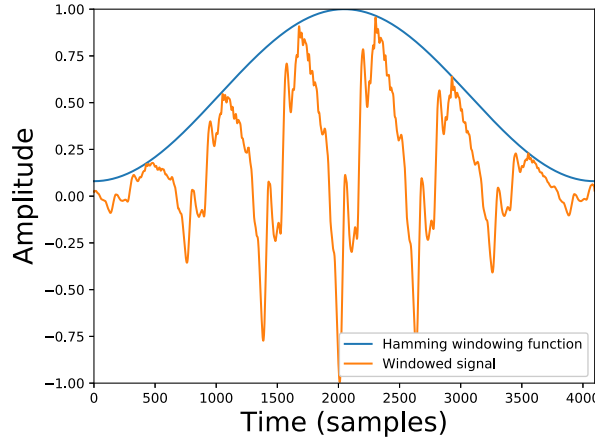


Figure 2.2: The Hamming windowing function applied to the 100 ms time-frame of the cello recording.

By using the above matrix expressions, the functionality of Eq. (2.1) remains the same. However, the representation  $\mathbf{z}$  is replaced by the matrix  $\mathbf{Z}$  of size  $T' \times M$ . Each column in  $\mathbf{Z}$  is the representation of each corresponding column (time-frame) in the matrix  $\mathbf{X}$ . The main benefit of the segmentation process is that the basis functions, contained in  $\mathbf{W}$ , and the corresponding representation have dramatically smaller dimensions, leading to more computationally efficient processing of audio and music signals. The signal information with respect to time variation is acquired from the time-frames of the representation, allowing the analysis and modelling of the temporal structure of music signals.

### 2.2.2 Signal Representations

The central point of the previous section is the synthesis operation of a signal  $\mathbf{x}$ . In contrast, this section focuses on computing the representation  $\mathbf{z}$ . Computing signal representations can be understood as the decomposition, also known as *analysis*, of the signal  $\mathbf{x}$  into a finite set of components. Since the representations are computed for each time-frame and the matrix notation (using Eq. (2.3)) will cause a clutter in the notation, the data-set notation is introduced. Specifically, instead of dealing with two different matrices, one for the signal and one for the representation, and accessing each corresponding matrix's column to denote a single time-frame vector, each time-frame vector is considered an individual example (data-point) in the data-set  $\mathcal{D}$ . To distinguish between the  $M$  time-frame vectors, i.e., the examples, a bracketed superscript to each vector is used, i.e.,  $\mathcal{D} \in \{\mathbf{x}^{(i)}, \mathbf{z}^{(i)}\}_{i=1}^M$ .

For audio and music signal processing applications, universally optimal signal representations do not exist. That is because optimal representations are dependent on the provided or computed basis functions, and the basis functions are adapted towards the requirements of the signal processing application [20]. For computing signal representations, two general frameworks can be distinguished from relevant research literature [15,

Chapter 9], [20], [21]. In the first framework, the aim is to find the best representation  $\mathbf{z}_o^{(i)}$  given the basis functions  $\mathbf{W}$ , by solving the following constrained optimization problem:

$$\begin{aligned} \mathbf{z}_o^{(i)} &= \underset{\mathbf{z}^{(i)}}{\operatorname{argmin}} \|\mathbf{z}^{(i)}\|_0, \\ \text{subject to } \mathbf{x}^{(i)} &= \mathbf{W}\mathbf{z}^{(i)}. \end{aligned} \quad (2.4)$$

In Eq. (2.4) the objective is to find the representation that has the smallest number of non-zero elements, i.e., the *sparsest* representation, that satisfies Eq. (2.1). The number of non-zero elements is given by the  $\ell_0$  vector norm ( $\|\cdot\|_0$ ). However, the computation of the  $\ell_0$  norm used in Eq. (2.4) is NP-hard. To alleviate this, the  $\ell_0$  norm is often replaced by an approximation. Most commonly the  $\ell_1$  norm ( $\|\cdot\|_1$ ) is used [20]. The replacement by the  $\ell_1$  norm relaxes the problem of Eq. (2.4) to a linear program that can be solved using a wide range of algorithms. Examples of algorithms that have been successfully applied in audio and music signal processing are the basis pursuit algorithm [22], the iterative thresholding algorithm [23], and greedy approaches such as the matching pursuit algorithm [24], [25]. This category of approaches is extensively used and investigated in the research fields of sparse coding and compressed sensing [14], [20].

Regarding the second framework, computing representations can be seen from a more general point of view that encompasses the first framework. More specifically, the second framework relies on energy-based learning<sup>4</sup> [21]. Particularly to the problem of learning signal representations, energy-based learning allows to learn models, i.e., functions for analyzing and synthesizing the signal(s) of interest. The learning of models is performed by using an energy function denoted by  $\mathcal{E}(\cdot)$ . The function  $\mathcal{E}(\cdot)$  yields a scalar value that is very small, ideally 0, when the signal estimated by the model is very close or ideally identical to  $\mathbf{x}^{(i)}$ . In contrast, the energy function yields a high value when the signal constructed by the model is not equal or similar to  $\mathbf{x}^{(i)}$ . In many practical scenarios of digital signal processing the energy function is implemented as a divergence function or a loss function  $\mathcal{L}(\cdot)$  that evaluates how close or similar two signals are. Typical loss functions are computed using a (vector) norm of the residual between the constructed and the target signal.

The above described framework finds many applications and is general enough to include the extremely wide range of algorithms, methods, and models that have been devised in order to analyze and process signals. For example, energy-based learning can be seen as a minimization problem with the main objective to obtain the output of the energy function as close as possible to 0. Taking that into account, the problem of Eq. (2.4) can be re-formulated using energy-based learning as

$$\mathbf{z}_o^{(i)} = \underset{\mathbf{z}^{(i)}}{\operatorname{argmin}} \mathcal{E}(\mathbf{x}^{(i)}, \mathbf{W}, \mathbf{z}^{(i)}) \quad (2.5)$$

$$\mathcal{E}(\mathbf{x}^{(i)}, \mathbf{W}, \mathbf{z}^{(i)}) = \mathcal{L}(\mathbf{x}^{(i)}, \mathcal{M}(\mathbf{W}, \mathbf{z}^{(i)})) . \quad (2.6)$$

The energy function defined in Eq. (2.6) can be also sparsity promoting, with respect to  $\mathbf{z}^{(i)}$ , as in Eq. (2.4). This can be done by adding the  $\ell_p$  vector norm as an additional term in the loss function  $\mathcal{L}(\cdot)$ . In Eq. (2.6),  $\mathcal{M}(\cdot)$  is an auxiliary function that allows an extra degree of freedom for interacting and operating on the corresponding arguments, i.e.,  $\mathbf{W}$

---

<sup>4</sup>Not to be confused with energy-based models discussed in [21].



and  $\mathbf{z}^{(i)}$ . Specifically for this case, the auxiliary function is defined as the vector-matrix product between the representation  $\mathbf{z}^{(i)}$  and the matrix  $\mathbf{W}$

$$\mathcal{M}(\mathbf{W}, \mathbf{z}^{(i)}) := \mathbf{W}\mathbf{z}^{(i)} .$$

It should be stated that the auxiliary function  $\mathcal{M}(\cdot)$  can be used to express an arbitrary or even non-linear operation applied to  $\mathbf{z}$ .

Since the objective of energy-based learning is to push down the output of the energy function, or in other words minimize the loss  $\mathcal{L}(\cdot)$ , additional directions in learning representations emerge. These directions aim at overcoming the limitations that might be imposed when learning of representations is performed using a *fixed* set of basis functions. Learning representations with a fixed set of basis functions could potentially lead to representations that do not accurately characterize the signals of interest [26], [27]. To alleviate this issue, the minimization described in Eq. (2.5) can be performed with respect to a set of variables  $\mathcal{W}$  instead of considering only the representation  $\mathbf{z}^{(i)}$ . More formally, Eq (2.5) can be revisited using

$$\begin{aligned} \mathcal{W}_o = \underset{\mathcal{W}}{\operatorname{argmin}} \mathcal{E}(\mathbf{x}^{(i)}, \mathcal{W}) , \text{ where} \\ \mathcal{W} = \{\mathbf{z}^{(i)}, \mathbf{W}\} \end{aligned} \tag{2.7}$$

is a set that contains both the representation  $\mathbf{z}$  and the matrix  $\mathbf{W}$ . The important thing to notice here is that by using Eq. (2.7), both the basis functions and the representation can be learned with respect to the signal  $\mathbf{x}^{(i)}$ . The task of learning new basis functions  $\mathbf{W}$ , is commonly referred to as *dictionary learning* [26], and goes hand-in-hand with the task of learning (sparse) signal representations [20], [28].

In addition to the previously mentioned joint learning of dictionaries and representations, Eq. (2.5) and Eq. (2.6) can be further extended. Two frequently used extensions in digital signal processing and deep learning are the explicit formulation or design of the auxiliary function  $\mathcal{M}(\cdot)$  in Eq. (2.6) and the learning of basis functions that describe the whole signal data-set  $\mathcal{D}$ . The learning of basis functions that describe the whole data-set can be achieved by minimizing the output of the energy function  $\mathcal{E}(\cdot)$  averaged over the data-set. This type of minimization is commonly referred to as *empirical loss minimization* [21] and it is formally expressed along the lines of Eq. (2.6) as

$$\mathcal{E}(\mathcal{D}, \mathcal{W}) = \frac{1}{M} \sum_{i=1}^M \mathcal{L}(\mathbf{x}^{(i)}, \mathcal{M}(\mathbf{W}, \mathbf{z}^{(i)})) , \tag{2.8}$$

where  $M$  is the number of paired-vectors, i.e.,  $\mathbf{x}^{(i)}$  and  $\mathbf{z}^{(i)}$ , contained in  $\mathcal{D}$ .

In addition to the above, many approaches have considered replacing the explicit optimization with respect to the representation  $\mathbf{z}^{(i)}$  by a proxy. An extensively used proxy is to express the representation  $\mathbf{z}^{(i)}$  as a function of the signal  $\mathbf{x}^{(i)}$ . The function of the signal  $\mathbf{x}$  is usually parameterized by an additional set of basis functions that is shared among all  $M$  examples. By letting the additional basis functions be denoted by  $\mathbf{W}_a$ , in the simplest case the representation  $\mathbf{z}^{(i)}$  can be computed as a function of the signal  $\mathbf{x}^{(i)}$  using:

$$\mathbf{z}^{(i)} = \mathbf{W}_a \mathbf{x}^{(i)} . \tag{2.9}$$

In Eq. (2.9) the matrix  $\mathbf{W}_a$  *transforms* the signal  $\mathbf{x}$  into the corresponding representation. The subscript “a” is used for  $\mathbf{W}_a$  in Eq. (2.9) to denote the analysis operator of the signal  $\mathbf{x}^{(i)}$ . The goal now is to find the set of both analysis and synthesis basis functions  $\mathcal{W} = \{\mathbf{W}, \mathbf{W}_a\}$  that minimize the output of the energy function. There are two benefits for learning suitable analysis and synthesis basis functions with respect to an objective that is expressed as a minimization problem of the output of the energy function. The first benefit is that by using the learned analysis basis functions to transform an audio mixture signal, the resulting representation can be used to detect the sources contained in the mixture signal. The detection of sources is often easier in the representation domain, computed using the analysis basis functions, compared to the time-domain of audio signals [20]. The second benefit, is that ideally the learned synthesis basis functions can be used to reconstruct the transformed signal without any loss of information. This is particularly useful when the representation is processed, e.g., the interfering sources are detected and removed by an appropriate operation<sup>5</sup>, and then the representation is used to obtain the time-domain signal with the help of the synthesis basis functions. With this in mind, there are two important directions that are of high relevance to this thesis; namely, these are (handcrafted) audio signal transforms and learned transforms by neural networks.

### 2.2.3 Audio Signal Transforms

A very common choice for selecting the analysis basis functions  $\mathbf{W}_a$  is based on the local structure of the audio and music signals, described in Section 2.2.1. The weak stationarity and specifically the periodicity observed in the short-time segments of audio and music signals, implies that the class of periodic functions could serve as a reasonable starting point for selecting the basis set  $\mathcal{W} = \{\mathbf{W}, \mathbf{W}_a\}$ . In particular, periodic functions that enable an intuitive transformation of audio signals are sinusoidal functions, i.e., cosine and sine functions. Sinusoidal functions contain a very limited frequency content, i.e., frequency bandwidth, of a single frequency. This enables the transformation of an audio signal into a representation that conveys information about frequencies. Often this transformation is related to the human auditory system [13], and the modelling of music instruments [7], making sinusoidal functions an attractive choice for separating music sources. That is because, music source separation can be also seen as a frequency filtering operation [3], [29].

The most well-known and commonly used transform in music source separation that employs sinusoidal functions is the discrete Fourier transform (DFT). The DFT is a complex-valued transform that uses the analysis matrix  $\mathbf{W}_a \in \mathbb{C}^{N \times N}$  as

$$W_{a[k,n]} = \frac{1}{\sqrt{N}} \left( \cos\left(\frac{2\pi kn}{N}\right) - i \sin\left(\frac{2\pi kn}{N}\right) \right), \quad (2.10)$$

where  $\frac{1}{\sqrt{N}}$  is a normalization scalar that is useful in obtaining the basis functions for the synthesis  $\mathbf{W}$ ,  $i = \sqrt{-1}$ , and  $\cos(\cdot)$  and  $\sin(\cdot)$  denote the cosine and sine functions, respectively. In Eq. (2.10),  $k$  and  $n$  are integers  $k, n \in [0, 1, \dots, N-1]$  that denote the index of the frequency and time-sample of each function, respectively. Although using the DFT implies that complex numbers have to be used to process real-valued time-domain

---

<sup>5</sup>Examples of operations are given in Section 2.3.

signals, the convenience of using a complex-valued representation is the computation of the magnitude and the phase of the signal. The magnitude is the modulus of the complex valued representation applied to each frequency component, or sub-band, and the phase is the complex angle computed in each sub-band.

Subject to the goal of minimizing the output of the energy function  $\mathcal{E}(\cdot)$ , expressed in Eq. (2.6), the complex-valued exponential functions employed in the DFT, can be handy. That is because the employed functions are orthogonal to each other, forming an orthogonal basis over  $N$  samples [16]. This means that the (real-valued) product between  $\mathbf{W}_a$  and its Hermitian transpose  $\mathbf{W}_a^H$  is the identity matrix  $\mathbf{I}_N$ . From an algebraic perspective, by computing  $\mathbf{W}_a$  using Eq. (2.10) and by allowing the synthesis basis functions be

$$\mathbf{W} = \mathbf{W}_a^H ,$$

any audio and music signal can be analyzed and synthesized *without* losing any information for that signal, i.e.,  $\mathbf{x}$  is perfectly reconstructed resulting into a nullification of the output of the energy function.

The DFT can be also used to compute a time-localized transformation of the signal. This transform is denoted as the short-time Fourier transform (STFT), and it employs the above described analysis and synthesis matrices  $\{\mathbf{W}_a, \mathbf{W}\}$ . The main difference between the DFT and the STFT is that the STFT applies the DFT to all short-time segments available in the signal. This is done by using Eq. (2.3), where the time-domain signal  $\mathbf{x} \in \mathbb{R}_{[-1,1]}^T$ , of  $T$  samples, is segmented into  $M$  time-frames, each consisting of  $T'$  samples. At each time-frame, a windowing function is applied, leading to the yielded matrix form  $\mathbf{X}^{T' \times M}$  of the signal  $\mathbf{x}$ . By ensuring that the number of samples in each time-frame of  $\mathbf{X}$  is equal to the number of basis functions  $N$ , i.e.,  $T' = N$ , a straightforward way to perform the STFT analysis and synthesis is by using:

$$\mathbf{Z} = \mathbf{W}_a \mathbf{X} \tag{2.11}$$

$$\mathbf{X} = \mathcal{R}(\mathbf{WZ}) , \tag{2.12}$$

where  $\mathcal{R}(\cdot)$  is the operator retaining the real-valued elements of a matrix.

The process of computing the STFT, involves selecting two transformation parameters. The transformation parameters are the number of basis functions  $N$  and the step-size of the segmentation. The number of basis functions  $N$  affects the resolution of the transform, i.e., how many sinusoidal functions are used to describe the signal. On the other hand, the step-size controls the time resolution of the STFT, i.e., how many time-frames are used to describe the variation of the signal with respect to time. Both hyper-parameters are important as there is always a trade-off between favoring frequency analysis over time. In music source separation, it is more common to favor frequency resolution over high time resolution, as the sources can be distinguished better and therefore are easier to separate, up to a certain extent [30]. To obtain more accurate analysis estimates, with respect to frequency, signal up-sampling techniques, such as zero-padding, can be used [18].

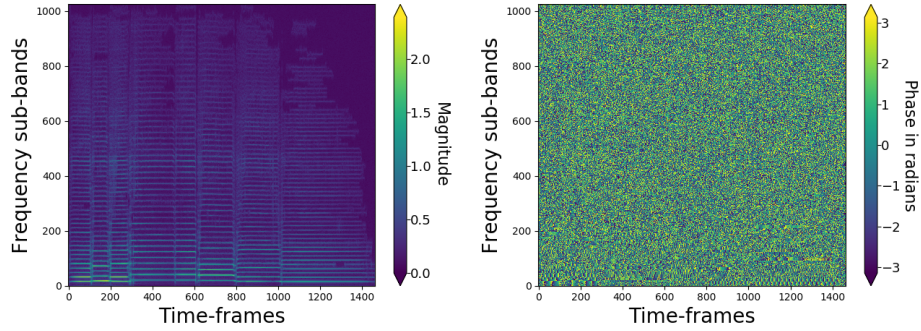
Focusing on the complex-valued representation computed using the STFT (Eq. (2.11)), i.e.,  $\mathbf{Z} \in \mathbb{C}^{N \times M}$ , the magnitude  $|\mathbf{Z}| \in \mathbb{R}_{\geq 0}^{N \times M}$  and the phase  $\mathbf{Z}_{\angle} \in \mathbb{R}^{N \times M}$  information is

computed using

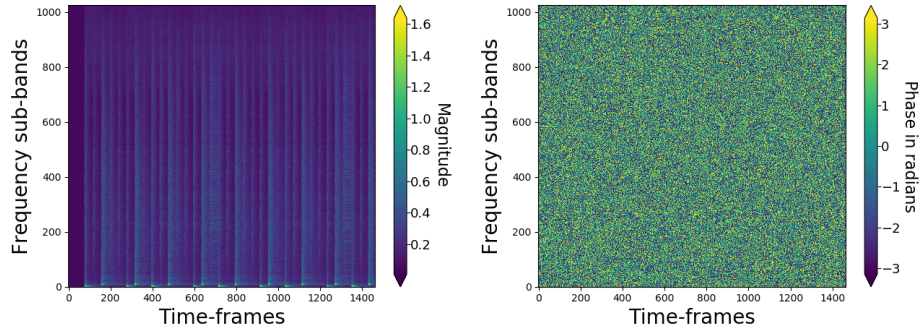
$$|\mathbf{Z}| = \sqrt{\mathcal{R}(\mathbf{Z})^{\odot 2} + \mathcal{I}(\mathbf{Z})^{\odot 2}} \quad (2.13)$$

$$\mathbf{Z}_{\angle} = \arctan\left(\frac{\mathcal{I}(\mathbf{Z})}{\mathcal{R}(\mathbf{Z})}\right). \quad (2.14)$$

In Eqs.(2.13)–(2.14),  $\mathcal{I}(\cdot)$  is the operator that retains only the imaginary values from the vector/matrix, and  $\arctan$  in Eq. (2.14) is the arc-tangent function. The basis functions used in the DFT and the STFT are conjugate symmetric between positive (the lower half of the DFT sub-band indices) and negative frequencies (the upper half of the DFT sub-band indices). Consequently, the representation  $\mathbf{Z}$  of a real-valued signal is redundant.



(a) Magnitude (*left*) and phase (*right*) spectrogram representations of a cello recording.



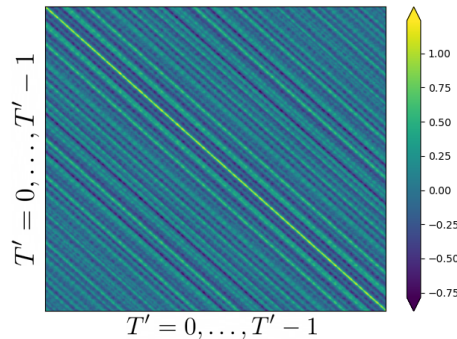
(b) Magnitude (*left*) and phase (*right*) spectrogram representations of a drum-set recording.

Figure 2.3: Magnitude and phase spectrogram representations of a cello recording (a) and a drum-set recording (b). The STFT is computed using a window size of 2048 samples, a Hamming windowing function, and step size of 256 samples. The redundant information of the negative frequency sub-bands is omitted.

This particular redundancy of the DFT and the STFT and the necessity of processing complex-valued representations is sometimes not desired in applications such as (sparse) audio coding [31]. For such applications, real-valued transforms are used [31]. Real-valued transforms employ only cosine functions. Examples of real-valued transforms include the

discrete cosine transform (DCT) and variants, such as the modified discrete cosine transform (MDCT), and modulated transforms, such as the pseudo-quadrature mirror filterbanks (PQMF) [32]. Although there is a great variety of real-valued transforms, music source separation research has mostly focused on using the STFT [A9]. An explanation to this is that the magnitude information computed using the STFT conveys the essential information of the underlying structure of the music sources and exhibits temporal attributes, such as the slow time-variance for sustained music signals that can be used for separation [11], [33]–[35]. An illustration of the magnitude and phase spectrogram representations of a cello and a drum-set recording is given in Figure 2.3.

From a general point of view, all the above described transforms that are based on sinusoidal functions rely on an assumption of some sort of optimality with respect to the coordinate space used to describe audio signals. In the case of sinusoidal functions, the assumption is that sinusoids form the eigenvectors of the normalized auto-correlation (covariance) matrix of the audio signals. However, this can be true only if the covariance matrix of the audio signals to be transformed, i.e., the covariance of the matrix  $\mathbf{X}$ , is a circulant matrix<sup>6</sup> as the (auto) covariance of a stationary first-order Markov process [36], [37]. That is because sinusoidal functions are the solutions to the matrix diagonalization problem of a circulant matrix [37]. Therefore, sinusoidal functions serve as characteristic functions for forming the coordinate space of a linear transformation. Nonetheless, the covariance matrix of music signals is usually not strictly circulant, and consequently sinusoidal functions do not form the ideal transform, even if they allow perfect reconstruction of the signal(s). An illustration of the covariance matrix of a music signal recording is given in Figure 2.4. The covariance matrix is computed using a cello recording of 8



(a) Covariance matrix of a cello recording.

Figure 2.4: An illustration of the covariance matrix of a cello recording.

seconds long and segmented using Eq. (2.3) for  $T' = 2048 (\sim 50\text{ms})$ .

In Figure 2.4, it can be observed that the non-negative entries of the covariance matrix exhibit strong correlations between non-neighbouring samples of an audio signal. That is in contrast to the structure of circulant covariance matrices suggesting that only adjacent samples are correlated. This in turn, makes the assumption of a first-order Markov process

<sup>6</sup>A circulant matrix is a special case of a Toeplitz matrix. Covariance matrices that are Toeplitz, suggest that the underlying signals are WSS.

with a circulant covariance matrix only a fair proxy to the empirical statistics of audio signals. It also raises the suspicion that sinusoidal functions might not be the best choice for audio signal transforms, even if these transforms offer an intuitive and informative way to process audio and music signals.

Considering the above, the learning of audio signal transforms is an emerging research direction. Learned audio signal transforms can be computed using a variety of algebraic methods including the Karhunen-Lo  ve transform, that is similar to singular value decomposition (SVD) and principal component analysis (PCA). Another approach towards learning audio signal transforms is artificial neural networks (ANNs). The latter approach allows many more degrees of freedom in learning signal transforms. The learning of the transforms can be performed using the framework of energy-based learning described in Section 2.2.2. For example, the signal transformations can be non-linear, enabling the modelling of more complicated signal structures. Also, the signal transform can be hierarchical, i.e., a composition of different and cascaded transforms, enabling the discovery of more abstract descriptions of audio signals [1].

### 2.2.4 Artificial Neural Networks

ANNs have been initially developed as a mathematical model of biological brains, and particularly the way that information is processed by biological neurons [38], [39]. Although ANNs are a parsimonious model of biological neurons, they have received a lot of attention in research communities, due to their good performance in the hierarchical processing of information using a series of cascaded operators [40]–[42]. The hierarchical processing of information is useful in problems that can be formally expressed as classification, pattern recognition, or regression problems [1]. The fundamental structure of an ANN consists of a set (a network) of nodes that are connected. The connection between the nodes is determined by a set of weights, that are adapted towards minimizing the provided loss function using signal observations (data points). Over the last decades many types of ANNs have been proposed. The most well established and used ones in music source separation are (deep) feed-forward neural networks (FNNs)<sup>7</sup> [38], [42], convolutional neural networks (CNNs) [43], and recurrent neural networks (RNNs) with extensions such as the long short-term memory (LSTM) networks, and gated recurrent units (GRUs) [44].

#### Feed-forward Neural Networks

FNNs can be seen as a learned function that aims at mapping an input vector  $\mathbf{x} \in \mathbb{R}^N$  to an output target vector  $\mathbf{y} \in \mathbb{R}^{N_o}$ . In the context of music source separation,  $\mathbf{x}$  is often the mixture signal and  $\mathbf{y}$  is the target source signal. The mapping function  $\mathbf{x} \rightarrow \mathbf{y}$  is mainly parameterized by weights that establish the connection between multiple nodes in the network. Towards computing  $\mathbf{y}$ , FNNs use multiple functions in series and in practice FNNs yield an approximation of  $\mathbf{y}$  that is denoted by  $\hat{\mathbf{y}} \in \mathbb{R}^{N_o}$ . The output of each one of those functions is denoted as the *hidden* or *latent* representation of the input to that function.

---

<sup>7</sup>FNNs are also known as the multi-layer perceptron, when the output of the (multi-layer) FNN corresponds to the probability of the input belonging to a certain class, event or pattern [1].

Non-linear FNNs that compute a single latent representation of sufficiently high dimensionality, i.e., equal to the intrinsic dimension of the input data, can approximate any continuous function<sup>8</sup> [45]. If the network uses two or more latent representations for computing  $\hat{\mathbf{y}}$ , the neural network is referred to as *deep*; otherwise it is referred to as *shallow*. Additional computations of latent representations extend the approximation capabilities of the FNN. However, in many applications the choice of the dimensionality for each latent representation and the number of computed representations remains a heuristic choice that is verified empirically [1].

More formally, using a deep FNN to compute  $L$  latent representations and the output  $\hat{\mathbf{y}}$ , let  $\mathbf{W}^{(l)} \in \mathbb{R}^{N_l \times N_{l-1}}$  denote the  $l$ -th weights matrix used to compute  $l$ -th latent representation  $\mathbf{z}^{(l)} \in \mathbb{R}^{N_l}$ . The dimensionality of the previously computed representation is denoted by  $N_{l-1}$  and  $N_l$  is the dimensionality of the succeeding representation. The representations are computed using a series of vector-matrix products as

$$\tilde{\mathbf{z}}^{(l)} = \mathbf{W}^{(l)} \mathbf{z}^{(l-1)} + \mathbf{b}^{(l)} , \quad (2.15)$$

and the application of an activation function  $g(\cdot)$

$$\mathbf{z}^{(l)} = g^{(l)}(\tilde{\mathbf{z}}^{(l)}) . \quad (2.16)$$

The activation function  $g^{(l)}(\cdot)$  is an element-wise non-linear function that is used to compute the  $l$ -th representation and  $\mathbf{b}^{(l)} \in \mathbb{R}^{N_l}$  is the bias vector that allows shifting the output of the vector-matrix product prior to the application of the non-linear function. For  $l = 0$  in Eq. (2.15), it is assumed that  $\mathbf{z}^{(0)} = \mathbf{x}$  and  $N_{(l=0)} = N$ . The output of the FNN is then computed by another vector-matrix product followed by the application of a non-linear function as

$$\tilde{\mathbf{y}} = \mathbf{W}^{(o)} \mathbf{z}^{(L)} + \mathbf{b}^{(o)} \quad (2.17)$$

$$\hat{\mathbf{y}} = g^{(o)}(\tilde{\mathbf{y}}) \quad (2.18)$$

In Eqs. (2.17), (2.18) the superscript “ $(o)$ ” is used to denote the output weights and biases, that are different from the weights and biases used to computed the previous representation.

Common choices for the element-wise and non-linear activation functions  $g(\cdot)$  are the hyperbolic tangent  $\tanh(\cdot)$  function defined as

$$\tanh(x) = \frac{\exp(2x) - 1}{\exp(2x) + 1} , \quad (2.19)$$

the (logistic) sigmoid  $\sigma(\cdot)$  function defined as

$$\sigma(x) = \frac{1}{1 + \exp(-x)} , \quad (2.20)$$

and the rectified linear unit (ReLU) [46] function defined as

$$\text{ReLU}(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise} . \end{cases} \quad (2.21)$$

---

<sup>8</sup>The approximation can be performed to arbitrary precision, given that the input and output domains are defined on a compact numerical set.

Using Eqs. (2.19)–(2.21) to compute the latent representation(s) and the output  $\hat{\mathbf{y}}$ , makes the FNN non-linear and flexible to find non-linear classification boundaries or approximate complex-structured functions [1].

In order for the FNN to yield reliable predictions  $\hat{\mathbf{y}}$ , i.e., the predictions are similar to the target  $\mathbf{y}$  subject to a given metric, the parameters of the FNN are updated. The updates are performed in an iterative way during the training process. The goal of the training process is to estimate the parameters, i.e., the weights and biases, of the FNN so that the difference between the predicted  $\hat{\mathbf{y}}$  and targeted output  $\mathbf{y}$  is minimized. Since the functions expressed in Eqs. (2.15)–(2.21), i.e., the vector-matrix products and the element-wise activation functions, are differentiable<sup>9</sup>, the parameters of the FNN can be optimized by using gradient descent. A well-known algorithm based on gradient descent is the *back-propagation* algorithm [42], [48]. Back-propagation uses the chain rule of differentiation to compute partial derivatives with respect to all parameters of the FNN. More formally, allow  $\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})$  be a *differentiable* loss function that yields the error E, i.e.,

$$E = \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) .$$

Then, the partial derivatives of the FNN parameters are calculated as

$$\frac{\partial E}{\partial \mathbf{W}^{(o)}} = \Delta \frac{\partial \tilde{\mathbf{y}}}{\partial \mathbf{W}^{(o)}} \quad (2.22)$$

$$\frac{\partial E}{\partial \mathbf{b}^{(o)}} = \Delta \frac{\partial \tilde{\mathbf{y}}}{\partial \mathbf{b}^{(o)}} , \quad (2.23)$$

where

$$\Delta := \frac{\partial E}{\partial \tilde{\mathbf{y}}} = \frac{\partial E}{\partial \hat{\mathbf{y}}} \frac{\partial \hat{\mathbf{y}}}{\partial \tilde{\mathbf{y}}} , \quad (2.24)$$

and  $\frac{\partial \hat{\mathbf{y}}}{\partial \tilde{\mathbf{y}}}$  is the first derivative of the selected activation function  $g^{(o)}(\cdot)$  with respect to  $\tilde{\mathbf{y}}$ . In the above notation, the functional dependence of E and  $\mathcal{L}(\cdot)$  is used for convenience and it admits  $\frac{\partial E}{\partial \tilde{\mathbf{y}}} := \frac{\partial \mathcal{L}(\mathbf{y}, \hat{\mathbf{y}})}{\partial \tilde{\mathbf{y}}}$ . From Eq. (2.15) and Eq. (2.17) it follows that the partial derivative with respect to the bias vector(s) is the all-ones vector  $\mathbf{1}_{N_{l-1}}$ . Similar to the above, the partial derivatives with respect to the parameters that compute the  $l$ -th latent representation, i.e.,  $\mathbf{W}^{(l)}$  and  $\mathbf{b}^{(l)}$  are computed<sup>10</sup>as

$$\frac{\partial E}{\partial \mathbf{W}^{(l)}} = \Delta \frac{\partial \tilde{\mathbf{y}}}{\partial \mathbf{z}^{(L)}} \frac{\partial \mathbf{z}^{(L)}}{\partial \tilde{\mathbf{z}}^{(L)}} \cdots \frac{\partial \tilde{\mathbf{z}}^{(l+1)}}{\partial \mathbf{z}^{(l)}} \frac{\partial \mathbf{z}^{(l)}}{\partial \tilde{\mathbf{z}}^{(l)}} \frac{\partial \tilde{\mathbf{z}}^{(l)}}{\partial \mathbf{W}^{(l)}} \quad (2.25)$$

$$\frac{\partial E}{\partial \mathbf{b}^{(l)}} = \Delta \frac{\partial \tilde{\mathbf{y}}}{\partial \mathbf{z}^{(L)}} \frac{\partial \mathbf{z}^{(L)}}{\partial \tilde{\mathbf{z}}^{(L)}} \cdots \frac{\partial \tilde{\mathbf{z}}^{(l+1)}}{\partial \mathbf{z}^{(l)}} \frac{\partial \mathbf{z}^{(l)}}{\partial \tilde{\mathbf{z}}^{(l)}} \frac{\partial \tilde{\mathbf{z}}^{(l)}}{\partial \mathbf{b}^{(l)}} , \quad (2.26)$$

for  $l > 0$ . After the computation of the partial derivatives, the parameters  $(\mathbf{W}^{(l)}, \mathbf{b}^{(l)})$ ,

---

<sup>9</sup>The gradients for the ReLU function (Eq. (2.21)) can be calculated using a subgradient method. For  $x = 0$  an intended gradient [47] value of 0 is used, following the work presented in [46].

<sup>10</sup>In software packages for automatic differentiation, such as the one used in [49], the vectors and matrices of Eq. (2.25) and Eq. (2.26) are not computed anew for every layer. The above equations are used as an example of the involved calculations.



$\mathbf{W}^{(o)}$ , and  $\mathbf{b}^{(o)}$  are updated using:

$$\mathbf{W}_{u+1}^{(*)} = \mathbf{W}_u^{(*)} - \eta \frac{\partial E}{\partial \mathbf{W}_u^{(*)}} \quad (2.27)$$

$$\mathbf{b}_{u+1}^{(*)} = \mathbf{b}_u^{(*)} - \eta \frac{\partial E}{\partial \mathbf{b}_u^{(*)}} , \quad (2.28)$$

where  $u > 0$  is the iteration index (gradient step), and  $\eta$  is a scalar known as the learning rate. The asterisk “ $(*)$ ” is used to replace the superscript identifiers “ $(l)$ ” and “ $(o)$ ” in order to denote the applicability of Eq. (2.27) and Eq. (2.28) to all parameters of the network.

Recent works in deep learning involve solvers that re-calculate the evaluated partial derivatives used in Eq. (2.27) and Eq. (2.28). The re-calculation is based on higher order moments (aggregating information of past gradient steps) of the partial derivatives in order to adaptively weight the values of the partial derivatives. A solver that is extensively used in this thesis is the *Adam* algorithm [50], [51]. Adam is a popular choice due to its convergence performance in stochastic optimization, where multiple update steps  $u$  are performed using data-points from a given data-set. The pseudocode for Adam is given in Appendix B.

### Recurrent Neural Networks

Data such as audio signals exhibit strong temporal patterns. At a given time instance an audio signal exhibits strong dependencies on past time instances. An important type of ANNs that takes into account those dependencies in time is the RNN. Unlike FNNs, RNNs also process past information of the input data. Specifically, instead of mapping input to output vectors, as in the case of the FNNs, RNNs in principle map from the previously observed input vectors to each output. More formally, let  $\mathbf{X} \in \mathbb{R}^{T \times N}$  be a matrix that consists of a sequence of  $T$  vectors, and each vector  $\mathbf{x}_t \in \mathbb{R}^N$  has a size  $N$

$$\mathbf{X} = [\mathbf{x}_0, \dots, \mathbf{x}_{T-1}] .$$

In many music source separation applications, the matrix  $\mathbf{X}$  can be thought of as a sequence of time-frames obtained from segmenting audio signals, as in the segmentation process by using Eq. (2.3). It should be noted that the RNN is not restricted to the usage of the above described organization of time-frames in  $\mathbf{X}$ , but instead refers to a rather general process of sequential information [52]. Using the RNN, without bias vectors, the latent representation  $\mathbf{z}_t^{(h)} \in \mathbb{R}^{N_h}$  at each time-step  $t$  is computed by recursively applying:

$$\tilde{\mathbf{z}}_t^{(h)} = \mathbf{W}^{(i-h)} \mathbf{x}_t + \mathbf{W}^{(h-h)} \mathbf{z}_{t-1}^{(h)} \quad (2.29)$$

$$\mathbf{z}_t^{(h)} = g^{(h)}(\tilde{\mathbf{z}}_t^{(h)}) . \quad (2.30)$$

In Eq. (2.29),  $\mathbf{W}^{(i-h)} \in \mathbb{R}^{N_h \times N}$  and  $\mathbf{W}^{(h-h)} \in \mathbb{R}^{N_h \times N_h}$  are the weights of the RNN that compute the input-to-hidden (i-h) and hidden-to-hidden (h-h) representations, respectively. The dimensionality of the latent representation is  $N_h$ . Furthermore, it is assumed that for the initial time-step  $t = 0$ ,  $\mathbf{z}_{t-1}^{(h)}$  is the null vector. The output of the RNN

$\hat{\mathbf{y}}_t \in \mathbb{R}^{N_o}$  at each time-step  $t$ , is computed using

$$\hat{\mathbf{y}}_t = \mathbf{W}^{(h-o)} \mathbf{z}_t^{(h)}, \quad (2.31)$$

where  $\mathbf{W}^{(h-o)} \in \mathbb{R}^{N_o \times N_h}$  is the hidden-to-output weight matrix, and  $N_o$  is the dimensionality of the output vector.

For computing the derivatives with respect to the parameters of the RNN,  $\mathbf{W}^{(i-h)}$ ,  $\mathbf{W}^{(h-h)}$ , and  $\mathbf{W}^{(h-o)}$ , the back-propagation algorithm can be used. However, for calculating the partial derivatives of the RNN the back-propagation algorithm has to consider all the time-steps involved in the computation of the latent representation and output [52], [53]. This is commonly referred to as back-propagation through time (BPTT) and is introduced in [54]. By letting  $\mathbf{Y} \in \mathbb{R}^{T \times N_o}$  and  $\hat{\mathbf{Y}} \in \mathbb{R}^{T \times N_o}$  be the matrices that contain the sequences of the target  $\mathbf{y}_t$  and predicted  $\hat{\mathbf{y}}_t$  vectors, respectively, and

$$E_t = \mathcal{L}(\mathbf{y}_t, \hat{\mathbf{y}}_t)$$

be the outcome of a differentiable loss function at time-step  $t$ , the derivative of the total sum of errors  $E$  with respect to  $\mathbf{W}^{(h-o)}$  is calculated as

$$\frac{\partial E}{\partial \mathbf{W}^{(h-o)}} = \sum_{t=0}^{T-1} \frac{\partial E_t}{\partial \hat{\mathbf{y}}_t} \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{W}^{(h-o)}}. \quad (2.32)$$

For calculating the derivatives of  $E$  with respect to  $\mathbf{W}^{(h-h)}$  and with respect to  $\mathbf{W}^{(i-h)}$ , let  $\Theta^{(h)}$  be a matrix used to denote either  $\mathbf{W}^{(h-h)}$  or  $\mathbf{W}^{(i-h)}$ . Then, the derivative of  $E$  with respect to  $\Theta^{(h)}$  can be calculated by summing the derivatives for all time-steps  $T$

$$\frac{\partial E}{\partial \Theta^{(h)}} = \sum_{t=0}^{T-1} \frac{\partial E_t}{\partial \Theta^{(h)}}. \quad (2.33)$$

Then, each derivative at time-step  $t$  is calculated using the temporal contribution of all the previous time-steps as

$$\frac{\partial E_t}{\partial \Theta^{(h)}} = \sum_{t'=0}^t \frac{\partial E_t}{\partial \hat{\mathbf{y}}_t} \frac{\partial \hat{\mathbf{y}}_t}{\partial \mathbf{z}_t} \frac{\partial \mathbf{z}_t}{\partial \mathbf{z}_{t'}} \frac{\partial^+ \mathbf{z}_{t'}}{\partial \Theta}, \quad (2.34)$$

where  $\frac{\partial^+ \mathbf{z}_{t'}}{\partial \Theta}$  is the instantaneous partial derivative. The instantaneous partial derivative considers the time-step prior to  $t'$  as a constant, so that  $\frac{\partial^+ \mathbf{z}_{t'}}{\partial \Theta}$  computes only the derivative of  $\mathbf{z}_{t'}$  with respect to  $\Theta$  at time-step  $t'$ . As it can be seen from Eq. (2.29), each  $\mathbf{z}_t$  is dependent on its previous state  $\mathbf{z}_{t-1}$ . Therefore,  $\frac{\partial \mathbf{z}_t}{\partial \mathbf{z}_{t'}}$  has to consider the dependencies on the previous state and is calculated as

$$\frac{\partial \mathbf{z}_t}{\partial \mathbf{z}_{t'}} = \prod_{t''=t'+1}^t \frac{\partial \mathbf{z}_{t''}}{\partial \mathbf{z}_{t''-1}}, \quad (2.35)$$

where  $t''$  is an index introduced to compute the products up to time-step  $t$ . Then, the update of the RNN parameters,  $\mathbf{W}^{(h-o)}$ ,  $\mathbf{W}^{(h-h)}$ , and  $\mathbf{W}^{(i-h)}$ , can be performed as previously shown in Eq. (2.27).

Eqs. (2.33)–(2.34) suggest that computing derivatives with respect to the parameters of the RNNs involves computing sum-of-products for each time-step  $t$ . By inspecting Eq. (2.34) closer and considering  $\mathbf{W}^{(h-h)}$  as the parameter to be optimized, it can be seen that a recursive product is computed between  $\mathbf{W}^{(h-h)}$  and the first derivative of the employed non-linear function  $g^{(h)}(\cdot)$  with respect to the state  $\mathbf{z}_{t-1}$ . The recursive computation of the product is problematic in many cases, as for long sequences the derivatives can explode or vanish [52], [53], [55], meaning that the norm of the calculated gradients goes to zero or reaches exceedingly high values. Consequently, training RNNs becomes extremely difficult [52], [53], [56]. To alleviate this issue, the LSTM architecture is proposed in [57].

The LSTM architecture mitigates the exploding or vanishing gradients problem, by using memory and gating units. Memory units are linear functions for recurrent self-connectivity. Gating units control the flow of information from and to the memory unit. These two units allow the network to better control the gradient values at each time-step. The control of the gradient values is performed by using suitable parameter updates that are computed by the gating units. The capabilities of the LSTM architecture to solve problems where long temporal dependencies have to be modelled are demonstrated in [52]. A less computationally demanding alternative to the LSTM architecture, that is based on the gating mechanisms of the LSTM architecture, is the GRU architecture presented in [44]. The GRU architecture has comparable performance to the LSTM [58] in modelling long temporal dependencies of the data, but requires computing fewer gating mechanisms than the LSTM. Due to the comparable performance and the reduction in the number of parameters, the GRU is used in this thesis.

More specifically, the GRU architecture employs two gating units at each time-step  $t$ . The gating units are the update gate  $\mathbf{h}_t \in \mathbb{R}_{[0,1]}^{N_h}$ , that controls the information fed into the memory, and the forget gate  $\mathbf{r}_t \in \mathbb{R}_{[0,1]}^{N_h}$ , that controls the information flowing out of the memory of the GRU. In more details, the representation<sup>11</sup>  $\mathbf{z}_t \in \mathbb{R}_{[-1,1]}^{N_h}$  at time-step  $t$  is computed as

$$\mathbf{z}_t = (1 - \mathbf{h}_t) \odot \mathbf{z}_{t-1} + \mathbf{h}_t \odot \tilde{\mathbf{z}}_t, \quad (2.36)$$

where the update gate  $\mathbf{h}_t$  is computed as

$$\mathbf{h}_t = \sigma(\mathbf{W}^{(i-h)}\mathbf{x}_t + \mathbf{W}^{(z-h)}\mathbf{z}_{t-1}), \quad (2.37)$$

and  $\tilde{\mathbf{z}}_t \in \mathbb{R}_{[-1,1]}^{N_h}$  is the *candidate* hidden state. The candidate state is processed by the application of the forget gate  $\mathbf{r}_t$  at time-step  $t$ , and is computed using

$$\tilde{\mathbf{z}}_t = \tanh(\mathbf{W}^{(i-z)}\mathbf{x}_t + \mathbf{r}_t \odot (\mathbf{W}^{(h-h)}\mathbf{z}_{t-1})). \quad (2.38)$$

Practically, Eq. (2.38) for  $\mathbf{r}_t = \mathbf{1}_{N_h}$  and  $\forall t \in [0, 1, \dots, T-1]$  boils down to the conventional RNN, from Eq. (2.29) and Eq. (2.30), that uses the  $\tanh(\cdot)$  as a non-linear function. To this end, the forget gate  $\mathbf{r}_t$  is computed at time-step  $t$  using

$$\mathbf{r}_t = \sigma(\mathbf{W}^{(i-r)}\mathbf{x}_t + \mathbf{W}^{(z-r)}\mathbf{z}_{t-1}). \quad (2.39)$$

---

<sup>11</sup>Compared to the paper introducing the GRU architecture [44], in this section the used notation between the latent representation  $\mathbf{z}$  and the update gate  $\mathbf{h}$  has been swapped, in order to preserve consistency in the notation from previous sections, regarding the latent representations.

The weight parameters of the GRU that are optimized, are  $\mathbf{W}^{(i-h)}, \mathbf{W}^{(i-z)}, \mathbf{W}^{(i-r)} \in \mathbb{R}^{N_h \times N}$ , and  $\mathbf{W}^{(z-h)}, \mathbf{W}^{(h-h)}, \mathbf{W}^{(z-r)} \in \mathbb{R}^{N_h \times N_h}$ . Optionally, three bias vectors  $\mathbf{b}^{(h)}$ ,  $\mathbf{b}^{(z)}$ , and  $\mathbf{b}^{(r)}$  can be added to Eq. (2.37), Eq. (2.38), and Eq. (2.39), respectively, prior to the application of each respective non-linear function. The bias vectors are omitted from the above described equations for brevity. By using the representation computed by the GRU, an additional FNN can be applied to obtain the outputs of the GRU at each time-step, as previously done in Eq. (2.31). Then, by providing a differentiable loss function  $\mathcal{L}(\cdot)$ , the previously mentioned parameters of the GRU can be optimized using the BPTT algorithm.

All previously mentioned RNN architectures process the input sequence using a direction that starts from  $t = 0$  and advances to  $t = T - 1$ , that is the termination step of the recursion. This effectively models data dependencies, at a current time-step  $t$ , using only past information. However, information from future time-steps can be also useful in various applications of sequence modelling [52], [59]. To that aim, RNNs can be extended to process sequences using two sequence directions rather than only one. In that case, the RNN is denoted as bi-directional. To do so, two RNNs with different parameters are used to compute the time-dependent latent information. The first RNN accepts as input the input sequence  $\mathbf{X}$  and processes it according to the previously mentioned recursive equations and the second RNN accepts as input the time-reversed version of the input sequence  $\mathbf{X}$ , that is commonly denoted as  $\overleftarrow{\mathbf{X}}$ . For clarity, the original sequence is also denoted as  $\overrightarrow{\mathbf{X}}$ . The latent information yielded from the two RNNs is concatenated, forming a new latent representation of higher dimensionality. This higher dimensional latent information contains richer information with respect to time-dependencies of the input data [52], [59].

### Convolutional Neural Networks

CNNs, introduced in [43], have been extremely popular in image processing and recognition, due to their performance in detecting patterns and shapes in images [60]. The architecture of the CNN is in a sense similar to the FNN as the CNN does not process temporal data recursively as in the case of RNNs. Instead, a convolution<sup>12</sup> operation is applied between the input signal and a set of kernels. The assumption behind the usage of CNNs is that the structure of the input signal(s) is characterized by spatial information, e.g., neighboring pixels of an image or adjacent time-samples of an audio signal. The spatial information is exploited by the CNNs by means of the convolution operation, resulting in learning representations that encode certain information about the structure of the input signal(s).

In more details, the application of a single one-dimensional (1D) CNN to the input signal  $\mathbf{x} \in \mathbb{R}^N$ , using a set of  $C$  kernels  $\mathbf{w}_c \in \mathbb{R}^{L_k}$ , is described as

$$Z_{[t,c]} = \sum_{l_k=0}^{L_k-1} x_{[St+l_k]} w_{c[l_k]} , \quad (2.40)$$

---

<sup>12</sup>The term convolution is used in a loose sense. From a signal processing point of view, a cross-correlation is computed between a signal and a set of kernels. However, kernels are often randomly initialized and the order reversal of the samples in each kernel is omitted in the notation.

where  $L_k$  is the length of the kernels,  $l_k \in [0, 1, \dots, L_k - 1]$  is the sample position index in each kernel  $\mathbf{w}_c$ , and  $S$  is the stride size that allows stepping over  $S$  samples in the signal  $\mathbf{x}$ . In Eq. (2.40),  $t$ ,  $0 < t \leq \lceil N/S \rceil$ , is an integer that indexes the temporal resolution of the outcome of the convolution.

The latent representation  $\mathbf{Z} \in \mathbb{R}^{T \times C}$  consists of  $C$  components and it can be further processed using CNNs. In this case, an additional set of  $N_o$  kernels is employed and the kernels are indexed by  $c'$ . Each kernel  $\mathbf{W}_{c'}^{(o)} \in \mathbb{R}^{L'_k \times C}$  is a matrix composed of  $C$  column vectors, one for each component in the representation  $\mathbf{Z} \in \mathbb{R}^{T \times C}$ . Also, the kernels have a temporal length  $L'_k$ , and each kernel computes the contribution of every  $c$  in the representation, to the output of the CNN,  $\hat{\mathbf{Y}} \in \mathbb{R}^{T' \times N_o}$  using

$$\hat{\mathbf{Y}}_{[t', c']} = \sum_{c=0}^{C-1} \sum_{l'_k=0}^{L'_k-1} \mathbf{Z}_{[S't' + \phi l'_k, c]} \mathbf{W}_{c'}^{(o)}_{[l'_k, c]} . \quad (2.41)$$

In Eq. (2.41),  $S'$  is the stride hyper-parameter of the second convolutional network, and  $\phi$  is the dilation factor [61]. The dilation factor is an optional hyper-parameter that allows each kernel  $\mathbf{W}_{c'}^{(o)}$  to aggregate information from proceeding time-frames, enabling each kernel to “be stretched” in the temporal domain of the representation by a factor  $\phi$ . The dilation can also be applied to Eq. (2.40) but it is neglected for clarity in the presentation of the CNNs. The parameters of the CNN can be optimized by employing a differentiable loss function  $\mathcal{L}(\cdot)$  and the target output  $\mathbf{Y} \in \mathbb{R}^{T' \times N_o}$ . Then, the back-propagation algorithm can be applied as in the case of the FNNs, with the only peculiarity that the sample order in each kernel is reversed when the partial derivatives are computed<sup>13</sup> [43].

To compute the temporal length of the output of any CNN, termed as  $T_o$ , the stride, the dilation factor, the kernel length, and the temporal length  $T_i$  of the signal or representation to be convolved are taken into account. Specifically, the output  $T_o$  is computed as

$$T_o = \lceil \frac{T_i - L_k - (L_k - 1)(\phi - 1) + 2P}{S} + 1 - 0.5 \rceil ,$$

where  $P$  is an integer that denotes the number of zeroes that are appended before and after the signal or representation to be convolved. In this thesis, all convolutional operations use the above formula to determine the zero-padding factor  $P$ . This is performed in order to maintain the expected time-resolution, dependent on the application.

### Network Initialization

All the above mentioned weight and bias parameters of the described neural networks are initialized and then optimized by gradient descent-based algorithms. In this thesis all considered bias vectors are initialized with zero values. For weight parameters, a random initialization procedure is performed. During this procedure, values are randomly drawn from a normal or a uniform distribution and scaled according to a selected scaling strategy. The scaling is performed because it plays a crucial role in gradient-based optimization, as the norm of the gradients can be dauntingly high leading to a degenerate training procedure. The strategies employed in this thesis are the one presented in [62] for the

---

<sup>13</sup>CNNs that use kernels whose sample ordered is reversed are also loosely denoted as transposed CNNs.

RNNs and/or GRUs, and the scaling strategy presented in [63] and in [64] for FNNs and for CNNs. Each selected distribution and strategy is given in the corresponding sections of each chapter that describe the followed experimental procedure.

### Practical Considerations

The previously presented computations of latent information or outputs by using DNNs, and the calculation of partial derivatives are computed using numerical software packages. These packages include automatic differentiation mechanisms to keep track of all the variables and the respective partial derivatives. In this thesis Pytorch [49] is used, due to its powerful automatic differentiation mechanism and its flexibility towards developing novel modules for deep learning-based research. Due to the high dimensionality of audio and music signals and the usage of a data-set that contains hours of audio material, all the computations presented in this thesis are performed using graphics processing units (GPUs). Specifically, all described experiments are using either an Nvidia Titan X or an Nvidia GTX 1050Ti GPU with single precision floating point operations.

## 2.3 Mixing Models & Separation by Masking

Apart from the models that aim at characterizing or representing individual signals, such as the ones presented in Section 2.2, mixing models aim at characterizing the function for mixing individual signals, i.e., the *mixing function*. Towards that aim, two mixing models are commonly assumed: the instantaneous (linear) and the convolutive mixing model. Each mixing model can be characterized as time-invariant or time-variant.

### 2.3.1 Instantaneous Mixing Model

The main assumption behind the instantaneous mixing model, is that the sources' signals have been processed only by amplitude scaling, prior to the computation of their mixture. More formally, let  $j \in [1, \dots, J]$  be an integer denoting the  $j$ -th source  $\mathbf{x}_j \in \mathbf{R}_{[-1,1]}^T$ ,  $c_h \in [1, 2]$  be the index of the channel of a signal, and  $\mathbf{x}_{c_h \text{ m}} \in \mathbf{R}_{[-1,1]}^T$  be the mixture of  $J \geq 2$  sources in channel  $c_h$ . The instantaneous mixing model is formally described as

$$\mathbf{x}_{c_h \text{ m}} = \sum_{j=1}^J \alpha_{c_h \text{ } j} \mathbf{x}_j$$

where  $\alpha_{c_h \text{ } j}$  is a scalar value, denoted as the *gain*, that modifies the amplitude of each source in channel  $c_h$ . The instantaneous mixing model can be further extended to take into account possible time fluctuations of the gain values. In this case, the instantaneous mixing model is characterized as time-variant and is formally expressed as

$$x_{c_h \text{ m}}[t] = \sum_{j=1}^J \alpha_{c_h \text{ } j}[q] x_j[t] ,$$

where  $q$  is an index that is computed as a function of  $t$  and is used to denote the slower, compared to  $t$ , time-variation of the gain values<sup>14</sup>. For the case of the instantaneous mixing model, the goal of source separation methods is to estimate the target sources  $\mathbf{x}_j$  from the observed mixture  $\mathbf{x}_m$ . Due to the simplicity of the instantaneous mixing model, it has been assumed in many unsupervised or blind source separation methods [4], [9].

### 2.3.2 Convolutional Mixing Model

The convolutional mixing model assumes that prior to the mixing of the sources, each source has been recorded in a room with particular acoustic characteristics (e.g., reverberation) or processed by a digital signal processing operation, such as a delay, a filter, or a reverberation effect. The previously mentioned assumptions are commonly modeled by the convolution of each source signal with a particular finite impulse response. More formally, let  $\mathbf{h}_{c_h j} \in \mathbb{R}^{T'}$  denote the finite impulse response that is applied to the  $j$ -th source and in channel  $c_h$ . Then, the convolutional mixing model is expressed as

$$\mathbf{x}_{c_h m}[t] = \sum_{j=1}^J \sum_{t'=0}^{T-1} \alpha_{c_h j} (\mathbf{x}_j[t-t'] \mathbf{h}_{c_h j}[t']) ,$$

where the sum in the convolution is assumed to be computed using indices  $t-t'$  for which the source signal is defined.

In addition to the above formula, the convolutional mixing model can be extended in its time-variant form. For the time-variant case, each source can be convolved by a set of  $Q'$  finite impulse responses  $\mathbf{H}_{c_h j} \in \mathbb{R}^{T' \times Q'}$  that depend on channel  $c_h$ . The difference compared to the time-invariant case, is that the  $j$ -th source  $\mathbf{x}_j$  is convolved with a different response, i.e., the  $q'$ -th finite impulse response of  $\mathbf{H}_{c_h j}$ , depending on the time instance  $t$ . The variation of the finite impulse response with respect to time, allows the convolutional model to describe moving sources, where the movement of the source imposes a change of the source's observed signal characteristics. Examples of signal characteristics include the delay of the source or the magnitude of individual frequency sub-bands, that are described by different finite impulse responses. The time-variant convolutional mixing model can be formally expressed as

$$\mathbf{x}_{c_h m}[t] = \sum_{j=1}^J \sum_{t'=0}^{T'-1} \alpha_{c_h j} (\mathbf{x}_j[t-t'] \mathbf{H}_{c_h j}[t', q']) ,$$

where  $q$  is an integer that denotes the usage of the  $q$ -th finite impulse response contained in  $\mathbf{H}_{c_h j}$ . The integer  $q$  is computed as a function of time  $t$ .

As the convolution operation is applied prior to the addition of the sources, the difficulty of separating the target source(s) is increased. To simplify this problem, audio signal transforms are used [9]. That is because signal transforms such as the STFT can be used to simplify the convolutions as element-wise multiplications, in the respective domain. However, the number of samples used for the STFT should be at least twice as

---

<sup>14</sup>The computation of  $q$  depends on the number of predetermined gain values at time indices  $t$ . Most of the times, these factors depend on artistic usage for simulating the position of source in the panoramic field composed by the two available channels [12], [65].

big as the length of the impulse responses [66], posing another benefit of using the STFT instead of real-valued transforms such as the MDCT. In the case of the convolutive mixing model, the goal of the separation methods is to recover the  $j$ -th target source after the application of the convolution and the gain. The target source signal after the previously described operations is referred to as the target source image, that essentially describes the contribution of a source to the mixture [9], [67].

### 2.3.3 Relaxed Mixing Model

Music mixtures available from various media streaming services are not the outcome of a simple sum of signals that have been either convolved with an impulse response or scaled by a gain value. In contrast to the previously introduced mixing models, the music sources in this case have undergone a production stage [65]. Specifically, each source is processed by a series of signal processing operations that include digital audio effects [65]. Many of the digital audio effects used for audio mixing are non-linear [65], [68]. Consequently, the assumption that the mixing function can be modelled by the convolutive mixing model falls short. The mixing model that yields the mixture of produced music sources can be formally expressed as

$$\mathbf{x}_{c_h m} = \sum_{j=1}^J \mathcal{G}_{c_h j}(\tilde{\mathbf{x}}_j) ,$$

where  $\mathcal{G}_{c_h j}(\cdot) : \mathbb{R}^T \mapsto \mathbb{R}^T$  is a function that applies a series of audio effects to the initial source signal  $\tilde{\mathbf{x}}_j$ . The function  $\mathcal{G}_{c_h j}$  can be different for each source and channel<sup>15</sup> and is applied to  $\tilde{\mathbf{x}}_j$  according to subjective criteria [65]. In nearly all realistic scenarios, the function  $\mathcal{G}_{c_h j}(\cdot)$  is unknown, i.e., there is no information regarding what audio effects have been applied to each source. Consequently, the previously described mixing models fall short since devising the inverse operation that estimates  $\tilde{\mathbf{x}}_j$  from  $\mathbf{x}_j$  is difficult and it is an open research problem. To mitigate this difficulty, the problem of music source separation is often relaxed. The relaxation follows the principle used in the convolutive mixing model and the goal is to separate the target source image  $\mathbf{x}_{c_h j}$  that includes the application of  $\mathcal{G}_{c_h j}(\cdot)$ , i.e.,

$$\mathbf{x}_{c_h j} = \mathcal{G}_{c_h j}(\tilde{\mathbf{x}}_j) .$$

In music production [65] or data-sets for music source separation [A8],  $\mathbf{x}_{c_h j}$  is often denoted as the source track.

The above relaxation, leads to the following (relaxed) mixing model

$$\mathbf{x}_{c_h m} = \sum_{j=1}^J \mathbf{x}_{c_h j} .$$

Although the relaxed mixing model may seem simpler than the instantaneous mixing model, presented in Section 2.3.1, the non-linear functions  $\mathcal{G}_{c_h j}(\cdot)$  applied to the corresponding sources increase the difficulty for separating the target source(s) from the

---

<sup>15</sup>The function  $\mathcal{G}_{c_h j}$  is different for each channel for expressing multi-channel audio effects. It should be noted that  $\mathcal{G}_{c_h j}$  can also consider inter-channel dependencies, but the explicit conditioning on channel information is dropped from the notation for brevity.



observed mixture. That is because each  $\mathcal{G}_{c_h j}(\cdot)$  is often composed by audio effects [65] that are often non-linear [68] and entangle the mixing process more than the application of gain values or the convolution with impulse responses. However, the relaxed mixing model does not consider additional (non-linear) signal processing operations applied to the mixture signal  $\mathbf{x}_{c_h m}$ , after the addition of the sources and the application of each  $\mathcal{G}_{c_h j}(\cdot)$ . The process of further modifying the mixture is commonly denoted as (audio) mastering [69], and is used most of the times in distributed music content. The process of audio mastering could potentially increase the difficulty of separating music sources.

### 2.3.4 Music Source Separation via Masking

For separating music sources from the observed single-channel (monaural) or two-channel (stereo) music mixtures, filtering is a widely used process. More specifically, a time-frequency representation of the monaural<sup>16</sup> mixture signal is computed using the STFT  $\mathcal{F}_{\text{STFT}}(\cdot)$  yielding the complex-valued matrix  $\mathbf{Y}_m \in \mathbb{C}^{N \times T'}$  of  $N$  frequency sub-bands and  $T'$  time-frames

$$\mathbf{Y}_m := \mathcal{F}_{\text{STFT}}(\mathbf{x}_m) .$$

Then, a source-specific filter, henceforth denoted as the *mask*, is applied to the mixture representation. The estimated time-domain signal of the target source  $j$  can be computed using the inverse short-time Fourier transform (ISTFT) and the filtered mixture representation as

$$\mathbf{x}_j \approx \hat{\mathbf{x}}_j = \mathcal{F}_{\text{ISTFT}}(\mathbf{Y}_m \odot \mathbf{M}_j^*) ,$$

where “ $\odot$ ” is the element-wise multiplication,  $\mathcal{F}_{\text{ISTFT}}(\cdot)$  denotes the ISTFT and  $\mathbf{M}_j^* \in \mathbb{R}^{N \times T'}$  is the computed mask.

There are multiple methods for computing source specific masks. All masking methods require that the time-frequency representation, most commonly the magnitude representation, of the target and interfering sources is estimated. In the last years, the most common way to estimate these spectrograms is by using deep learning [A9]. Deep learning-based approaches to music source separation are discussed in detail in Chapter 3 and Chapter 4. For simplicity, in this subsection it is assumed that the sources are provided by optimized DNNs, such as the ones described in Section 2.2.4. To that aim, let  $|\hat{\mathbf{Y}}_j| \in \mathbb{R}_{\geq 0}^{N \times T'}$  denote the estimated magnitude spectrogram of the  $j$ -th source.

Using the estimated magnitude representations, the first method computes a binary mask (BM),  $\mathbf{M}^{\text{BM}} \in \{0, 1\}^{N \times T'}$ , using the following fraction of magnitude information

$$\mathbf{M}_j^{\text{BM}} = g \left( |\hat{\mathbf{Y}}_j| \oslash \left( \sum_{j' \neq j} |\hat{\mathbf{Y}}_{j'}| \right) \right) . \quad (2.42)$$

In Eq. (2.42), “ $\oslash$ ” is the element-wise division and  $j'$  is an index that denotes only the interfering source(s) contained in the mixture and not the target source  $j$ . The element-wise function  $g(\cdot)$  is defined as

$$g(y) = \begin{cases} 1, & \text{if } y \geq 0.5 \\ 0, & \text{otherwise} \end{cases} .$$

---

<sup>16</sup>The focus is given on monaural signals since the most common masking operations operate on each individual channel.

The function  $g(\cdot)$  outputs a binary indicator of the dominance, with respect to the magnitude, of the target source in each frequency sub-band for every time-frame.

The application of BMs in music source separation leads in many cases to unwanted reconstruction artifacts, such as music distortion [70]. That is due to the abrupt changes of the mask values between adjacent time-frames of the representation. Nonetheless, computing BMs can yield some insight information regarding the overlap that the target and interfering sources have in the computed representation. Specifically, in [17] the (windowed) disjoint-orthogonality (W-DO) objective measure is introduced. The W-DO measure uses the computed BM to evaluate the orthogonality between target and interfering sources, and how well the target source can be separated, enabling the evaluation of representations subject to the task of music source separation [A5], [30]. The computation of the W-DO measure is given in the Appendix C.

In an attempt to alleviate the music distortions introduced by the application of the BM to the mixture signal, many approaches for music source separation consider soft-masking (SM) methods [A9]. An SM method allows the values of the computed mask to vary continuously between zero and one, instead of being binary. To that aim, the mask of the  $j$ -th source using a SM method,  $\mathbf{M}_j^{\text{SM}} \in \mathbb{R}_{[0,1]}^{N \times T'}$ , is computed as the ratio of  $\alpha$ -power magnitude spectrograms

$$\mathbf{M}_j^{\text{SM}} = |\hat{\mathbf{Y}}_j|^{\odot \alpha} \oslash \left( \sum_{j'=1}^J |\hat{\mathbf{Y}}_{j'}|^{\odot \alpha} \right). \quad (2.43)$$

In Eq. (2.43), “ $\alpha$ ” is the exponent that is applied element-wise to the magnitude spectrogram of each source. For  $\alpha = 2$ , Eq. (2.43) boils down to the generalized Wiener filtering [29], [70], [71]. Soft masks considered for separating a music source make the assumption that the  $\alpha$ -power magnitude spectrograms are *additive*, i.e., the sum of all  $|\hat{\mathbf{Y}}_j|^{\odot \alpha}$  is equal to the  $\alpha$ -power magnitude spectrogram of the observed mixture  $|\mathbf{Y}_m|^{\odot \alpha}$ . However, the assumption of source additivity is often violated, especially in real-world examples of music mixtures.

In an attempt to avoid the source additivity assumption, two alternative approaches are commonly considered. The first approach is to compute ideal amplitude masks (IAMs) defined as

$$\mathbf{M}_j^{\text{IAM}} = |\hat{\mathbf{Y}}_j| \oslash |\mathbf{Y}_m|, \quad (2.44)$$

commonly followed by a truncation operation [72]. The truncation is necessary, since the IAM is only optimal, i.e., it maximizes the source separation performance, when there are no phase differences between the mixture and the  $j$ -th source signals. However, this is often violated in real-world mixtures of music sources, and a truncation method that uses the phase difference information is presented in [72].

For the second approach, two strategies for SM have been proposed. The first strategy, presented in [29], replaces the magnitude spectrograms in the numerator and denominator of Eq. (2.43) by the computation of a divergence for each frequency sub-band and time-frame. Specifically, the numerator is replaced by the element-wise divergence between the target source and the observed mixture spectrogram. The denominator is replaced by the sum of all divergence output values between the interfering source(s) and the mixture. Depending on the selected divergence in [29], either the generalized Kullback-Leibler (KL) or the Itakura-Saito (IS) divergence, the numerator and denominator are

element-wise exponentiated either by  $\alpha = 1$  or  $\alpha = 2$ , respectively. The choice for the exponent  $\alpha$  is based on the statistical assumptions of the employed divergence, i.e., for the KL and IS, respectively [73], and of the  $\alpha$ -power spectrograms [70].

More recently, the work based on divergences [29] has been revisited in [70] from a statistical perspective that takes into account the (heavy tailed) distributions of magnitude spectrograms. Specifically, it is proposed in [70] to compute the soft-mask(s) using Eq. (2.43), but with the usage of an ideally optimal, with respect to additivity, exponent  $\alpha^o$ . More specifically, it is proposed to solve the following problem

$$\alpha^o = \underset{\alpha}{\operatorname{argmin}} |||\mathbf{Y}_m|^{\odot\alpha} - \sum_{j=1}^J |\hat{\mathbf{Y}}_j|^{\odot\alpha}|||_1^{1/\alpha}, \quad (2.45)$$

by using a grid-search in the interval  $(0, 2]$ . Although the strategy of finding optimal exponents for soft-mask computation improves music source separation performance [70], the assumption that the choice of the exponent is based on magnitude spectrogram distributions serves only as a proxy. That is because phase information determines the choice of the exponent, as suggested by the work presented in [74]. However, phase information of other music sources is in many cases unavailable, since each source has to be estimated accurately first and then a phase retrieval algorithm has to be applied. Consequently, the computation of soft-masks using source phase information, such as selecting the exponent for Eq. (2.43) based on phase distributions of the sources or the usage of phase-sensitive masks [72], is often impractical. In addition to this, the studies presented in [29], [70], [74] suggest directions for computing soft-masks, highlighting the fact that the mask computation should be subject to optimization, depending on the separation method.

It should be noted that all the masking methods discussed above can be used with the spectrogram information of the true sources, that can be obtained from data-sets [A8]. In this case,  $|\hat{\mathbf{Y}}_j|$  can be replaced by the true source magnitude spectrogram  $|\mathbf{Y}_j|$  in Eqs. (2.42)–(2.45). Furthermore, the mask computation above can be extended to multiple channels of music signals [75]. This is done by first computing the corresponding mask, most commonly using the generalized Wiener filtering, for the monaural estimate(s) of the target source(s). Then, the spatial covariance matrix of the multi-channel mixture signal is computed [75], [76] for each frequency sub-band. The spatial covariance matrix is used to compute the multi-channel mask that is applied to each channel of the mixture signal’s STFT, yielding a multi-channel estimate of the target source.

## 2.4 Source Separation Performance Evaluation

In order to assess the performance of music source separation approaches, research has also focused on the development of appropriate evaluation methodologies. These methodologies can be grouped into objective and subjective. Objective evaluation methodologies aim at a systematic assessment of source separation approaches, using objective metrics. This leads to a standardized evaluation procedure that can be adopted by music source separation research so that a comparison between separation approaches can be made. However, objective measures often cannot reflect the perceptual quality of audio and music signals [77]. To that aim, research has focused on the development of perceptually motivated objective measures for audio source separation evaluation [78] and routines for

the subjective evaluation of music source separation methods [79]. Subjective evaluation methodologies aim at qualitatively assessing the performance of separated music sources. The caveat of qualitative assessment is that it relies on listening tests that require multiple and often trained human participants. This makes qualitative assessment cumbersome for large-scale evaluation studies. It is important to note that both groups of methodologies require a data-set of isolated music sources and the corresponding mixtures. In this thesis, the employed data-set for evaluation is the MUSDB18 [A8] that consists of two-channel music mixtures.

### 2.4.1 Objective Assessment

The objective assessment of music source separation approaches is based on the computation of measures that are fractions of energies between computed components of the separated source. These measures are namely the signal-to-distortion ratio (SDR), the signal-to-interference ratio (SIR), and the signal-to-artifacts ratio (SAR), and are the standard measures for source separation evaluation [80], [81]. The computation of the measures is based on the BSSEval framework presented in [80]. More specifically, the evaluation metrics reported in this thesis are computed using the fourth version of the BSSEval framework, that is presented in [81] and originally proposed in [67]. That version uses the source images for computing the source components used in the evaluation, and considers two-channel (stereo) signals<sup>17</sup>.

More formally, let  $c_h \in [1, 2]$  be the index of the channel of a source or signal, then the  $j$ -th estimated source  $\hat{\mathbf{x}}_{c_h j}$  is modeled as the sum of the true source  $\mathbf{x}_{c_h j}$  and source specific error components  $\mathbf{e}_{c_h j}^{(*)} \in \mathbb{R}^T$

$$\hat{\mathbf{x}}_{c_h j} = \mathbf{x}_{c_h j} + \mathbf{e}_{c_h j}^{(\text{spat})} + \mathbf{e}_{c_h j}^{(\text{interf})} + \mathbf{e}_{c_h j}^{(\text{artif})}, \quad (2.46)$$

where  $\mathbf{e}_{c_h j}^{(\text{spat})}$ ,  $\mathbf{e}_{c_h j}^{(\text{interf})}$ , and  $\mathbf{e}_{c_h j}^{(\text{artif})}$  are the source error components for spatial, interference, and artifacts or distortions, respectively. The source error components are computed using all the true and estimated sources  $\mathbf{x}_{c_h j}$  and  $\hat{\mathbf{x}}_{c_h j}$ , respectively, by first calculating two types of least-squares projectors. The projectors are based on linear filtering. The first projector type is calculated for the target source  $\mathcal{P}_j(\cdot)$  and the second one is calculated for the interference(s)  $\mathcal{P}_{j''}(\cdot)$ . These two types of projectors<sup>18</sup> are source-specific and are computed across channels  $c_h$ . Essentially, the projection operators compute the error components as filtered versions of  $\mathbf{x}_{c_h j}$  by projecting the estimated sources onto the corresponding signal sub-spaces as

$$\mathbf{e}_{c_h j}^{(\text{spat})} := \mathcal{P}_j(\hat{\mathbf{x}}_{c_h j}) - \mathbf{x}_{c_h j} \quad (2.47)$$

$$\mathbf{e}_{c_h j}^{(\text{interf})} := \mathcal{P}_{j''}(\hat{\mathbf{x}}_{c_h j}) - \mathcal{P}_j(\hat{\mathbf{x}}_{c_h j}) \quad (2.48)$$

$$\mathbf{e}_{c_h j}^{(\text{artif})} := \hat{\mathbf{x}}_{c_h j} - \mathcal{P}_{j''}(\hat{\mathbf{x}}_{c_h j}). \quad (2.49)$$

Using the above error components, the SDR, SIR, and SAR metrics for the  $j$ -th source

---

<sup>17</sup>Two-channels are used due to the structure of the employed data-set [A8].

<sup>18</sup>In this context, the projectors are constructed by using a set of signals, e.g.,  $\{\mathbf{x}_{c_h 1}, \mathbf{x}_{c_h 2}, \dots, \mathbf{x}_{c_h J}\}$ .

are computed as

$$\text{SDR}_j = 10 \log_{10} \left( \frac{\sum_{c_h} \|\mathbf{x}_{c_h j}\|_2^2}{\sum_{c_h} \|\mathbf{e}_{c_h j}^{(\text{spat})} + \mathbf{e}_{c_h j}^{(\text{interf})} + \mathbf{e}_{c_h j}^{(\text{artif})}\|_2^2} \right) \quad (2.50)$$

$$\text{SIR}_j = 10 \log_{10} \left( \frac{\sum_{c_h} \|\mathbf{x}_{c_h j} + \mathbf{e}_{c_h j}^{(\text{spat})}\|_2^2}{\sum_{c_h} \|\mathbf{e}_{c_h j}^{(\text{interf})}\|_2^2} \right) \quad (2.51)$$

$$\text{SAR}_j = 10 \log_{10} \left( \frac{\sum_{c_h} \|\mathbf{x}_{c_h j} + \mathbf{e}_{c_h j}^{(\text{spat})} + \mathbf{e}_{c_h j}^{(\text{interf})}\|_2^2}{\sum_{c_h} \|\mathbf{e}_{c_h j}^{(\text{artif})}\|_2^2} \right). \quad (2.52)$$

The above metrics are expressed in dB (the higher the better), and are computed over segments of the respective signals. Following signal separation and evaluation campaign (SiSEC) rules [81], the length of the segments is 2 seconds with an overlap of half a second between segments. The overall reported value for each corresponding metric is most commonly the median or average value, across all segments within each track and then across the number of tracks in the data-set.

A limitation of the above metrics and especially the SDR, Eq. (2.50), which is the most used separation metric [A9], [81], is that the metrics are sensitive to scale perturbations [82]. For example, this can be seen by substituting Eq. (2.46) into Eq. (2.50) the SDR metric becomes the logarithmic-scaled signal-to-noise ratio (SNR), with the SNR being defined as

$$\text{SNR}_j = \frac{\|\mathbf{x}_j\|_2^2}{\|\mathbf{x}_j - \hat{\mathbf{x}}_j\|_2^2}. \quad (2.53)$$

The SNR metric (the lower the better) is sensitive to scale perturbations. Consequently, the above metrics do not fully reflect the actual separation performance of an approach, as the metrics could be further increased by potentially considering gain manipulations of the separated source<sup>19</sup> [82]. To tackle this problem, the scale-invariant versions of the metrics are presented in [82]. In this thesis, and particularly in Chapter 4, the non-scale-invariant metrics are used in order to preserve a consistent comparison to previously proposed approaches for music source separation. The scale-invariant SDR is used in Chapter 5.

## 2.4.2 Subjective Assessment

The subjective assessment of music source separation approaches focuses on the perceptual evaluation of the quality of the separated source(s). This is achieved by means of listening tests. In audio and music source separation research, listening tests are not that common, although studies suggest that listening tests should be highly considered [79]. To conduct a listening test, a standard procedure is followed, and there exist various standards to do so. A particularly useful, to music source separation, standard is the International Telecommunication Union (ITU) standard (ITU-R BS.1534-1) [84] denoted as multiple stimulus with hidden reference and anchors (MUSHRA). In this thesis, a web-based graphical user interface (GUI) for conducting the MUSHRA tests is used. The

---

<sup>19</sup>Metrics that are *variant* to scale could be also informative in music source separation evaluation. That is the case when the separated sources are used for (automated) re-purposing tasks [A2], [83].

implementation is based on the work presented in [85]. An illustration of the GUI used to conduct the listening test is given in Figure 2.5.

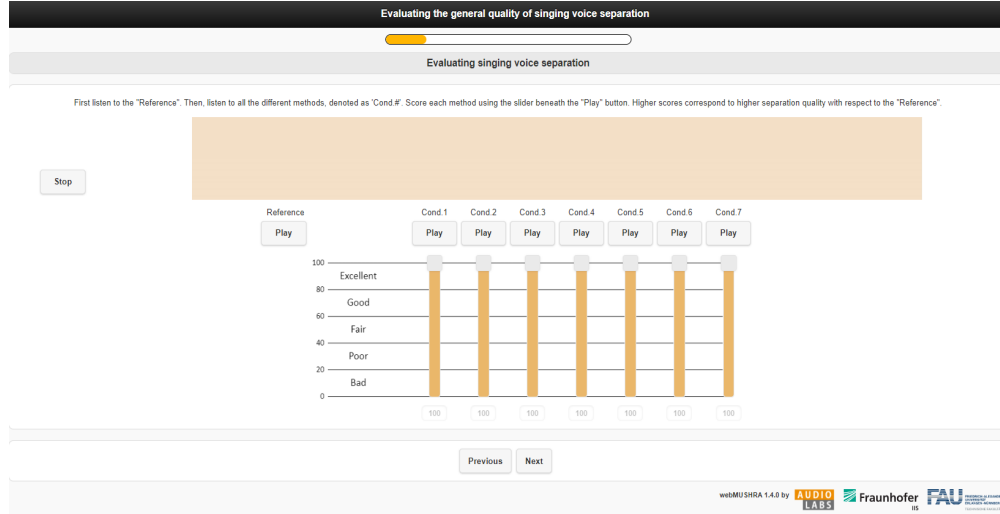


Figure 2.5: Illustration of the GUI that is used to conduct listening tests. The GUI relies on the framework presented in [85].

For evaluating the performance of music source separation approaches, listening tests based on MUSHRA qualitatively evaluate the degradation of the estimated music source(s). The evaluation is based on the score assigned by the listener. The score is given in comparison to the original signal of the target music source. According to the MUSHRA standard, a hidden anchor signal is additionally used during the listening tests. The anchor signal is a degraded version of the original music source, serving as the low-bound for the score. Particularly for audio and music source separation, the degradation used in MUSHRA tests is computed by either using the mixture signal directly as the hidden anchor, or more diligently, by synthetically generating source separation artifacts, interference, and music distortion. The method to synthesize the separation artifacts relies on the linear filtering operators [78], i.e., the Eqs. (2.47)–(2.49), that are used to decompose the estimated signal into artifacts, interference, and distortion components. These components are then added to the clean target source signal to compute the degraded version.

## Chapter 3

# Neural Network-Based Music Source Separation

### Preface

In this chapter, the focus is given on the separation of music sources by using (deep) neural networks. A particular neural network model that has led to several breakthroughs in music source separation and has been extended to that aim is the denoising autoencoder (DAE) model presented in [86], [87]. To examine the effectiveness of the DAE model and its published extensions in music source separation literature, this chapter formally describes the DAE model, including the corresponding extensions, and presents an algorithm for investigating what source separation models, based on the DAE, learn from spectral music data. The presented algorithm is denoted as the neural couplings algorithm (NCA) and the content of this chapter is based on [A22]. The corresponding source code is publicly available<sup>1</sup>.

### 3.1 Introduction

The separation of music sources using deep learning is an active research area that has attracted a lot of attention and many approaches have been proposed in related literature [A9], [81], [88]. A commonly used practice that has led to breakthroughs in music source separation performance, is the usage of the time-frequency representation of the mixture signal as input to the deep neural network (DNN) [A9], [81]. Depending on the target output signal, three different classes of approaches can be identified<sup>2</sup>. The

---

<sup>1</sup>Algorithm gist: [https://github.com/Js-Mim/nca\\_mss](https://github.com/Js-Mim/nca_mss), experimental code and trained models: <https://zenodo.org/record/2629650>

<sup>2</sup>It could be argued that there is a fourth class of approaches that is based on deep clustering [89]. However and according to the results presented in [89], deep clustering in music source separation relies on mask prediction to a great degree. Mask prediction approaches are the second class of approaches discussed here. Deep clustering approaches are discussed from a neural network perspective in Section 4.2.

approaches in the first class, referred to as *spectral approximation* methods, use the time-frequency representation of the target source as the target output of the DNN. This input-output relationship of signals follows the seminal work of the DAE model presented in [86], [87]. The DAE in [86], [87] is presented as a model for signal recovery from corrupted observations. It should be denoted that the DAE model, as originally defined in [86], is not restricted to symmetric encoding-decoding functions with dimensionality reduction, commonly referred to as bottleneck [86][Sec. 5]. Following the definition from [86], the term DAE refers to methods for signal recovery from corrupted signals.

For the second class of approaches, the target output signal is the pre-computed and source-dependent time-varying filter, i.e., the mask, that is used for filtering the mixture input signal [90]–[92]. The methods in this category will be referred to as *mask prediction* methods. As described in Section 2.3.4 of Chapter 2, the usage of pre-computed masks requires the information of at least two sources, the target and the interfering source(s) contained in the mixture [90], [91]. That is different from the first class of approaches that require only the time-frequency representation of the target source [90], [93]. Furthermore, the computation of masks imposes various assumptions regarding the additive properties of the sources [70], [74]. In practice, those assumptions do not hold true. Consequently, it is hard to define what the mask target should be. In some cases, the masks predicted by the DNN are sub-optimal and require additional DNN (re)training procedures [90]. These training procedures aim at improving the separation performance [90], suggesting that the mask computation should be subject to optimization.

Although the mask prediction approaches are not the central focus of this thesis, their relevance is that they inspired the third class of approaches. Conceptually, the third class combines spectral approximation and mask prediction, and are referred to in this thesis as skip-filtering connections. Specifically, these methods allow deep learning models to implicitly mask the input mixture signal by using the output of the deep learning model [A6], [94]–[96]. This operation yields a filtered version of the mixture signal, that serves as an estimate of the target source signal and is used to optimize the corresponding model [A6], [94], [96]. The optimization is performed using a signal reconstruction objective as done in the spectral approximation. In contrast to mask prediction methods [A6], [A10], [A11], [94], [96], methods based on skip-filtering connections do not require pre-computed masks. While the terminology of skip-filtering connections is used in the context of music source separation [A6], [A10], [A11], the speech enhancement and separation community often refers to methods in the third class as the signal approximation method [94], [97].

Subject to the estimation of the target source signal(s), the spectral approximation methods usually rely on an additional post-processing step using the generalized Wiener filtering [A9], [A10], [76], [81], [90], [93], [98], [99]. The usage of this additional post-processing is an empirical strategy to obtain target signals of better perceptual quality than the direct outputs of the DAEs [93], [98]. In contrast, approaches based on skip-filtering connections, which implicitly mask the mixture signal, yield competitive, yet not superior, results compared to the spectral approximation approaches, without the need of post-processing using the generalized Wiener filtering<sup>3</sup> [A11], [96]. Furthermore, approaches based on skip-filtering connections tend to result in better separation per-

---

<sup>3</sup>The performance of the approaches based on skip-filtering connections can be further improved by employing post-processing based on the generalized Wiener filtering [100].



formance than mask prediction approaches [A6], [A10]<sup>4</sup>. This experimental evidence is also reflected in source separation evaluation studies, like for instance, the large-scale study presented in [81]. However, from this experimental evidence it is not clear why approaches that focus on spectral approximation of the target source [76], [93], [99] require the post-processing step of generalized Wiener filtering. It is also intriguing to provide an explanation on why methods based on skip-filtering connections [A10], [94], [96] work well in practice. With this in mind, the first research question that this chapter aims at answering is ***RQ1 - Why is masking important in approaches based on the DAE model?***

Additionally, the work presented in [101] underlines the tendency of the encoding and decoding functions of the DAE to become symmetric during training. The *composition* of symmetric encoding and decoding functions yields another function, i.e., the *mapping function* for transforming the input signal to the corresponding output, that shares many similarities with the identity function. For spectral-based denoising, the similarities of the mapping function with the identity function result into a *trivial* scaling of the input mixture spectrogram. That could potentially result in poor estimation of the target source spectra, unless the learned function is derived from an ideal and time-variant frequency mask [102], computed using an appropriate time-frequency masking technique [72]. Subsequently, the second research question that this chapter aims at answering is ***RQ2 - Do DAEs commonly employed in music source separation learn trivial solutions for the given problem?***

To answer these research questions, the focus of this chapter is source separation models that operate on the magnitude spectra of the observed music mixture. Furthermore, it is proposed to examine the mapping functions of DNN-based music source separation approaches to estimate the magnitude spectra of the singing voice. Since nearly all music separation approaches are non-linear, the computation of the mapping function is not straightforward. To tackle that, an experimentally derived algorithm is presented and used. The algorithm approximates the mapping function of the non-linear model previously optimized for source separation. The result of the algorithm is a matrix that is utilized to linearly map the magnitude information of the mixture to the target source magnitude spectra. The algorithm is denoted as the *neural couplings algorithm* (NCA).

The spectral approximation and the skip-filtering connection methods for music source separation are clustered together under the general family of DAEs, since these approaches follow the exact same principle as the DAE model presented in [87]; that is, to recover a target signal from its corrupted version. Specifically, this chapter focuses on three particular and fundamental extensions of the DAE model for music source separation:

1. **DAE:** The DAE model presented in [87], as it forms the baseline that source separation approaches have built upon.
2. **MSS-DAE:** The music source separation denoising autoencoder (MSS-DAE), that is a multi-layered extension of the DAE applied to music source separation following the pioneering works presented in [76], [93].
3. **SF:** The implicit mask prediction via the skip-filtering (SF) connections presented in [A6], [94], [96].

---

<sup>4</sup>Supporting results to this claim are also presented in Chapter 4.

For these three models, the rectified linear unit activation function (ReLU) is used, as it is experimentally shown to perform well in music source separation tasks [93]. The target signal to be estimated by each model is the singing voice magnitude spectra. For assessing the mapping functions of each model the outcome of the NCA is used. The couplings matrix, computed using the NCA, is used to objectively compute a fraction of the magnitude contained in the main and off-diagonal elements of the corresponding matrix, and is explained in detail in the following sections. The rest of this chapter is organized as follows: Section 3.2 provides background information on the DAEs and the extensions proposed for the problem of music source separation tasks. The proposed algorithm for computing the mapping function is described in Section 3.3. The followed experimental procedure followed by the findings are given in Section 3.4. Section 3.5 summarizes this chapter and points out the limitations of the conducted study.

## 3.2 Denoising Autoencoders

### 3.2.1 Background

Music source separation based on DAEs relies on a supervised learning scenario. Formally, given a data-set  $\mathcal{D} = \{\tilde{x}^{(i)}, x^{(i)}\}_{i=1}^K$ , comprised of  $K \in \mathbb{Z}_+$  training examples indexed by  $i$ , the goal is to learn a denoising function  $f(\cdot)$  (or unmixing function in the context of source separation). The function  $f(\cdot)$  is parameterized by  $\theta$ , and estimates the clean  $x$  from the noisy  $\tilde{x}$  observation, i.e.,  $f : \theta \times \tilde{x} \mapsto x$ . Obtaining  $\tilde{x}$  involves a mixing process, which for audio signals is commonly assumed to be the addition of the interfering or noise signal  $x_n$  and the target source signal  $x$  [A10], i.e.,  $\tilde{x} = x + x_n$ . The mixing process using the interfering source signal  $x_n$  is different from the corruption process used originally for DAEs in [87]. Specifically, the signal  $x_n$  can be characterized as a generic multi-modal distribution-based noise, that is different from the pre-defined and constant distribution-based noise used in the DAE model [87] for unsupervised learning.

The learning of the parameters  $\theta$  relies on the energy-based learning framework presented in Chapter 2. More specifically, given a reconstruction loss function  $\mathcal{L}(\cdot)$  the parameters  $\theta$  are optimized using

$$\theta^o = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^K \mathcal{L}(x^{(i)}, \hat{x}^{(i)}), \quad (3.1)$$

where  $\hat{x}$  is an estimate of  $x$ , and  $\theta^o$  is the (ideally optimal) parameter or set of parameters, that minimizes the empirical loss. The updates of the parameters towards obtaining  $\theta^o$  are carried out using stochastic gradient descent over samples drawn from the data-set  $\mathcal{D}$  and optional heuristics, such as early stopping [52].

In [87], two functions are employed by the DAE model in order to approximate the unmixing function  $f(\cdot)$ . These two functions are namely the encoding  $f_{\text{enc}} : \theta_{\text{enc}} \times \tilde{x} \mapsto z$  and the decoding function  $f_{\text{dec}} : \theta_{\text{dec}} \times z \mapsto x$ . The set of parameters for the encoding and decoding functions is defined as

$$\theta := \{\theta_{\text{enc}}, \theta_{\text{dec}}\}$$

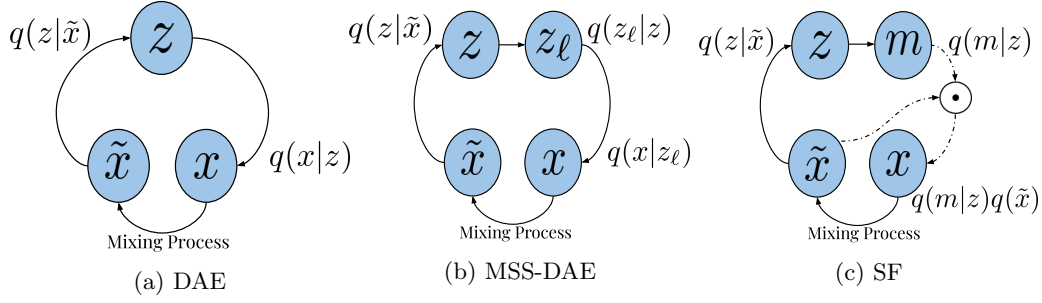


Figure 3.1: Illustration of the probabilistic graphical models<sup>5</sup> of encoder-decoder configurations examined in this chapter. The Illustration is based on [A22]. (a) *DAE*: a denoising auto-encoder model [87]. (b) *MSS-DAE*: a three layer example of a DAE model adapted to music source separation [76], [93]. (c) *SF*: skip-filtering connections [A6], [A10], [94], [96]. Solid arrows are functions computed by neural networks. Dashed arrows are the identity function. In the context of the input and latent variables, the symbol “ $\odot$ ” refers to the multiplication of the corresponding variables.

and is optimized according to Eq. (3.1). In the context of music source separation, the motivation is to learn the empirical (unnormalized) distribution

$$q(x|\tilde{x}) \quad (3.2)$$

through the usage of a latent representation  $z$ , and the utilization of the decoding process  $f_{\text{dec}}(\cdot)$ . An illustration of the DAE model is given in Figure 3.1a. In Eq. (3.2) the empirical distribution of the corruption or mixing process is considered as a known constant [87], [103], [104] that is independent from the clean target  $x$ , and is neglected from the explicit formulation. However, music source separation approaches based on the DAE [76], [93], [98] also assume that the corruption process is a constant, which does not hold true in practice.

The benefit of incorporating the latent variable  $z$  into the model is that it provides a representation or feature space that is useful for denoising auto-encoding [87]. In music source separation, an often used extension of the DAE model includes the usage of extra latent variables, computed using additional computational layers [76], [93]. The graphical model of a three layer example of the DAE model, employed in music source separation, is denoted as *MSS-DAE* and is illustrated in Figure 3.1.b. The additional layers are experimentally justified for improved separation performance [76], [90], [93], [98]. Specifically, the performance of the methods and the approximation of the target source  $x$  is shown empirically to be based on the computation of  $z_\ell$ , that is a deeper and hidden representations of  $z$  that leads to the conditional distribution  $q(z_\ell|z_{\ell-1})$  [76], [90], [93]. The subscript  $\ell \in \{1, 2, \dots, L\}$  denotes the layer and the depth of the computed, hidden representations.

<sup>5</sup>The purpose of the graphical models is to denote the dependencies between variables of computational graphs used in music source separation. The circular dependencies should not to be confused with any type of recurrence.

As in the case of the DAE, the music source separation approaches based on skip-filtering connections use the same two functions, i.e.,  $f_{\text{enc}}(\cdot)$  and  $f_{\text{dec}}(\cdot)$ . The conceptual difference with the DAE is that the decoder yields the mask variable  $m$ , i.e.,  $f_{\text{dec}} : \theta_{\text{dec}} \times z \mapsto m$ , and the clean source is computed as

$$x = m \odot \tilde{x}. \quad (3.3)$$

Eq. (3.3) shows the simplicity of implementing the skip-filtering connections, which allow  $\tilde{x}$  to be propagated to the encoding and to the last decoding function of the model [A6], [94], [97]. Subject to the target source  $x$  and since both the mask  $m$  and the target signal  $x$  are computed as a function of the signal  $\tilde{x}$ , i.e.,

$$x = f_{\text{dec}}(f_{\text{enc}}(\tilde{x})) \odot \tilde{x},$$

the separation model based on skip-filtering connections leads to the following empirical distribution<sup>6</sup>:

$$q(x|\tilde{x})q(\tilde{x}). \quad (3.4)$$

The above empirical distribution is conceptually different from the one of the DAE and MSS-DAE, which model  $q(x|\tilde{x})$  directly. Essentially, the SF model naively takes into account the empirical distribution of the corruption or mixing process, by considering the mixture’s empirical distribution. That is because the masking operation is considered as a part of the computational graph through the variable  $m$ . For the SF model and its corresponding conditional, the product between distributions is the product of the (unnormalized) probability values between the outcome of the DAE and  $\tilde{x}$ .

### 3.2.2 Connection to Music Source Separation & Related Work

The most widely adopted way to perform music source separation using the family of DAEs and (deep) neural network models, is to employ the magnitude spectral representations computed using the short-time Fourier transform (STFT). This is performed in order to reduce the overlap that the sources exhibit in the time-domain signal representation [20], and to exploit the wide-sense stationarity and the phase-invariant structure(s) of specific types of music sources [8]. Consequently, the variables of the DAE, MSS-DAE, and SF models can be seen as vectors containing magnitude spectral information. More specifically,  $\tilde{\mathbf{x}}, \hat{\mathbf{x}}, \mathbf{x} \in \mathbb{R}_{\geq 0}^N$ , and  $\mathbf{z} \in \mathbb{R}_{\geq 0}^F$  are the mixture signal, the estimated target<sup>7</sup> source signal, the target source signal, and the latent representation (after the application of the ReLU function), respectively.  $N$  denotes the dimensionality of the input comprising the frequency sub-bands of a time-frame computed using the STFT and  $F$  denotes the dimensionality of the hidden representations. It should be stated that since the phase information is not considered, the additive properties of the mixing process for computing  $\tilde{x}$  do not longer hold, i.e.,  $\tilde{x} \neq x + x_n$ , where  $x_n$  is the interfering source. Furthermore, the symbol “ $\odot$ ” now refers to the Hadamard (element-wise) product that scales each frequency sub-band of the previously mentioned vector(s).

<sup>6</sup>The empirical distribution is expressed with respect to the input-output relationship of the model and thus, differs from the dependencies of the mask variable  $m$  shown in Figure 3.1c.

<sup>7</sup>Joint separation of sources can be performed with the described models. However, a single target is considered here following the pioneering works presented in [76], [90], [93], [98].

For estimating music sources using DAEs, the authors in [76] propose the use of multi-layered feed-forward neural networks, using context information of past and future STFT magnitude spectra. Context information for spectral-based denoising via feed-forward neural networks is also proposed in [93]. Aiming to model the dependencies of adjacent time-frames, the work in [98] proposes to use bi-directional RNNs instead of feed-forward neural networks. In both [98] and [76] the estimated sources are further processed using the multi-channel Wiener filtering.

The skip-filtering connections are a straightforward extension of the denoising source separation (DSS) framework (in the spectral domain), presented in [102]. In the DSS framework, it is proposed to perform spectral-based denoising by learning a sparse matrix with non-zero elements only on the main diagonal. These elements allow a scalar filtering operation of each corresponding frequency sub-band [105]. In music source separation, the frequency sub-band scaling is achieved by employing the Hadamard product between the mask and the corresponding frequency-domain signal. The usage of the Hadamard product instead of the diagonal matrix, proposed in the DSS framework, enables the music source separation approaches based on deep learning to generate time-varying frequency masks and to use neural architectures that exploit time-varying information of the signals, such as recurrent neural networks (RNNs) and/or convolutional neural networks (CNNs). The skip-filtering connections are also similar to the highway connection networks presented in [106] for training very deep neural networks. Specifically, highway connection networks employ the Hadamard products between inputs and outputs of neural network layers. In the case of highway connection networks, the Hadamard products are used as the gating mechanisms in RNN architectures, such as the long short-term memory (LSTM) network. The main difference between the skip-filtering connections and the highway network connections, is that for the skip-filtering connections the predicted masks applied to the corresponding inputs are dependent on latent representations, computed by the network. In contrast, highway connection networks condition the predicted masks directly on the input signal. The usage of latent representations plays an important role in the denoising performance of the corresponding model, as shown in [86].

### 3.2.3 Implementation of the Models

Focusing on the graphical models presented in Figure 3.1, the problem is simplified to feed-forward neural network (FNN) layers, and constrained to the minimization of the mean squared error (MSE) loss function, defined as:

$$\mathcal{L}_{\text{MSE}}(\mathbf{x}^{(i)}, \hat{\mathbf{x}}^{(i)}) = \frac{1}{N} \|\mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)}\|_2^2, \quad (3.5)$$

where  $\|\cdot\|_2$  denotes the  $\ell_2$  vector norm. Stochastic gradient descent is performed to optimize the model parameters with respect to Eq. (3.5). This training configuration including the MSE was adopted from state-of-the-art approaches in music source separation [A6], [A14], [76], [93], [98], [99].

An example of calculating the representation  $\mathbf{z}$  and approximating the  $i$ -th data example of the target source  $\hat{\mathbf{x}}$ , using the mixture signal  $\tilde{\mathbf{x}}$  and the corresponding encoding and decoding functions is given in Eqs. (3.6)–(3.12) for the DAE, MSS-DAE, and SF models.

$$\mathbf{z}_{\text{DAE}}^{(i)} = g(\mathbf{W}_{\text{enc}} \tilde{\mathbf{x}}^{(i)} + \mathbf{b}_{\text{enc}}), \quad (3.6)$$

$$\hat{\mathbf{x}}_{\text{DAE}}^{(i)} = g(\mathbf{W}_{\text{dec}} \mathbf{z}_{\text{DAE}}^{(i)} + \mathbf{b}_{\text{dec}}), \quad (3.7)$$

$$\mathbf{z}_{\text{MSS-DAE}}^{(i)} = g(\mathbf{W}^{(\ell=1)} g(\mathbf{W}_{\text{enc}} \tilde{\mathbf{x}}^{(i)} + \mathbf{b}_{\text{enc}}) + \mathbf{b}^{(\ell=1)}), \quad (3.8)$$

$$\hat{\mathbf{x}}_{\text{MSS-DAE}}^{(i)} = g(\mathbf{W}'_{\text{dec}} \mathbf{z}_{\text{MSS-DAE}}^{(i)} + \mathbf{b}'_{\text{dec}}), \quad (3.9)$$

$$\mathbf{z}_{\text{SF}}^{(i)} = g(\mathbf{W}''_{\text{enc}} \tilde{\mathbf{x}}^{(i)} + \mathbf{b}''_{\text{enc}}), \quad (3.10)$$

$$\hat{\mathbf{x}}_{\text{SF}}^{(i)} = g(\mathbf{W}''_{\text{dec}} \mathbf{z}_{\text{SF}}^{(i)} + \mathbf{b}''_{\text{dec}}) \odot \tilde{\mathbf{x}}^{(i)}, \text{ where} \quad (3.11)$$

$$g(x) = \max(0, x). \quad (3.12)$$

The equations Eq. (3.6), Eq. (3.8), and Eq. (3.10) describe the encoding functions and the equations Eq. (3.7), Eq. (3.9), and Eq. (3.11) describe the decoding functions of each model. The weight matrices and bias terms are denoted by  $\mathbf{W}$  and  $\mathbf{b}$ , respectively. The subscripts “enc” and “dec” stand for encoder and decoder layers, respectively. The superscripts “'” and “''” in the weights and biases are used to distinguish between the parameters of different models (e.g. between  $\mathbf{W}$  of MSS-DAE and DAE models). In more detail, Eq. (3.6) and Eq. (3.7) express the encoding and decoding functions for the DAE model. An example of a three layered MSS-DAE model, with a single decoding function, for approximating the target source is provided by Eq. (3.8) and Eq. (3.9). The SF model in Eq. (3.10) and Eq. (3.11) employs the same encoding and decoding configuration as the DAE, with the only difference the output of the decoding is element-wise multiplied with the input to the model. The important thing to notice here, is that Eqs. (3.6)–(3.11) suggest that the encoding and decoding functions are realized as a series of linear operators and element-wise applications of the ReLU activation function, expressed in Eq. (3.12).

### 3.3 Computing Mapping Functions

#### 3.3.1 Motivation & Related Work

The motivation behind the neural couplings algorithm (NCA) is to compute a linear mapping function that describes how the input data are transformed to obtain the desired target source, according to the approach under examination. The NCA differs from methods that aim at explaining the neural network decisions, like for instance the layer-wise relevance propagation method presented in [107], or explaining the predictions of a classifier [108], [109]. However, the NCA shares many similarities with the knowledge distillation concept presented in [110]. The conceptual difference between the NCA and the previously stated methods is that the NCA specifically approximates the mapping function of a pre-trained neural network model. It does not aim at compressing the neural network model as in [110], or at pin-pointing input spectral features that affect the choice of the neural network model as in [107]. Instead, the couplings matrix computed using the NCA expresses the transfer of energy from one frequency sub-band to another,

differentiating from the distance-dependent mapping(s) between data distributions as in the case of optimal transportation theory [111], [112] in the discrete case.

Furthermore, the NCA is conceptually similar to the extended data Jacobian matrix (EDJM) framework, presented in [113], that analyses the functionality of DNNs. The difference between the EDJM and NCA, is that NCA seeks a simpler linear matrix that can be easily evaluated using digital signal processing. In contrast, the EDJM focuses on the spectral analysis of the resulting mapping matrix, which is not that informative for the particular problem of examining music source separation approaches based on neural networks. Another difference is that the computations used by the NCA, formally described in the next sub-section, are simpler than the explicit computation of latent gradient information used in the EDJM [113]; an important factor considering the high dimensionality of audio spectrogram data. The formulation used by the NCA, shares many similarities with the methodology used in [114] for the examination of the performance of randomly parameterized DNNs, based on the computation of super-masks<sup>8</sup>.

### 3.3.2 The Neural Couplings Algorithm

The NCA is an iterative method for approximating the mapping function of a non-linear source separation model. The mapping function is defined as a *local* affine transformation of magnitude spectral data. The affine transformation is constrained to be linear, time-invariant<sup>9</sup>, and model-dependent in order to enable an intuitive examination of what each source separation model has learned. The affine transformation is represented by a matrix that is denoted as the *couplings matrix*  $\mathbf{C} \in \mathbb{R}^{N \times N}$ . The couplings matrix  $\mathbf{C}$  is used to transform, the input mixture magnitude spectrum  $\tilde{\mathbf{x}}$  to the output  $\mathbf{y} \in \mathbb{R}_{\geq 0}^N$  of the last layer of each corresponding model including the non-linearity, i.e., the output of the decoding matrix followed by the ReLU function. The vector  $\mathbf{y}$  is used to denote the output of the decoding function in each model. Specifically, the output for the DAE and MSS-DAE models is the singing voice spectra, and for the SF model is the derived frequency mask. The indexing by  $i$  in  $\tilde{\mathbf{x}}$  and  $\mathbf{y}$  is dropped in order to denote the usage of spectral data that are not sampled from the training data-set.

The reason for using the mask instead of singing voice spectra for the SF model follows naturally from the graphical models illustrated in Figure 3.1. Particularly for the SF model, illustrated in Figure 3.1c, the information of the mixture spectra is also necessary after the decoding process in order to estimate the source via masking using the Hadamard product expressed in Eq. (3.11). As masking is an additional operator that heavily depends on the mixture data, the computation of a single affine transformation would fail to approximate both the masking and the mapping function of the model. A solution to this is given by knowledge distillation [110]; that is, to use the last hierarchical variable that is computed using the model’s parameters. This leads to a fair usage of information during the approximation of the NCA among source separation models. Furthermore, the ReLU function applied to the decoding stage, is also considered by the NCA since an algebraic

<sup>8</sup>The work presented in [114] and the NCA both propose methods to compute super-masks, i.e., weight matrices for scaling the parameters of the DNN. However, the areas of applications of the NCA and of the work presented in [114] are different.

<sup>9</sup>The source separation models are also time-invariant and thus, only frequency dependencies are considered.

expression for the ReLU function is given in [115], and its relevance in the computation of the couplings matrix is explained later in this section.

In the ideal case in which each model is linear, the couplings matrix, and thus the mapping function, is expressed algebraically as the product of the corresponding encoding and decoding matrices. That product is denoted as the *linear composition*. Neglecting the bias terms for brevity in the notation, the linear composition is computed for each model as follows:

$$\mathbf{C}_{\text{linear-DAE}}^o = \mathbf{W}_{\text{dec}} \mathbf{W}_{\text{enc}}, \quad (3.13)$$

$$\mathbf{C}_{\text{linear-MSS-DAE}}^o = \mathbf{W}'_{\text{dec}} \mathbf{W}^{(\ell=1)}_{\text{enc}} \mathbf{W}'_{\text{enc}}, \text{ and} \quad (3.14)$$

$$\mathbf{C}_{\text{linear-SF}}^o = \mathbf{W}''_{\text{dec}} \mathbf{W}''_{\text{enc}}. \quad (3.15)$$

Since the DAE, MSS-DAE, and SF models are non-linear, the direct application of Eqs. (3.13)–(3.15) would result into rather crude approximations of each model’s mapping function. Even the linear behaviour of the ReLU function in the non-negative range, and of the vector-matrix products expressed in Eqs. (3.6)–(3.11), is not sufficient for the above linear composition functions to hold. The ReLU function performs a thresholding operation on the variables that yield the latent  $\mathbf{z}$  and output  $\mathbf{y}$  vectors that are learned through observations drawn from the training data-set. This in turn makes each model highly non-linear [27].

An algebraic expression of the ReLU function that is related to the concept of thresholding of negative values is presented in [115]. In [115, Section 3 and Eq. (3)], a single application of the ReLU function for a given input vector  $\tilde{\mathbf{x}}$  can be expressed as a *binary diagonal* matrix, that sets to 0 any negative value of the encoded vector. The binary diagonal matrix is denoted by  $\mathbf{G}$ . Consequently, to obtain the couplings matrix of the DAE, MSS-DAE, and SF models, it is necessary to compute as many matrices  $\mathbf{G}$  as the number of application of the ReLU function in the DAE, MSS-DAE, and SF models:

$$\mathbf{C}^o = \mathbf{G}_{\text{dec}} \mathbf{W}_{\text{dec}} \dots \mathbf{G}_{\text{enc}} \mathbf{W}_{\text{enc}}. \quad (3.16)$$

Furthermore, to compute each matrix  $\mathbf{G}_*$  in Eq. (3.16), it is necessary to learn the model specific dependencies that are captured by the DAE, MSS-DAE, and SF models during the supervised training [27], [115]. The asterisk “\*” in the notation is used for brevity, and replaces the subscripts and/or superscripts of the layer identifiers initially used in Eqs. (3.6)–(3.11). The data dependencies expressed by each  $\mathbf{G}_*$  refer to the algebraic operations between the mixture  $\tilde{\mathbf{x}}$  or the corresponding latent vectors  $\mathbf{z}$ , i.e., the encoding or decoding matrices  $\mathbf{W}_*$ , and the corresponding bias terms  $\mathbf{b}_*$  as in Eqs. (3.6)–(3.12).

A straightforward way to learn the data dependencies can be derived from the knowledge distillation concept presented in [110]. In knowledge distillation, a neural network, i.e., the *student*, is optimized by means of (stochastic) gradient descent to predict the output of a more complicated model (e.g. the non-linear DAE, MSS-DAE and SF). Subject to the goal of this work, gradient descent is employed similarly to the knowledge distillation [110], and the student network is constrained to be a linear, affine transformation from  $\tilde{\mathbf{x}}$  to  $\mathbf{y}$ . That transformation is based on the product of the mixture spectra  $\tilde{\mathbf{x}}$  and the couplings matrix  $\mathbf{C}$ . For computing the couplings matrix  $\mathbf{C}$  it is proposed to solve



the following optimization problem:

$$\mathbf{C}^o = \underset{\mathbf{C}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{C}\tilde{\mathbf{x}}\|, \quad (3.17)$$

where  $\|\cdot\|$  is the  $\ell_1$  norm and  $\tilde{\mathbf{x}}$  is sampled from the testing data-set<sup>10</sup>. The output vector  $\mathbf{y}$  is computed by using  $\tilde{\mathbf{x}}$  as an input to the corresponding model. The  $\ell_1$  norm is used instead of the  $\ell_2$  norm in the above optimization, because the vector  $\mathbf{y}$  is expected to be sparse, due to the application of the ReLU function [27]. Commonly, the  $\ell_1$  norm offers an attractive objective for error minimization between sparse vectors, when the choice of the regularization strength parameter in the sparse aware setting of  $\ell_2$ -based optimization is difficult [15, Chapter 9]. By minimizing the  $\ell_1$  norm of errors it can be interpreted as expecting the reconstruction error to follow an exponential distribution [15, Chapter 9]. Given that Eq. (3.17) is inspired by the student network of the knowledge distillation concept [110], this strategy is denoted as the *student*.

To compute  $\mathbf{C}^o$ , let  $E$  denote the  $\ell_1$  norm of the error of Eq. (3.17) and computed as

$$E = \|\mathbf{y} - \mathbf{C}\tilde{\mathbf{x}}\|.$$

For the student strategy the following partial derivatives are used:

$$\Delta := \frac{\partial E}{\partial \mathbf{C}} = \frac{\partial E}{\partial \mathbf{C}\tilde{\mathbf{x}}} \frac{\partial \mathbf{C}\tilde{\mathbf{x}}}{\partial \mathbf{C}} \quad (3.18)$$

From Eq. (3.18), it follows that the gradient signal  $\Delta$  that is used to update  $\mathbf{C}$  in an iterative manner is given by

$$\Delta = \operatorname{sgn}(\mathbf{C}\tilde{\mathbf{x}} - \mathbf{y})\tilde{\mathbf{x}}^\top,$$

where  $\operatorname{sgn}(\cdot)$  is the signum element-wise function and  $\cdot^\top$  is the vector/matrix transposition. The gradient signal  $\Delta$  suggests that the optimal  $\mathbf{C}^o$  lies over the least affinity between the mixture magnitude spectra  $\tilde{\mathbf{x}}$  and  $\mathbf{C}\tilde{\mathbf{x}} - \mathbf{y}$ . This means that the updates of  $\mathbf{C}$  only favor the minimization of the reconstruction error term. Although this strategy could yield simplified and robust surrogates of possibly deep and complex models for source separation (in terms of reconstruction errors) [110], it neglects the learned data dependencies contained in the encoding and decoding matrices of Eq. (3.16).

According to [113], the learned data dependencies of the encoding and decoding functions are the key ingredient to characterize the functionality of a non-linear model. In [113], it is also shown that those dependencies are described by linear systems that can be computed using observations of  $\tilde{\mathbf{x}}$ ,  $\mathbf{y}$  and the corresponding weight matrices. Given that the current goal is to approximate the functionality of the model, i.e., including the knowledge captured by the encoding/decoding matrices  $\mathbf{W}_*$  and the bias terms  $\mathbf{b}_*$ , an alternative strategy is proposed and is denoted as the *compositional*. The proposed strategy composes the couplings matrix  $\mathbf{C}$  similar to the composition expressed in Eq. (3.16) and hence, the name compositional. In contrast to the method presented in [113], the compositional approach does not require the explicit computation of the latent information  $\mathbf{z}_*$  of each model. Instead, it uses directly  $\mathbf{G}_*$  to extract data dependencies contained in each  $\mathbf{W}_*$ , capturing both the information of the encoding/decoding matrices and the spectral

<sup>10</sup>The training data-set can also be used. For testing the generalization of the approach, the testing data-set is used.

$$\mathbf{C} = \left( \mathbf{W}_{\text{dec}} \odot \mathbf{G}_{\text{dec}} \right) \cdot \cdot \cdot \left( \mathbf{W}_{\text{enc}} \odot \mathbf{G}_{\text{enc}} \right)$$

Figure 3.2: An illustration of how the couplings matrix  $\mathbf{C}$  is computed using the compositional strategy. The compositional strategy takes into account the encoding/decoding matrices  $\mathbf{W}_*$  by preserving and scaling the relevant corresponding matrix elements using the matrices  $\mathbf{G}_*$ . Only for illustration purposes, the matrices are assumed to be non-negative and continuously vary between zero and one. White color is used for matrix elements whose values are zero and (dark) purple color is used for values close to one.

data. In contrast to EDJM framework [113], the NCA practically allows the computation of the mapping function(s) at a lower computational cost, due to the fewer required algebraic operations. Furthermore, the computation using the dependencies contained in each  $\mathbf{W}_*$  enables the NCA to compute the mapping function that describes multiple data observations, that is particularly useful for spectral-based music source separation.

More specifically, it is proposed to exploit the fact that the ReLU function behaves linearly in the non-negative range where the mixture  $\tilde{\mathbf{x}}$ , the output  $\mathbf{y}$ , and the latent  $\mathbf{z}$  vectors reside in. Therefore, and according to Eqs. (3.6)–(3.11), the relevant components that affect which elements are thresholded by the ReLU function for computing the latent  $\mathbf{z}$  and output  $\mathbf{y}$  vectors, are: i) the row-vectors contained in each  $\mathbf{W}_*$  and ii) the corresponding bias term  $\mathbf{b}_*$  applied after each vector-matrix product and before the application of the ReLU function. Inspired by [115], that models the ReLU function as a diagonal matrix  $\mathbf{G}_*$  that scales the row-vectors of  $\mathbf{W}_*$  in Eq. (3.16), the NCA builds the couplings matrix for the compositional strategy as follows:

$$\mathbf{C} = (\mathbf{W}_{\text{dec}} \odot \mathbf{G}_{\text{dec}}) \cdot \cdot \cdot (\mathbf{W}_{\text{enc}} \odot \mathbf{G}_{\text{enc}}). \quad (3.19)$$

The Hadamard product in Eq. (3.19) accounts for each individual element of the row-vectors contained in the corresponding  $\mathbf{W}_*$ , rather than having a single scalar value per row-vector as in the case of the matrix product using the diagonal matrix. Practically, this mitigates the usage of binary diagonal matrices and allows more degrees of freedom into the approximation, as the operator is applied to all the elements of the corresponding matrix vectors [115]. A visual example of the compositional strategy (Eq. (3.19)) is given in Figure 3.2.

The reason for accounting for all the elements, is that there are two important features that carry the essential information that characterize the model-dependent processing of non-negative vectors. These two features are: i) the sign and ii) the magnitude of each corresponding element in each  $\mathbf{W}_*$ . To consider the influence of the sign of the elements in each  $\mathbf{W}_*$ , each matrix  $\mathbf{G}_*$  is restricted to be non-negative, i.e.,  $\mathbf{G}_* \in \mathbb{R}_{\geq 0}^{N \times N}$ . To do so, and to account for the influence of the bias terms, each  $\mathbf{G}_*$  is computed as

$$\mathbf{G}_* = g(\hat{\mathbf{G}}_*) \quad (3.20)$$

$$\hat{\mathbf{G}}_* = \mathbf{P}_*(\mathbf{W}_* + \mathbf{b}_* \mathbf{1}_N^\top)^\top. \quad (3.21)$$

In Eq. (3.21) the  $N$ -th dimensional vector  $\mathbf{1}_N^\top$  and the bias vector  $\mathbf{b}_*$  are used in order to add the bias vector to each column-vector of  $\mathbf{W}_*$ .

By the application of  $\mathbf{G}_*$  via the Hadamard product, the elements in  $\mathbf{W}_*$  are either preserved and scaled or nullified. This depends on the individual element's relevance for mapping the mixture magnitude spectra under linear constraints. The relevance itself is dependent on the sign and the magnitude, and their effect on mapping the magnitude information through the matrix  $\mathbf{C}$ . The latter is enforced through Eq. (3.17). For computing the relevance, it is proposed to learn an additional affine transformation computed for each matrix  $\mathbf{W}_*$  plus the corresponding bias term  $\mathbf{b}_*$ . This affine transformation is denoted by the matrix  $\mathbf{P}_*$  and its purpose is to discover the correlations in the shifted versions of the column-vectors of  $\mathbf{W}_*$  that are computed by adding the bias term  $\mathbf{b}_*$ . These correlations are useful for determining the relevance of each matrix element in  $\mathbf{W}_*$  for computing  $\mathbf{C}$  using Eq. (3.19). The corresponding basis vectors of  $\mathbf{P}_*$  are unknown for the compositional strategy. To jointly compute the unknowns of the compositional strategy, back-propagation is used.

To further analyze how the previously mentioned data dependencies are learned using the compositional strategy, let us consider the case in which a single encoding and decoding matrix is used, as in the DAE and SF models. For the compositional strategy, it is necessary to compute two matrices  $\mathbf{P}_{\text{enc}}$  and  $\mathbf{P}_{\text{dec}}$  whose corresponding partial derivatives are defined as

$$\frac{\partial \mathbf{E}}{\partial \mathbf{P}_{\text{dec}}} = \frac{\partial \mathbf{E}}{\partial \mathbf{C}} \frac{\partial \mathbf{C}}{\partial \mathbf{G}_{\text{dec}}} \frac{\partial \mathbf{G}_{\text{dec}}}{\partial \mathbf{P}_{\text{dec}}} \quad (3.22)$$

$$\frac{\partial \mathbf{E}}{\partial \mathbf{P}_{\text{enc}}} = \frac{\partial \mathbf{E}}{\partial \mathbf{C}} \frac{\partial \mathbf{C}}{\partial \mathbf{G}_{\text{enc}}} \frac{\partial \mathbf{G}_{\text{enc}}}{\partial \mathbf{P}_{\text{enc}}}. \quad (3.23)$$

Using Eq. (3.18) and by calculating the partial derivatives of Eqs. (3.19), (3.20), and (3.21) with respect to  $\mathbf{G}_*$  and  $\mathbf{P}_*$  respectively, the above partial derivatives lead to the following update rules for  $\mathbf{P}_{\text{enc}}$  and  $\mathbf{P}_{\text{dec}}$ :

$$\frac{\partial \mathbf{E}}{\partial \mathbf{P}_{\text{dec}}} = \left( \Delta (\mathbf{W}_{\text{enc}} \odot \mathbf{G}_{\text{enc}}) \odot \mathbf{W}_{\text{dec}} \odot g'(\hat{\mathbf{G}}_{\text{dec}}) \right) (\mathbf{W}_{\text{dec}} + \mathbf{b}_{\text{dec}} \mathbf{1}_N^\top) \quad (3.24)$$

$$\frac{\partial \mathbf{E}}{\partial \mathbf{P}_{\text{enc}}} = \left( (\mathbf{W}_{\text{dec}} \odot \mathbf{G}_{\text{dec}})^\top \Delta \odot \mathbf{W}_{\text{enc}} \odot g'(\hat{\mathbf{G}}_{\text{enc}}) \right) (\mathbf{W}_{\text{enc}} + \mathbf{b}_{\text{enc}} \mathbf{1}_N^\top), \quad (3.25)$$

where  $g'(x)$  is the first derivative of the ReLU element-wise function, that is approximated by a function that is equal to 1 for positive inputs and 0 otherwise.  $\mathbf{G}_{\text{dec}}$  and  $\mathbf{G}_{\text{enc}}$  are computed using Eq. (3.21).

In Eq. (3.24) and Eq. (3.25), instead of considering only the gradient for minimizing the reconstruction error  $\Delta$ , the models' optimized parameters partake into the optimization of the compositional strategy. Specifically,  $\mathbf{W}_*$  and  $\mathbf{b}_*$  contribute to the optimization in a similar vein as the linear compositions described in Eqs. (3.13)–(3.15). This can be seen, by the products between the outer parentheses that surround the encoding and decoding matrices in Eqs. (3.24) and (3.25). The Hadamard products including the  $g'(x)$  inside the first parentheses of Eq. (3.24) and Eq. (3.25) can be understood as the search of elements, contained in the decoder and encoder matrices, that contribute to the mapping of the mixture to the corresponding output. From the above, it can be said that the compositional strategy applies a layer-wise restriction. The layer-wise restriction forces the

couplings matrix to be computed using the parameters of each model, similar to the linear composition that algebraically corresponds to the mapping function. This is in contrast to the student strategy, which does not take into account the information in each  $\mathbf{W}_*$  and  $\mathbf{b}_*$ . It should be mentioned that regardless of the strategy, i.e., student or compositional, the computed matrices  $\mathbf{C}_*$  are allowed to retain negative values. That is because the destructive property of the negatives values during the computation of the vector-matrix products are helpful in estimating the target source’s magnitude information. The pseudo-algorithm of the NCA for both strategies is given in Algorithm 1.

---

**Algorithm 1** The Neural Couplings Algorithm

---

**Require:** Mixture spectra  $\tilde{\mathbf{x}}$ , model  $\mathcal{M}(\cdot)$ , model’s parameters  $\mathbf{W}_*, \mathbf{b}_*$ ,  $N \times N$  identity matrix  $\mathbf{I}_N$ , total number of layers in model  $L'$ , number of iterations  $N_{it}$ , strategy  $S \in \{\text{student, compositional}\}$ , random generator function  $rnd$ , optimizer/solver  $\mathcal{A}(\cdot)$

```

1:  $\mathbf{y} \leftarrow \mathcal{M}(\tilde{\mathbf{x}})$ 
2: if  $S$  is student then
3:    $\mathbf{C} \leftarrow rnd$ 
4: else
5:    $\mathbf{C} \leftarrow \mathbf{I}_N$ 
6:   for  $l' := 1$  to  $L'$  do
7:      $\mathbf{P}_{l'} \leftarrow rnd$ 
8:      $\mathbf{G}_{l'} \leftarrow g(\mathbf{P}_{l'}(\mathbf{W}_{l'} + \mathbf{b}_{l'}\mathbf{1}_N^\top)^\top)$ 
9:      $\mathbf{C} \leftarrow (\mathbf{W}_{l'} \odot \mathbf{G}_{l'})\mathbf{C}$ 
10:   end for
11: end if
12: for  $i = 1$  to  $N_{it}$  do
13:    $E \leftarrow \|\mathbf{y} - \mathbf{C}\tilde{\mathbf{x}}\|$ 
14:   if  $S$  is student then
15:      $\mathbf{C} \leftarrow \mathbf{C} - \mathcal{A}(i, \frac{\partial E}{\partial \mathbf{C}})$ 
16:   else
17:      $\mathbf{C} \leftarrow \mathbf{I}_N$ 
18:     for  $l' := 1$  to  $L'$  do
19:        $\mathbf{P}_{l'} \leftarrow \mathbf{P}_{l'} - \mathcal{A}(i, \frac{\partial E}{\partial \mathbf{P}_{l'}})$ 
20:        $\mathbf{G}_{l'} \leftarrow g(\mathbf{P}_{l'}(\mathbf{W}_{l'} + \mathbf{b}_{l'}\mathbf{1}_N^\top)^\top)$ 
21:        $\mathbf{C} \leftarrow (\mathbf{W}_{l'} \odot \mathbf{G}_{l'})\mathbf{C}$ 
22:     end for
23:   end if
24:    $\mathbf{C}^o \leftarrow \mathbf{C}$ 
25: end for
26: return  $\mathbf{C}^o$ 

```

---

### 3.4 Experimental Procedure

#### 3.4.1 Training & Assessing Separation Models

To optimize the parameters used by the operations described in Eqs. (3.7)–(3.11), the 100 two-channel multi-tracks available in the the MUSDB18 data-set [A8] are used. The multi-track recordings are sampled at 44100 Hz. For each multi-track, the mixture and singing voice signals in the data-set are used. A monaural mixing is performed for each signal by averaging the two available channels. For constructing the data-set  $\mathcal{D}$ , the STFT analysis is performed for each mixture and corresponding source signal, using a hamming windowing 46 ms long, a factor of 2 for zero-padding, a hop-size of 8.7 ms, and a frequency analysis of  $N' = 4096$ . Due to the redundancies of the discrete Fourier transform, only the first  $N = 2049$  frequency sub-bands are retained for constructing the data-set. After computing the magnitude of the complex representation, each frequency sub-band is normalized to have a unit variance with respect to the time frames.

A single encoding and decoding layer is used for all models and in all approaches. The number of hidden layers for MSS-DAE is set to  $L = 2$  (MSS-DAE has  $L' = 4$  layers in total). The dimensionality through the layers is preserved the same in order to avoid any implicit model regularization [101], [116], [117]. The number of layers for the MSS-DAE model was chosen experimentally according to the saturation in minimizing Eq. (3.5) during the training process, with respect to the number of hidden layers. All the weight matrices are initialized with samples drawn from a normal distribution and scaled by  $\sqrt{\frac{1}{N}}$  as proposed in [64]. The bias terms are initialized to zero.

The data-set  $\mathcal{D}$  is randomly shuffled, and the training is performed using batches of 128 time-frames. For gradient-based optimization, the Adam algorithm [50] is used with the initial learning rate set to  $1e - 3$ , and decreased by half if no improvement to the loss was observed for two consecutive iterations over all available training data. The exponential decay rates for the first and second-order moments of the Adam algorithm are set to 0.9 and 0.999, respectively, following the proposed settings presented in [50]. The training is terminated after no improvement is observed over four consecutive iterations throughout the training data. The previously described training procedure is repeated 50 times using different random initialization states. This is performed in an attempt to minimize the induced bias of randomly initializing the models' layers, and therefore more reliably addressing the second research question “*RQ2 - Do DAEs commonly employed in music source separation learn trivial solutions for the given problem?*”.

To assess the source separation models, it is proposed to evaluate the ability of each model to exploit the structure in music spectral representations. To that aim, the couplings matrix computed by the NCA is used. The couplings matrix acts as a data and model-specific filtering operator. According to [105], there are two broad classes of filtering operators. The first class is the *vector* filtering operator, where the matrix responsible for filtering, i.e., the couplings matrix in the studied case, contains high values of magnitude on off-diagonal elements. The main benefit of off-diagonal elements is that they allow the exploitation of inter-frequency relationships of the spectral data.

In contrast, the second class of filtering operators is denoted as *scalar* filters. Scalar filters are characterized by magnitude only on the main diagonal of the couplings matrix. Each element on the main diagonal scales individually the corresponding frequency sub-

band. The latter operation is equivalent to the application of a masking strategy [72]. In practice, source separation models are optimized using many training examples. Consequently, learning a mapping function with activity on the main diagonal could imply a limited performance in estimating the singing voice spectra for multiple examples in the data-set.

Based on the arguments expressed in the above paragraphs, the trace-to-off-diagonal-ratio (TOD-R) objective measure [A22] is employed. The TOD-R is computed as follows:

$$\text{TOD-R}(\mathbf{C}_*) = \sqrt{N} \frac{\text{tr}(|\mathbf{C}_*|)}{\|\mathbf{C}_* \odot (\mathbf{J}_N - \mathbf{I}_N)\|}, \quad (3.26)$$

where  $\text{tr}(\cdot)$  is the trace function,  $\mathbf{I}_N$  is the  $N \times N$  identity matrix, and  $\mathbf{J}_N$  is the  $N \times N$  matrix with all its elements equal to one. The scaling by  $\sqrt{N}$  is performed in order to compensate for the expected high values of the denominator in Eq. (3.26) as the norm is computed using far more matrix elements than the trace in the numerator. The element-wise absolute  $|\cdot|$  is computed prior to the computation of the trace to avoid biasing the ratio due to the norm (sum of absolute values) in the denominator of Eq. (3.26). Small values of TOD-R indicate that the off-diagonal elements of the couplings matrix retain higher magnitude values than the elements on the main diagonal and vice versa. High off-diagonal activity suggests that the mapping function of the model exploits more inter-frequency relationships rather than estimating values that scale the mixture spectra as in scalar-based filtering and frequency masking [105].

### 3.4.2 Computing & Assessing the Neural Couplings

For computing the neural couplings, it is required to solve the optimization problem expressed in Eq. (3.17). To that aim, the test subset from the MUSDB18 data-set [A8] is used. The test subset comprises 50 additional two-channel, multi-tracks sampled at 44100 Hz. For computing the STFT the same parameters are employed as previously used during the construction of the training data-set  $\mathcal{D}$ , reported in Sec. 3.4.1. Since the separation models are able to process multiple time-frame vectors of mixture spectra rather than a single instance drawn from the data-set, the mapping functions are computed using a batch of adjacent magnitude vectors. The batch is drawn from the test subset and the size is set to  $T = 350$  ( $\sim 3.1$  seconds long) time-frames. The hyper-parameter  $T$  is experimentally chosen based on the available computational resources. Generally, the larger the  $T$  becomes the more general are the computed mapping functions.

Following the above usage of a batched spectral data, the NCA can be simply reformulated using matrix notation:  $\tilde{\mathbf{X}} \in \mathbb{R}_{\geq 0}^{N \times T}$  instead of  $\tilde{\mathbf{x}} \in \mathbb{R}_{\geq 0}^N$  and  $\mathbf{Y} \in \mathbb{R}_{\geq 0}^{N \times T}$  instead of  $\mathbf{y} \in \mathbb{R}_{\geq 0}^N$ . For the solver denoted as  $\mathcal{A}(\cdot)$  in Algorithm 1, the Adam algorithm (Algorithm 4) is used with a learning rate equal to  $4e^{-4}$  and the same exponential decay rates as in Section 3.4.1. The total number of iterations is set to 600, and the random function (*rnd* in Algorithm 1) refers to drawing samples from a normal distribution scaled by  $\sqrt{\frac{1}{N}}$  as proposed in [64]. The above mentioned hyper-parameters were chosen experimentally.

In comparison to the optimization of the music source separation models presented in Section 3.4.1, the couplings matrix is computed for each model and for each batch of adjacent magnitude vectors drawn from the test subset of MUSDB18 data-set [A8]. The

pre-trained parameters of each one of the three models are randomly drawn from one of the 50 training instances. In a series of initial experiments it was observed that the silent segments in the used multi-tracks led to randomly structured and sparse, i.e., low  $\ell_1$  norm values, row-vectors of the computed couplings matrix  $\mathbf{C}^o$ . Although the observed convergence of the NCA was satisfactory for the silent segments, it significantly biased the TOD-R measure in an unpredictable manner. Therefore, from each multi-track 30 seconds are selected, where all music sources, available in the data-set, are active. The selection of the active waveform regions, is based on the generator<sup>11</sup> used in the music source separation evaluation campaign [81].

The convergence of the NCA is analysed quantitatively. The analysis aims at assessing the ability of the NCA to accurately *approximate the models' outputs*. For intuitively evaluating the accuracy of the approximation the signal-to-noise ratio (SNR), expressed in dB, is used. For the DAE and MSS-DAE models, the SNR is computed using the singing voice magnitude spectra estimated by the NCA, and the singing voice magnitude spectra estimated by the corresponding models. For the SF model, the predicted mask using the NCA is first applied to the input mixture, and the outcome is used to compute the SNR. This is done because the SF model does not employ any pre-computed mask during its optimization that could be used for evaluating the NCA. For baseline comparison, the linear composition and the identity function are used as proxies to the couplings matrix, i.e.,  $\mathbf{C}$  is computed using Eqs. (3.13)–(3.15) and  $\mathbf{C} = \mathbf{I}_N$ , respectively. Since the SNR is computed using the approximations of the NCA and the models' outputs, different SNR values across the source separation models are expected.

In addition to the above, the SNR is also computed using the NCA approximation and the true source singing voice spectra. Then, the SNR values are compared with the SNR computed using the models' outputs and the true source singing voice spectra. This is done in order to intuitively quantify the loss of information induced by the NCA. The previously mentioned analysis is performed for each strategy, model, and segment in each multi-track. It is important to note that the goal here is to understand the power of the NCA to deliver an accurate approximation of the model's output.

### 3.4.3 Results & Discussion

To address the second research question “*RQ2: Do DAEs commonly employed in music source separation learn trivial solutions for the given problem?*”, the linear composition functions are computed for the three models (DAE, MSS-DAE, and SF) using Eqs. (3.13)–(3.15). The average result across the 50 performed experimental iterations from the composition functions is illustrated in Figure 3.3. The computation of the linear compositions draws inspiration from the findings presented in [101] that underlines the tendency of encoding and decoding functions to become symmetric, that practically leads to learning scalar filtering operators.

By observing Figure 3.3 it is evident that the two models that map directly the mixture magnitude spectra to the singing voice spectra (DAE and MSS-DAE) have a prominent main diagonal structure. This means that through training, the corresponding encoding and decoding functions tend to become symmetric [101], in an attempt to provide a solution in the MSE sense. Therefore, it is plausible that the DAE and MSS-DAE models

<sup>11</sup>Available at: <https://github.com/sigsep/sigsep-mus-cutlist-generator>

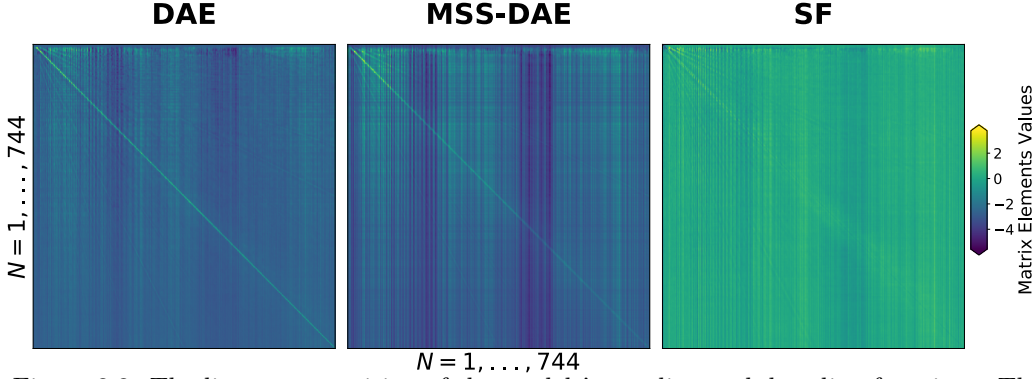


Figure 3.3: The linear composition of the models’ encoding and decoding functions. The compositions are computed using the Eqs. (3.13)–(3.15), and averaged across the 50 experimental iterations. The first 744 frequency sub-bands ( $\sim 8\text{kHz}$ ) are displayed for clarity. *Left Column:* Composition for the denoising auto-encoder model [87] (DAE). *Middle Column:* Composition for the four layer extension of the DAE model, adapted to music source separation (MSS-DAE) [76], [93]. *Right Column:* Composition for the skip connections for filtering the input mixture (SF) [A6], [94], [96]. Illustration is based on [A22].

learned trivial solutions to the problem of singing voice separation. This could result in severely restricting the overall separation performance. On the other hand, employing the skip-connections as performed in the SF model, it can be observed from the right column of Figure 3.3, that the activity has been repelled from the main diagonal. This can be explained by recalling that the estimation of the singing voice spectra can be performed also by scaling the individual frequency sub-bands of the mixture, which in turn is expressed by a diagonal matrix. The above observations combined with the latter statement provide a simple explanation on why skip-connections [118] and end-to-end learning [119], where the time-domain signals are used instead of spectrograms, are emerging directions in music source separation.

Aiming to address the other research question “***RQ1: Why is masking important in approaches based on the DAE model?***”, the attention is directed at the computed mapping functions using the NCA, and the model assessment via the TOD-R metric. Table 3.1 summarizes the TOD-R results for each model and for both strategies, i.e., student and compositional. The average TOD-R across all the segments of the test-subset is reported. Inside the parentheses the standard deviation of the corresponding measurements is provided. Bold faced numbers indicate the smallest obtained value for the TOD-R (smaller is better), implying that the model has exploited a richer inter-frequency structure. The results of Table 3.1 show that the SF model provides the smallest TOD-R value for both strategies. More specifically, the TOD-R for the compositional strategy and for the SF model is 12 times smaller than the TOD-R for the DAE model, which has equal number of encoding and decoding layers. In comparison to the MSS-DAE model that comprises two additional hidden layers, the TOD-R value of the SF model is decreased by approximately 4.6 times. Therefore, it could be concluded that the skip-filtering connections can be seen as a simple method to repel source separation approaches from learning solutions that concentrate most of the activity on the main diagonal of their



Table 3.1: Assessing the mapping functions. The TOD-R metric (Eq. (3.26)) for each strategy and model. Lower TOD-R values indicate high off-diagonal activity and thus, the model’s ability to learn inter-frequency relationships.

Strategy	Model		
	DAE	MSS-DAE	SF
Student	0.03 ( $\pm 0.00$ )	0.03 ( $\pm 0.00$ )	<b>0.02 (<math>\pm 0.00</math>)</b>
Compositional	0.36 ( $\pm 0.13$ )	0.14 ( $\pm 0.04$ )	<b>0.03 (<math>\pm 0.01</math>)</b>

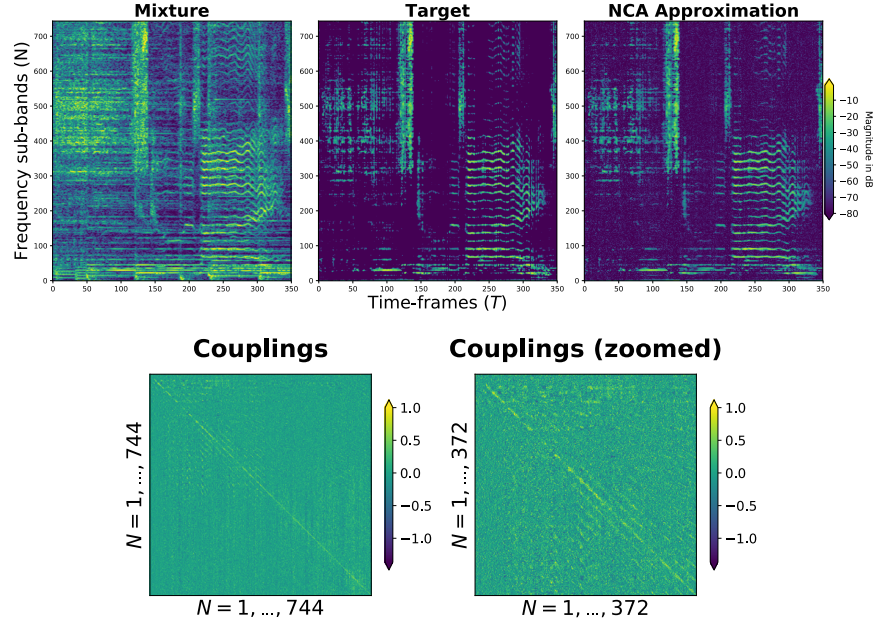
Table 3.2: Assessing the approximation performance of the mappings computed by the NCA, compared to the models’ outputs. The mean and standard deviation of the SNR, expressed in dB, are reported. For comparison, the linear composition and identity function are used. Bold faced values denote the best approximation performance.

Strategy	Model		
	DAE	MSS-DAE	SF
Student	<b>6.51(<math>\pm 1.79</math>)</b>	9.11( $\pm 1.09$ )	<b>6.53(<math>\pm 2.31</math>)</b>
Compositional	4.78( $\pm 2.69$ )	<b>9.25(<math>\pm 1.13</math>)</b>	5.98( $\pm 2.24$ )
Baseline	DAE	MSS-DAE	SF
Linear Comp.	0.01( $\pm 0.01$ )	−174.9( $\pm 18.4$ )	0( $\pm 0$ )
Identity Funct.	2.38( $\pm 0.81$ )	3.05( $\pm 0.66$ )	2.51( $\pm 1.03$ )

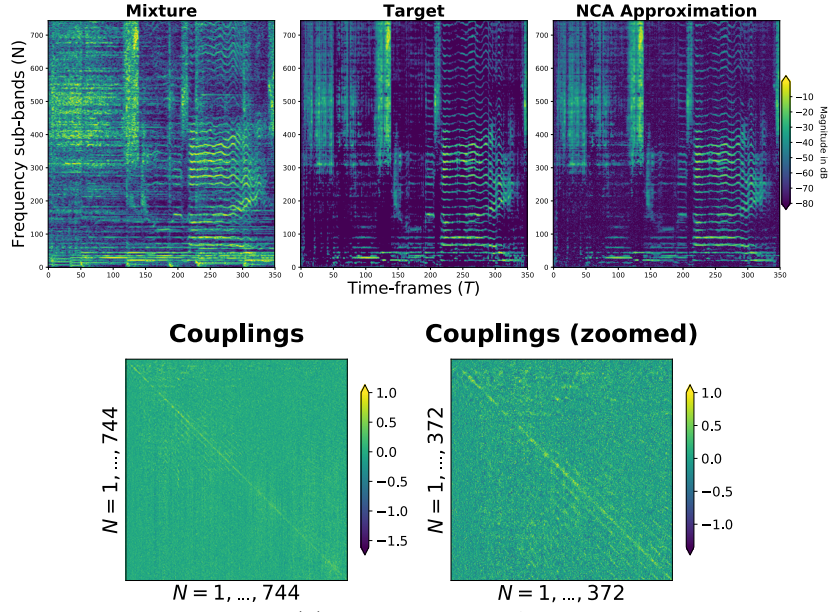
corresponding mapping function(s).

Additionally, Table 3.2 presents the results from the evaluation of the approximation performance of the NCA strategies by means of the SNR. The corresponding values are reported in dB. The average approximation performance of the NCA for the DAE and SF models and both strategies, outperforms the linear composition by approximately 6dB. For the case of the MSS-DAE model the approximation performance of the NCA is better than the linear composition by a very large margin that is greater than 100 dB. Closer experimental inspections suggest that the large margin is due to the exceeding norm of the spectra approximated by the linear composition. The exceeding norm of the spectra is itself due to the high norm of the row-vectors of  $\mathbf{C}$ , that is computed using the linear composition of the MSS-DAE’s weight matrices and bias vectors. This in turn, shows that by increasing the number of layers in non-linear source separation models, the linear composition is a poor proxy of the models’ mapping functions. As for the identity function, i.e., using the mixture instead of the models’ output spectral estimates, it is outperformed by the NCA on average, across strategies and models, by  $\sim 4$ dB. The student strategy outperforms the compositional strategy on average across the source separation models by 0.7 dB. This is due to the fact that the student strategy employs gradient updates that are related only to the reconstruction error minimization.

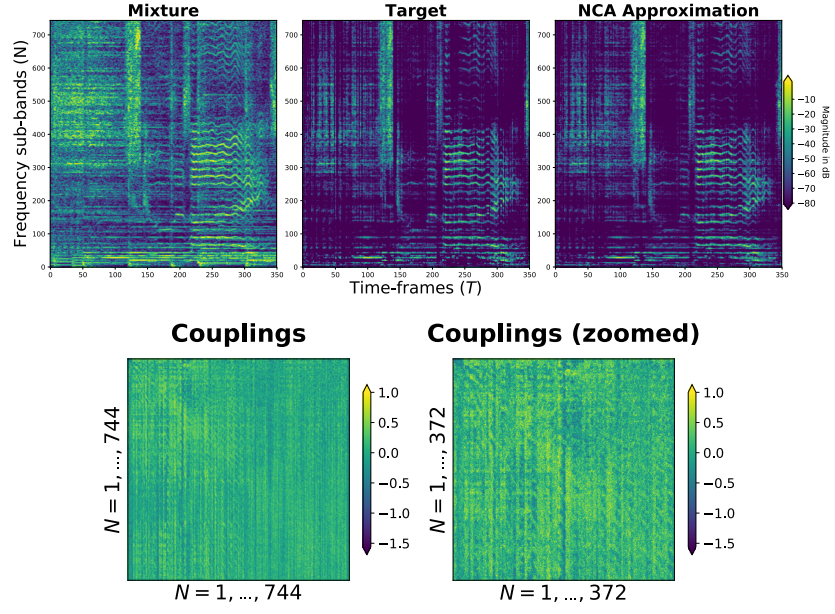
To further understand the approximation performance of the NCA, the employed segments in the test subset are used to calculate the SNR values between the NCA approximation and the true singing voice spectra. The results from this assessment are presented in Table 3.3. By observing the results in Table 3.3 it is highlighted that the NCA approximations are marginally better than those obtained directly with the models’ outputs.



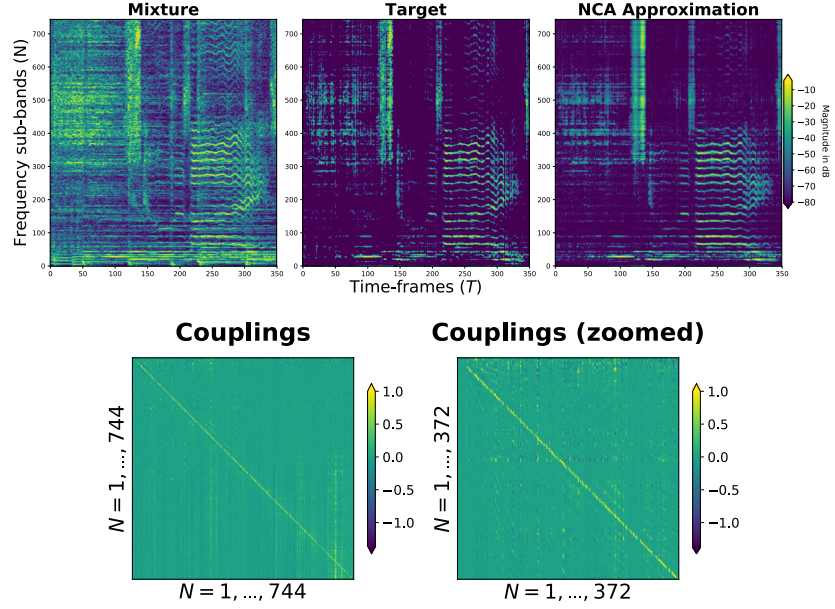
(a) Student: DAE



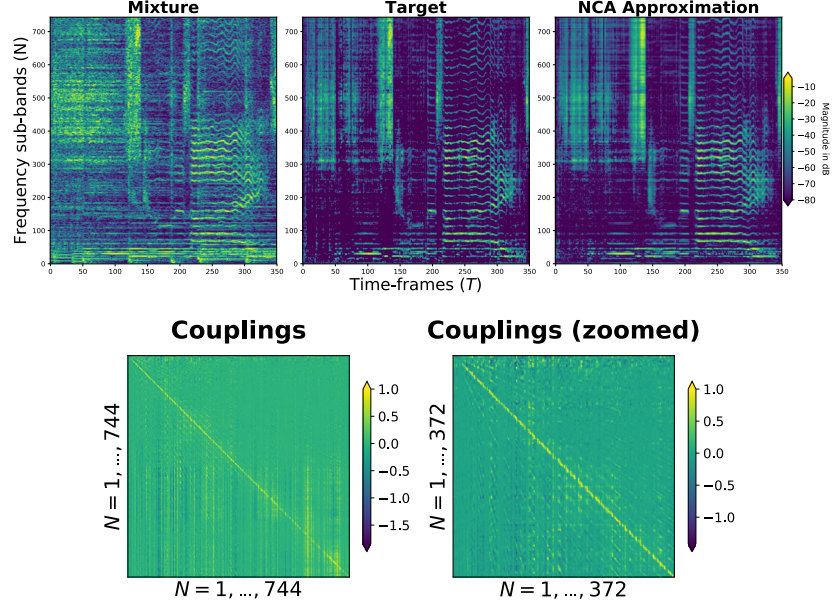
(b) Student: MSS-DAE



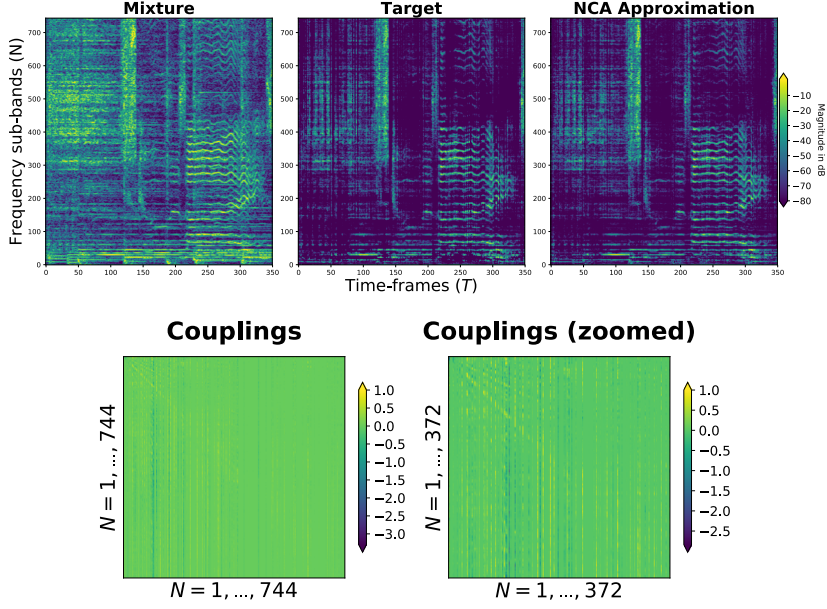
(c) Student: SF



(d) Compositional: DAE



(e) Compositional: MSS-DAE



(f) Compositional: SF

Figure 3.4: The outcome of the NCA for the DAE, MSS-DAE, and SF models using a  $\sim 3$  seconds excerpt from the file *Al James - Schoolboy Fascination* (test subset of MUSDB18) [A22]. Sub-figures (a)–(c): The couplings matrices and the corresponding spectral estimates using the *student* strategy. Sub-figures: (d)–(f): The couplings matrices and the corresponding spectral estimates using the *compositional* strategy.

Table 3.3: Assessing the approximation performance of the mappings computed by the NCA, compared to the true singing voice spectra. The mean and standard deviation of the SNR, expressed in dB, are reported. For comparison, the three separation models’ outputs are also reported.

Strategy	Model		
	DAE	MSS-DAE	SF
Student	$-4.02(\pm 2.43)$	$-1.94(\pm 2.03)$	$-4.88(\pm 3.06)$
Compositional	$-5.50(\pm 3.41)$	$-2.36(\pm 2.29)$	$-4.15(\pm 3.41)$
<b>Model Output</b>	$-5.05(\pm 3.05)$	$-2.17(\pm 2.43)$	$-4.74(\pm 3.15)$

More specifically and on average across strategies and models, a small improvement of  $\sim 0.17$  dB is observed. It should be denoted that the results presented in Table 3.2 and Table 3.3 do not aim at showing a correlation between the ability of a separation model to learning inter-frequency relationships and the objective performance of singing voice separation. Instead, Table 3.2 and Table 3.3 demonstrate the approximation performance of the NCA for various strategies and baselines.

Focusing on the structure of the mapping functions that are computed using the NCA via the student strategy, illustrated in the first row of Figure 3.4, it can be seen that the couplings matrices, serving as the corresponding mapping functions, are nearly identical between the DAE and MSS-DAE models, and marginally different from the SF model. The marginal differences can be explained by the fact that the couplings matrix for the SF model was optimized according to Eq. (3.17) using the output masks of the SF model as target function(s)  $\mathbf{Y}$ , as opposed to the DAE and MSS-DAE models that use the corresponding singing voice spectral estimates as target function(s).

The above tendency is also demonstrated in Table 3.1, where the statistics of the TOD-R metric for the student strategy are exactly the same for the DAE and the MSS-DAE models. This shows that the mappings are nearly identical between the models. This can be explained by recalling the Eq. (3.17) and Eq. (3.18) that depict the approximation of the mapping functions by observing only input-output and model-dependent relationships of spectral representations. This in turn, neglects the parameters of the model. The disuse of the model’s parameters during the optimization of the NCA is a convex problem that is experimentally shown to lead to almost the same solution during the experimental realization. However, according to [113] the disuse of the model parameters leads to system solutions that do not characterize the functionality of the non-linear model. Consequently, the attention is given on the compositional strategy, that includes the knowledge of the parameters of each model.

In the second row of Table 3.1 and in Figure 3.4d and Figure 3.4e, a pattern that is evident among the mapping functions of the DAE and MSS-DAE models is the high diagonal activity. As observed from Figure 3.4f, the main diagonal activity is not apparent in the mapping function of the SF model that employs the skip-filtering connections. Specifically, the mapping function of the SF model has pushed most of its activity away from the main diagonal. For small  $N$ , i.e., in the first matrix rows of the zoomed mapping function displayed in Figure 3.4f, it can be seen that elementary spectral structures are formed. Those spectral structures are related to the target spectra of Figure 3.4f as both



the spectral and the mapping function illustrations concentrate magnitude information in same frequency sub-band regions. According to the graphical model and its expected conditional, i.e., Figure 3.1c and  $q(x|\tilde{x})q(\tilde{x})$ , it can be underlined that the mapping function of the SF model captures the relevant structure of the target source that has been observed in the mixture. That structure is then used to construct time-frequency mask that can be applied to suppress the interfering sources. Nonetheless, the presented results do not provide evidence regarding the capacity of the SF model in learning spectral structures from the training data.

To this end, Figure 3.4 also shows that the additional layers that the MSS-DAE model employs are essentially used to model additional inter-frequency relationships, compared to the DAE model that comprises only two layers and concentrates most of its activity on the main diagonal. The MSS-DAE model not only pushes its activity to off-diagonal elements but also forms a structured matrix, mostly seen in the zoomed couplings matrix of Figure 3.4e, that is *roughly* similar to a circulant matrix with sparse entries. Those types of matrices are commonly used in digital signal processing for convolutional operators. This observation somewhat justifies the advantage of incorporating additional computational layers into a source separation model as proposed in [76], [93], and serves as an explanation on why convolutional layers are attractive choices in source separation models [99]. On the other hand, the DAE model has a simpler structure similar to a band matrix, with the minor off-diagonal activity denoting the spectral relationships, i.e., quasi-harmonic structures of the singing voice.

### 3.5 Summary

In this chapter, the main theme is the separation of singing voice separation by means of neural networks and particularly the most commonly used neural network model in music source separation, i.e., the denoising autoencoder (DAE) presented in [87]. To understand how DAEs process music mixture signals in order to separate music sources two research questions were formalized: **RQ1** - “*Why is masking important in approaches based on the DAE model?*”, and **RQ2** - “*Do DAEs commonly employed in music source separation learn trivial solutions for the given problem?*”.

In an attempt to answer those questions it is proposed to examine the mapping functions of the corresponding models applied to the particular problem of singing voice separation. For computing the mapping functions, an experimentally derived algorithm was proposed. The algorithm is denoted as the neural couplings algorithm (NCA), and two strategies are investigated for assisting the NCA to compute the mapping functions. The first strategy, denoted as the *student*, is based on the neural network distillation concept presented in [110]. It is observed that the student strategy leads to ambiguous results regarding the approximation of the mapping functions, as it does not account for the model’s parameters. As an alternative, the *compositional strategy* is proposed for taking into account the model’s already optimized parameters for the problem of singing voice separation. The compositional strategy encompasses, into the optimization objective, the algebraic definition of the mapping function of a model.

Using the compositional strategy and the computed mapping functions, the DAE model is investigated among its multi-layered extension employed in relevant tasks (MSS-DAE) [76], [93], [98], and the DAE with skip-filtering (SF) connections that are used to

mask the mixture spectra similarly to a time-frequency filtering operation [A6], [94], [96]. By examining the overall structure of the mapping functions, it can be concluded that the source separation models learn data-driven filtering functions when they are optimized for singing voice separation. Specifically, the DAE model learns trivial solutions because the corresponding encoding and decoding functions become symmetric during the training procedure. Consequently, the filtering functions learned by the DAE act as scalar filters in the frequency domain potentially limiting the overall source estimation performance.

Furthermore, employing the skip-connections as in the SF model can be seen as a simple method to enforce DAEs to learn richer inter-frequency dependencies compared to the DAE. This can justify the empirically observed performance boost over DAEs in previous works like [A10], [A11], [94], [96]. Finally, the additional computational layers employed in the MSS-DAE model, seem to promote the learning of filter kernels with a sparse and circulant structure roughly similar to convolutional operations. However, those kernels share many similarities with scalar filters, as in the case of the DAE, which reduce the overall filtering performance and support the experimental results in [93] that promote the usage of masking as a post-processing step for estimating the target source(s).

Although this study does not fully reflect the current trends in deep learning-based music source separation, it serves as a first step towards understanding what non-linear models learn from data when they are optimized to separate the singing voice. For instance, a model that uses skip connections is presented in [96]. That model is based on ladder-like concatenations. According to the results presented in [96], the improved performance in music source separation is observed when the skip-filtering connections were introduced to the model. In addition to this, the skip-filtering connections are also an important ingredient in end-to-end learning approaches. An example is the the Conv-TasNet model [120] that has surpassed the oracle performance in speech signal separation.

Directions for future research include the examination of another important type of skip connections that are commonly referred to as residual connections. Residual connections play an important role in signal enhancement and denoising [118] and have been used in music source separation [99] and evaluation [98]. The reason that residual connections have not been studied here is that the skip-filtering have played a much more important role in music source separation, due to their connection to masking that is a vastly used time-frequency operation. Finally, expanding the proposed study to more advanced architectures is also emerging. That is because the current study has solely focused on data-dependent inter-frequency relationships, whereas it is well known that temporal information conveys much information in music signals. It should be stated though, that the mappings, with respect to the frequency structure, for recurrent and convolutional neural networks are not expected to deviate significantly from the mappings presented in this work. That is because the signal estimation is based on vector and matrix products as the models examined in this chapter.





## Chapter 4

# Masker-and-Denoiser Architecture

### Preface

This chapter focuses on a learning algorithm, based on a neural network architecture, that learns masks for music source separation. That neural network architecture is denoted as the Masker-and-Denoiser (MaD) and it relies on the skip-filtering connections. The usage of these connections is based on the findings presented in Chapter 3, where the skip-filtering connections are shown to repel deep learning approaches to music source separation from learning trivial filtering operations for the problem of music source separation. The performance of the MaD architecture is assessed in tasks including monaural singing voice separation and harmonic-percussive source separation (HPSS). The content of this chapter is based on the publications [A6], [A9]–[A12], [A14]. The corresponding source code for the experiments, including audio examples for the separated source(s), can be accessed online<sup>1</sup>. Additional results for this chapter are included in Appendix D.

### 4.1 Introduction

Many deep learning approaches for music source separation rely on the computation of source-dependent masks [81], [88]. The application of source-dependent masks is most commonly performed in the frequency domain using the short-time Fourier transform (STFT), as described in Section 2.3.4 of Chapter 2 for the case of informed source separation. The difference with the case of informed source separation is that the mask is inferred, either explicitly or implicitly, by the deep neural network (DNN) that is conditioned on the information of the mixture signal.

There are three categories of approaches for inferring the source-dependent masks from the mixture signal using deep learning. These categories are: i) the *spectral approximation*, ii) the (explicit) *mask prediction*, and iii) a combination of the two previous categories

---

<sup>1</sup>Latest MaD models: [https://github.com/Js-Mim/mss\\_pytorch](https://github.com/Js-Mim/mss_pytorch), MaD with Twin Networks: <https://github.com/Js-Mim/mad-twinnet>, and HPSS experiments: <https://github.com/Js-Mim/phase-hpss>

that incorporate the masking operation as a part of the computational graph. The latter category uses the skip-filtering connections. Following the experimental results presented in Chapter 3, the usage of skip-filtering connections is justified as a simple method for DNNs to capture the structure of the target source from within the observed mixture signal. This chapter proposes a neural network architecture for music source separation based on these connections. The architecture is denoted as the MaD architecture and is evaluated for its performance in singing voice and accompaniment source separation and HPSS. The rest of this chapter is organized as follows: Section 4.2 provides information regarding previous works in audio and music source separation that are related to the proposed architecture. Section 4.3 gives a detailed description of the proposed architecture including extensions regarding the mask inference. Section 4.4 provides technical details regarding the training objectives of the proposed architecture. Section 4.5 describes the followed experimental procedure, followed by Section 4.6 that presents and discusses the results from the conducted experiments. Section 4.7 summarizes this chapter.

## 4.2 Related Work

### 4.2.1 Spectral Approximation Approaches

The approaches in this category try to first estimate the target and interfering source(s) from the mixture signal, and then post-process the estimates by means of generalized Wiener filtering. The output of the post-processing step is the final output of these approaches. More formally, let  $\mathbf{Y}_m \in \mathbb{C}^{T \times N'}$  be the STFT of the mixture signal, with  $N'$  frequency sub-bands and  $T$  time-frames. Furthermore, let  $|\mathbf{Y}_j|^\alpha \in \mathbb{R}_{\geq 0}^{T \times N'}$  be the  $\alpha$ -power magnitude spectrogram of the  $j$ -th source computed using the STFT. The goal of the approaches in this category is to learn a single or multiple functions  $f_j^{(1)}(\cdot)$ , depending on the number of the target sources, for estimating the magnitude of  $J$  sources. The input to the functions is the mixture's magnitude spectrograms, learning the following mapping:

$$f_j^{(1)} : |\mathbf{Y}_m|^\alpha \mapsto |\mathbf{Y}_j|^\alpha, \forall j \in J.$$

The above mapping strategy is widely adopted, since it's implementation is straightforward by using the denoising autoencoder (DAE) model [86], [87]. Differentiating from the DAE model, the additive noise corresponds to the addition of other sources. Nonetheless, the experimental findings presented in Chapter 3 suggest that the DAE model learns trivial solutions to the problem of singing voice separation, leading to sub-optimal separation performance. One way to improve the separation performance is to post-process the estimated magnitude spectrograms by applying source-dependent masks. This post-processing step has been employed by nearly all the approaches in the first category. The masks computed for the post-processing step are using the estimated source magnitude spectrograms [76], [93], i.e., the outputs of each  $f_j^{(1)}(\cdot)$  after the training procedure. The computation of the masks is based on a soft-masking (SM) method, by using each source's estimated magnitude spectrogram  $|\mathbf{Y}_j|^\alpha$  to Eq. (2.43).

More specifically, in [93] it is proposed to train multiple, one for each target source, multi-layered DAEs, using feed-forward neural network (FNN). Each multi-layered DAE estimates the magnitude spectrogram of the corresponding target source and is trained

in a greedy and layer-wise fashion<sup>2</sup>. The training is performed using audio signals from a non-public data-set containing various music instruments, including the singing voice source. After the training procedure the estimated target sources' spectrograms, from each multi-layered DAE, are used to compute source-dependent masks that are applied to the mixture. Similarly in [121], it is proposed to extend the usage of DAEs by an adaptive fine-tuning scheme that uses normal auto-encoders<sup>3</sup>. The auto-encoders fine-tune the predictions of the DAEs for the target sources. Then, the fine-tuned spectrograms for each source are used to compute source-dependent soft-masks that are applied to the mixture. The latter results into the final estimates of various audio and music sources, including the singing voice and accompaniment.

Similarly, in [76] it is proposed to train multiple, one for each target source, multi-layered DAEs using FNNs, to predict the power spectral density (PSD), i.e., the power magnitude spectrograms for  $\alpha = 2$ , of the target sources. The PSD of the monaural mixture is used as input to the multi-layered DAE. The estimated sources' PSDs are then used to compute the multi-channel, two-channel in this case, Wiener filtering using the model presented in [75] and refined using an iterative scheme based on expectation-maximization. Combining the ideas from [93] and [76], the work presented in [98] proposes to use bi-directional long short-term memory (LSTM) layers to predict the PSDs of the target sources. Then, the time-domain sources are first estimated using the inverse short-time Fourier transform (ISTFT) and the mixture phase information. Finally, the sources are refined by recomputing the STFT of the estimated time-domain source signals, and using the multi-channel Wiener filtering to obtain the refined spectral estimates. The refined spectral estimates of each target source are then transformed to the time-domain using the ISTFT. Then, the reconstructed time-domain signals are linearly mixed with the corresponding estimated sources from the exact method presented in [93]. In an attempt to decrease the extensive computations used by the above mentioned separation approaches, in [122] it is proposed to use two-dimensional convolutional neural network (CNN), to estimate the PSDs of the target sources, that are then refined by the multi-channel Wiener filtering, as in [76], [98]. Tow-dimensional CNNs are also proposed by the work presented in [99], where different convolutional kernels, employed by the two-dimensional CNNs, are applied to sub-sets of frequency sub-bands of the mixture's magnitude spectrogram. The estimated target sources' magnitude spectrograms, are then used as input to the multi-channel Wiener filtering to yield the final spectral estimates of the target sources. The approach presented in [99] has led to state-of-the-art results in music source separation, with the reported performance relying on additional training data that are not publicly available.

---

<sup>2</sup>In the greedy and layer-wise training setup and in the context of the work presented in [93], each computational layer in the corresponding DAE, i.e., each individual FNN, is optimized to yield the magnitude spectrogram of the target source. The input to that FNN is the output of the previously trained FNN. In the case of the first layer, the mixture magnitude spectrogram is used. After all FNNs have been trained, the FNNs are cascaded to form the multi-layered DAE model, that is then fine-tuned with the objective of estimating the magnitude spectrogram of the target source. The greedy layer-wise technique has been extensively used prior to the research that examines better initialization strategies for DNNs [64].

<sup>3</sup>Similar to the DAE model but without any corruption or process applied to the input signal.

### 4.2.2 Mask Prediction Approaches

The approaches of the second category aim at explicitly predicting the source-dependent mask(s). More specifically, allow  $\mathbf{M}_j^*$  to denote the  $j$ -th mask computed by utilizing an appropriate masking method, such as a soft-masking (SM) method or a binary mask (BM). Then, the goal of the mask prediction approaches is to learn a function  $f^{(2)}(\cdot)$ , that maps the mixture magnitude spectrogram to the source dependent mask  $\mathbf{M}_j^*$  of size  $T \times N'$ , i.e.,

$$f^{(2)} : |\mathbf{Y}_m| \mapsto \mathbf{M}_j^*, \forall j \in J.$$

Although the strategy of explicitly predicting the source-dependent masks is attractive, as there is no need to estimate the interfering source(s) in order to compute the target source's mask, there are two severe limitations to this approach. The first limitation is that defining a loss function for minimization is an open question. The second limitation is that the learning of the function  $f^{(2)}(\cdot)$  relies on a training procedure that uses sub-optimal masks as the target output. The target output masks are pre-computed using a data-set of multi-track audio signals. The sub-optimality is due to the fact that the non-linear mixing parameters of the target source are unknown, i.e., the function  $\mathcal{G}_{c_h, j}(\cdot)$  from the assumed relaxed mixing model<sup>4</sup> is not known. In practice, it is very common to pre-compute BMs or masks derived from a SM method, and then use these masks as training target output(s) [90], [123]–[125]. This choice disregards that BMs induce music distortions in the separated signal and the computation of SM assumes that the sources are additive. For instance, in [123] it is proposed to use a DNN comprising of two-dimensional CNNs for predicting the BMs for the singing voice and accompaniment sources, respectively. Similarly, in [124] it is proposed to use multiple DNNs, based on FNNs, to estimate both BMs and soft-masks for each target source. The estimated masks are then linearly combined together to compute a soft-mask, that reduces the musical distortions and artifacts, that are induced from the application of the predicted BM to the mixture spectrogram. To this end, the method presented in [90] and [125] builds on the previously mentioned method of [124], by introducing a post-processing step of the masked mixture spectrograms using an optimized DAE. The DAE is optimized similarly to the spectral approximation approaches, and it is shown in [124] to improve the separation performance of singing voice and accompaniment sources in terms of signal-to-distortion ratio (SDR) and signal-to-interference ratio (SIR). An alternative approach that is based on spectral clustering [126] is presented in [89]. In [89] a supervised approach to clustering, using deep bi-directional LSTMs, is used to compute features for discriminating between the singing voice and the accompaniment sources. These discriminative features are used to predict both binary and soft masks for the separation of the previously mentioned music sources.

### 4.2.3 Skip-filtering connection Approaches

In order to alleviate the problem of using pre-computed and sub-optimal masks as training targets for the DNN, the strategy employing the skip-filtering connections<sup>5</sup> combines

<sup>4</sup>Interested readers are referred to Section 2.3.3 of Chapter 2.

<sup>5</sup>The term skip-filtering connections is described in Chapter 3. This term is mostly used in music source separation. For speech enhancement and separation literature, the same approach is denoted as the signal approximation method; a term easily confused with spectral approximation.

the two previous categories of approaches, i.e., the spectral approximation and the explicit mask prediction approaches, by allowing the masking operation to be a part of the optimization. The goal of this category of approaches is to learn a function  $f^{(3)}(\cdot)$  that predicts the source-dependent and optimized, according to a loss function, mask  $\mathbf{M}_j^o$  given the magnitude spectrogram of the mixture signal, i.e.,

$$f^{(3)} : |\mathbf{Y}_m| \mapsto \mathbf{M}_j^o, \text{ subject to } |\mathbf{Y}_j| \approx \mathbf{M}_j^o \odot |\mathbf{Y}_m| \text{ and } \forall j \in J.$$

The strategy with the skip-filtering connections serves as a simple extension to the DAE model, that allows the DAE model to implicitly mask the input mixture signal. The applied mask is provided by the output of the model and the application of the mask results into a filtered version of the mixture signal. The filtered signal is used as an approximation of the true target source’s magnitude spectrogram. In contrast to the mask prediction, the approaches based on skip-filtering connections do not require pre-computed masks to be optimized.

In the context of neural network-based source separation, this strategy is proposed in [94] for the separation of speech signals. The work presented in [94] involves the training of deep LSTM networks to yield the source-dependent<sup>6</sup> masks that filter the input mixture magnitude spectrum. The parameters of the deep LSTM network are optimized using the magnitude spectrogram of the target speaker(s) as target output. Focusing on singing voice and accompaniment source separation, the work presented in [95] proposes to use RNNs to predict the magnitude spectrograms of the singing voice and the accompaniment source, similar to the spectral approximation approach. Then, the magnitude estimates are given to a deterministic function that involves the computation of a soft-mask, based on the generalized Wiener filtering. The output mask is applied to the mixture. According to [95], although the approximation of the time-frequency mask is subject to optimization, the optimization is based on the ability of the neural network layers to estimate the spectrogram of the sources. Consequently, the method presented in [95] does not allow the deep RNNs to learn the masking process, but rather to output magnitude estimates that can be used to compute soft-masks as a ratio of magnitude spectrograms.

With the main ambition to also learn the masking process using DNNs, the author’s work, previously published in [A6], proposes to directly mask the mixture magnitude spectrum with the output of the model based on bi-directional RNNs and particularly gated recurrent units (GRUs). This approach has been extended in [A10], [A11], introducing the architecture denoted as the MaD and extensions such as the recurrent inference [A10] and the usage of twin networks [A11]. The technical details regarding the MaD architecture and the extensions are given in the following sections. The MaD architecture is considered one of the state-of-the-art approaches in SiSEC 2018 campaign [127]. It should be stated that the skip-filtering connections have been proposed also in [96] for singing voice and accompaniment source separation. The main difference between [96] and the proposed architecture is that in [96] it is proposed to use two-dimensional CNNs instead of RNNs, employed in [A6], [A10], [A11], and the usage of additional skip-connections that leak, by means of concatenation, the representations computed by previous layers to the following ones. Furthermore, the experimental results presented in [96] suggest that the skip-filtering connections are mostly responsible for the observed improvements in the separation performance of the singing voice.

<sup>6</sup>In the case of speech separation, each source refers to a single speaker.

More recent approaches, employ the skip-filtering connections for semi-supervised learning in singing voice separation [128]. In addition to that, the skip-filtering connections are the main ingredient of current state-of-the-art approaches to music source separation that yield remarkable results [100], [129], [130]. For instance, in [100], [129] it is proposed to use the two-dimensional CNNs, as in [96], combined with post-processing steps based on the generalized Wiener filtering<sup>7</sup>. Furthermore, the model presented in [100], [129] is trained on additional and non publicly available data. To this end, in [130] it is proposed to use bi-directional LSTMs and (data) normalization layers, combined with post-processing steps based on the generalized Wiener filtering as in [100] and [129].

## 4.3 Proposed Architecture

### 4.3.1 Overview of the Architecture

The proposed architecture takes as input the time-domain samples of the mixture signal and outputs the time-domain samples of the target source. The architecture consists of four modules. An illustration of the proposed architecture and its corresponding modules is given in Figure 4.1. The first module performs the signal analysis and the pre-processing of the analyzed signal. The second module accepts as input the analyzed and pre-processed signal and predicts the source-dependent mask. The mask is applied to the analyzed signal, creating a filtered version of the corresponding signal. The filtered version is the first estimate of the target source that is then given to the third module. The third module enhances the first estimate of the target source by predicting and applying a denoising filter. Finally, the output of the third module is given to the fourth module that constructs the time-domain samples of the target source. The second module of the proposed architecture is denoted as the “*Masker*”, and the third module is denoted as the “*Denoiser*”<sup>8</sup>. The Masker and the Denoiser modules are denoted separately, because the Masker is optimized to predict the source-dependent time-frequency mask that yields the first estimate of the source, whereas the Denoiser is optimized to enhance the output of the Masker by additionally predicting and applying a time-frequency mask.

The Masker consists of an *encoder*, a *decoder*, and the skip-filtering connections between the input to the Masker and the output of the decoder. More specifically, the encoder is implemented using a single layer of a bi-directional RNN [59] and is denoted as the  $\text{RNN}_{\text{enc}}(\cdot)$ . The decoder consists of a single layer of a uni-directional RNN, denoted as the  $\text{RNN}_{\text{dec}}(\cdot)$ , and the sparsifying transform implemented using a feed-forward neural network (FNN) layer. The Denoiser is implemented using two FNN layers that are used for encoding and decoding the frequency information from the Masker, and are denoted as the  $\text{FNN}_{\text{enc}}(\cdot)$  and the  $\text{FNN}_{\text{dec}}(\cdot)$ , respectively. The output of the  $\text{FNN}_{\text{dec}}(\cdot)$  is used to mask the input to the Denoiser, using the the skip-filtering connections. The Masker and

---

<sup>7</sup>The post-processing steps based on the generalized Wiener filtering, include estimating all the sources with multiple, one for each target source, DNNs with skip-filtering connections. These estimates are then used to compute soft-masks that are applied to the mixture. This particular post-processing step is understood as a fusion of information from multiple separation models, based on DNNs, that have been trained to separate different music sources. The fusion of separation models is empirically shown to yield further improvements to the overall separation quality.

<sup>8</sup>Not to be confused with the definition of the denoising autoencoder (DAE) [86].

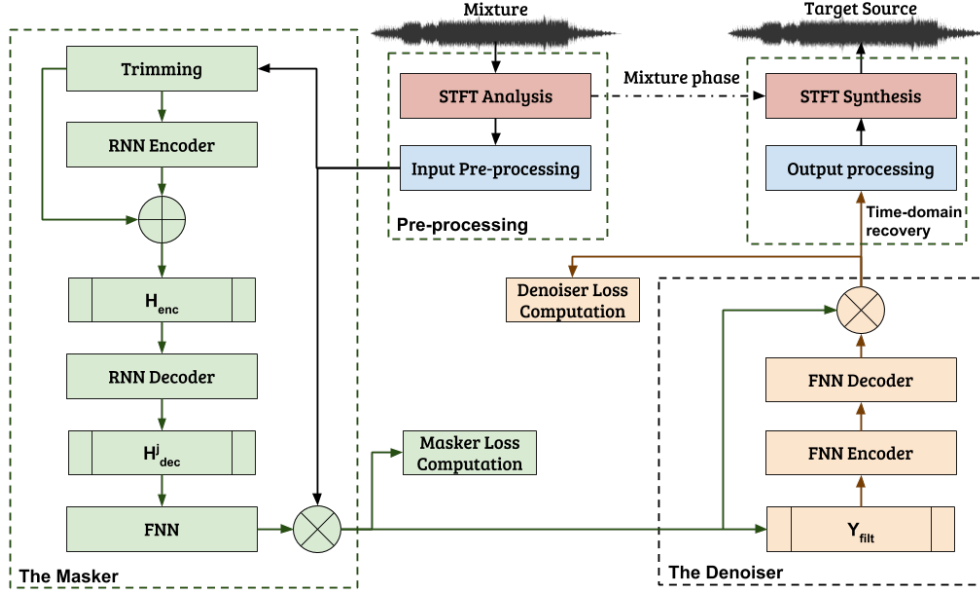


Figure 4.1: Overview of the Masker-and-Denoiser (MaD) architecture for music source separation. Circled crosses denote element-wise multiplication. Illustration reproduced from [A10].

the Denoiser modules are jointly optimized using stochastic gradient in a supervised fashion. All the previously mentioned RNNs in the proposed architecture are gated recurrent units (GRUs), following the Eqs. (2.36)–(2.39) discussed in Chapter 2.

### 4.3.2 Signal Analysis & Input Pre-processing

Let  $\mathbf{x}_m \in \mathbb{R}_{[-1,1]}^{T'}$  be the vector containing  $T'$  time-domain samples of the monaural mixture of  $J$  sources, sampled at 44.1 kHz. The monaural signal  $\mathbf{x}_m$  is used to obtain a time-frequency representation of the mixture signal using the STFT. The STFT is computed using a Hamming windowing function, that is 2049 samples long (46 ms). The step-size of the STFT is 384 samples (8.7 ms). Prior to the application of the discrete Fourier transform (DFT) matrix, each segmented time-frame is zero-padded to  $N' = 4096$  samples. This is performed in order to increase the frequency resolution that leads to higher disjointness of the sources in the computed time-frequency representation [30], without smearing the time resolution [18].

Subsequent to the STFT of  $\mathbf{x}_m$ , only the positive frequencies including the DC term are retained, i.e., the first  $N = 2049$  frequency sub-bands are used and the rest are omitted. This yields the complex-valued time-frequency representation of  $\mathbf{x}_m$  that is denoted as  $\mathbf{Y}_m \in \mathbb{C}^{T \times N}$ . Using the complex-valued representation of the mixture, the phase and the magnitude spectrograms are computed. The mixture signal's phase spectrogram is used for reconstructing the time-domain samples of the estimated target source. The mixture's magnitude spectrogram, denoted as  $|\mathbf{Y}_m| \in \mathbb{R}_{\geq 0}^{T \times N}$ , is then split in  $B = \lceil T/M \rceil$  sub-

sequences, where  $T$  is the number of time-frames and  $M$  is the length of the sequence. The splitting of  $|\mathbf{Y}_m|$  into sub-sequences is performed to train the GRUs in a computationally efficient manner. That is because the sequence length of music signal spectrograms is most of the times very large. By recalling the necessary recursive Eqs. (2.36)–(2.39) used by the GRUs to compute the hidden representations, it can be deduced that the process of high dimensional and lengthy time-frequency information leads to extremely slow computations of the gradients required for optimizing the corresponding parameters of the corresponding GRU.

The computed sub-sequences overlap by an empirical factor of  $2L$  time-frames, where  $L$  is an integer denoting the overlap factor in time-frames. The overlap between sub-sequences is introduced in order to use the overlapping time-frames as context information for the encoding stage performed by the  $\text{RNN}_{\text{enc}}(\cdot)$ . Each overlapping sub-sequence contains  $M$  frames of the mixture’s magnitude spectrogram  $|\mathbf{Y}_m|$  and is denoted as  $\mathbf{Y}_{\text{in}} \in \mathbb{R}_{\geq 0}^{M \times N}$ , where the subscript “ $\cdot_{\text{in}}$ ” denotes the input to the Masker. Specifically, each  $\mathbf{Y}_{\text{in}}$  is computed as

$$\mathbf{Y}_{\text{in}} = \begin{bmatrix} |\mathbf{Y}_m[t', 0]| & \cdots & |\mathbf{Y}_m[t', N-1]| \\ \vdots & \ddots & \vdots \\ |\mathbf{Y}_m[t'+M-1, 0]| & \cdots & |\mathbf{Y}_m[t'+M-1, N-1]| \end{bmatrix}, \quad (4.1)$$

where  $t'$  is an integer indicating the time-frame index in the mixture magnitude spectrogram  $|\mathbf{Y}_m|$ . The integer  $t'$  is calculated using the following expression:

$$t' = (b-1)M - (b-1)2L,$$

where  $b$  is an integer smaller or equal than the total number of sub-sequences  $B$ , i.e.,  $b \in \mathbb{Z}_{[1, B]}$ . The explicit indication of  $b$  in  $\mathbf{Y}_{\text{in}}$  is dropped for clarity in the notation. In the case that the above expression results into time-frame indices that exceed the number of time-frames  $M$  in the mixture spectrogram, zero-padding is applied, i.e., if  $t' > M$ , then  $|\mathbf{Y}_m[t', n]| := 0 \forall n \in [0, 1, \dots, N-1]$ . The usage of overlapping time-frames can be seen as having temporal context information of  $L$  time-frames before and  $L$  time-frames after (a total of  $2L$  time-frames), that is used for the  $\text{RNN}_{\text{enc}}(\cdot)$ . The usage of temporal context information is useful for both deep learning based music source separation [93], [98] and for discovering meaningful temporal patterns in sequences using RNNs [52].

### 4.3.3 The Masker

The input to the Masker is each sub-sequence  $\mathbf{Y}_{\text{in}}$  that is computed from the input pre-processing module. Only for, and prior to the encoding of each  $\mathbf{Y}_{\text{in}}$  a low-bandwidth version of  $\mathbf{Y}_{\text{in}}$  is computed. This is done by preserving only the first  $F$  frequency sub-bands in each time-frame, yielding the frequency-reduced sequence of magnitude information that is denoted as  $\mathbf{Y}_{\text{tr}} \in \mathbb{R}_{\geq 0}^{M \times F}$ . This operation is denoted as trimming in Figure 4.1, hence the subscript “ $\cdot_{\text{tr}}$ ”. The trimming operation is useful for reducing the number of trainable parameters of the Masker and is dependent on the hyper-parameter  $F$ . The hyper-parameter  $F$  is chosen according to the expected frequency bandwidth that is necessary to detect the target source(s) from the observed mixture spectrogram. Particularly for the prediction of the mask that separates the singing voice or the harmonic sources,



the first  $F = 744$  frequency sub-bands are retained. This set of frequency sub-bands includes bandwidth information up to 8 kHz, where the most relevant spectral content is located for harmonic and quasi-harmonic music instruments [7].

The trimmed magnitude spectrogram of the mixture  $\mathbf{Y}_{\text{tr}}$  is used as input to the encoder  $\text{RNN}_{\text{enc}}(\cdot)$  that is implemented as a single-layered bi-directional GRU. The bi-directional GRU consists of a forward GRU and a backward GRU. The parameters of each GRU, i.e., the matrices and biases, are not shared between the forward and the backward GRUs. The forward GRU takes  $\vec{\mathbf{Y}}_{\text{tr}}$  as input, which is equal to the sequence  $\mathbf{Y}_{\text{tr}}$ , and the backward GRU takes as input the sequence-reversed version of  $\mathbf{Y}_{\text{tr}}$ , denoted as  $\overleftarrow{\mathbf{Y}}_{\text{tr}}$ . The time-reversed version of  $\mathbf{Y}_{\text{tr}}$  is computed as

$$\overleftarrow{\mathbf{Y}}_{\text{tr}} = \begin{bmatrix} \mathbf{Y}_{\text{tr}[M-1,0]} & \cdots & \mathbf{Y}_{\text{tr}[M-1,F-1]} \\ \vdots & \ddots & \vdots \\ \mathbf{Y}_{\text{tr}[0,0]} & \cdots & \mathbf{Y}_{\text{tr}[0,F-1]} \end{bmatrix}.$$

The arrows “ $\vec{\cdot}$ ” and “ $\overleftarrow{\cdot}$ ” indicate the direction of the sequence processed by the forward and backward GRU, respectively. After the processing of the sequences by the forward and the backward GRUs, the resulting hidden representations<sup>9</sup> are updated by means of residual connections [131]. The residual connections are computed between  $\vec{\mathbf{Y}}_{\text{tr}}$ ,  $\overleftarrow{\mathbf{Y}}_{\text{tr}}$  and the corresponding outputs from the forward and backward GRU, respectively. The residual connections assist in the more stable training of the bi-directional GRU [52]. After the application of the residual connections, the updated representations are concatenated together to construct  $\tilde{\mathbf{H}}_{\text{enc}} \in \mathbb{R}^{M \times 2F}$ . More formally,  $\tilde{\mathbf{H}}_{\text{enc}}$  is computed as

$$\tilde{\mathbf{H}}_{\text{enc}} = \begin{bmatrix} \vec{\mathbf{H}}_{[0,0]} + \vec{\mathbf{Y}}_{\text{tr}[0,0]} & \cdots & \vec{\mathbf{H}}_{[M-1,0]} + \vec{\mathbf{Y}}_{\text{tr}[M-1,0]} \\ \vdots & \ddots & \vdots \\ \vec{\mathbf{H}}_{\text{tr}[0,F-1]} + \vec{\mathbf{Y}}_{\text{tr}[0,F-1]} & \cdots & \vec{\mathbf{H}}_{[M-1,F-1]} + \vec{\mathbf{Y}}_{\text{tr}[M-1,F-1]} \\ \overleftarrow{\mathbf{H}}_{[0,0]} + \overleftarrow{\mathbf{Y}}_{\text{tr}[0,0]} & \cdots & \overleftarrow{\mathbf{H}}_{[M-1,0]} + \overleftarrow{\mathbf{Y}}_{\text{tr}[M-1,0]} \\ \vdots & \ddots & \vdots \\ \overleftarrow{\mathbf{H}}_{[0,F-1]} + \overleftarrow{\mathbf{Y}}_{\text{tr}[0,F-1]} & \cdots & \overleftarrow{\mathbf{H}}_{[M-1,F-1]} + \overleftarrow{\mathbf{Y}}_{\text{tr}[M-1,F-1]} \end{bmatrix}^{\top},$$

where  $\vec{\mathbf{H}}, \overleftarrow{\mathbf{H}} \in \mathbb{R}_{[-1,1]}^{M \times F}$  are the outputs of the forward and of the backward GRU, respectively. The residual connections are used to allow the gradients to flow through the encoder directly, resulting into a more stable gradient-based optimization of the parameters of the MaD architecture [131].

With the main ambition to enforce the decoder  $\text{RNN}_{\text{dec}}(\cdot)$  to focus only on the sequence frames that are relevant to the decoding of the source-dependent and optimized mask,  $\tilde{\mathbf{H}}_{\text{enc}}$  is further processed by a sub-sampling operation. The sub-sampling operation drops the first  $L$  and the last  $L$  time-frames, that are used as context information for the Masker.

<sup>9</sup>The term hidden representation follows the definition of the GRU given in the Eqs.(2.36)–(2.39) of Chapter 2. It should be mentioned that *hidden state(s)* is an alternative and frequently used term in related literature.

This results in the output of the encoder  $\mathbf{H}_{\text{enc}} \in \mathbb{R}^{M' \times 2F}$ , i.e.,

$$\mathbf{H}_{\text{enc}} := \text{RNN}_{\text{enc}}(\vec{\mathbf{Y}}_{\text{tr}}, \overleftarrow{\mathbf{Y}}_{\text{tr}})$$

with  $M' < M$ , and  $\mathbf{H}_{\text{enc}}$  is computed by sub-sampling  $\tilde{\mathbf{H}}_{\text{enc}}$  as

$$\mathbf{H}_{\text{enc}} = \begin{bmatrix} \tilde{\mathbf{H}}_{\text{enc}}[L, 0] & \cdots & \tilde{\mathbf{H}}_{\text{enc}}[L, 2F-1] \\ \vdots & \ddots & \vdots \\ \tilde{\mathbf{H}}_{\text{enc}}[M-1-L, 0] & \cdots & \tilde{\mathbf{H}}_{\text{enc}}[M-1-L, 2F-1] \end{bmatrix}.$$

The output of the encoder  $\mathbf{H}_{\text{enc}}$  is then given to the decoder  $\text{RNN}_{\text{dec}}(\cdot)$ . The  $\text{RNN}_{\text{dec}}(\cdot)$  is implemented using a single-layered GRU which outputs the hidden representation  $\mathbf{H}_{\text{dec}}^j \in \mathbb{R}_{[-1,1]}^{M' \times 2F}$ , i.e.,

$$\mathbf{H}_{\text{dec}}^j = \text{RNN}_{\text{dec}}(\mathbf{H}_{\text{enc}}).$$

Then,  $\mathbf{H}_{\text{dec}}^j$  is used as input to the sparsifying transform that yields an approximation of the  $j$ -th source's optimized mask. The approximated mask is denoted as  $\tilde{\mathbf{M}}_j \in \mathbb{R}_{\geq 0}^{M' \times N}$  and has the full spectral bandwidth as the mixture magnitude spectrogram. The sparsifying transform is implemented using a FNN, followed by the application of the element-wise rectified linear unit (ReLU)(Eq. (3.12)) function as

$$\tilde{\mathbf{M}}_j = \text{ReLU}(\mathbf{H}_{\text{dec}}^j \mathbf{W}_{\text{mask}} + \mathbf{1}_{M'} \mathbf{b}_{\text{mask}}^\top). \quad (4.2)$$

In Eq. (4.2)  $\mathbf{W}_{\text{mask}} \in \mathbb{R}^{2F \times N}$  is the weight matrix of the sparsifying transform and  $\mathbf{b}_{\text{mask}} \in \mathbb{R}^N$  is the corresponding bias vector of the sparsifying transform. The bias vector is added to all time-frames  $M'$  after its product with the vector  $\mathbf{1}_{M'}$ . The transform is characterised as sparsifying due to the application of the ReLU function [27] that sets to zero the negative values after the above described operations using  $\mathbf{W}_{\text{mask}}$  and  $\mathbf{b}_{\text{mask}}$ .

The output of the Masker is obtained by using the skip-filtering connections between the predicted mask  $\tilde{\mathbf{M}}_j$  and the sub-sampled version of the mixture magnitude spectrogram  $\mathbf{Y}_{\text{in}}$ . The sub-sampling is performed with respect to the time-frames and is denoted as  $\mathbf{Y}'_{\text{in}} \in \mathbb{R}_{\geq 0}^{M' \times N}$ . Formally,  $\mathbf{Y}'_{\text{in}}$  is computed as

$$\mathbf{Y}'_{\text{in}} = \begin{bmatrix} \mathbf{Y}_{\text{in}}[L, 0] & \cdots & \mathbf{Y}_{\text{in}}[L, N-1] \\ \vdots & \ddots & \vdots \\ \mathbf{Y}_{\text{in}}[M-1-L, 0] & \cdots & \mathbf{Y}_{\text{in}}[M-1-L, N-1] \end{bmatrix},$$

and is then filtered by the predicted mask  $\tilde{\mathbf{M}}_j$  as

$$\mathbf{Y}_{\text{filt}} = \tilde{\mathbf{M}}_j \odot \mathbf{Y}'_{\text{in}}. \quad (4.3)$$

The output of Eq. (4.3) is the filtered version of the sub-sampled mixture magnitude spectrogram, denoted as  $\mathbf{Y}_{\text{filt}} \in \mathbb{R}_{\geq 0}^{M' \times N}$ , and serves as a first estimate of the target source's magnitude spectrogram. The subscript “ $\cdot_{\text{filt}}$ ” is used to denote that the corresponding magnitude spectrogram has undergone a filtering process.

#### 4.3.4 The Recurrent Inference Algorithm

In many cases the effective modeling and processing of long sequences relies on the usage of multiple computational layers of RNNs [1], [52]. Experimental results in speech and music source separation research presented in [94], [98], suggest that the additional computational layers of RNNs increase the objective separation performance of the corresponding architecture. However, the number of computational layers is chosen heuristically. In addition to this, the increase of computational layers employing RNNs and particularly either GRUs or LSTMs, dramatically increases the number of parameters to be optimized and makes the gradient-based optimization of the added layers harder [52], [53]. To avoid the previously described problems and being inspired by deep learning methods where the number of computational layers is stochastic [132], the *recurrent inference algorithm* is proposed. The recurrent inference algorithm is based on the method presented in [132], but with the difference that the parameters to be optimized, i.e., the weight matrices and bias vectors, are shared among the computational layers used by the proposed algorithm.

---

##### Algorithm 2 The Recurrent Inference Algorithm

---

**Require:** Function of the recurrent decoder  $\text{RNN}_{\text{dec}}(\cdot)$ , output of the encoder  $\mathbf{H}_{\text{enc}}$ , maximum number of iterations  $iter$ , and the termination threshold  $\tau_{\text{term}}$ ,

```

1:  $\mathbf{S}_0^j \leftarrow \text{RNN}_{\text{dec}}(\mathbf{H}_{\text{enc}})$ 
2:  $i \leftarrow 0$ 
3: while  $i \leq iter$  do
4:    $\mathbf{H}_{\text{dec}}^j \leftarrow \text{RNN}_{\text{dec}}(\mathbf{S}_{i-1}^j)$ 
5:   if  $\|\mathbf{S}_{i-1}^j - \mathbf{H}_{\text{dec}}^j\|_2^2 \geq \tau_{\text{term}}$  then
6:      $\mathbf{S}_i^j \leftarrow \mathbf{H}_{\text{dec}}^j$ 
7:      $i \leftarrow i + 1$ 
8:   else
9:     Terminate the process
10:  end if
11: end while
12: return  $\mathbf{H}_{\text{dec}}^j$ 
```

---

Subject to the MaD architecture, the recurrent inference algorithm is used in the Masker and processes the latent representation computed by  $\text{RNN}_{\text{dec}}(\cdot)$ . The latent representation computed using  $\text{RNN}_{\text{dec}}(\cdot)$  affects the predicted mask. The recurrent inference algorithm is employed by the Masker with the main ambition to improve the separation performance of the Masker. The recurrent inference is an iterative process and consists in reevaluating the source-dependent latent representation  $\mathbf{H}_{\text{dec}}^j$ , produced by the  $\text{RNN}_{\text{dec}}(\cdot)$ . The iterative process is terminated when the convergence criterion is reached. This circumvents the need for specifying a fixed number of application of the  $\text{RNN}_{\text{dec}}(\cdot)$ . The stopping criterion is based on the threshold value  $\tau_{\text{term}}$ , operating on the  $L_2$  matrix norm ( $\|\cdot\|_2^2$ ) of the differences between the consecutive estimates, i.e., the estimates of  $\mathbf{H}_{\text{dec}}^j$  from one iteration to the other. The comparison between consecutive estimates of  $\mathbf{H}_{\text{dec}}^j$  practically indicates how much did  $\mathbf{H}_{\text{dec}}^j$  change compared to its previous evaluation using  $\text{RNN}_{\text{dec}}(\cdot)$ . A maximum number of iterations ( $iter$ ) is used to avoid having infinite iterations until the convergence criterion is met, i.e., the  $L_2$  matrix norm of dif-

ferences is smaller than  $\tau_{\text{term}}$ . The recurrent inference is given in Algorithm 2 and its benefits in singing voice separation are presented and discussed in Section 4.6 among a few hyper-parameter choices.

### 4.3.5 The Denoiser

The output of the masker  $\mathbf{Y}_{\text{filt}}$  is further processed by the Denoiser. The Denoiser is used because the Masker is expected to allow interferences from other active sources to the magnitude spectrogram of the target source<sup>10</sup>. One explanation to the allowed interferences is because the Masker consists of only three computational layers, the encoder, the decoder, and the sparsifying transform, that could potentially decrease the learning capabilities of the Masker [1]. In more details, the Denoiser is a two-layer denoising auto-encoder [86] with skip-filtering connections [A22]. The Denoiser consists of the FNN encoder ( $\text{FNN}_{\text{enc}}(\cdot)$ ), and the FNN decoder ( $\text{FNN}_{\text{dec}}(\cdot)$ ). The parameters of the  $\text{FNN}_{\text{enc}}(\cdot)$  and of the  $\text{FNN}_{\text{dec}}(\cdot)$  are shared between the time-frames of the spectrogram sequence. The output of the Denoiser is denoted as  $\hat{\mathbf{Y}}_{\text{seq}}^j \in \mathbb{R}_{\geq 0}^{M' \times N}$  and is the estimated magnitude spectrogram of the target source. The subscript “ $_{\text{seq}}$ ” is used to indicate that the output of the Denoiser is a sub-sequence of the initial magnitude spectrogram. Formally, the output of the Denoiser  $\hat{\mathbf{Y}}_{\text{seq}}^j$  is computed as

$$\hat{\mathbf{Y}}_{\text{seq}}^j = \text{FNN}_{\text{dec}}(\text{FNN}_{\text{enc}}(\mathbf{Y}_{\text{filt}})) \odot \mathbf{Y}_{\text{filt}}, \quad (4.4)$$

where

$$\text{FNN}_{\text{enc}}(\mathbf{Y}_{\text{filt}}) := \text{ReLU}(\mathbf{Y}_{\text{filt}} \mathbf{W}_{\text{enc}} + \mathbf{1}_{M'} \mathbf{b}_{\text{enc}}^{\top}),$$

$\mathbf{W}_{\text{enc}} \in \mathbb{R}^{N \times \lfloor N/2 \rfloor}$  is the weight matrix of the FNN encoder, and  $\mathbf{b}_{\text{enc}} \in \mathbb{R}^{\lfloor N/2 \rfloor}$  is the bias vector that is added to the  $M'$  time-frames of the encoded information of  $\mathbf{Y}_{\text{filt}}$ . The dimensionality of the encoded information is approximately half of  $N$ . This is done in an attempt to enforce the Denoiser to encode the relevant information of the filtered, by the Masker, magnitude spectrogram [117]. The decoding process of the Denoiser is defined as

$$\text{FNN}_{\text{dec}}(\text{FNN}_{\text{enc}}(\mathbf{Y}_{\text{filt}})) := \text{ReLU}(\text{FNN}_{\text{enc}}(\mathbf{Y}_{\text{filt}}) \mathbf{W}_{\text{dec}} + \mathbf{1}_{M'} \mathbf{b}_{\text{dec}}^{\top}),$$

where  $\mathbf{W}_{\text{dec}} \in \mathbb{R}^{\lfloor N/2 \rfloor \times N}$  is the weight matrix of the FNN decoder and  $\mathbf{b}_{\text{dec}} \in \mathbb{R}^N$  is the bias vector that is added to the  $M'$  time-frames of the Denoiser’s decoded mask.

### 4.3.6 Output Processing & Synthesis

All the previously mentioned operations performed by the Masker and the Denoiser are applied to all  $B$  overlapping sub-sequences, that are computed using Eq. (4.1). Each output sub-sequence of the Denoiser  $\hat{\mathbf{Y}}_{\text{seq}}^j \in \mathbb{R}_{\geq 0}^{M' \times N}$  is combined with the rest of the Denoiser’s output sub-sequences. Due to the induced overlap between the sub-sequences, the inverse operation of Eq. (4.1) is a time-frame wise concatenation of the sub-sequences. The result of the concatenation is the full-sequence magnitude spectrogram of the  $j$ -th source denoted as  $\hat{\mathbf{Y}}^j \in \mathbb{R}_{\geq 0}^{M \times N}$ . By exploiting the symmetry between positive and negative frequency sub-bands in the representation computed by the STFT and by using the phase information of the mixture, the time-domain samples of the target source  $\hat{\mathbf{x}}_j \in \mathbb{R}_{[-1,1]}^{T'}$  are computed using the ISTFT.

<sup>10</sup>The expectation is verified using experimental results demonstrated in Section 4.6.

## 4.4 Training Objectives

To optimize the parameters of the Masker and the Denoiser, i.e., the weight matrices and the bias vectors employed in the operations described in the previous sub-sections, a training process is performed. The training process aims at minimizing the empirical loss of the MaD architecture between the estimated and the true spectrogram of the target source. The empirical loss is computed as a linear combination of two terms that are namely: i) *the reconstruction* term, and ii) *the penalty* term. The empirical loss is termed as  $E_*^{\text{MaD}} \in \mathbb{R}_{\geq 0}$  and is computed as

$$E_*^{\text{MaD}} = \underbrace{E_*^{\text{M}} + E_*^{\text{D}}}_{\text{reconstruction}} + \underbrace{\omega E_*^{\text{R}}}_{\text{penalty}}, \quad (4.5)$$

where  $E_*^{\text{M}} \in \mathbb{R}_{\geq 0}$  and  $E_*^{\text{D}} \in \mathbb{R}_{\geq 0}$  are the loss values that are specific to the Masker and to the Denoiser, respectively. The computation of the Masker-specific loss value  $E_*^{\text{M}}$  follows the probabilistic graphical model of the skip-filtering connections, illustrated in Figure 3.1c of Chapter 3. In practice, the computation of  $E_*^{\text{M}}$  enforces the Masker to predict a source-dependent mask rather than a time-variant gain envelope, that is applied to the mixture’s magnitude spectrogram  $\mathbf{Y}_{\text{in}}$ . The mask prediction is enforced by training the Masker to reconstruct the magnitude spectrogram of the target source. The computation of the Denoiser-specific loss value  $E_*^{\text{D}}$ , in addition to  $E_*^{\text{M}}$ , provides the the objective to the Denoiser to further enhance the output of the Masker, resulting into an increased separation performance of the MaD architecture.

In Eq. (4.5),  $\omega$  is a scalar that controls the strength of the penalty term in the computation of the empirical loss  $E_*^{\text{MaD}}$ . The penalty’s loss value is denoted as  $E_*^{\text{R}} \in \mathbb{R}_{\geq 0}$  and is computed using the Masker. The superscript  $\cdot^{\text{R}}$  is used to distinguish the penalty term from the Masker and the Denoiser specific objectives. The computation of the loss value is performed only for the Masker and not for the Denoiser, since the Masker consists of far more many trainable parameters than the Denoiser, that might lead to poor separation performance during the evaluation stage of the MaD architecture. Also, the Masker is implemented using RNNs that are most of the times very difficult to optimize for effectively processing long temporal sequences [52], [53], [133], thus reducing the performance of RNNs in processing music spectrograms.

The asterisk “\*” in Eq. (4.5) is used for brevity and it replaces the subscripts that denote the different loss functions used for optimizing the MaD architecture. In particular, two different loss functions are considered for computing the loss values of the reconstruction term in Eq. (4.5). These two loss functions are the mean squared error (MSE), denoted as  $\mathcal{L}_{\text{MSE}}(\cdot)$ , and the generalized Kullback-Leibler (KL) divergence, that is denoted as  $\mathcal{L}_{\text{KL}}(\cdot)$ . The  $\mathcal{L}_{\text{MSE}}(\cdot)$  is examined because it is employed by state-of-the-art approaches in music source separation [76], [93], [98], [99]. On the other hand, the KL divergence is examined because it is shown in [29], [70] that the KL divergence is a good criterion for comparing the magnitude spectrograms of music sources, following the assumption that the distribution of the magnitude spectrograms are approximated by tailed distributions such as the Laplacian distribution.

More formally, allow  $\mathcal{F}_{\text{Masker}}(\cdot)$  and  $\mathcal{F}_{\text{Denoiser}}(\cdot)$  to denote the (parameterized) func-

tions of the Masker and the Denoiser, respectively, so that

$$\begin{aligned}\mathbf{Y}_{\text{filt}} &:= \mathcal{F}_{\text{Masker}}(\mathbf{Y}_{\text{in}}), \text{ and} \\ \hat{\mathbf{Y}}_{\text{seq}}^j &:= \mathcal{F}_{\text{Denoiser}}(\mathbf{Y}_{\text{filt}}).\end{aligned}$$

Then, the loss values specific to the Masker and to the Denoiser using the MSE loss function  $\mathcal{L}_{\text{MSE}}(\cdot)$  are computed as

$$\begin{aligned}E_{\text{MSE}}^{\text{M}} &= \mathcal{L}_{\text{MSE}}(|\mathbf{Y}_{\text{seq}}^j|, \mathbf{Y}_{\text{filt}}) \\ E_{\text{MSE}}^{\text{D}} &= \mathcal{L}_{\text{MSE}}(|\mathbf{Y}_{\text{seq}}^j|, \hat{\mathbf{Y}}_{\text{seq}}^j),\end{aligned}$$

where  $|\mathbf{Y}_{\text{seq}}^j| \in \mathbb{R}_{\geq 0}^{M' \times N}$  is the sub-sequence of the target source's magnitude spectrogram, computed using the STFT followed by Eq. (4.1), and the sequence sub-sampling operation used by the Masker. The MSE between two matrices  $\mathbf{A}, \hat{\mathbf{A}} \in \mathbb{R}^{M' \times N}$  is defined as

$$\mathcal{L}_{\text{MSE}}(\mathbf{A}, \hat{\mathbf{A}}) := \frac{1}{M' N} \|\mathbf{A} - \hat{\mathbf{A}}\|_2^2, \quad (4.6)$$

where  $\|\cdot\|_2$  denotes the  $L_2$  matrix norm. The loss values using the  $\mathcal{L}_{\text{KL}}(\cdot)$  are computed as

$$\begin{aligned}E_{\text{KL}}^{\text{M}} &= \mathcal{L}_{\text{KL}}(|\mathbf{Y}_{\text{seq}}^j|, \mathbf{Y}_{\text{filt}}) \\ E_{\text{KL}}^{\text{D}} &= \mathcal{L}_{\text{KL}}(|\mathbf{Y}_{\text{seq}}^j|, \hat{\mathbf{Y}}_{\text{seq}}^j).\end{aligned}$$

The KL divergence between two non-negative matrices  $\mathbf{B}, \hat{\mathbf{B}} \in \mathbb{R}_{\geq 0}^{M' \times N}$  is defined as

$$\mathcal{L}_{\text{KL}}(\mathbf{B}, \hat{\mathbf{B}}) := \frac{1}{M' N} \|\mathbf{B} \odot \log\left(\frac{\mathbf{B}}{\hat{\mathbf{B}} + \epsilon}\right) - \mathbf{B} + \hat{\mathbf{B}}\|_1, \quad (4.7)$$

where  $\log(\cdot)$  is the element-wise logarithmic function,  $\|\cdot\|_1$  is the  $L_1$  matrix norm, and  $\epsilon = 1e^{-24}$  is a small value to ensure numerical stability. It should be noted that the usage of  $\mathcal{L}_{\text{KL}}(\cdot)$  treats the frequency sub-band magnitude values, in the employed magnitude spectrograms  $|\mathbf{Y}_{\text{seq}}^j|$ ,  $\mathbf{Y}_{\text{filt}}$ , and  $\hat{\mathbf{Y}}_{\text{seq}}^j$ , as un-normalized probability values.

To compute the loss value  $E_{*}^{\text{R}}$  of the penalty term, two loss functions are examined. These loss functions are namely the  $L_1$  based loss function, that yields the loss value  $E_{L_1}^{\text{R}} \in \mathbb{R}_{\geq 0}$ , and the loss function computed using the Twin Network (TwinNet) technique, presented in [133]. The first loss function is based on the  $L_1$  matrix norm that is computed using the weight matrix of the sparsifying transform employed by the Masker, i.e., the  $\mathbf{W}_{\text{mask}} \in \mathbb{R}^{2F \times N}$ . The purpose of this loss function is to further promote the sparsity of the predicted target-source mask. Formally, the loss value using the normalized  $L_1$  matrix norm is computed as

$$E_{L_1}^{\text{R}} = \frac{1}{2F N} \|\mathbf{W}_{\text{mask}}\|_1. \quad (4.8)$$

Using the above definitions for computing the corresponding loss values and Eq. (4.5), the following training objectives are formed to train the MaD architecture:

$$E_{\text{MSE}/L_1}^{\text{MaD}} = E_{\text{MSE}}^{\text{M}} + E_{\text{MSE}}^{\text{D}} + \omega E_{L_1}^{\text{R}} \quad (4.9)$$

$$E_{\text{KL}/L_1}^{\text{MaD}} = E_{\text{KL}}^{\text{M}} + E_{\text{KL}}^{\text{D}} + \omega E_{L_1}^{\text{R}}. \quad (4.10)$$



spectrograms of music mixtures. Furthermore, twin networks offer an alternative and less computationally expensive technique to effectively process sequences compared to the recurrent inference described in Section 4.3.4. The  $\text{RNN}_{\text{dec}}(\cdot)$  is trained with the loss function computed using the TwinNet, hypothesizing that the decoded time-frequency mask, for separating the target source, should remain the same regardless the direction of the sequence computed using the  $\text{RNN}_{\text{dec}}(\cdot)$ . In other words, the direction in time, e.g., forward or backward, that one chooses to process the mixture’s magnitude spectrogram should be independent from the structure of the time-frequency mask that separates the target source.

More formally, let  $\text{RNN}_{\text{twin}}(\cdot)$  be the twin of the  $\text{RNN}_{\text{dec}}(\cdot)$ . The input sequence to  $\text{RNN}_{\text{dec}}(\cdot)$ , denoted as  $\mathbf{H}_{\text{enc}}$ , is also given to  $\text{RNN}_{\text{twin}}(\cdot)$  but reversed, i.e.,

$$\mathbf{H}_{\text{twin}}^j = \text{RNN}_{\text{twin}}(\overleftarrow{\mathbf{H}_{\text{enc}}}) .$$

The output of  $\text{RNN}_{\text{twin}}(\cdot)$ , denoted as  $\mathbf{H}_{\text{twin}}^j \in \mathbb{R}_{[-1,1]}^{M' \times 2F}$  is used to compute the loss value for  $\text{RNN}_{\text{dec}}(\cdot)$  as

$$E_{\text{twin}}^{\text{R}} = \|\mathbf{H}_{\text{dec}}^j \mathbf{W}_{\text{aff}} - \mathbf{H}_{\text{twin}}^j\|_2, \quad (4.11)$$

where  $\mathbf{W}_{\text{aff}} \in \mathbb{R}^{2F \times 2F}$  is a trainable, affine, and linear transformation of  $\mathbf{H}_{\text{dec}}^j$ , that allows small perturbations to the output of  $\text{RNN}_{\text{dec}}(\cdot)$ . The affine transformation is used only for the computation of the  $L_2$  matrix norm of the differences between  $\mathbf{H}_{\text{dec}}^j$  and  $\mathbf{H}_{\text{twin}}^j$ . Using  $E_{\text{twin}}^{\text{R}}$  in Eq. (4.11) the last examined objective for the MaD architecture is defined as

$$E_{\text{KL/twin}}^{\text{MaD}} = E_{\text{KL}}^{\text{M}} + E_{\text{KL}}^{\text{D}} + \omega E_{\text{twin}}^{\text{R}}. \quad (4.12)$$

As shown in [133], the effectiveness of the twin network to penalize the  $\text{RNN}_{\text{dec}}(\cdot)$  is based on the ability of the twin network to reconstruct the target sequence from the observed reversed latent representation<sup>11</sup>. To do so, the twin network also duplicates the sparsifying transform and the skip-filtering connections that are used to compute the magnitude of the target source. That magnitude estimate is used to compute a reconstruction loss value, that is denoted as  $E_{\text{KL}}^{\text{twin}} \in \mathbb{R}_{\geq 0}$ . The loss value  $E_{\text{KL}}^{\text{twin}}$  is computed as

$$E_{\text{KL}}^{\text{twin}} = \mathcal{L}_{\text{KL}}(|\mathbf{Y}_{\text{seq}}^j|, \mathbf{Y}_{\text{twin}}), \quad (4.13)$$

where  $\mathbf{Y}_{\text{twin}} \in \mathbb{R}_{\geq 0}^{M' \times N}$  is the spectrogram output of the twin network, that is computed using the duplicated sparsifying transform followed by the skip-filtering connections, as shown in Fig. 4.2. More specifically,  $\mathbf{Y}_{\text{twin}} \in \mathbb{R}_{\geq 0}^{M' \times N}$  is a filtered version of the mixture magnitude spectrogram  $\mathbf{Y}'_{\text{in}}$  and is computed as

$$\mathbf{Y}_{\text{twin}} = \text{ReLU}(\mathbf{H}_{\text{twin}}^j \mathbf{W}_{\text{twin}} + \mathbf{1}_{M'} \mathbf{b}_{\text{twin}}^{\top}) \odot \mathbf{Y}'_{\text{in}}. \quad (4.14)$$

In Eq. (4.14)  $\mathbf{W}_{\text{twin}} \in \mathbb{R}^{2F \times N}$  and  $\mathbf{b}_{\text{twin}} \in \mathbb{R}^N$  are the weight matrix and the bias vector of the duplicated sparsifying transform, respectively. The loss value  $E_{\text{KL}}^{\text{twin}}$  is detached from the calculation of gradients used for optimizing the parameters of the MaD architecture, and therefore it does not contribute to the calculation of  $E_{\text{KL/twin}}^{\text{MaD}}$  using Eq. (4.12). This is

<sup>11</sup>In [133] it is proposed to maximize the log-likelihood of the observed data given the reversed sequence. In the context of this work, the log-likelihood maximization corresponds to reconstructing the spectrogram of the target source.



done in order to prevent the preceding differentiable modules of the Masker in assisting the minimization of Eq. (4.11). This enforces only the parameters of the twin network to be optimized using Eq. (4.12) and thus computing representations  $\mathbf{H}_{\text{dec}}^j$  that are meaningful for penalizing the  $\text{RNN}_{\text{dec}}(\cdot)$ . After the training procedure of the MaD architecture, the twin network is discarded and it does not contribute to the separation of the target source.

## 4.5 Experimental Procedure

The goal of the experimental procedure is to experimentally justify the suitability of the MaD architecture in music source separation tasks. The considered music source separation tasks are singing voice separation and HPSS, and the employed data-set is the MUSDB18 data-set [A8]. For the experiments focusing on singing voice and accompaniment separation, the accompaniment signal is computed as the sum of the bass, drums, and other music instruments, such as synthesizers, contained in the MUSDB18 data-set. For the HPSS, the harmonic source is computed as the sum of the singing voice, the bass, and the other music instruments. The percussive source is the drums track of the previously mentioned data-set.

With a focus on singing voice separation, the perceptual relevance of the previously described training objectives is also investigated. The investigation is performed by conducting listening tests based on the multiple stimulus with hidden reference and anchors (MUSHRA) standard. The degradation of the hidden anchor relies on the synthetic computation of artifacts, interference, and distortion components using the clean target source signal, as previously proposed in [78], [79] and described in Section 2.4.2. To this end, the best performing model, based on the obtained experimental results, is compared to other competitive methods from the spectral approximation and mask prediction approaches. It should be noted, that the comparison presented herein serves only the purpose of highlighting the potentials of the proposed architecture, given the most competitive and related, to the MaD architecture, approaches.

### 4.5.1 Training

The MaD architecture is trained multiple times using various target sources, training objectives, and using extensions, such as the recurrent inference. To that aim, the 100 two-channel multi-tracks from the MUSDB18 data-set are used. The multi-tracks are sampled at 44.1 kHz. For each multi-track, the time-domain signals of the monaural version of the mixture and of the target source, either the singing voice or the harmonic or the percussive source, are down-mixed to a monaural version by averaging the two available channels<sup>12</sup>. The time-frequency representations of the monaural time-domain signals are computed by the input pre-processing module of the proposed architecture described in Section 4.3.2. Then, the time-frequency representations of the respective signals are segmented into sub-sequences using Eq. (4.1).

For computing  $B = \lceil T/M \rceil$  sub-sequences that are used to optimize the parameters of the MaD architecture, the length of each sub-sequence is set to  $M = 60$  time-frames,

---

<sup>12</sup>Initial experiments indicated that training with time-frequency representations of monaural signals yields equivalent separation performance, with respect to the corresponding objective metrics, compared to the training using the information from the two available channels.

modelling a sequence of approximately 0.5 seconds. The hyper-parameter that controls the number of time-frames used as context information for the sequence processing using the Masker is  $L = 10$ , i.e., the first 10 and the last 10 time-frames in each sub-sequence. For the recurrent inference algorithm, described in Section 4.3.4, the termination threshold is set to  $\tau_{\text{term}} = 1e^{-3}$  and two iteration values  $iter = [3, 10]$  are examined. All of the previously mentioned hyper-parameters are chosen based on an small scale experimental procedure involving a grid search of the respective hyper-parameters, the evaluation of the separation performance using a smaller sub-set of the training data-set, and the available computational resources.

In the training objectives, two scalar values are considered for controlling the strength of the penalty term. These values are  $\omega = 0$  and 0.5. The value 0 is used to examine if the penalty term has any effect on the separation performance, and the value 0.5 was chosen experimentally according to the balance between the loss values of the reconstruction and penalty terms in Eqs. (4.9)–(4.10), and Eq.(4.12). The conducted experiments for each one of the above considered variations of the MaD architecture, i.e., different hyper-parameter for the recurrent inference or a different training objective, use a random initialization with fixed seed to ensure a fair comparison. For all the considered RNNs contained in the Masker, the hidden-to-hidden matrices are initialized with random values and scaled according to the method presented in [62]. All the other matrices are initialized with values drawn from a normal distribution and scaled using the method presented in [64]. The respective bias vectors are initialized with zeroes. All the parameters are jointly optimized using the Adam algorithm [50], with a learning rate equal to  $1e^{-3}$  and the default coefficients for the first and second-order moments. The batch size is set to 16 sub-sequences. Prior to each gradient update, the  $L_2$  norm of each computed gradient is clipped to 0.5 for ensuring a stable training of the employed RNNs [52]. The total number of iterations throughout the whole training data-set is set to 100, and the training procedure is terminated if the average training loss value has not been decreased after 3 consecutive iterations.

### 4.5.2 Evaluation

For evaluating the separation performance of the MaD architecture, including the extensions, the rest of the 50 tracks from the MUSDB18 data-set are used. From each track the left and right channels are given independently as input to the MaD architecture, yielding two-channel estimates of the target sources. This is done in an attempt to assess the performance of the MaD architecture *without* any post-processing or enhancement steps. To estimate the accompaniment source in the singing voice separation case study, spectral subtraction is employed after the estimation of the singing voice magnitude spectrogram. For HPSS the MaD architecture is trained for each target source independently. Then, the estimated sources from the proposed architecture and the true sources, from the evaluation sub-set of MUSDB18, are used to compute the signal-to-distortion ratio (SDR), the signal-to-interference ratio (SIR), and the signal-to-artifacts ratio (SAR) objective measures. The computation of the objective measures is performed for one-second segments of the corresponding files, and the median values, across segments and tracks, of the measures are reported.

For assessing the effect of the recurrent inference algorithm, the focus is given on

singing voice and accompaniment source separation. Each optimized version of the MaD that uses a different values for  $iter$ , is used to compute the SDR, SIR, and SAR values. In addition to these objective measures, the average time required to process a sequence using the  $RNN_{dec}(\cdot)$ , employed by the Masker, is calculated. Specifically, the average time in ms is calculated for the  $RNN_{dec}(\cdot)$  to process the output of the bi-directional GRU encoder  $\mathbf{H}_{enc}$ , yielding the source dependent latent information of the mask  $\mathbf{H}_{dec}^j$ . The length of the sequences is approximately 0.5 seconds long, and the calculations are performed on a computer with an 8-core Intel Xeon CPU (3.50GHz), 32 GB of RAM, and the Nvidia Titan X GPU.

For showcasing the performance of the MaD architecture, a comparison with state-of-the-art approaches for deep learning-based music source separation is also conducted. To limit the number of possible evaluation configurations the best performing variation of the MaD architecture is used and compared with the following approaches: i) *UHL* [98] – A spectral approximation approach involving multiple three-layered DAEs using bi-directional LSTMs data-augmentation, during the training procedure, and followed by a post-processing step involving the multi-channel Wiener filtering for separating the singing voice and accompaniment sources, ii) *GRA* [90] – An approach for (explicit) mask prediction using FNNs for predicting both binary and soft masks that are then linearly combined for singing voice and accompaniment source separation, iii) *HUA+* [95] – The first approach that incorporates the masking process into the computational graph of the DNN, re-implemented using a three-layered bi-directional LSTM that yields the magnitude spectrograms of the singing voice and accompaniment sources that are then used to compute soft-masks that are applied to each channel, iv) *LPR* [129] – A state-of-the-art and very recent model for music source separation that uses the skip-filtering connections and two-dimensional CNNs, followed by a post-processing step that fuses the outcomes of multiple separation models that are optimized with different target sources, and v) *LOR* [134] – A state-of-the-art approach in deep learning based HPSS, that uses two-dimensional CNNs and the skip-filtering connections, without post-processing steps. In the case that the compared approaches have been engineered and trained on the previous version of MUSDB18 data-set, i.e., the DSD100 data-set that comprises fewer multi-tracks for training, the previously described steps for training and evaluation are repeated for the MaD architecture using the DSD100 data-set. In that case, the computation of the objective measures is based on the BSSEval version 3, following the SiSEC evaluation rules. Due to the increased number of experimental results, the results from the comparison with other supervised approaches for singing voice and accompaniment separation are presented and discussed in the Appendix D.

### Listening Tests

The listening tests are based on the MUSHRA standard [84] and are conducted to cross-validate whether perceptual quality improvements for singing voice separation, could be perceived when using any of the examined training objectives. The training objectives employed in the subjective evaluation are the KL-based objective  $E_{KL/L_1}^{MaD}$ , the MSE-based objective  $E_{MSE/L_1}^{MaD}$ , and the KL-based objective with the twin network  $E_{KL/twin}^{MaD}$ , as described in Section 4.4. The choice of these objectives is based on the singing voice separation performance, reported and discussed in Section 4.6. The singing voice separation is considered

as a case study due to its high importance in music source separation [81], [88].

To that aim, a total number of 7 participants (after post-screening) participated in the listening tests. All of the participants have an experience in audio signal processing. A selection of six 10-second segments from the test sub-set of MUSDB18 data-set was used as test corpus in the listening tests<sup>13</sup>. The segments used in the listening tests were chosen such that they include 3 male and 3 female singers, ensuring timbre and genre diversity. The anchor signals were generated to resemble distortions produced by source separation algorithms, following the procedure described in Section 2.4.2. The participants of the listening tests were asked to rate the *general separation quality*, in the context of the experimental findings presented in [135] for the subjective definition of separation quality. All participants were asked to rate the test content using a scale from 0 to 100, divided into 5 equal intervals with the following quality descriptors: *bad* [0-20], *poor* [20-40], *fair* [40-60], *good* [60-80], and *excellent* [80-100]. Each participant went through a training phase for getting familiar with the goal of the listening tests and the GUI used in the tests. The average rating with respect to the test segments is reported.

## 4.6 Results & Discussion

### 4.6.1 Singing Voice & Accompaniment Source Separation

#### The effect of the Denoiser

To justify the usage of the Denoiser module in the MaD architecture, Table 4.1 and Table 4.2 demonstrate the separation performance for singing voice and accompaniment source separation, respectively. The separation performance is based on the calculation of the median values for the SDR, SIR, and SAR objective measures in dB (the higher the better), by either omitting (“✗”) or incorporating (“✓”) the Denoiser in the separation of the target source during the evaluation<sup>14</sup>.

Table 4.1: The effect of the Denoiser module on the objective singing voice separation performance, using two training objectives. Both training objectives are computed using  $\omega = 0.5$ . Values in boldface denote the best obtained performance.

Objective	Denoiser	SDR (dB)	SIR (dB)	SAR (dB)
$E_{\text{KL}/L_1}^{\text{MaD}}$	✗	3.92	8.41	<b>4.59</b>
	✓	4.02	8.63	4.53
$E_{\text{MSE}/L_1}^{\text{MaD}}$	✗	4.02	7.81	4.28
	✓	<b>4.10</b>	<b>8.68</b>	4.41

The results in Table 4.1 show that the Denoiser further enhances the output estimates of the Masker by marginally improving the overall SDR and decreasing the interferences

<sup>13</sup>Audio corpus can be publicly accessed from <https://zenodo.org/record/1476866>.

<sup>14</sup>Training the Denoiser separately from the Masker and then evaluating the MaD with and without the Denoiser, leads to identical results with training the Masker combined with the Denoiser and omitting the Denoiser only at the evaluation stage.

Table 4.2: The effect of the Denoiser module on the objective performance in accompaniment separation, computed using the objective evaluation measures and the MUSDB18 data-set. Values in boldface denote the best obtained performance.

Objective	Denoiser	SDR (dB)	SIR (dB)	SAR (dB)
$E_{\text{KL}/L_1}^{\text{MaD}}$	$\times$	9.93	12.49	<b>12.94</b>
	$\checkmark$	10.03	12.60	12.91
$E_{\text{MSE}/L_1}^{\text{MaD}}$	$\times$	9.93	12.67	12.21
	$\checkmark$	<b>10.11</b>	<b>12.99</b>	12.53

from other sources. In particular for the  $E_{\text{KL}/L_1}^{\text{MaD}}$  training objective, by incorporating the Denoiser the median SDR is improved by 0.10 dB, the SIR is improved by 0.22 dB, but the SAR decreases by 0.06 dB. Also, for the  $E_{\text{MSE}/L_1}^{\text{MaD}}$  training objective, the effect of the Denoiser is more prominent than  $E_{\text{KL}/L_1}^{\text{MaD}}$ , since the median SIR value is increased by 0.87 dB, compared to the usage of only the Masker for separating the target source. When the Denoiser is used marginal differences in the SDR and SAR values are observed. Specifically, SDR is improved by 0.08 dB and the SAR is decreased by 0.13 dB.

For the separation of accompaniment music sources, the results from Table 4.2 show that the Denoiser is beneficial as in the case of singing voice separation. Focusing on the  $E_{\text{KL}/L_1}^{\text{MaD}}$  training objective, the usage of the Denoiser leads to improvements in the median SDR by 0.10 dB, in the SIR by 0.11 dB. For the SAR measure a marginal decrease of 0.03 dB is observed. However, for the  $E_{\text{MSE}/L_1}^{\text{MaD}}$  training objective, the usage of the Denoiser improves the separation performance of the MaD architecture with respect to all the objective measures. Specifically, the SDR is improved by 0.18 dB and the SIR and SAR are both improved by 0.32 dB. From Tables 4.1 and 4.2 it is evident that the effect of the Denoiser is to eliminate remaining interferences from the Masker, enhancing the overall performance of the MaD architecture.

### The effect of the recurrent inference algorithm

Focusing on the effect of the recurrent inference algorithm, i.e., Algorithm 2, Table 4.3 presents the results from objectively evaluating the MaD architecture (including the Denoiser module) optimized using the KL-based training objective  $E_{\text{KL}/L_1}^{\text{MaD}}$  with  $\omega = 0.5$ . In addition to the SDR, SIR, and SAR objective measures, the processing time in ms (the lower the better) is also reported. From the results presented in Table 4.3 two observations can be made. The first observation is that the recurrent inference algorithm dramatically increases the computation time that is required to obtain the latent representation  $\mathbf{H}_{\text{dec}}^j$ . Specifically, the required processing time for  $iter = 3$  is 156 ms that is approximately equal three times the baseline time, i.e., for  $iter = 0$ . Furthermore, for  $iter = 10$  the processing time is approximately equal to seven times the baseline time. The higher standard deviation in processing time observed for  $iter = 10$ , compared to smaller values for  $iter$ , and the incremental increase of processing time from  $iter = 3$  to  $iter = 10$ , suggest that the maximum number of iterations to 10 is not all the times reached, meaning that that  $iter = 10$  is a heuristically safe upper bound for the iterative process.

The second observation is that the recurrent inference algorithm is useful only for

Table 4.3: The effect of the recurrent inference algorithm, used by the  $\text{RNN}_{\text{dec}}(\cdot)$  Masker, in singing voice and accompaniment source separation, including the processing time required (standard deviation in parentheses). The objective evaluation measures and the average processing time are reported on the MUSDB18 data-set. Values in boldface denote the best obtained performance subject to the target source.

Target Source	$iter$	SDR (dB)	SIR (dB)	SAR (dB)	Time (ms)
Sing. Voice	0	<b>4.02</b>	8.63	<b>4.53</b>	<b>50(<math>\pm 4</math>)</b>
	3	4.02	7.85	4.53	156 ( $\pm 1$ )
	10	3.88	<b>9.22</b>	4.11	355 ( $\pm 11$ )
Accompaniment	0	<b>10.03</b>	12.60	<b>12.91</b>	<b>50(<math>\pm 4</math>)</b>
	3	9.92	14.38	11.86	156 ( $\pm 1$ )
	10	9.94	<b>14.94</b>	11.55	355 ( $\pm 11$ )

increasing the SIR by 0.59 dB and by 2.34 dB for the singing voice and accompaniment separation, respectively. This observation can be made for  $iter = 10$ . In contrast to the SIR, the SDR and SAR values are shown to decrease by 0.14 dB and 0.42 dB for singing voice separation, and by 0.09 and 1.36 dB for the separation of the accompaniment source. In the case that  $iter = 3$  there are not any significant improvements for singing voice separation, but only an increase of the SIR by 1.78 dB for the accompaniment source separation and compared to  $iter = 0$ . In summary, the recurrent inference algorithm can be seen as an effective way to improve only the interference reduction capabilities of the MaD architecture. However, that comes at the cost of increased computation time, that is important especially during the training phase, and the decrease of the SDR that is considered a very important objective measure in source separation.

### The effect of the training objectives

The results from the objective assessment are presented in Table 4.4 for the task of singing voice separation. The corresponding results for the accompaniment source are given in the Appendix D. In more details, the median values for the SDR, SIR, and SAR objective measures are reported for the MaD architecture that was trained with the MSE-based objective  $E_{\text{MSE}/L_1}^{\text{MaD}}$ , the KL-based objective  $E_{\text{KL}/L_1}^{\text{MaD}}$ , and the KL-based objective using the TwinNet  $E_{\text{KL}/\text{twin}}^{\text{MaD}}$ , for two values of  $\omega = 0$  and 0.5 that controls the strength of the penalty term in the training objective.

As can be seen in Table 4.4, including the penalty term in the training objective of the MaD architecture results in an overall improvement of the SDR and SIR objective measures. For the SAR measure, a marginal decrease of the median value is observed. In more details, using the  $L_1$  matrix norm as a penalty on the weights of the sparsifying transform, i.e., Eq. (4.8), a negligible improvement in the SDR of 0.01 dB and 0.02 dB is observed for the  $E_{\text{KL}/L_1}^{\text{MaD}}$  and the  $E_{\text{MSE}/L_1}^{\text{MaD}}$  training objectives, respectively. For the SIR, the penalty computed using Eq. (4.8) leads to an improvement of 0.13 dB for the KL-based objective  $E_{\text{KL}/L_1}^{\text{MaD}}$  and 0.55 dB for the MSE-based objective  $E_{\text{MSE}/L_1}^{\text{MaD}}$ . The difference in the observed improvements of the median SIR measure between  $E_{\text{KL}/L_1}^{\text{MaD}}$  and  $E_{\text{MSE}/L_1}^{\text{MaD}}$ , can be explained due to the different matrix norms employed in the KL divergence (Eq. (4.7))

Table 4.4: The effect of various training objectives on the performance in singing voice separation, computed using the objective evaluation measures and the MUSDB18 dataset. Values in boldface denote the best obtained performance.

Objective	$\omega$	SDR (dB)	SIR (dB)	SAR (dB)
$E_{\text{KL}/L_1}^{\text{MaD}}$	0	4.01	8.50	<b>4.55</b>
	0.5	4.02	8.63	4.53
$E_{\text{MSE}/L_1}^{\text{MaD}}$	0	4.08	8.13	4.43
	0.5	4.10	8.68	4.41
$E_{\text{KL}/\text{twin}}^{\text{MaD}}$	0.5	<b>4.16</b>	<b>9.96</b>	4.26

and the MSE (Eq. (4.6)). The  $L_2$  matrix norm used in Eq. (4.6) could yield smaller values for the error signal allowing the  $L_1$  matrix norm, used to compute the penalty term (Eq. (4.8)), to have a greater impact on the computation of the gradients. For the SAR, Table 4.4 suggests that the usage of a penalty term decreases the respective objective measure by 0.02 dB for both  $E_{\text{KL}/L_1}^{\text{MaD}}$  and  $E_{\text{MSE}/L_1}^{\text{MaD}}$ . In addition to this, the usage of the KL diverge to compute the reconstruction term yields the best results with respect to the SAR.

By comparing the objectives  $E_{\text{KL}/L_1}^{\text{MaD}}$ ,  $E_{\text{MSE}/L_1}^{\text{MaD}}$ , and  $E_{\text{KL}/\text{twin}}^{\text{MaD}}$ , with respect to the SDR, SIR, and SAR measures, it can be seen in Table 4.4 that the usage of the TwinNet yields the best results with respect to the median SDR and SIR measures. Specifically, the  $E_{\text{KL}/\text{twin}}^{\text{MaD}}$  marginally improves the SDR by 0.14 dB and 0.06 dB in comparison to the  $E_{\text{KL}/L_1}^{\text{MaD}}$  and  $E_{\text{MSE}/L_1}^{\text{MaD}}$  objectives, respectively, for  $\omega = 0.5$ . However, the usage of  $E_{\text{KL}/\text{twin}}^{\text{MaD}}$  leads to improvements with respect to the SIR that are 1.33 dB and 1.28 dB, compared to  $E_{\text{KL}/L_1}^{\text{MaD}}$  and  $E_{\text{MSE}/L_1}^{\text{MaD}}$  objectives, respectively. This observation considers the penalty term with  $\omega = 0.5$ . Also, it can be observed that for  $\omega = 0$  the SAR decreases by 0.29 dB and 0.17 dB compared to  $E_{\text{KL}/L_1}^{\text{MaD}}$  and  $E_{\text{MSE}/L_1}^{\text{MaD}}$  objectives, respectively. The case of  $\omega = 0$  is considered since it results in the highest median SAR according to the Table 4.4.

In addition to the above, the usage of the TwinNet yields superior performance to the recurrent inference algorithm, without the necessity of increasing the processing time of a sequence during the estimation of the target source. This observation can be made by comparing the corresponding results of the recurrent inference presented in Table 4.3 and the results of the TwinNet in Table 4.4. In particular and in comparison to the recurrent inference algorithm, using the the TwinNet technique the SIR is increased by 0.44 dB without the decrease of the SDR objective measure. From these observations it can be concluded that the usage of the TwinNet technique results in improvements of two important objective measures in music source separation, namely the SDR and the SIR.

### The qualitative effect of the training objectives

From a qualitative perspective, a closer inspection of the previously mentioned marginal differences between the training objectives with respect to the SDR, SIR, and SAR measures is given in Fig. 4.3. Specifically, Fig. 4.3 illustrates the average score (the higher the better), that is obtained from the listening tests highlighting trends of the the *perceptual* effects that the training objectives have on the singing voice separation performance using

the MaD architecture. The average score is computed within the 95% confidence intervals for the most competitive training objectives, with respect to the SDR and SIR measures.

As it can be seen in Fig. 4.3, the MaD architecture optimized with the three training objectives has obtained score values between the range of 40 and 60. The obtained scores suggest that a *fair* separation quality is achieved by the MaD architecture, according to the MUSHRA specifications. In contrast to the marginal improvements of  $E_{\text{KL}/\text{twin}}^{\text{MaD}}$

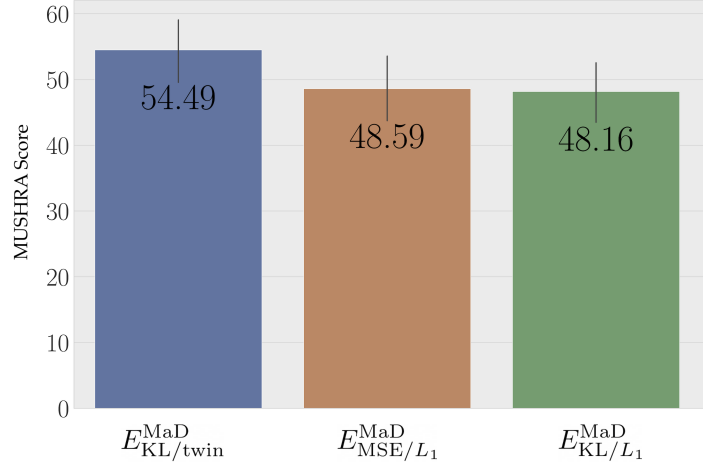


Figure 4.3: Average scores obtained from the listening tests for the three training objectives:  $E_{\text{KL}/L_1}^{\text{MaD}}$ ,  $E_{\text{MSE}/L_1}^{\text{MaD}}$ , and  $E_{\text{KL}/\text{twin}}^{\text{MaD}}$  for  $\omega = 0.5$ .

with respect to the SDR and SIR observed in Table 4.4, the usage of the objective with the TwinNet technique has an effect on the perceptual quality of singing voice separation. Specifically,  $E_{\text{KL}/\text{twin}}^{\text{MaD}}$  received an average score of 54.49, that is approximately 6 points higher than the  $E_{\text{KL}/L_1}^{\text{MaD}}$  and  $E_{\text{MSE}/L_1}^{\text{MaD}}$ , on average. This suggests that the usage of the TwinNet technique leads to perceptually relevant improvements in the singing voice separation quality. Even though that the number of participants in the listening test is not sufficiently large to argue about significant differences between objective and subjective evaluation scores, the reported differences allow a more thorough inspection of the marginal improvements observed of the objective measures reported in Table 4.1. It should be noted that the reported discrepancy between objective and subjective evaluation measures follows similar trends that are presented in [79], where results from listening tests, that were performed at a larger scale, do not always coincide with the SDR, SIR, and SAR measures.

### Comparison with other supervised approaches

For demonstrating benchmark results in comparison to other deep-learning based approaches in music source source separation, Table 4.5 and Table 4.6 provide the median SDR, SIR, and SAR objective measures for the MaD architecture, using the TwinNet



technique (*MaD-TwinNet*). More specifically, Table 4.5 demonstrates results in comparison to legacy approaches and Table 4.6 focuses on recent state-of-the-art approaches in music source separation.

Table 4.5: Results from the objective evaluation for singing voice and accompaniment source separation using the SDR, SIR, and SAR objective measures. Results reported on the DSD100 data-set and in comparison to legacy approaches for mask prediction. Values in boldface denote the best obtained performance.

Target Source	Approach	SDR (dB)	SIR (dB)	SAR (dB)
Sing. Voice	<i>GRA</i> [90]	0.92	6.28	3.49
	<i>HUA+</i> [95]	2.27	<b>8.28</b>	3.71
	<i>MaD-TwinNet</i>	<b>4.57</b>	8.17	<b>5.61</b>
Accompaniment	<i>GRA</i> [90]	4.52	8.78	<b>14.67</b>
	<i>HUA+</i> [95]	6.33	<b>13.71</b>	8.22
	<i>MaD-TwinNet</i>	<b>9.58</b>	13.09	11.67

The results presented in Table 4.5 suggest that the separation performance using the MaD architecture with the TwinNet technique leads to superior results for both singing voice and accompaniment separation, compared to legacy approaches that either predict (un-optimized) masks explicitly (*GRA*), or using the magnitude estimates of the RNNs to compute soft-masks and jointly optimize the parameters of the RNNs (*HUA+*). Specifically, the MaD-TwinNet outperforms the explicit mask prediction approach (*GRA*) by an average of 4.36 dB and 3.10 dB in the SDR and the SIR objective measures, respectively. Subject to the SAR, the MaD-TwinNet is outperformed by 0.88 dB on average, with respect to the two target sources. In comparison to the *HUA+* approach, that does not consider optimizing the predicted masks for the target sources, the MaD-TwinNet yields superior results only with respect to the SDR and SAR measures. Specifically, on average across the two target sources the MaD-TwinNet outperforms the *HUA+* approach by 2.68 dB, in both the SDR and the SAR measures. Subject to SIR, the *HUA+* approach outperforms the MaD-TwinNet by 0.39 dB. One interpretation of the above results is that allowing supervised approaches to yield and optimize source-dependent mask, as the MaD-TwinNet does, yields superior superior results without the necessity of any post-processing steps.

For providing benchmark results compared to more recent approaches, the median SDR, SIR, and SAR values are presented in Table 4.6 for the singing voice and accompaniment sources. From Table 4.6 it can be seen that the usage of data-augmentation, used in UHL, combined with the fusion of multiple source separation approaches optimized with various target sources, a strategy followed by both LPR and UHL, yield significantly better SDR, SIR, and SAR measures than the MaD-TwinNet. Particularly, the UHL approach yields the best results surpassing the MaD-TwinNet by 1.96 dB and the LPR by 1.60 dB in the SDR measure and on average with respect to the two target sources. For the SIR and the SAR measures a similar trend can be seen in Table 4.6. On average with respect the singing voice and accompaniment sources, the UHL approach outperforms the LPR approach by 2.84 dB and 4.10 dB, and the MaD-TwinNet by 6.07 dB and 2.59 dB for the SIR and the SAR measures, respectively.

Table 4.6: Results from the objective evaluation for singing voice and accompaniment source separation using the SDR, SIR, and SAR objective measures. Results reported on the MUSDB18 data-set and in comparison to state-of-the-art approaches. Values in boldface denote the best obtained performance.

Target Source	Approach	SDR (dB)	SIR (dB)	SAR (dB)
Sing. Voice	<i>MaD-TwinNet</i>	4.16	9.96	4.26
	<i>LPR</i> [129]	4.32	<b>12.62</b>	4.10
	<i>UHL</i> [81], [98]	<b>5.93</b>	11.69	<b>6.28</b>
Accompaniment	<i>MaD-TwinNet</i>	10.09	12.89	12.86
	<i>LPR</i> [129]	10.65	13.46	11.51
	<i>UHL</i> [81], [98]	<b>12.23</b>	<b>17.23</b>	<b>13.43</b>

From the above observations, two evident trends can be highlighted by recalling that UHL is a spectral approximation approach and LPR employs the skip-filtering connections plus the fusion of multiple trained source separation models. The first trend is that the spectral approximation benefit from using data-augmentation and the fusion of multiple models, by means of generalized Wiener filtering, yielding superior results. The second trend is that source separation models employing the skip-filtering connections also benefit from post-processing steps involving the application of soft-masks by fusing the outputs of multiple separation models. The latter trend is mostly evident for interference reduction, i.e., the increase of the SIR, that can be observed by comparing LPR that uses the previously mentioned post-processing technique and the MadTwin-Net that does not include any post-processing steps.

#### 4.6.2 Harmonic & Percussive Separation

For demonstrating the potential of the proposed architecture to separate other music sources other than the singing voice or accompaniment, Table 4.7 presents the objective performance of the proposed architecture in harmonic and percussive source separation. In particular, the median SDR, SIR, and SAR values are reported for the MaD-TwinNet using the MUSDB18 data-set and compared to a state-of-the-art approach to HPSS [134], that is denoted as *LOR*. The MaD-TwinNet is considered in the evaluation, as it was shown in the above presented results that the MaD-TwinNet yields perceptually relevant improvements to the separation performance of the proposed architecture.

The results from Table 4.7 suggest that the MaD-TwinNet performs better than the LOR approach with respect to the suppression of percussive in the harmonic source, as indicated by the SIR that is by 2.61 dB higher than the LOR for harmonic sources. Furthermore, it can be seen that the MaD-TwinNet produces less separation artefacts for the percussive source, compared to the LOR approach. This is supported by the 0.7 dB difference in the SAR objective measure between the MaD-TwinNet and LOR. Recalling that the LOR approach is based on two-dimensional CNNs and the skip-filtering connections, whereas the Mad-TwinNet relies on RNNs and the skip-filtering connections, the results of Table 4.7 show that the usage of CNNs is beneficial for improving the separation quality of both harmonic and percussive sources. This can be seen from the

Table 4.7: Results from the objective evaluation for harmonic and percussive source separation using the SDR, SIR, and SAR objective measures. Results reported on the MUSDB18 data-set and compared to a state-of-the-art HPSS approach. Values in bold-face denote the best obtained performance subject to the target source.

Target Source	Approach	SDR (dB)	SIR (dB)	SAR (dB)
Harmonic	<i>MaD-TwinNet</i>	8.93	<b>13.09</b>	11.46
	<i>LOR</i> [134]	<b>9.71</b>	10.48	<b>13.32</b>
Percussive	<i>MaD-TwinNet</i>	3.61	4.84	<b>6.05</b>
	<i>LOR</i> [134]	<b>3.70</b>	<b>5.84</b>	5.35

SDR objective measure of the LOR approach, that is superior to the Mad-TwinNet by 0.78 dB and by 0.09 dB for the harmonic and the percussive source, respectively.

## 4.7 Summary

This chapter presented a neural network architecture that relies on two concepts. The first concept is the skip-filtering connections, that serves as an extension of the denoising autoencoder (DAE) model and is particularly useful for music source separation in the time-frequency domain. The second concept is to treat the time-frequency representations of music signals as sequences with rich temporal information, that can be exploited for music source separation. To that aim, recurrent neural networks (RNNs) and particularly gated recurrent units (GRUs) were employed in the proposed neural network architecture. The proposed architecture is denoted as the Masker-and-Denoiser (MaD) and several extensions to the architecture have been presented. These extensions include the usage of alternative training objective functions but also techniques that aim at increasing the capabilities of the proposed architecture, for modelling and processing sequences of music signals' time-frequency representations. These techniques are namely the recurrent inference algorithm and the twin network technique [133].

To assess the performance of the MaD architecture in music source separation, an experimental procedure was conducted. The experimental procedure involved the optimization of the proposed architecture for estimating various music sources that include the singing voice, the accompaniment, the harmonic, and the percussive sources. Furthermore, the optimized MaD architecture, for each target music source, was used in an evaluation procedure that involves the computation of objective measures, such as the signal-to-distortion ratio (SDR), the signal-to-interference ratio (SIR), and the signal-to-artifacts ratio (SAR). For the singing voice and accompaniment separation listening tests with trained listeners were conducted, highlighting the perceptual relevance of the employed enhancement to the MaD architecture. The experimental results showed three main trends. The first trend is that the MaD architecture produces *fair* separation results for singing voice separation, while the enhancements regarding the sequence processing and particularly the usage of the twin network technique [133], yielded to perceptually relevant improvements in the overall separation quality. The second trend is that proposed architecture outperformed the approaches that aim at explicitly predicting

a pre-computed and un-optimized time-frequency masks. However, the MaD architecture was surpassed by approaches that rely on post-processing steps for estimating the target sources. Those post-processing steps commonly include the fusion of predictions from multiple deep learning-based source separation models via the usage of generalized Wiener filtering and or linear mixing. Finally, the third trend is that the optimization of RNNs subject to the spectral-based music source separation is cumbersome, and without additional techniques, such as the twin network or fusion of multiple networks, it is difficult to achieve the current performance standards in music source separation.

## Chapter 5

# Learning Representations for Separation

### Preface

In the previous chapters the focus is given on supervised approaches to music source separation. Those approaches are optimized using the magnitude representations of the mixture and of the target sources' signals as input and output targets respectively. The used magnitude representations are computed using the short-time Fourier transform (STFT). The motivation for using the STFT is that the separation is understood as a filtering operation, which is an intuitive process. The goal of this chapter is to go beyond the usage of STFT, and explore methods for learning signal representations directly from time-domain music signals. Furthermore, the computed representations consist of attributes that are useful for the task music source separation. To evaluate the benefits of the learned representations, the separation of the singing voice source from the accompaniment is considered as the downstream task. The work presented in this chapter is based on the works [A21], [A24]. The corresponding source code, based on the PyTorch framework [49], is available online<sup>1</sup>. Additional results are given in Appendix E.

### 5.1 Introduction

Recent advances in music source separation rely on deep learning approaches that can be discriminated in two categories. In the first category the separation approaches operate in the STFT domain [100], [130], and are denoted as spectral-based approaches. In the second category the separation approaches operate directly on the waveform signals [136], [137], i.e., the approaches are trained end-to-end, and are denoted as waveform-based approaches. Spectral and waveform based approaches have in common that they implicitly compute source-dependent masks that are applied to the mixture signal, prior to the

---

<sup>1</sup>[https://github.com/Js-Mim/rl\\_singing\\_voice](https://github.com/Js-Mim/rl_singing_voice)

reconstruction of the target signals [100], [130], [136], [137]<sup>2</sup>.

Although the implicit masking is shown to be a simple and robust method to learn source dependent patterns for source separation [A22], one could expect that waveform based approaches would significantly outperform the spectral ones. That is because waveform based approaches are optimized using time-domain signals that also contain the phase information, that unarguably carries important signal information [A13], [3], [11] and has been neglected by many spectral based approaches [A10], [A11], [100], [130]. Nonetheless, previously conducted experiments and reported results suggest that spectral based approaches have comparable or marginally better separation performance to the waveform ones [130], [136], [137]. Since both waveform and spectral approaches rely on deep neural networks (DNNs) and for both approaches a considerable engineering effort has been directed to the employed neural architecture, it is evident that the difference in the performance between the two different approaches can be attributed to the utilized signal representation that is used for separation.

For the spectral-based approaches the utilized representation is the non-negative signal representation offered by the magnitude of the STFT. For the waveform-based approaches the representation is computed by trainable encoding functions, commonly neural networks. The parameters of the encoding functions are optimized jointly with the rest of the separation model. The optimization of the separation model, and thus also the encoding functions that compute the representations, is based on minimizing loss objectives that assess the reconstruction of the signals of the target sources given the mixture signal as input [136], [137]. In this case and subject to the representations, the learning is performed using solely discriminative optimization objectives, that aim at distinguishing between the mixture and the target sources. As shown in [104], this could potentially impose severe limitations in the generalization capabilities of the learned representations, as the learning process based on discriminative objectives does not aim at capturing the essential structure of the signals [86], [87]. Furthermore, the learned representations obtained by approaches utilizing end-to-end training are not easily nor intuitively interpreted, compared to the pre-computed signal representations that utilize the STFT.

In an attempt to learn music signal representations that capture the structure of the music signals, are interpretable, and consist of attributes that are useful for music source separation, the focus is given on neural-based representation learning [138]. The following sections present a new and simple method for learning representations of time-domain music signals. The proposed method is characterized as unsupervised because the optimization of the method does not depend on labelled categorical data, i.e., labels for distinguishing between the music sources, and the representation attributes (discussed later) are learned using unsupervised training objectives. Furthermore, these training objectives do not target the learning of the unmixing function, i.e., the mapping from the mixture to the target source signal as done in the previous chapters. This in turn, alleviates the need of having either labelled or paired training data (i.e., matched multi-track audio data of each corresponding source). However, the proposed method still requires isolated source's audio signals, but this information is more accessible than paired multi-track data.

The rest of this chapter is organized as follows: Section 5.2 provides information re-

---

<sup>2</sup>Subject to the masking strategy, the thesis refers to the adaptation of Conv-TasNet [120] for music signals also presented in [136].

garding previously published research that is related to representation and interpretable representation learning for audio and speech processing and enhancement. The proposed method for learning representations is described in Section 5.3, followed by the experimental procedure described in Section 5.4. Section 5.5 presents and discusses the results obtained from the experimental procedure, including visualizations of the obtained representation(s). Section 5.6 summarizes the findings presented in this chapter.

## 5.2 Related Work

The proposed method is based on the denoising autoencoder (DAE) model [87] that can be used also for unsupervised learning of signal features and representations, other than trained in a supervised way to separate music sources as previously done in Chapters 3 and 4. The DAE can efficiently learn the empirical distribution of the signal of interest, i.e., the signal to be denoised [87], [104]. This is achieved by optimizing the DAE with the unsupervised objective to reconstruct the clean signal from a noisy version. Subject to this work, the underlying assumption is that a model capable of computing representations that characterize the clean signal of interest can be obtained by learning the empirical distribution of the observed data. Contrary to the DAE, the proposed method uses a simple and real-valued sinusoidal-based model for the *decoding* functions. The sinusoidal model consists of amplitude-modulated cosine functions, and whose parameters are jointly optimized with the rest of the DAE. The motivation behind using a sinusoidal model as a decoding function is to guide via back-propagation the encoding layers of the DAE to learn and convey information regarding the energy of specific cosine functions that compose the audio signal. This leads to interpretable representations akin to the STFT.

Employing a vastly used digital signal processing operation for decoding functions is inspired by two works. The first work introduces the concept of differentiable digital signal processing [139] where the parameters of common digital signal processing functions are optimized by means of back-propagation. In the case of the proposed method, back-propagation is applied with respect to the parameters of a simple signal model that is based on sinusoidal functions. The second work that the proposed method is inspired from, is the Sinc-Network presented in [140]. The Sinc-Network uses sinc functions in the *encoding* layers of convolutional kernels for interpretable deep learning. The Sinc-Network has been extended to complex-valued representations for speaker separation [141].

The proposed method differs from [141] as the representation of the proposed method is real-valued, alleviating the cumbersome signal processing operations on complex numbers. Furthermore, the proposed method differs from approaches that initialize the front-end parts of the networks with cosine functions [142] that are then updated by means of back-propagation. The difference is that the proposed method inherits the cosine functions as a part of the model to be optimized. Finally, the proposed method is similar to the sound source separation method presented in [143]. In [143] an encoder gets as an input the signals of the sources and their corresponding mixture, and outputs latent representations of the signals of each source and the mixture. Then, using these latent representations, the method calculates and applies source dependent masks to the latent representation of mixture. The result of the application of the masks is given as an input to the decoder, which outputs an estimation of the signal of each source. The encoder and the decoder are jointly optimized to minimize the reconstruction error between the ground truth and

the estimated signals of each source, i.e., a discriminative training is performed. However, using reconstruction objectives in a discriminative setting for separating only specific sources, could severely restrict the representation learning capabilities of encoder-decoder methods [104]. In contrast, the proposed method uses information from the mixture and target source signals using unsupervised and non-discriminative training objectives that aim at capturing the structure of the music signals.

### 5.3 Proposed Method

The proposed method employs an encoder  $E(\cdot)$  and a decoder  $D(\cdot)$ . The input to the method is a music signal,  $\mathbf{x} \in \mathbb{R}_{[-1,1]}^N$ , with  $N$  time-domain samples. The output of the method is the learned non-negative representation of  $\mathbf{x}$ ,  $\mathbf{A} \in \mathbb{R}_{\geq 0}^{C \times T}$ , with  $T$  templates of  $C$  features. The  $C$  features can be viewed as analogous to the frequency bins and the  $T$  templates as the analogous to the time-frames in a time-frequency representation.  $\mathbf{A}$  is computed by the encoder  $E(\cdot)$ , and is interpreted as the magnitude information for a real-valued, sinusoidal-based model, employed by the decoder  $D(\cdot)$ .

To optimize  $E(\cdot)$ , the decoder  $D(\cdot)$  is used. In addition to this, a data-set of monaural (single channel) recordings of singing voice,  $\mathbf{x}_v \in \mathbb{R}_{[-1,1]}^N$ , and accompanying musical instruments  $\mathbf{x}_{ac} \in \mathbb{R}_{[-1,1]}^N$  is used. Using  $\mathbf{x}_v$  two synthetic signals are created. The first synthetic signal,  $\tilde{\mathbf{x}}_m \in \mathbb{R}_{[-1,1]}^N$ , is the result of an additive corruption process, where the accompanying musical instruments such as drums, guitars, synthesizers, and bass (i.e. a generic multi-modal distribution-based noise) are added to  $\mathbf{x}_v$ :

$$\tilde{\mathbf{x}}_m = \mathbf{x}_v + \mathbf{x}_{ac} .$$

The second synthetic signal,  $\tilde{\mathbf{x}}_v \in \mathbb{R}_{[-1,1]}^N$ , is also the result of a corruption process, where Gaussian noise is added to  $\mathbf{x}_v$ , independently of the amplitude of  $\mathbf{x}_v$ .

During training, the encoder  $E(\cdot)$  computes two non-negative representations  $\mathbf{A}_m \in \mathbb{R}_{\geq 0}^{C \times T}$  and  $\mathbf{A}_v \in \mathbb{R}_{\geq 0}^{C \times T}$ , using the two above mentioned synthetic signals,  $\tilde{\mathbf{x}}_m$  and  $\tilde{\mathbf{x}}_v$ , respectively.  $\mathbf{A}_v$  is used as input to  $D(\cdot)$ , and  $D(\cdot)$  outputs an approximation of the clean singing voice signal  $\mathbf{x}_v$ , denoted as  $\hat{\mathbf{x}}_v$ .  $\mathbf{A}_m$  is solely used to calculate an additional loss function. This is done in order to enforce  $E(\cdot)$  to learn about the additive multi-modal noise. Essentially, the encoder is informed about more realistic corruption process that might occur in realistic data-sets. An illustration of the training procedure is shown in Figure 5.1.

After the training process of the proposed method,  $E(\cdot)$  can take as an input any musical signal  $\mathbf{x}$ , and will output the representation of  $\mathbf{x}$ , denoted as  $\mathbf{A}$ . Furthermore, an approximation of the signal  $\mathbf{x}$  can be computed using the decoder  $D(\cdot)$ . The benefits for doing so, is that  $\mathbf{A}$  has good music signal representation attributes that include interpretability (post-hoc [144]), non-negativity<sup>3</sup>, and structured spectrogram-like representations. Furthermore, the inputted signal  $\mathbf{x}$  can be approximated from  $\mathbf{A}$  using  $D(\cdot)$ , with a small reconstruction error for the parts that the singing voice signal is active. Consequently, the method could be effectively used in the downstream task of singing voice separation, but not limited to.

<sup>3</sup>Non-negativity or in other words the disregard of phase information in music source separation has played an important role in devising models and approaches [A9].



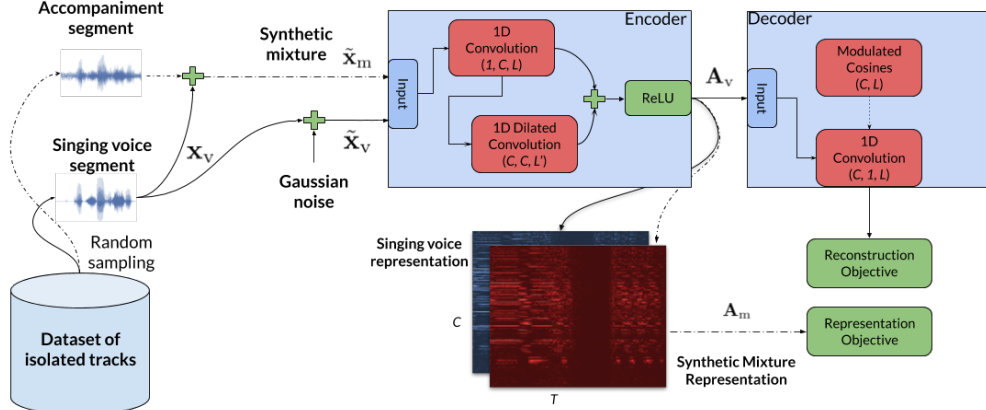


Figure 5.1: Overview of the proposed method for representation learning.

### 5.3.1 The Encoder

The encoder  $E(\cdot)$  computes the representation(s) using two one-dimensional (1D) convolutions with strides and in series. The first 1D convolution uses a stride size  $S$  and a set of  $C$  kernels,  $\mathbf{k}_c \in \mathbb{R}^L$ , where  $c = [0, 1, \dots, C-1]$  is the kernel index. The temporal length of each kernel  $\mathbf{k}$  is  $L$  samples. As input to the first convolution, the signals  $\tilde{\mathbf{x}}_m$  and  $\tilde{\mathbf{x}}_v$  are used. The outputs are the *latent* representations  $\tilde{\mathbf{H}}_m \in \mathbb{R}_{\geq 0}^{C \times T}$  and  $\tilde{\mathbf{H}}_v \in \mathbb{R}_{\geq 0}^{C \times T}$ , respectively. More formally, the latent representations are computed as

$$\tilde{\mathbf{H}}_{m[c,t]} = \sum_{l=0}^{L-1} \tilde{\mathbf{x}}_{m[St+l]} \mathbf{k}_c[l] \quad (5.1)$$

$$\tilde{\mathbf{H}}_{v[c,t]} = \sum_{l=0}^{L-1} \tilde{\mathbf{x}}_{v[St+l]} \mathbf{k}_c[l], \quad (5.2)$$

where  $t \in [0, 1, \dots, T-1]$  and  $l \in [0, 1, \dots, L-1]$  are integers denoting the time-frame and the kernel sample indices, respectively. Appropriate zero-padding is applied to  $\tilde{\mathbf{x}}_m$  and  $\tilde{\mathbf{x}}_v$ , so that  $T = \lceil N/S \rceil$ . Each latent representation is used as an input to the second 1D convolution, which uses another set of  $C$  kernels,  $\mathbf{K}'_{c'} \in \mathbb{R}^{L' \times C}$ , with a temporal length of  $L'$  samples, that is  $L' \ll L$ . The output channels are indexed by  $c'$ , where  $c' = [0, 1, \dots, C-1]$ .

The outputs of the second convolution using the previously computed representations, are denoted by  $\mathbf{H}_m \in \mathbb{R}^{C \times T}$  and  $\mathbf{H}_v \in \mathbb{R}^{C \times T}$ , respectively. The second convolution is performed with a dilation factor of  $\phi$  and a unit stride  $S = 1$ , as

$$\mathbf{H}_{m[c',t]} = \sum_{c=0}^{C-1} \sum_{l'=0}^{L'-1} \tilde{\mathbf{H}}_{m[c,t+\phi l']} \mathbf{K}'_{c'}[l',c] \quad (5.3)$$

$$\mathbf{H}_{v[c',t]} = \sum_{c=0}^{C-1} \sum_{l'=0}^{L'-1} \tilde{\mathbf{H}}_{v[c,t+\phi l']} \mathbf{K}'_{c'}[l',c]. \quad (5.4)$$

Then, the representations  $\mathbf{A}_m$  and  $\mathbf{A}_v$  are computed using the previously computed latent representations,  $\mathbf{H}_m$  and  $\mathbf{H}_v$  respectively, and by means of residual connections, followed by the application of the rectified linear unit (ReLU) function [46] as

$$\mathbf{A}_m = \text{ReLU}(\mathbf{H}_m + \tilde{\mathbf{H}}_m) \quad (5.5)$$

$$\mathbf{A}_v = \text{ReLU}(\mathbf{H}_v + \tilde{\mathbf{H}}_v) . \quad (5.6)$$

The application of the ReLU function promotes non-negative and sparse representations by preserving positive values and setting the rest to zero [27], and is shown to be particularly useful in general modelling of audio signals [145]. Another targeted and useful attribute of the learned representation is that of smoothness [142], [145], especially useful when real-valued cosine functions are involved in auto-encoding or separation models [142]. Smoothness refers to the slow time variation of the representation, and is useful for general audio signal modelling. That is because the modelling of audio signals based on cosine functions requires the phase information for reconstruction. Phase information is usually encoded as the sign (positive or negative value) of the real-valued representation, that varies along the time-frames of the representation. Since the negative values are nullified by the application of the ReLU function, neighbouring time-frames, that convey similar information for music signals are expected to be non-smooth. To compensate for the expected non-smoothness, the second convolution operation uses dilated convolutions that aggregate temporal information from neighboring time-frames [61] and updated using residual connections.

In order to enforce the learning of smooth representations, a representation objective is introduced. The introduced objective is a loss function that the encoder has to minimize. The most straightforward way to enforce the smoothness is to compute the norm of the first-order differences of the representation [146]. To do so, the (anisotropic) total-variation denoising loss is used. Specifically, the representation of  $\tilde{\mathbf{x}}_m$ ,  $\mathbf{A}_m$ , is used to compute the total variation denoising ( $\mathcal{L}_{TV}(\cdot)$ ) as

$$\begin{aligned} \mathcal{L}_{TV}(\mathbf{A}_m) = \frac{1}{CT} & \left( \sum_{c=1}^{C-1} \sum_{t=0}^{T-1} |A_{m[c,t]} - A_{m[c-1,t]}| \right. \\ & \left. + \sum_{t=1}^{T-1} \sum_{c=0}^{C-1} |A_{m[c,t]} - A_{m[c,t-1]}| \right). \end{aligned} \quad (5.7)$$

Practically,  $\mathcal{L}_{TV}(\cdot)$  penalizes  $E(\cdot)$  by the norm of the first order difference across both time-frames  $T$  and templates  $C$ . The former promotes slow time varying representations as the magnitude of the STFT representation, and the latter promotes a grouping of the template activity. It should be noted that the smoothness, as a representation objective, is not expected to smear any transient information contained in the time-domain signal. That is because transients, in the computed representations, are expected to be active only in a few time-frames. By optimising with longer signals and thus, increasing the number of time-frames of the representation, the first order difference does not exceed high values on average. The previously mentioned attributes of the desired representation attributes are formed from domain knowledge that is based on the STFT. Furthermore,  $\mathcal{L}_{TV}(\cdot)$  is an unsupervised objective that depends only on the representation  $\mathbf{A}_m$ , and does not imply any separation objective, i.e., estimating  $\mathbf{A}_v$  from  $\mathbf{A}_m$ .

Although  $\mathcal{L}_{\text{TV}}(\cdot)$  seems an attractive loss function due to its simple computation, it has a severe limitation. According to [147, Theorem 2] the total-variation distance, and in this particular case the sum of absolute differences employed in Eq.(5.7), is not a suitable loss function for data distributions supported by low-dimensional manifolds. Instead, optimal transportation distances are more suitable. Under the hypothesis that both the singing voice and the mixture signals, and their corresponding representations can be described by low-dimensional manifold(s), an alternative unsupervised objective to  $\mathcal{L}_{\text{TV}}(\cdot)$  is also examined.

Sinkhorn distances  $\mathcal{L}_{\text{SK}}(\cdot)$  allow an efficient computation of optimal transportation loss [148]. The goal here is to minimize the distribution differences between local time-frames. The locality of the computation is dependent on the considered number of time-frames  $T$ . More specifically and subject to the goal of this work, Sinkhorn distances are computed as

$$\mathcal{L}_{\text{SK}}(\mathbf{A}_m) = \langle \mathbf{P}_\lambda, \psi(\mathbf{A}_m) \rangle, \quad (5.8)$$

where “ $\langle \cdot, \cdot \rangle$ ” is the Frobenius dot-product and  $\psi : \mathbb{R}_{\geq 0}^{C \times T} \mapsto \mathbb{R}_{\geq 0}^{T \times T}$  is a function that computes the matrix  $\mathbf{M} \in \mathbb{R}_{\geq 0}^{T \times T}$  of pair-wise distances, i.e.,  $\mathbf{M} = \psi(\mathbf{A}_m)$ . More specifically, the pair-wise distances are computed as

$$M_{[t,t']} = \left( \sum_{c=0}^{C-1} (|A_{m[c,t]} - A_{m[c,t']}|)^p \right)^{1/p}. \quad (5.9)$$

In Eq. (5.9)  $t, t' \in [0, \dots, T-1]$  are indices that are used to compute the pair-wise distance between the time-frames  $T$  of the representation. For the computation of the pair-wise distances the time-frames  $T$  are considered instead of the features  $C$ . That is because the goal here is to compute slowly time-varying representations and that can be directly enforced by comparing time-frame vectors at various time instances within  $T$ . Furthermore, Eq. (5.9) considers  $p = 1$  for computing the distances. It should be denoted, that only for, and prior to, the computation of the loss matrix  $\mathbf{M}$ , the representation  $\mathbf{A}_m$  is normalized so that the sum of the features at each time-frame  $t$  sum up to unity. More formally, the normalized representation  $\mathbf{A}_m^o$  is computed as

$$A_{m[c,t]}^o = \frac{A_{m[c,t]}}{\sum_c (A_{m[c,t]} + \frac{1}{C})}.$$

This is done in order to treat  $\mathbf{A}_m$  as a probability simplex in which the computation of the optimal transportation loss can be computed.

In Eq.(5.8),  $\mathbf{P}_\lambda \in \mathbb{R}_{\geq 0}^{T \times T}$  is the transportation plan that is computed by solving the following minimization problem

$$\mathbf{P}_\lambda = \arg \min_{\mathbf{P} \in \mathbb{U}(r,c)} \langle \mathbf{P}, \psi(\mathbf{A}_m) \rangle - \frac{1}{\lambda} H(\mathbf{P}). \quad (5.10)$$

In the above minimization problem,  $\lambda > 0$  is a scalar the controls the strength of the entropic regularization, and  $H(\cdot)$  denotes the entropy function that is computed as

$$H(\mathbf{P}) = - \sum_{t,t'=0}^{T-1} P_{[t,t']} \log(P_{[t,t']}).$$

---

**Algorithm 3** Computation of the transportation plan, using Sinkhorn-Knopp’s iterative matrix scaling operation [148], [149]

---

**Require:** loss matrix  $\mathbf{M}$ , entropic regularization scalar  $\lambda$ , dimensionality  $T$ , vector of ones  $\mathbf{1}_T$ , number of iterations  $iter$ , termination threshold  $\tau$

```

1: Initialize:  $K = \exp(-\lambda \mathbf{M})$ ,  $\mathbf{u} = \mathbf{1}_T \oslash T$ ,  $\mathbf{v} = \mathbf{1}_T \oslash T$ ,  $\mathbf{K}_{\mathbf{u}} = \text{diag}(\mathbf{u}) \mathbf{K}$ 
2: for all  $iter$  do
3:    $\mathbf{v} \leftarrow T \oslash (\mathbf{K}^\top \mathbf{u})$ 
4:    $\mathbf{u} \leftarrow \mathbf{1} \oslash (\mathbf{K}_{\mathbf{u}} \mathbf{v})$ 
5:    $o = \|\text{diag}(\mathbf{u}) \mathbf{K} \text{diag}(\mathbf{v})\|_1$ 
6:   if  $\|o - (\mathbf{1}_T \oslash T)\|_1^2 < \tau$  then
7:     stop iterating
8:   end if
9: end for
10:  $\mathbf{P}_\lambda = \text{diag}(\mathbf{u}) \mathbf{K} \text{diag}(\mathbf{v})$ 
11: return Transportation plan  $\mathbf{P}_\lambda$ 

```

---

In addition to this,  $\mathbb{U}(r, c)$  is the set of non-negative matrices of size  $T \times T$  whose rows and columns sum up to  $r$  and  $c$ , respectively. It is further assumed that  $r = c = 1$ . For solving the minimization problem of Eq.(5.10) the proposed method for representation learning employs the algorithm presented in [148] that is based on the Sinkhorn-Knopp iterative matrix scaling operator [149] and its pseudo-algorithm is given in Algorithm 3. In Algorithm 3,  $\|\cdot\|_p$  is the  $p$ -th vector or matrix norm.

In Eq.(5.7) and Eq.(5.10) only the representation  $\mathbf{A}_m$  is used to compute the corresponding loss functions. This is performed in order to enforce the encoder  $E(\cdot)$  to yield smooth representations on the most realistic corruption scenario. This scenario is the additive generic multi-modal distribution-based noise  $\tilde{\mathbf{x}}_m$  that contains also the information regarding the singing voice signal  $\mathbf{x}_v$ . Thus, the smoothness for the representation of the singing voice is implicitly enforced.

### 5.3.2 The Decoder

The decoder  $D(\cdot)$  takes as an input the representation  $\mathbf{A}_v$  and yields an approximation of the clean singing voice signal  $\mathbf{x}_v$ , denoted as  $\hat{\mathbf{x}}_v \in \mathbb{R}_{[-1,1]}^N$ . Specifically,  $D(\cdot)$  models the clean singing voice as a sum of  $C$  modulated sinusoidal components that overlap in  $\mathbb{R}^N$ . The components are computed using transposed convolutions with strides of  $S$  and another set of  $C$  kernels,  $\mathbf{w}_c \in \mathbb{R}^L$ , as

$$\hat{X}_{v[l,t]} = \sum_{c=0}^{C-1} A_{v[c,t]} w_{c[l]} , \quad (5.11)$$

where  $\hat{\mathbf{X}}_v \in \mathbb{R}^{L \times T}$  is the matrix containing the modulated components that are used to compute  $\hat{\mathbf{x}}_v$  as

$$\hat{x}_v[n] = \sum_t \hat{X}_{v[n-tS,t]} \forall n \in [0, 1, \dots, N-1]. \quad (5.12)$$

Eq. (5.12) is the overlap-add process (explained in Section 2.2 and Eq.(2.2)) applied for recovering the entire signal of length  $N$  and follows the assumption that

$$\hat{X}_{v[n-tS, t]} = 0 \text{ if } (n - tS) \notin [0, 1, \dots, L - 1].$$

Similar to the Sinc-Net [140] and its complex-valued extension for speech enhancement [141], the proposed method does not allow each  $\mathbf{w}_c$  to be updated directly using back-propagation. Instead, each  $\mathbf{w}_c$  is re-parameterized by amplitude modulated sinusoidal functions. The back-propagation is computed with respect to the corresponding parameters of the modulated sinusoidal functions. More specifically, each  $\mathbf{w}_c$  is computed using

$$\mathbf{w}_c[l] = \cos(2\pi f_c^2 l + \rho_c) \mathbf{b}_c[l], \quad (5.13)$$

where  $\cos(\cdot)$  is the cosine function and  $l = [0, \dots, L - 1]$  is the time index. The parameters that are jointly learnt with the parameters of the encoder  $E(\cdot)$ , are the sampling-rate-normalized carrier frequency  $f_c$ , the phase  $\rho_c$  (in radians), and the modulating signal  $\mathbf{b}_c \in \mathbb{R}^L$ . The direct access to natural quantities like the above described, significantly boosts the interpretability of the representation learning method. Additionally,  $\mathbf{w}_c$  can be sorted according to the carrier frequency  $f_c$ , promoting intuitive representations. The non-linear squaring operation applied to  $f_c$  is motivated by experimental results presented in Section 5.5.1.

There are three reasons for using modulated cosine functions for decoding  $\mathbf{A}_v$ : a) cosine functions promote interpretability [140], i.e. the representation  $\mathbf{A}_v$  is expected to convey amplitude related information for driving a well established synthesis model based on sinusoidal functions [7], b) the auto-encoding operation shares many similarities with the STFT yet without having to deal directly with the phase information, for which supervised based separation works remarkably well [100], [130], and c) amplitude modulations allow an extra degree of freedom in reconstructing signals that cannot be described by pure sinusoidal functions [7]. The latter statement is supported by the convolution theorem which states that the element-wise product of two vectors can be expressed in the Fourier domain as their corresponding convolution. Since in the proposed re-parameterization scheme (i.e. Eq. (5.13)) one of the signals is a cosine function, then  $\mathbf{b}_c$  is expected to convey timbre information regarding the signal  $\mathbf{x}_v$  that was used to compute the reconstruction objective.

After the reconstruction of  $\hat{\mathbf{x}}_v$ , the negative signal-to-noise ratio (neg-SNR) [150], is computed as

$$\mathcal{L}_{\text{neg-SNR}}(\mathbf{x}_v, \hat{\mathbf{x}}_v) = -10 \log_{10} \left( \frac{\|\mathbf{x}_v\|_2^2}{\|\mathbf{x}_v - \hat{\mathbf{x}}_v\|_2^2} \right), \quad (5.14)$$

where  $\|\cdot\|_2$  is the  $\ell_2$  vector norm, and the negative sign is used to cast the logarithmic SNR as a minimization objective. Then, the overall overall minimization objective for  $E(\cdot)$  and  $D(\cdot)$  is computed using  $\mathcal{L}_{\text{TV}}(\cdot)$ , for baseline comparison, as

$$L_A = \mathcal{L}_{\text{neg-SNR}}(\mathbf{x}_v, \hat{\mathbf{x}}_v) + \omega \mathcal{L}_{\text{TV}}(\mathbf{A}_m), \quad (5.15)$$

or using  $\mathcal{L}_{\text{SK}}(\cdot)$  as

$$L_B = \mathcal{L}_{\text{neg-SNR}}(\mathbf{x}_v, \hat{\mathbf{x}}_v) + \omega \mathcal{L}_{\text{SK}}(\mathbf{A}_m), \quad (5.16)$$

where  $\omega$  is a scalar that weights the impact of the representation objective (either  $\mathcal{L}_{\text{TV}}(\cdot)$  or  $\mathcal{L}_{\text{SK}}(\cdot)$ ) in the gradient (learning signal) used for optimizing  $E(\cdot)$ . In addition to this,

$L_A$  and  $L_B$  are scalar values that contain the overall loss that is used for optimizing the encoder and the decoder. The decoder  $D(\cdot)$  computes  $\hat{\mathbf{x}}_v$  only from the singing voice representation  $\mathbf{A}_v$ . That is because the signal  $\mathbf{x}_v$  is expected to be more appropriately modelled by sinusoidal functions compared to  $\mathbf{x}_m$  and thus providing more meaningful gradient information. That is because  $\mathbf{x}_m$  is expected to include transient signals that cannot be easily modelled with sinusoidal functions. Consequently, and in the case that  $\mathbf{x}_m$  is used as the target signal, the gradient information passed to the encoder might not be useful for learning structured representations [1], [138], [140]. Furthermore, it is aimed at learning general representations in an unsupervised and non discriminative fashion, i.e., repelling from mapping the mixture to the target source. To achieve that by means of the DAE model [87], it is reasonably assumed that the distribution of the corruption process is constant for all segments in the data-set [104]. This cannot be assumed for music signal mixtures, as even the distribution of the accompaniment instruments can vary dramatically from one segment to another. Consequently, by making such an assumption it could lead to degenerate learning of representations. It should be stated however, that using only  $\mathbf{x}_v$  as a target signal for optimization does not imply that after training the decoding of  $\mathbf{A}_m$  leads to poor approximations of  $\mathbf{x}_m$ . According to the provided support material, the reconstructions of  $\mathbf{x}_m$  via decoders that have been optimized with the above criteria lead to reasonable reconstruction performance.

## 5.4 Experimental Procedure

### 5.4.1 Data-set

For training and testing the representation learning method the MUSDB18 data-set [A8] is used. The data-set consists of 150 two-channel professionally produced multi-tracks, i.e, the stereophonic signals of bass, drums, singing voice, and other music instruments, that comprise a music mixture. Every signal is sampled at 44100 Hz. The multi-tracks are split into training (100 multi-tracks) and testing (50 multi-tracks) subsets.

### 5.4.2 Initialization & Hyper-parameter Selection

#### Initialization

Before the training process, the kernels in the first convolutions are randomly initialized with values drawn from a uniform distribution. The bounds of the uniform distribution are  $(-\sqrt{\frac{3}{C}}, \sqrt{\frac{3}{C}})$ , following the initialization strategy presented in [63]. For the decoder, the phase values  $\rho_c$  are initialized to zero, and all the elements of the modulating vectors  $\mathbf{b}_c$  are initialized to the value of  $\frac{1}{C+L}$ . The initialization of the normalized frequencies  $f_c$  is inspired by [140] and is performed by first computing the center frequencies of the Mel scale, denoted as  $f_{\text{Mel}}$ , in the range of  $f_{\text{Hz}} \in [30, \dots, 22050]$  Hz with a step-size equal to  $C$ . Then,  $f_{\text{Mel}}$  is computed as

$$f_{\text{Mel}} = 2595 \log_{10} \left( 1 + \frac{f_{\text{Hz}}}{700} \right)$$

and the initial value for each component  $f_c$  is computed as

$$f_c = \frac{700 \cdot 10^{\frac{f_{\text{Mel}}}{2595}} - 1}{44100} .$$

### Hyper-parameter Selection

For selecting the hyper-parameters of the convolutional networks and the training procedure, a pilot experiment is conducted. During this experiment, 20 randomly selected tracks from the training sub-set of MUSDB18 data-set were used. The objective of the pilot experiment is to determine the learning rate and the batch size of the solving algorithm, the standard deviation of the additive Gaussian noise for the corruption processes, described in Section 5.3, and the convolutional hyper-parameters. To that aim, the proposed method was trained *without* the representation objective, with the only objective to reconstruct the singing voice signal from its corrupted version. The results from each experimental run were assessed by means of informal listening tests, focusing on the subjective quality of the reconstruction of the singing voice.

The results from the above described experimental procedure are the usage of the adam algorithm [50], with a learning rate equal to  $1e^{-4}$  and a batch size of 8. In addition to this, the following hyper-parameters for the convolutional layers: (number of kernels)  $C = 800$ , (stride size)  $S = 256$ , (temporal length of each kernel in the first encoding layer)  $L = 2048$ , (temporal length of each kernel in the second encoding layer)  $L' = 5$ , and (dilation factor for the second encoding layer)  $D = 10$  provided perceptually good reconstruction. Furthermore, it was observed that the method converges fast, so for the complete experimental procedure the total number of iterations throughout the whole data is set to 10. In similar vein, a standard deviation of  $1e^{-4}$  for the additive Gaussian noise was found to yield good and relatively fast results from a range of values  $[1e^{-5}, 5e^{-5}, 1e^{-4}, 5e^{-4}, 1e^{-3}, 5e^{-3}, 1e^{-2}]$ . Based on the available computational resources<sup>4</sup> each multi-track is partitioned in segments of  $N = 44100$  samples (1 second long) and the largest possible batch-size.

### 5.4.3 Training

During training, a set of four multi-tracks is sampled. For each multi-track the vocals and all the other music instrument sources are used collectively. The accompaniment source is computed by adding the bass, drums, and other music instrument sources. Then, each sampled multi-track is down-mixed to a single channel and is partitioned into overlapping segments of  $N = 44100$  samples. The overlap is 22050 samples. The segments for each source are independently and randomly shuffled. Then, the singing voice signal segments are corrupted using the shuffled segments of the accompaniment source. For the corruption by additive Gaussian noise, the standard deviation of the noise remains constant and is independent from the amplitude of the singing voice signal. For optimizing the parameters of the representation learning method, with respect to the minimization of Eq. (5.15) or Eq. (5.16), the adam algorithm [50] is used. To compute the Sinkhorn distance(s), Eq. (5.9) is applied to each  $\mathbf{A}_m$  contained within the batch, and

---

<sup>4</sup>An Nvidia GTX 1050Ti GPU with 6GB of memory.

the average distance is computed across each batch, as typically performed in mini-batch-based gradient descent [50].

#### 5.4.4 Evaluation

For evaluating the usefulness of the representation that is learned by the proposed method, the rest of the 50 tracks from the MUSDB18 data-set are used. Each track is down-mixed and partitioned into non-overlapping segments of  $N = 44100$  samples. Shuffling and random mixing are not performed at this stage. However, the silent segments in the singing voice tracks are discarded. Discarding silent singing voice segments is based on:

$$l_{x_v} = 10 \log_{10}(\|\mathbf{x}_v\|_2^2 + \epsilon) \begin{cases} \mathbf{x}_v : \text{active,} & \text{if } l_{x_v} \geq -10 \\ \mathbf{x}_v : \text{silent,} & \text{otherwise,} \end{cases}$$

where  $l_{x_v}$  is the thresholding value for discarding a segment. The thresholding value is empirically chosen by finding the minimum value, in the training sub-set of MUSDB18 for all segments, that can be used to preserve all active singing voice segments.

The representation is evaluated with respect to the three following criteria: i) reconstruction error of the proposed method to encode and decode the clean singing voice signal using the previously described methodology, ii) reconstruction error of the separated singing voice signal by binary masking, and iii) additivity of the representation. The first and second criteria are objectively measured with respect to the clean singing voice signal  $\mathbf{x}_v$  using the scale-invariant signal-to-distortion ratio (SI-SDR) [82]. The scale-invariant signal-to-distortion ratio (SI-SDR), expressed in dB, is computed for each segment as

$$\begin{aligned} \text{SI-SDR}(\mathbf{x}_v, \hat{\mathbf{x}}_v) &= 10 \log_{10} \left( \frac{\|\alpha \mathbf{x}_v\|_2^2}{\|\alpha \mathbf{x}_v - \hat{\mathbf{x}}_v\|_2^2} \right), \text{ for} \\ \alpha &= \frac{\tilde{\mathbf{x}}_v^\top \mathbf{x}_v}{\|\mathbf{x}_v\|_2^2}. \end{aligned} \quad (5.17)$$

Higher SI-SDR values indicate better reconstruction or separation performance. It should be noted that the first criterion is used only to evaluate the reconstruction capabilities of the proposed re-parameterization scheme and not the learning capabilities of the overall method for learning representations. That is because this reconstruction criterion does not support the claim of the proposed method to be unsupervised, since the reconstruction of the singing voice has been used as an optimization objective; yet it serves as an informative quality indicator for audio signals.

For performing the task of singing voice separation, informed binary masking is used. That is because masking is an important operation in audio and music source separation, and has been extensively used by deep learning based approaches and also representation learning [143]. The focus is given on informed separation, i.e., masks are computed by an oracle method using the information for all the mixture's sources available in the data-set. This is done in order to estimate the least-upper-bound performance of singing voice separation, for a learned representation. This alleviates the biases on the prior information that music source separation approaches have. Examples of biases include the source's structure and the existing neural architectures that are engineered for the representations computed using the STFT. Finally, binary masking is used because it is



an indicator of how disjoint (less overlap) two sources are, given a representation<sup>5</sup>. More specifically, the oracle binary masking is computed by encoding three signals. The first signal is the mixture  $\mathbf{x}_m$ , the second signal is the accompaniment source  $\mathbf{x}_{ac}$ , and the singing voice signal  $\mathbf{x}_v$ . The representations  $\mathbf{A}_m$ ,  $\mathbf{A}_{ac}$ , and  $\mathbf{A}_v$  of the signals  $\mathbf{x}_m$ ,  $\mathbf{x}_{ac}$ , and  $\mathbf{x}_v$ , respectively, are computed using the trained encoder  $E(\cdot)$ . The mask  $\mathbf{G}_v \in \mathbb{R}^{C \times T}$  is computed as

$$\mathbf{G}_v = g(\mathbf{A}_v \oslash \mathbf{A}_{ac}) ,$$

where  $g(\cdot)$  is defined as

$$g(x) = \begin{cases} 1, & \text{if } x \geq 0.5 \\ 0, & \text{otherwise} \end{cases} .$$

The approximation of the singing voice time-domain signal  $\hat{\mathbf{x}}_v$  using the decoder  $D(\cdot)$  and by means of binary masking is computed as

$$\hat{\mathbf{x}}_v = D(\mathbf{A}_m \odot \mathbf{G}_v) .$$

The additivity of the sources is computed using the following objective metric

$$\mathcal{A}(\mathbf{x}_m, \mathbf{x}_v, \mathbf{x}_{ac}) = 1 - \frac{\|E(\mathbf{x}_m) - E(\mathbf{x}_v) - E(\mathbf{x}_{ac})\|_1}{\|E(\mathbf{x}_m)\|_1 + \varepsilon} , \quad (5.18)$$

where  $\|\cdot\|_1$  is the  $L_1$  matrix norm,  $\varepsilon = 1e - 24$  is a small term for ensuring numerical stability, and  $\mathbf{x}_{ac}$  is the time-domain signal of the accompaniment music source that is computed by mixing the multi-tracks available in the testing subset. High values of  $\mathcal{A}(\cdot)$  that are close to 1 indicate that the representation of the mixture signal consists of additive sources (higher  $\mathcal{A}(\cdot)$  is better). The attribute of additivity is important for the computation of optimal separation masks [70], and in the unsupervised separation of music sources [33], [151].

### 5.4.5 Assessing Design Choices

Using the procedures that are described in Section 5.4.3 and Section 5.4.4, two additional experiments are conducted. For both experiments every model is optimized three times using different initial random seeds. For the first experiment, the modulated cosine functions (**mod-cos**) are examined for their applicability as synthesis model by measuring the reconstruction performance, after being optimized for the denoising task. It should be noted that for the first experiment the corruption process with the randomly shuffled segments of the accompaniment signal is not considered. Furthermore, an early stopping mechanism is used to terminate the training procedure if the model under examination has stopped decreasing the reconstruction objective (neg-SNR), expressed in Eq. (5.14), on average with respect to the batches in the previous iteration. For comparison, various modifications to the presented method for representation learning and decoding strategies from related literature are considered in this experiment. Specifically, the squaring of the normalized frequencies  $f_c$  is examined, among other decoding strategies such as non-modulated cosine functions (**cos**), and common one-dimensional convolutional networks

<sup>5</sup>For the detailed connection between disjointness and binary masking see Appendix C.

(**conv**) with and without the tanh non-linearity applied at the last stage of the decoding process. In addition to this, Sinc-Net [140] (**sinc**) Sinc-Net is examined as the first encoding stage as proposed in [140]. For this experiment,  $C$  is adapted for each model so that the same number of parameters is used by the models.

The best combination of the decoding and non-linear functions from the first experiment are further investigated in the second experiment. In this experiment the following values for the number of components  $C \in [400, 800, 1600]$  are examined. Furthermore, the effect of the representation objective is examined with respect to the usage of information either from the additive corruption by multi-modal noise or the additive corruption by Gaussian noise, i.e., using either  $\mathbf{A}_v$  or  $\mathbf{A}_m$ . For this experiment, the (an-isotropic) total-variation denoising (Eq. (5.7)) objective is used, as the Sinkhorn distances are computationally very demanding and significantly slow-down the training procedure. For comparison, the STFT is employed by performing the above described operations of analysis, masking, and synthesis. The STFT uses a hop-size of 256 samples, a window size of 2048 samples, and the hamming windowing function.

## 5.5 Results & Discussion

### 5.5.1 Results from Design Choices Evaluation

Table 5.1 demonstrates the median values of SI-SDR expressed in dB (the higher the better) yielded by the first experiment, with additional information regarding the various setups for the encoder  $E(\cdot)$  and the decoder  $D(\cdot)$ , the number of parameters  $N_P$  (in millions M), the used number of components  $C$ , and the employed non-linearities. The results in Table 5.1 highlight three trends. First, the application of the non-linearity to the normalized frequencies  $f_c$  results into better reconstruction performance compared to the linear case. The observed improvement is of  $\sim 3$ dB on average across experimental configurations. Secondly, the modulated cosine functions serve as a good differentiable synthesis model for singing voice signals, outperforming simple cosine functions by approximately 8 dB on average, with respect to the two experimental configurations (with and without frequency scaling of the normalized frequency), and by 1.4 dB the best configuration of convolution based model (**conv**). Since SI-SDR is invariant to scale modifications of the assessed signal, 1.4 dB is a significant improvement of signal quality and does not imply a simple matching of the gain that the model based on modulated cosine functions might have exploited. Thirdly, Sinc-Net [140] does not bring further improvements to the proposed method.

Focusing on the separation performance of the obtained representations, Table 5.2 presents the median SI-SDR values of the binary masking separation scenario, for three values for the hyper-parameter  $C$  and two strategies for computing the representation objective. These strategies consider two different signal representations that are either the corrupted by Gaussian noise  $\mathbf{A}_v$  or the synthetic mixtures using the accompaniment signals  $\mathbf{A}_m$ . The obtained results are compared to the common convolutional encoder/decoder setup used in Table 5.1 and the STFT that has perfect reconstruction properties and masking techniques work very well in practice [A9]. The results of Table 5.2 mainly underline two experimental findings. The main finding is that using the representation objective and information from the realistic corruption process  $\mathbf{A}_m$ , it can

Table 5.1: Results reflecting the decoding performance, by means of SI-SDR. Bold-faced numbers denote the best performance.

$E(\cdot)/D(\cdot)$ Setup	Non-linearity	$C$	SI-SDR	$N_P$
conv/cos	N/A	952	20.83	6.483M
	$f_c^2$		22.34	
conv/conv	N/A	800	31.25	6.476M
	$\tanh(\text{decoder})$		30.50	
conv/mod-cos	N/A	800	28.72	6.478M
	$f_c^2$		<b>32.62</b>	
sinc/mod-cos	$f_c^2$	952	26.82	6.487M

Table 5.2: SI-SDR for informed separation by binary masking (BM). Bold-faced numbers denote the best performance.

$E(\cdot)/D(\cdot)$ Setup	$C$	$\mathcal{L}_{TV}(\mathbf{A}_m/\mathbf{A}_v)$	SI-SDR	SI-SDR-BM	$N_P$
conv/mod-cos	400	$\mathbf{A}_v$	30.46	3.66	2.439M
		$\mathbf{A}_m$	30.73	5.93	
	800	$\mathbf{A}_v$	<b>32.28</b>	4.39	6.478M
		$\mathbf{A}_m$	32.11	6.28	
	1600	$\mathbf{A}_v$	31.94	4.68	19.356M
		$\mathbf{A}_m$	31.54	6.68	
conv/conv	800	$\mathbf{A}_v$	31.25	2.89	6.476M
		$\mathbf{A}_m$	31.13	4.95	
STFT/iSTFT	1025	N/A	N/A	<b>8.80</b>	N/A

be used to improve the reconstruction of the masked mixture signals without additional supervision, as previous studies suggest [143]. This claim is supported by the observed improvement of  $\sim 2$  dB, on average across models of various components  $C$ , when the synthetic mixtures are used for the unsupervised representation objective. Furthermore, the proposed re-parameterization scheme improves by approximately 1.6 dB the separation performance compared to typical convolutional networks. Nonetheless, there is much room for improvements in order to obtain the quality of the STFT/iSTFT approach that outperforms the best masked approximation of the proposed method by 2.12 dB.

## 5.5.2 Representation Learning Results

Table 5.3 presents the average and standard deviation values of the additivity measure  $\mathcal{A}(\cdot)$ , the SI-SDR for the reconstruction and the separation objective performance in dB, and the values of the hyper-parameters  $\omega$  and  $\lambda$  used to compute the two representation objectives. The results in Table 5.3 are discussed according to the SI-SDR value (higher is better), because SI-SDR assesses the reconstruction and separation performance.

There are two observable trends in Table 5.3. The first trend is that when using  $L_B$ , i.e., Eq. (5.16), small values of  $\lambda$  marginally improve the SI-SDR, compared to the best SI-SDR when using  $L_A$  (i.e., Eq. (5.15) for  $\omega = 0.5$  and SI-SDR=31.49). Specifically,

Table 5.3: Results from objectively evaluating the learned representations. Values in boldface denote the best obtained performance.

Objective	$\omega$	$\lambda$	SI-SDR (dB)	SI-SDR-BM (dB)	$\mathcal{A}(\cdot)$
$L_A$	0.5	N/A	31.49 ( $\pm 2.98$ )	4.43 ( $\pm 4.98$ )	0.76 ( $\pm 0.10$ )
	1.0	N/A	31.39 ( $\pm 3.16$ )	4.66 ( $\pm 4.92$ )	0.76 ( $\pm 0.10$ )
	1.5	N/A	31.01 ( $\pm 3.13$ )	4.97 ( $\pm 4.93$ )	0.75 ( $\pm 0.10$ )
	2.0	N/A	30.96 ( $\pm 2.98$ )	4.65 ( $\pm 4.90$ )	0.76 ( $\pm 0.10$ )
	4.0	N/A	31.40 ( $\pm 2.83$ )	5.06 ( $\pm 4.97$ )	0.76 ( $\pm 0.10$ )
$L_B$	1.0	0.1	31.28( $\pm 2.98$ )	5.40( $\pm 5.31$ )	0.76( $\pm 0.09$ )
	1.0	0.5	<b>31.61</b> ( $\pm 3.38$ )	<b>5.63</b> ( $\pm 5.29$ )	0.77( $\pm 0.09$ )
	1.0	1.0	31.29( $\pm 3.25$ )	4.33( $\pm 5.28$ )	0.86( $\pm 0.08$ )
	1.0	1.5	29.98( $\pm 3.48$ )	0.06 ( $\pm 6.43$ )	<b>0.89</b> ( $\pm 0.08$ )
	1.0	2.0	31.13( $\pm 3.66$ )	-0.02( $\pm 6.44$ )	<b>0.89</b> ( $\pm 0.08$ )

when using  $L_B$  as the representation objective and for  $\lambda = 0.5$ , the SI-SDR and SI-SDR-BM are improved by 0.12 dB and 1.20 dB, respectively, compared to the case of using  $L_A$  and  $\omega = 0.5$ . Additionally, with the same  $\lambda = 0.5$  for  $L_B$ , an improvement of 0.57 dB SI-SDR-BM can be observed, compared to the best obtained SI-SDR-BM using  $L_A$  with  $\omega = 4.0$ . This trend shows that when using the Sinkhorn distances as an objective (i.e.,  $L_B$ ) with a small entropic regularization weight, i.e., small values of  $\lambda$ , there is a marginal improvement of the reconstruction performance for the singing voice (measured with SI-SDR-BM), but also the learned representations yield better results for singing voice separation (measured with SI-SDR).

The second trend observed in Table 5.3 is that when using  $L_B$  and  $\lambda > 1$ , specifically for  $\lambda \in [1.5, 2.0]$ , the SI-SDR for binary masking drops by more than 5 dB, compared to  $L_B$  with  $\lambda = 0.5$ . This indicates that the separation by binary masking fails, suggesting that the singing voice and accompaniment are completely overlapping in the representation of the mixture  $\mathbf{A}_m$ . That is expected since entropy expresses the uncertainty about the representation of the mixture signal. This means that during training, all the components of the representation are equally probable to be active when the mixture signal is encoded. Interestingly enough, that uncertainty in the encoding process comes with the observed effect that the sources become additive in the learned representation.

To further investigate the effect of entropic regularization with respect to the additivity metric, the impact of the weight  $\omega$  on  $L_B$  is examined. To that aim, the best  $\lambda = 1.5$  from Table 5.3 is chosen as a fixed hyper-parameter and  $\omega$  is varied. The corresponding results are given in Table 5.4 and are compared to the magnitude representation computed using the STFT, that is the most commonly employed representation for music source separation. The results from Table 5.4 suggest that by increasing the weight  $\omega$  that affects the strength of the representation objective in the learning signal, the learned mixture representations consist of two almost additive representations, i.e., the singing voice and the accompaniment representations. This is observed for  $\omega = 4.0$ . Furthermore, nearly all representations computed using the Sinkhorn distances and the entropic regularization, outperform the magnitude of the STFT with respect to the objective measure of additivity in an unsupervised fashion, i.e., additivity was not explicitly enforced using

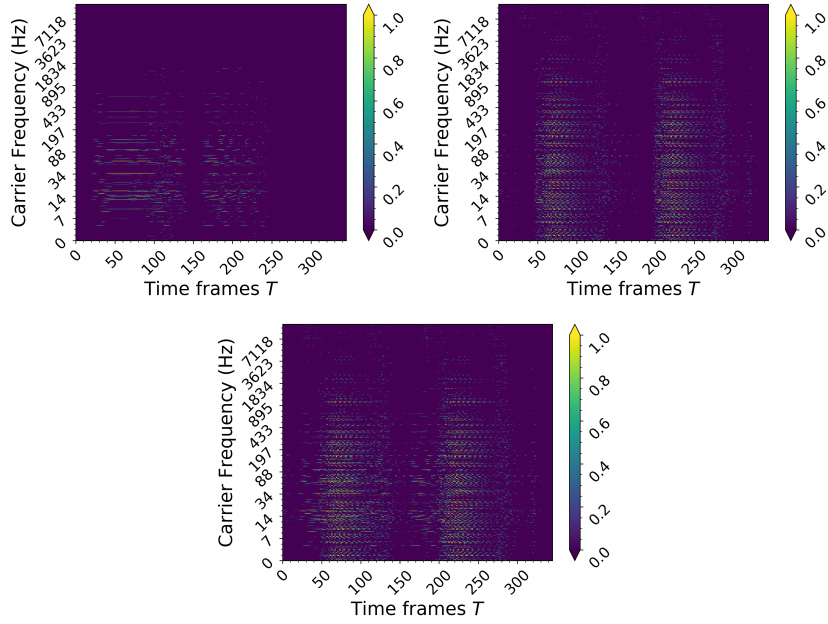
an optimization objective.

Table 5.4: Objective evaluation of the additivity of the learned representations.

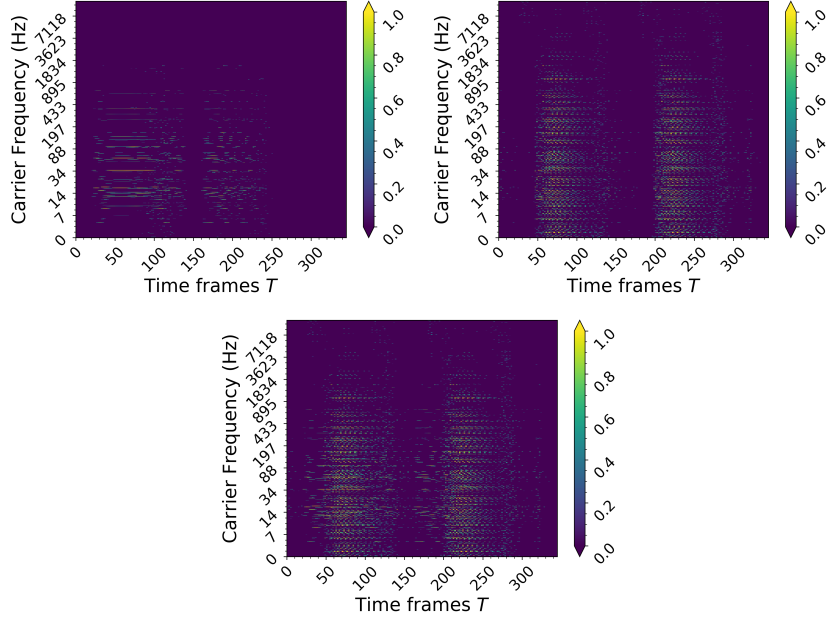
Objective	$\omega$	$\lambda$	$\mathcal{A}(\cdot)$
$L_B$	1.0	1.5	0.89 ( $\pm 0.08$ )
	1.5	1.5	0.90 ( $\pm 0.07$ )
	2.0	1.5	0.92 ( $\pm 0.07$ )
	4.0	1.5	<b>0.93</b> ( <b><math>\pm 0.06</math></b> )
STFT	N/A	N/A	0.86 ( $\pm 0.06$ )

To qualitatively assess the representations for the extreme case observed in Table 5.4, Figure 5.2 illustrates the learned representations for the mixture, singing voice, and the accompaniment signal using either  $L_A$  or  $L_B$ . The signals were acquired from a single multi-track segment contained in the testing sub-set of MUSDB18. For  $L_B$  the focus is given on two extreme cases of separation and additivity performance, that was observed in Table 5.3 and Table 5.4. In particular, Figure 5.2 illustrates the representations obtained for entropy values  $\lambda = 1.5$  and for  $\lambda = 0.5$ , that resulted in the best performance of additivity and masking, respectively. For comparison, the learned representations using  $L_A$  are displayed for  $\omega = 4.0$ , which yields the best separation performance according to Table 5.3.

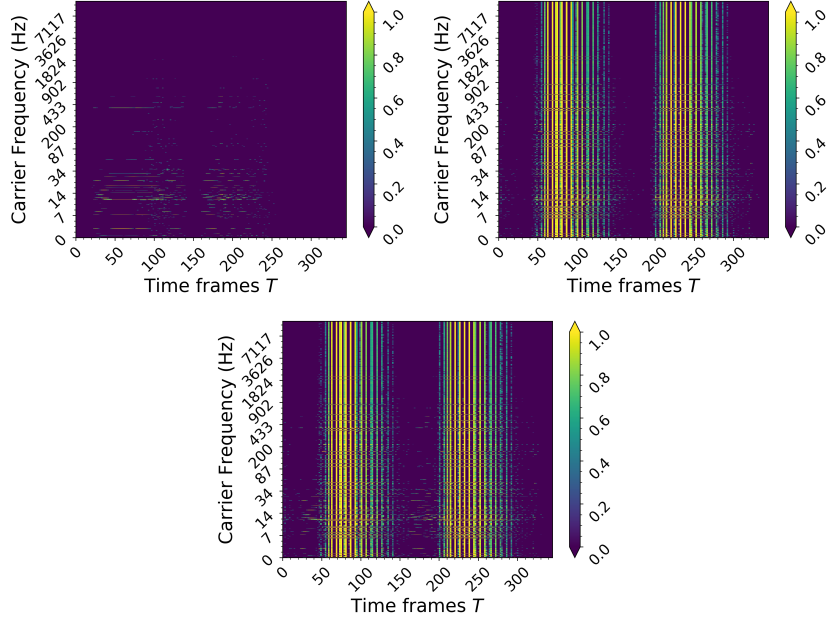
In Figure 5.2(a) it can be clearly observed that the usage of  $L_A$  (employing the total-variation denoising loss) leads to smooth representations. However, qualitatively the representation of the mixture and of the sources seem somewhat blurry, without distinct structure.



(a) Learned representations for the singing voice (top-left), the accompaniment (top-right), and the mixture (bottom-middle) signals using the  $E(\cdot)$  optimized with  $L_A$  for  $\mathcal{L}_{TV}(\cdot)$  with  $\omega = 4.0$



(b) Learned representations for the singing voice (top-left), the accompaniment (top-right), and the mixture (bottom-middle) signals using the  $E(\cdot)$  optimized with  $L_B$  for  $\mathcal{L}_{SK}(\cdot)$  with  $\omega = 1.0$ , and  $\lambda = 0.5$



(c) Learned representations for the mixture (left), the singing voice (middle), and the accompaniment (right) signals using the  $E(\cdot)$  optimized with  $L_B$  for  $\mathcal{L}_{SK}(\cdot)$  with  $\omega = 4.0$ , and  $\lambda = 1.5$

Figure 5.2: An illustration of the learned representations of a multi-track segment, by three encoders  $E(\cdot)$  optimized using various hyper-parameters for  $L_A$  and  $L_B$ .

Consequently, representations learned using  $L_A$  might impose difficulties for source separation methods that aim at capturing the structure of the target music source(s). On the other hand, the employment of  $L_B$  with the Sinkhorn distances and for  $\lambda = 0.5$ , leads to learned representations that at least for the singing voice signal a prominent structure of horizontal activity is observed. The interesting part comes when the entropy regularization weight is increased to  $\lambda = 1.5$ . Values of entropic regularization higher than 0.5, enable the learning of representations that for particular sources such as the accompaniment, exhibit distinct structure, i.e., vertical activity (activity with respect to  $C$ ). Furthermore, the representation of the singing voice is characterized by horizontal activity, i.e., a few components  $C$  are active and smoothly vary in time. The observed representation structures could be useful for unsupervised separation or audio in-painting methods, such as the deep audio prior [152] and the harmonic convolution(s) model [153].

### On Representation Interpretability

An important attribute of the learned representation(s), by using the proposed method, is the interpretability, i.e., the learned representations convey information about functions whose parameters have physical quantities such as frequency for example. This is the only guarantee for interpretability that the proposed method offers and it can be seen by inspecting closer Figure 5.2, where each component  $C$ , i.e., each row of the spectrogram-like illustration, has a carrier frequency that is expressed in Hz. This rationale can be seen as an analogous to common representations such as STFT that has been extensively used in audio signal processing. However, there are two main differences between the proposed method and the STFT.

The first difference is that the encoding functions of the proposed method are not forced to be symmetric to the decoding functions. This is in contrast to the DFT analysis (encoding) basis functions, employed by the STFT, that are symmetric to the synthesis (decoding) basis functions. This in turn, gives many more degrees of freedom to the encoder of the proposed method, to yield representations that can be optimized with specific objectives. As seen from Figure 5.2(c) the Sinkhorn distances, with some degree of entropy, allow the computed representations of the accompaniment source to be distinctly structured, something that would not be possible by using symmetric encoding functions. That is because the usefulness of symmetric functions is the perfect reconstruction of a signal after encoding and decoding and not the structure of the output of the encoding, i.e., the representation [32], [36]. However, this might impose the necessity of devising representation objectives for optimizing the encoder of the proposed method by using domain knowledge from audio and music signal processing.

The second difference is that the decoding functions employed by the proposed method are amplitude-modulated cosine functions as opposed to pure cosine functions that common (audio) transforms have. The main drawback in this case, is that the modulating signal is directly updated by using back-propagation and it might be hard to interpret after the training procedure. However, the difficulty in interpretation can be tackled by recalling Eq. (5.13), in which the signal that is being modulated, i.e., the carrier signal, is a cosine function. This in turn, makes the update rules, based on gradient descent, for the modulating signal to be the linear combination of sinusoidal functions convolved with some noise. That can be verified by evaluating the gradient of the reconstruction error



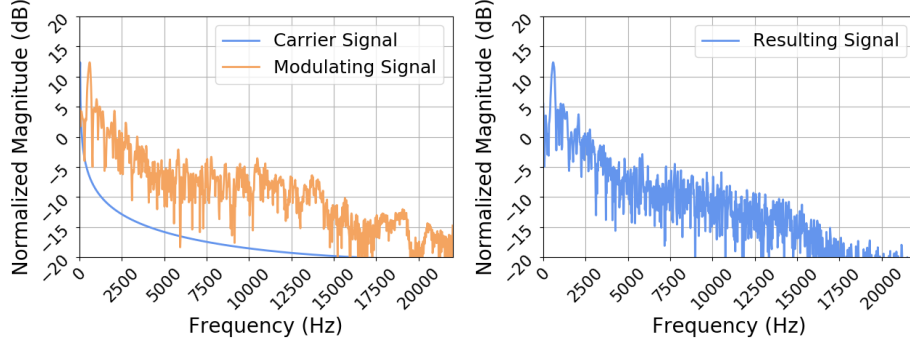
with respect to the modulating signal, that results into a sinusoidal function. Particularly, that function is the convolution of a sinusoid with the derivative of the reconstruction loss with respect to the reconstructed signal. In addition to this, the modulating signal allows an extra degree of freedom in reconstructing signals that cannot be described by pure sinusoidal functions [7], especially when additional representation attributes, such as non-negativity and smoothness, are intended to be learned.

To qualitatively assess the information that the modulating functions inherit from the training procedure, Figure 5.3 illustrates the frequency response of the carrier and the modulating signal for frequently used components  $\mathbf{w}_c$  that are in the lower carrier frequency region. The frequency response is obtained by computing the magnitude of the DFT for each corresponding signal. As it can be seen from Figure 5.3, the frequency response of the modulating signal (orange line) consists of a combination of sinusoidal components that have both harmonic structure, considering the position of the observed spectral peaks, but also a stochastic spectral structure. The stochastic structure reassembles formants and/or fricatives of the singing voice signal. This shows that the modulating signal increases the flexibility of the decoder, by allowing the decoder to capture information for formants and/or fricatives of the singing voice alongside the cosine functions. Nonetheless, this means that signal operations in the computed representation, like masking, will affect a greater proportion of the singing voice signal compared to typical sinusoidal functions employed by the STFT. This in turn, might not be ideal in general applications such as frequency equalization, where only the specific frequency regions have to be processed in a deterministic way.

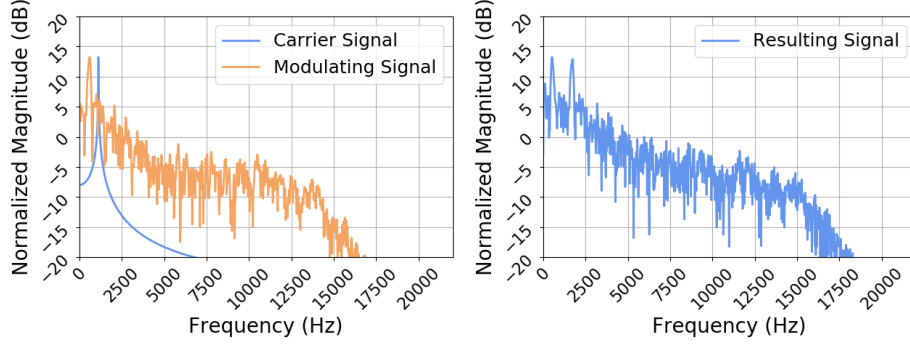
### 5.5.3 Sinkhorn Distances Results

To complement the results from using  $L_B$  that is computed using the Sinkhorn distances, Figure 5.4 presents results from the objective evaluation of the learned for a greater range of hyper-parameter values. Particularly, Figure 5.4 contains error plots for the following range of entropic regularization weights  $\lambda \in [0.1, 0.5, 1.0, 1.3, 1.5, 2.0, 5.0, 10.0]$  and for  $\omega = 1.0$ . To justify the choice for  $p = 1$  for computing the pair-wise distances matrix  $\mathbf{M}$ , used for computing the Sinkhorn distances, additional results for  $p = 1$  and  $p = 2$  are illustrated in Figure 5.4. By observing Figure 5.4, two observations can be highlighted. The first observation is that the computation of the loss matrix  $\mathbf{M}$  (Eq. (5.9)) for  $p = 2$  leads to marginally sub-optimal results, compared to  $p = 1$ , for nearly all  $\lambda$  values and with respect to all the evaluation metrics. Specifically, the reconstruction performance for  $p = 1$  is better than  $p = 2$  by 1 dB on average across  $\lambda$  values. Also, for  $p = 1$  an improvement of 0.6 dB on average with respect to the performance of separation by masking is observed in comparison to  $p = 2$ . For the additivity metric,  $p = 2$  marginally outperforms  $p = 1$  for a negligible difference of  $3e^{-3}$ . These results indicate the reason why the above presented results focus on  $p = 1$ .

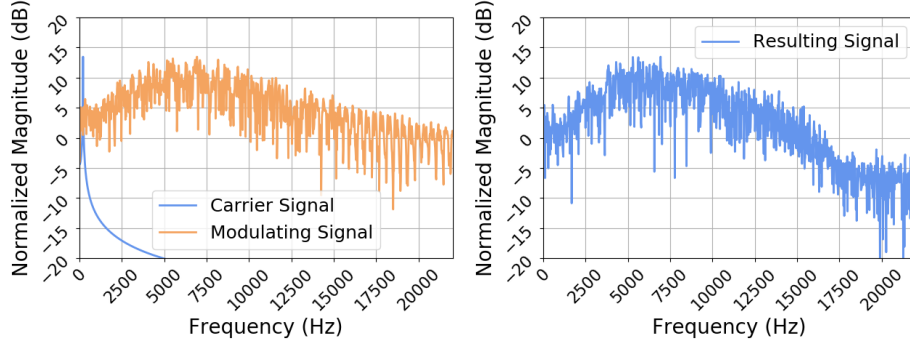
Another observation from Figure 5.4 is that for  $\lambda > 2$  the observed separation performance dip and additivity performance peak disappear in the area of  $\lambda \in [1.3, 1.5, 2.0]$ . In this area the examined method performs similarly to the values of low entropy, with respect to the examined metrics. This contradicts the expectations for the effect of entropic regularization. The only explanation to this behavior is that for values  $\lambda > 2$ , the exponential function used in the computation of the Sinkhorn distances and is applied



(a) Frequency response of a learned basis function, (left) carrier and modulating signal, (right) result of the modulation



(b) Frequency response of a learned basis function, (left) carrier and modulating signal, (right) result of the modulation, demonstrating a sinusoidal plus noise structure.



(c) Frequency response of a learned basis function, (left) carrier and modulating signal, (right) result of the modulation, demonstrating a harmonic plus noise structure.

Figure 5.3: The frequency response of three frequently used basis functions that are learned by the proposed method using  $L_B$  with  $\lambda = 1.5$  and  $\omega = 4.0$ . The frequency response is computed using the discrete Fourier transform, demonstrating a high-frequency comb-like filter.

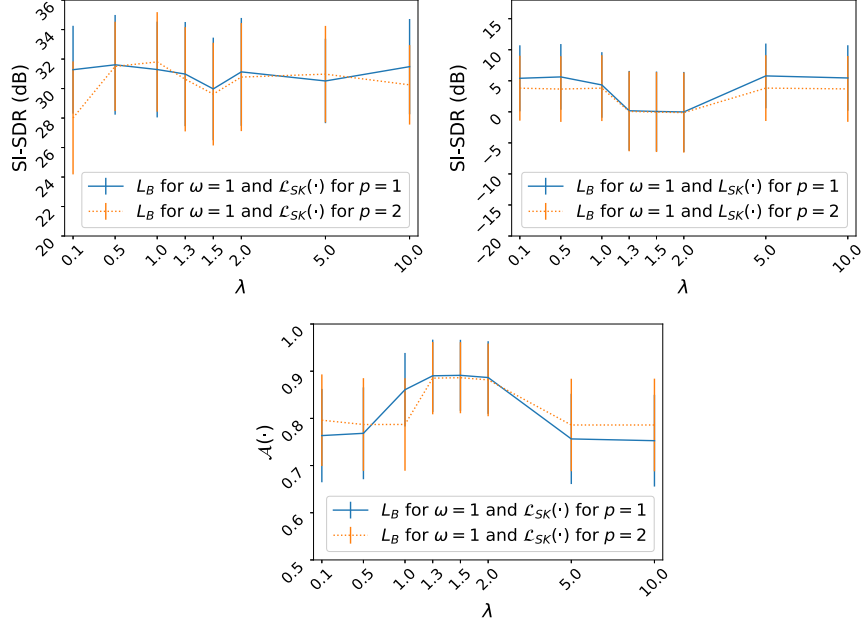


Figure 5.4: Performance evaluation of the learned representations by  $L_B$  that use the Sinkhorn distances. (top-left) Reconstruction of singing voice in SI-SDR, (top-right) oracle separation performance in SI-SDR, and (bottom) additivity objective measure. Horizontal and vertical lines denote the average and the standard deviation of the performance, respectively.

to  $\mathbf{M}$  (see Algorithm 3), yields saturated values that bias the overall minimization. The unexpected effect in the minimization of the computed loss values using the Sinkhorn distances for various values of  $\lambda$  is illustrated in Figure E.2 of Appendix E.

## 5.6 Summary

This chapter presented a method for learning representations of music signals that can be particularly useful for the task of music source separation. The presented method is based on the denoising autoencoder model [87] with modulated cosine functions for decoding bases, inspired by the differential digital signal processing concept [139]. The benefits of the proposed method are interpretability, due to the usage of the cosine functions for decoding, non-negativity promoting energy-informative representations akin to the magnitude of the STFT, and the fact that the proposed method can be trained in an unsupervised fashion, enabling the usage of unlabeled and unpaired multi-track data.

Focusing on the important problem of singing voice separation, the proposed method was investigated for its performance in separation, additivity of the sources' representations, and the reconstruction of the singing voice signal. Furthermore, representation objective functions were examined for improving the attributes and the performance of

the learned representations. Specifically, two objectives were examined, the (an-isotropic) total-variation denoising loss [146], and the family of Sinkhorn distances with entropic regularization [148]. The results from the experimental procedure, suggest that representations for music signals can be learned using unsupervised learning, leading representations that can be employed for the separation of singing voice by masking. In addition to this, Sinkhorn distances as an efficient computation for optimal-transportation distances, allow a flexible learning of representations in an unsupervised way, with the entropic regularization leading to sources' representations that are distinctly structured and are almost additive; attributes that are useful in music source separation.

## Chapter 6

# Conclusions

### 6.1 Thesis Summary

This thesis addressed the problem of music source separation using deep learning methods. The focus was given on learning (deep) models that are useful for separating music sources. The problem at hand was examined from three angles, namely the signal processing, the neural-architecture, and the signal representation angle. The presentation of the results and research findings from each angle has been organized in five chapters.

Chapter 1 provided an introduction to the problem of music source separation in the context of deep learning. This thesis categorized deep learning approaches to music source separation as a *domain informed* way to separate the target music source(s). Subject to *a priori* knowledge of the mixing process and the target source(s), domain informed approaches stand on the opposite side of the spectrum compared to *blind* or *semi-blind* approaches, that exploit none or minimum information, respectively. In related literature [A9], [88], information used by deep learning approaches has been shown to bring substantial improvements to the separation performance.

In Chapter 2, the focus was on the fundamental concepts that are commonly used in deep learning-based music source separation research. With the main goal being the introduction to concepts that were used throughout the thesis, this chapter started from basic signal models and built up to artificial neural networks, showing how artificial neural networks naturally emerge to tackle problems in audio signal processing. Specifically, it was demonstrated that the framework of *energy-based learning* can be used to leverage the design and the learning of signal representations, as well as the training of artificial neural networks to solve a particular and well-defined problem. In addition to this, commonly assumed mixing models were discussed and it was shown how the *filtering* operation can be used to separate music sources.

Chapter 3 proceeded to discuss the signal processing angle of this thesis. Specifically, it aimed at demonstrating what (deep) neural networks learn when they are trained to separate a music source. Furthermore, it investigated whether there is an analogous digital signal processing operation, that can be used to evaluate their learning capabilities. To do so, the focus was given to singing voice separation and to the estimation of the target sources' magnitude spectrogram, a strategy that has been employed by

many music source separation approaches. The deep neural networks that are trained to separate the target source signal from the mixture, were first grouped under the family of the denoising autoencoder (DAE) model [86]. Then, an algorithm was presented for computing the mapping function(s) of the DAE family models. The mapping function expresses how the magnitude information from each frequency component is transformed, by each model, to obtain the estimated magnitude information of the target source. The proposed algorithm, denoted as the neural couplings algorithm (NCA), showed that the source separation models learn data-driven filtering functions. By employing basic signal processing knowledge it was also shown that the basic DAE model learned non-optimal filters for singing voice separation. That observation explained why many music source separation approaches rely on post-processing or post-enhancement techniques to yield better estimates of the target source(s). It was also demonstrated that by allowing DAEs to predict and optimize source-dependent filters, by the usage of skip-filtering connections, it enforces the DAEs to learn richer inter-frequency dependencies, justifying experimental results published in previous works [A10], [A11], [94], [96].

From the neural-architecture angle, Chapter 4 proposed a neural network architecture denoted as the Masker-and-Denoiser (MaD). The development of the MaD architecture was based on two concepts. The first concept was the skip-filtering connections, that allows prediction of optimized target-source filters that are very useful in music source separation in the time-frequency domain. The second concept was to treat the time-frequency representations as sequences with structured temporal information. Based on the two concepts, the developed architecture consisted of recurrent neural networks (RNNs) and the skip-filtering connections combined with (back) propagation of reconstruction errors. The latter were used to improve the performance of the MaD architecture. In addition to that, extensions to the MaD architecture were presented and evaluated both objectively and subjectively, by means of listening tests. The extensions were used for improving both the learning but also the processing of the sequences of music signals' spectrograms, by the MaD architecture. From the conducted experiments, it was concluded that the proposed architecture yielded a fair separation quality, and some of the discussed extensions provided perceptually relevant improvements. Furthermore, the proposed architecture provided competitive, yet sub-optimal, results compared to very recent approaches. However, those approaches include data augmentation, post-processing, and enhancement stages that were not considered by the MaD architecture.

Finally, Chapter 5 focused on the learning of music signal representations and provided insights regarding the third angle examined in this thesis. Specifically, Chapter 5 proposed a (signal) representation learning method that is based on the DAE model [86] and the concept of differential digital signal processing [139]. The proposed method replaced the decoding functions with amplitude modulated cosine functions, whose parameters are learned from music signals. It was shown, that the replacement allowed the proposed method to yield non-negative and interpretable signal representations, that promoted the energy information with respect to frequency components, akin to commonly used transforms in audio signal processing. In addition to that, it was shown that by penalizing the proposed method using representation objectives, additional attributes were gained. Specifically, it was shown that the usage of entropic regularized optimal transport (OT) objectives enabled the proposed method to yield representations that were distinctly structured and almost additive, with respect to the target sources. The previously mentioned

attributes are important in music source separation by filtering.

## 6.2 Outlook

The work presented in this thesis was built upon and contributed to the research field of data-driven music source separation using deep learning. The methods that were analyzed, developed, and presented herein had the goal of merging the efficiency of deep learning algorithms with the domain knowledge of audio signal processing, not using deep learning as simply a powerful pre-processing step. That was performed with the long-term goal of developing audio signal processing pipe-lines that are intelligent, i.e., adaptive according to various high-level signal characteristics, and can be learned at a large scale.

To that aim, the thesis proposed an algorithm, the NCA in Chapter 3, for understanding what deep neural networks learn when they are optimized to separate music sources. With the proposed algorithm and through the examination conducted in the thesis, heuristic approaches in music source separation were validated. One of the main weaknesses of the proposed algorithm is that it does not scale, in a straightforward manner, to recently proposed architectures for music source separation. However, it should be noted that in the context of the study, described in Chapter 3, it was of high importance to understand and show what simpler architectures learn for the task of spectral-based music source separation. In other words, regardless of the simplicity of extending simple architectures with additional and non-linear computational operations, for the sake of improving objective scores, the understanding of the simple non-linearities is still an emerging aspect in music source separation. In addition to this, the idea of manipulating, by masking, the parameters of the deep neural networks, presented in Chapter 3, has received a lot of interest in multi-task learning [114] and distillation [110], leaving open directions for future research. For instance the core idea of the NCA could be further used for distilling big neural network models into smaller ones, that could operate on low-cost devices. Also, meta-learning for music source separation [137] could aim at predicting network parameters masks, that are dependent on the target source.

Towards the development of neural architectures that are efficient for music source separation, the work presented in Chapter 4 solely focused on the usage of RNNs. That was because the experimental evidence available during the development stages showed that RNNs are powerful information processing systems for modelling sequences of music signals. However, and as shown in Chapter 4, the usage of RNNs for processing time-frequency sequences can be tricky and various techniques should be considered for enforcing the RNNs to model long sequences. One of the employed techniques in this thesis, denoted as the TwinNet [133], was shown to yield perceptually relevant improvements to the overall separation quality of the singing voice. Furthermore, the necessary recursions and the high number of parameters used by the RNNs are not always optimal for separating music sources in the time-frequency domain, given the small publicly available data-set(s). One plausible direction for future research is the replacement of RNNs by dilated convolutional neural networks (CNNs), that could decrease the number of the employed parameters in deep models operating on audio and music signals [A23], [A25].

Finally, emerging topics in deep learning for audio and music signals are representation learning [138], interpretable deep learning [140], and differential digital signal processing [139]. Being inspired by these works, Chapter 5 described a method for learning

signal representations that are particularly useful for the stage of separation. The goal that was achieved is to represent music signals in an informative way, while enforcing some convenient attributes in the computed representation, such as non-negativity, additivity, and distinct structure of the target sources. These attributes have been of high importance in related music source separation works. With the proposed method for learning representations, legacy approaches in music source separation can be revisited but in a new signal domain. Furthermore, the results demonstrated in Chapter 5 serve as the first step towards building music signal representations that are interpretable and capture the structure of the music sources without the need of having annotated and aligned multi-track audio data. One plausible direction for future research is to revisit blind or semi-blind approaches to music source separation, by exploiting the convenient structure and the attributes of the computed representations of the presented method. An example of a separation approach is the usage of deep audio priors [152], [153] that is an unsupervised method for audio source separation. Furthermore, exploring additional re-parameterization schemes for the proposed method for learning representations, could assist in problems such as voice conversion or timbre-morphing between two arbitrary speech or audio signals. To this end, the distinct structure of the obtained representations could be beneficial in generative modelling with priors from convenient distributions.



# Appendix A

## List of Used Sound Examples

List of sound files used to compute the illustrations of Chapter 2.

1. Clarinet  
<https://freesound.org/people/tootoos24/sounds/418272/>
2. Snare drum  
<https://freesound.org/people/Theriavirra/sounds/270156/>
3. Singing voice  
<https://freesound.org/people/afleetingspeck/sounds/256994/>
4. Cello  
<https://freesound.org/people/xserra/sounds/242160/>
5. Drums sound  
<https://freesound.org/people/Krishmeister/sounds/413240/>

The sound file used to compute the representations illustrated in Fig. 5.2 of Chapter 5 is the track *Al James - Schoolboy Fascination* in the test sub-set of MUSDB18 [A8]. The extracted one second segment corresponds to the 35th and 36th second of the audio file.

# Appendix B

## Adam Algorithm

---

**Algorithm 4** *Adam* algorithm for stochastic optimization, based on [50] and with default values for the decay rates of the first and second order moment estimates  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ , respectively, learning rate  $\eta = 1e^{-3}$ , and  $\epsilon = 10e^{-8}$ .

---

**Require:** Gradient step  $u$  and the calculated gradient(s)  $\mathbf{D}_u$  at that step

- 1: **if**  $u = 1$  **then**
  - 2:   Initialize:  $\mathbf{M}_0 \leftarrow 0$ ,  $\mathbf{V}_0 \leftarrow 0$
  - 3: **end if**
  - 4:  $\mathbf{M}_u \leftarrow \beta_1 \odot \mathbf{M}_{u-1} + (1 - \beta_1) \odot \mathbf{D}_u$
  - 5:  $\mathbf{V}_u \leftarrow \beta_2 \odot \mathbf{V}_{u-1} + (1 - \beta_2) \odot \mathbf{D}_u^{\odot 2}$
  - 6: Store  $\mathbf{M}_u$  and  $\mathbf{V}_u$ , so that can be accessed at step  $u + 1$
  - 7:  $\hat{\mathbf{M}}_u \leftarrow \mathbf{M}_u \odot (1 - \beta_1^u)$
  - 8:  $\hat{\mathbf{V}}_u \leftarrow \mathbf{V}_u \odot (1 - \beta_2^u)$
  - 9:  $\mathbf{D}_{\text{out}} \leftarrow \eta \odot \hat{\mathbf{M}}_u \odot (\sqrt{\hat{\mathbf{V}}_u} + \epsilon)$
  - 10: **return** Updated gradient estimates  $\mathbf{D}_{\text{out}}$
- 

Algorithm 4 presents a slightly modified version of the original algorithm presented in [50]. The modifications affect only the clarity of the presented algorithm and are introduced so that Adam can be directly plugged into the previously described iterative schemes using pre-calculated gradients, e.g., Eq. (2.27), Eq. (2.28), and Algorithm 1. It should be noted that the estimation of the initial gradients and the update of the parameters is not included in Algorithm 4. It is assumed that the previously mentioned operations are performed during the iterative optimization of the corresponding parameters.

# Appendix C

## W-Disjointness Orthogonality

Let  $\mathbf{Y}_j \in \mathbb{C}^{N \times T'}$  be the complex-valued representation of the source's signal  $\mathbf{x}_j \in \mathbb{R}_{[-1,1]}^T$  computed using the STFT  $\mathcal{F}_{\text{STFT}}(\cdot)$ , where  $N, T$ , and  $T'$  are the number of frequency subbands in the STFT, the number of time-domain samples, and the number of time-frames respectively. Furthermore, assume that the STFT of the interfering source<sup>1</sup>  $\mathbf{x}_{j'} \in \mathbb{R}_{[-1,1]}^T$  is provided and is denoted as  $\mathbf{Y}_{j'} \in \mathbb{C}^{N \times T'}$ . Then, the BM,  $\mathbf{M}^{\text{BM}} \in [0, 1]^{N \times T'}$  is computed using the magnitude of the sources as

$$\mathbf{M}_j^{\text{BM}} = g(|\mathbf{Y}_j| \oslash |\mathbf{Y}_{j'}|), \text{ and}$$

$$g(y) = \begin{cases} 1, & \text{if } y \geq 0.5 \\ 0, & \text{otherwise} \end{cases}.$$

Then the W-DO measure is computed as

$$\text{W-DO} = \text{PSR} - \frac{\text{PSR}}{\text{SIR}}, \quad (\text{C.1})$$

where PSR and SIR are the preserved-signal-ratio and the source-to-interference ratio computed as

$$\text{PSR} = \frac{\|\mathbf{M}_j^{\text{BM}} \odot |\mathbf{Y}_j|\|_1^2}{\|\mathbf{Y}_j\|_1^2}, \text{ SIR} = \frac{\|\mathbf{M}_j^{\text{BM}} \odot |\mathbf{Y}_j|\|_1^2}{\|\mathbf{M}_j^{\text{BM}} \odot |\mathbf{Y}_{j'}|\|_1^2},$$

where  $\|\cdot\|_1$  is the unit matrix/vector norm. From the above expressions, it can be seen that for a W-DO value of one the sources are entirely disjoint, meaning that there is not overlap between the sources in the respective representation. In contrast, a W-DO value of zero means that the sources completely overlapped and the separation of the  $j$ -th source by binary masking is not possible. In the latter case, the inability of separating the  $j$ -th source is also reflected by extremely low PSR values indicating a poor reconstruction of the source after masking.

---

<sup>1</sup>The interfering source is the sum of all the sources in the mixture except the target source.

# Appendix D

## Supplementary Results: Masker-and-Denoiser Architecture

### The effect of the training objectives

The results from examining the effect of the training objectives in the performance of accompaniment separation are given in Table D.1. Specifically, the median values for the SDR, SIR, and SAR objective measures are reported for the MaD architecture that was trained with the MSE-based objective  $E_{\text{MSE}/L_1}^{\text{MaD}}$ , the KL-based objective  $E_{\text{KL}/L_1}^{\text{MaD}}$ , and the KL-based objective using the TwinNet  $E_{\text{KL}/\text{twin}}^{\text{MaD}}$ . For each objective two values of  $\omega = 0$  and 0.5 are reported. For  $E_{\text{KL}/\text{twin}}^{\text{MaD}}$ , the results are demonstrated only for  $\omega = 0.5$ , as for the proposed architecture is identical to  $E_{\text{KL}/L_1}^{\text{MaD}}$  for  $\omega = 0$ .

Table D.1: The effect of various training objectives on the performance in accompaniment separation, computed using the objective evaluation measures and the MUSDB18 dataset. Values in boldface denote the best obtained performance.

Objective	$\omega$	SDR (dB)	SIR (dB)	SAR (dB)
$E_{\text{KL}/L_1}^{\text{MaD}}$	0	10.01	12.61	12.89
	0.5	10.03	12.60	<b>12.91</b>
$E_{\text{MSE}/L_1}^{\text{MaD}}$	0	10.01	12.88	12.44
	0.5	<b>10.11</b>	<b>12.99</b>	12.53
$E_{\text{KL}/\text{twin}}^{\text{MaD}}$	0.5	10.09	12.89	12.86

Table D.1 shows that there is a minimal effect of various training objectives on the objective performance of accompaniment source separation. The notable differences are that the penalty term computed using the  $L_1$  matrix norm (Eq. (4.8)) improves the SDR and the SIR by 0.10 dB and by 0.11 dB, respectively, when the penalty term is combined with the MSE-based reconstruction term. The TwinNet technique improves marginally the SDR by 0.06 dB and the SIR by 0.29 dB in comparison to  $E_{\text{KL}/L_1}^{\text{MaD}}$  and for  $\omega = 0.5$ . The latter configuration yields higher SDR values compared to the case of  $\omega = 0$ . The  $E_{\text{MSE}/L_1}^{\text{MaD}}$  for  $\omega = 0.5$  surpasses all the other training objectives with respect to the SDR and SIR metrics. From the above it can be concluded that if the target is the accompaniment source, i.e., optimizing the MaD architecture for karaoke applications, the MSE-based objective combined with the  $L_1$  penalty term provides relatively good results.

# Appendix E

## Supplementary Results: Learning Representations for Separation

### Total-variation complimentary results

In Figure E.1 complimentary results for Table 5.3 are presented. Figure E.1 illustrates the obtained results using the (an-isotropic) total-variation denoising loss ( $\mathcal{L}_{TV}(\cdot)$ ), employed in the computation of the loss termed as  $L_A$ .

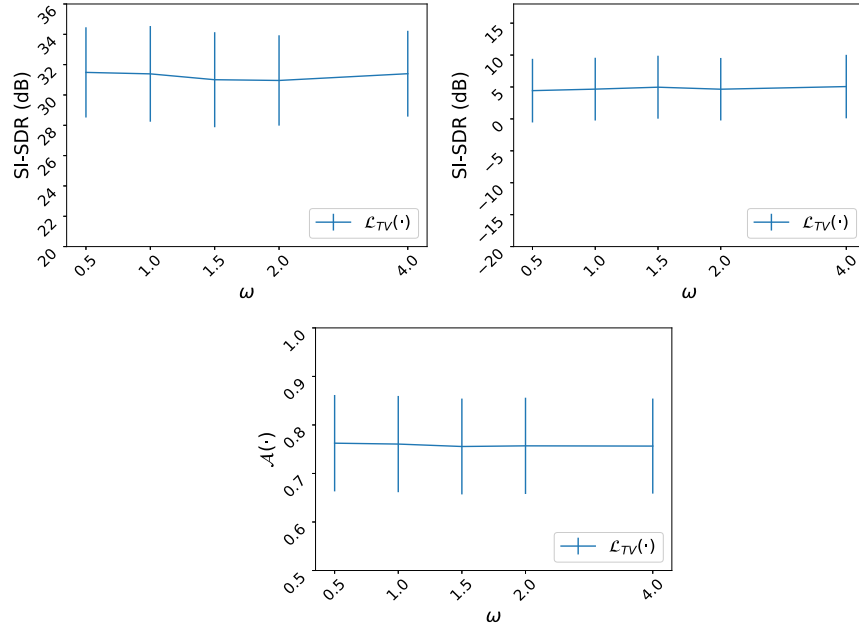


Figure E.1:  $L_A$  using total-variation denoising ( $\mathcal{L}_{TV}(\cdot)$ ) for various values of  $\omega$ : (top-left) Reconstruction of singing voice in SI-SDR, (top-right) oracle separation performance in SI-SDR, and (bottom) additivity objective measure.

## Sinkhorn distances complimentary results

Figure E.2 illustrates the output loss values of the  $\mathcal{L}_{\text{SK}}(\cdot)$  for the entropic regularization values  $\lambda = [1, 2, 5]$ . This figure serves as complimentary experimental results that show the saturation of the computed loss values for  $\lambda = [2, 5]$ , aiming at explaining the unexpected behavior of entropic regularization for high  $\lambda$  values discussed in Section 5.5.3.

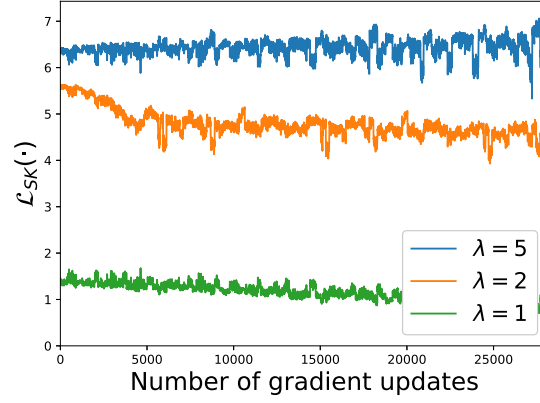


Figure E.2: The output values of  $\mathcal{L}_{\text{SK}}(\cdot)$  with  $p = 1$  and for various values for  $\lambda$  that are used to compute the loss  $L_B$ . The number of gradient updates corresponds to  $\sim 3$  full iterations throughout the whole training data-set. An eight-order quadratic smoothing filter with a window of 61 gradient updates is applied to the results for clarity.

# List of Figures

1.1	Three short-time excerpts of music signals, illustrating the structure of signals that exhibit (a) harmonic, (b) percussive, and (c) both signal characteristics. . . . .	4
2.1	A half-second signal recording of a cello music instrument. Zooming into a segment of 100 milliseconds (ms), a periodic and WSS signal is observed.	13
2.2	The Hamming windowing function applied to the 100 ms time-frame of the cello recording. . . . .	15
2.3	Magnitude and phase spectrogram representations of a cello recording (a) and a drum-set recording (b). The STFT is computed using a window size of 2048 samples, a Hamming windowing function, and step size of 256 samples. The redundant information of the negative frequency sub-bands is omitted. . . . .	20
2.4	An illustration of the covariance matrix of a cello recording. . . . .	21
2.5	Illustration of the GUI that is used to conduct listening tests. The GUI relies on the framework presented in [85]. . . . .	38
3.1	Illustration of the probabilistic graphical models <sup>1</sup> of encoder-decoder configurations examined in this chapter. The Illustration is based on [A22]. (a) <i>DAE</i> : a denoising auto-encoder model [87]. (b) <i>MSS-DAE</i> : a three layer example of a DAE model adapted to music source separation [76], [93]. (c) <i>SF</i> : skip-filtering connections [A6], [A10], [94], [96]. Solid arrows are functions computed by neural networks. Dashed arrows are the identity function. In the context of the input and latent variables, the symbol “ $\odot$ ” refers to the multiplication of the corresponding variables. . . . .	43
3.2	An illustration of how the couplings matrix $\mathbf{C}$ is computed using the compositional strategy. The compositional strategy takes into account the encoding/decoding matrices $\mathbf{W}_*$ by preserving and scaling the relevant corresponding matrix elements using the matrices $\mathbf{G}_*$ . Only for illustration purposes, the matrices are assumed to be non-negative and continuously vary between zero and one. White color is used for matrix elements whose values are zero and (dark) purple color is used for values close to one. . .	50

3.3	The linear composition of the models' encoding and decoding functions. The compositions are computed using the Eqs. (3.13)–(3.15), and averaged across the 50 experimental iterations. The first 744 frequency sub-bands ( $\sim 8$ kHz) are displayed for clarity. <i>Left Column</i> : Composition for the denoising auto-encoder model [87] (DAE). <i>Middle Column</i> : Composition for the four layer extension of the DAE model, adapted to music source separation (MSS-DAE) [76], [93]. <i>Right Column</i> : Composition for the skip connections for filtering the input mixture (SF) [A6], [94], [96]. Illustration is based on [A22]. . . . .	56
3.4	The outcome of the NCA for the DAE, MSS-DAE, and SF models using a $\sim 3$ seconds excerpt from the file <i>Al James - Schoolboy Fascination</i> (test subset of MUSDB18) [A22]. <i>Sub-figures (a)–(c)</i> : The couplings matrices and the corresponding spectral estimates using the <i>student</i> strategy. <i>Sub-figures: (d)–(f)</i> : The couplings matrices and the corresponding spectral estimates using the <i>compositional</i> strategy. . . . .	60
4.1	Overview of the Masker-and-Denoiser (MaD) architecture for music source separation. Circled crosses denote element-wise multiplication. Illustration reproduced from [A10]. . . . .	71
4.2	Overview of the Masker-and-Denoiser (MaD) architecture using the Twin-Net technique. The illustrated modules in magenta color are used only during the training procedure. Circled crosses denote element-wise multiplication. Illustration reproduced from [A11]. . . . .	79
4.3	Average scores obtained from the listening tests for the three training objectives: $E_{\text{KL}/L_1}^{\text{MaD}}$ , $E_{\text{MSE}/L_1}^{\text{MaD}}$ , and $E_{\text{KL}/\text{twin}}^{\text{MaD}}$ for $\omega = 0.5$ . . . . .	88
5.1	Overview of the proposed method for representation learning. . . . .	97
5.2	An illustration of the learned representations of a multi-track segment, by three encoders $E(\cdot)$ optimized using various hyper-parameters for $L_A$ and $L_B$ . . . . .	111
5.3	The frequency response of three frequently used basis functions that are learned by the proposed method using $L_B$ with $\lambda = 1.5$ and $\omega = 4.0$ . The frequency response is computed using the discrete Fourier transform, demonstrating a high-frequency comb-like filter. . . . .	114
5.4	Performance evaluation of the learned representations by $L_B$ that use the Sinkhorn distances. (top-left) Reconstruction of singing voice in SI-SDR, (top-right) oracle separation performance in SI-SDR, and (bottom) additivity objective measure. Horizontal and vertical lines denote the average and the standard deviation of the performance, respectively. . . . .	115
E.1	$L_A$ using total-variation denoising ( $\mathcal{L}_{\text{TV}}(\cdot)$ ) for various values of $\omega$ : (top-left) Reconstruction of singing voice in SI-SDR, (top-right) oracle separation performance in SI-SDR, and (bottom) additivity objective measure. . . . .	125



E.2	The output values of $\mathcal{L}_{\text{SK}}(\cdot)$ with $p = 1$ and for various values for $\lambda$ that are used to compute the loss $L_B$ . The number of gradient updates corresponds to $\sim 3$ full iterations throughout the whole training data-set. An eight-order quadratic smoothing filter with a window of 61 gradient updates is applied to the results for clarity. . . . .	126
-----	--	-----

# List of Tables

3.1	Assessing the mapping functions. The TOD-R metric (Eq. (3.26)) for each strategy and model. Lower TOD-R values indicate high off-diagonal activity and thus, the model's ability to learn inter-frequency relationships. . .	57
3.2	Assessing the approximation performance of the mappings computed by the NCA, compared to the models' outputs. The mean and standard deviation of the SNR, expressed in dB, are reported. For comparison, the linear composition and identity function are used. Bold faced values denote the best approximation performance. . . . .	57
3.3	Assessing the approximation performance of the mappings computed by the NCA, compared to the true singing voice spectra. The mean and standard deviation of the SNR, expressed in dB, are reported. For comparison, the three separation models' outputs are also reported. . . . .	61
4.1	The effect of the Denoiser module on the objective singing voice separation performance, using two training objectives. Both training objectives are computed using $\omega = 0.5$ . Values in boldface denote the best obtained performance. . . . .	84
4.2	The effect of the Denoiser module on the objective performance in accompaniment separation, computed using the objective evaluation measures and the MUSDB18 data-set. Values in boldface denote the best obtained performance. . . . .	85
4.3	The effect of the recurrent inference algorithm, used by the $\text{RNN}_{\text{dec}}(\cdot)$ Masker, in singing voice and accompaniment source separation, including the processing time required (standard deviation in parentheses). The objective evaluation measures and the average processing time are reported on the MUSDB18 data-set. Values in boldface denote the best obtained performance subject to the target source. . . . .	86
4.4	The effect of various training objectives on the performance in singing voice separation, computed using the objective evaluation measures and the MUSDB18 data-set. Values in boldface denote the best obtained performance. . . . .	87

4.5	Results from the objective evaluation for singing voice and accompaniment source separation using the SDR, SIR, and SAR objective measures. Results reported on the DSD100 data-set and in comparison to legacy approaches for mask prediction. Values in boldface denote the best obtained performance. . . . .	89
4.6	Results from the objective evaluation for singing voice and accompaniment source separation using the SDR, SIR, and SAR objective measures. Results reported on the MUSDB18 data-set and in comparison to state-of-the-art approaches. Values in boldface denote the best obtained performance. . . . .	90
4.7	Results from the objective evaluation for harmonic and percussive source separation using the SDR, SIR, and SAR objective measures. Results reported on the MUSDB18 data-set and compared to a state-of-the-art HPSS approach. Values in boldface denote the best obtained performance subject to the target source. . . . .	91
5.1	Results reflecting the decoding performance, by means of SI-SDR. Bold-faced numbers denote the best performance. . . . .	107
5.2	SI-SDR for informed separation by binary masking (BM). Bold-faced numbers denote the best performance. . . . .	107
5.3	Results from objectively evaluating the learned representations. Values in boldface denote the best obtained performance. . . . .	108
5.4	Objective evaluation of the additivity of the learned representations. . . . .	109
D.1	The effect of various training objectives on the performance in accompaniment separation, computed using the objective evaluation measures and the MUSDB18 data-set. Values in boldface denote the best obtained performance. . . . .	124

# Bibliography

## Publications as Co- or First Author

- [A1] C. Bönsel, J. Abeßer, S. Grollmisch, and **S. I. Mimitakis**, “Automatic Best Take Detection for Electric Guitar and Vocal Studio Recordings”, in *Proceedings of the 2nd Audio Engineering Society Workshop on Intelligent Music Production*, 2016.
- [A2] **S. I. Mimitakis**, E. Cano, J. Abeßer, and G. Schuller, “New Sonorities for Jazz Recordings: Separation and Mixing using Deep Neural Networks”, in *Proceedings of the 2nd Audio Engineering Society Workshop on Intelligent Music Production*, 2016.
- [A3] **S. I. Mimitakis**, K. Drossos, T. Virtanen, and G. Schuller, “Deep Neural Networks for Dynamic Range Compression in Mastering Applications”, in *Proceedings of the 140th Audio Engineering Society Convention*, AES, 2016.
- [A4] K. Drossos, **S. I. Mimitakis**, A. Floros, T. Virtanen, and G. Schuller, “Close Miking Empirical Practice Verification: A Source Separation Approach”, in *Proceedings of the 142nd Audio Engineering Society Convention*, 2017.
- [A5] **S. I. Mimitakis** and G. Schuller, *Investigating the Potential of Pseudo Quadrature Mirror Filter-Banks in Music Source Separation Tasks*, 2017. arXiv: 1706.04924 [cs.SD].
- [A6] **S. I. Mimitakis**, K. Drossos, T. Virtanen, and G. Schuller, “A Recurrent Encoder-Decoder Approach with Skip-filtering Connections for Monaural Singing Voice Separation”, in *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2017, pp. 1–6.
- [A7] J. Abeßer, **S. I. Mimitakis**, R. Gräfe, and H. Lukashevich, “Acoustic Scene Classification by Combining Autoencoder-Based Dimensionality Reduction and Convolutional Neural Networks”, in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2017 Workshop (DCASE2017)*, 2017, pp. 7–11.
- [A8] Z. Rafii, A. Liutkus, F. Stöter, **S. I. Mimitakis**, and R. Bittner, *The MUSDB18 Corpus for Music Separation*, 2017. [Online]. Available: <https://doi.org/10.5281/zenodo.1117372>.

- [A9] Z. Rafii, A. Liutkus, F. R. Stöter, **S. I. Mimitakis**, D. FitzGerald, and B. Pardo, “An Overview of Lead and Accompaniment Separation in Music”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 8, pp. 1307–1335, 2018.
- [A10] **S. I. Mimitakis**, K. Drossos, J.-F. Santos, G. Schuller, T. Virtanen, and Y. Bengio, “Monaural Singing Voice Separation with Skip-Filtering Connections and Recurrent Inference of Time-Frequency Mask”, in *Proceedings of the 43rd International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018)*, 2018.
- [A11] K. Drossos, **S. I. Mimitakis**, D. Serdyuk, G. Schuller, T. Virtanen, and Y. Bengio, “MaD TwinNet: Masker-Denoiser Architecture with Twin Networks for Monaural Sound Source Separation”, in *Proceedings of the 2018 IEEE International Joint Conference on Neural Networks (IJCNN)*, 2018.
- [A12] K. Drossos, P. Magron, **S. I. Mimitakis**, and T. Virtanen, “Harmonic-Percussive Source Separation with Deep Neural Networks and Phase Recovery”, in *Proceedings of the 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2018, pp. 421–425.
- [A13] P. Magron, K. Drossos, **S. I. Mimitakis**, and T. Virtanen, “Reducing Interference with Phase Recovery in DNN-based Monaural Singing Voice Separation”, in *Proc. Interspeech 2018*, 2018, pp. 332–336.
- [A14] **S. I. Mimitakis**, E. Cano, D. FitzGerald, K. Drossos, and G. Schuller, “Examining the Perceptual Effect of Alternative Objective Functions for Deep Learning Based Music Source Separation”, in *Proceedings of the 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 679–683.
- [A15] S. H. Hawley, B. L. Colburn, and **S. I. Mimitakis**, “Profiling Musical Audio Processing Effects with Deep Neural Networks”, *The Journal of the Acoustical Society of America*, vol. 144, no. 3, pp. 1753–1753, 2018.
- [A16] S. H. Hawley, B. Colburn, and **S. I. Mimitakis**, “SignalTrain: Profiling Audio Compressors with Deep Neural Networks”, in *Proceedings of the 147th Audio Engineering Society Convention*, AES, 2019.
- [A17] **S. I. Mimitakis**, C. Weiss, V. Arifi-Müller, J. Abeßer, and M. Müller, “Cross-version Singing Voice Detection in Opera Recordings: Challenges for Supervised Learning”, in *Machine Learning and Knowledge Discovery in Databases*, P. Cellier and K. Driessens, Eds., Springer International Publishing, 2020, pp. 429–436.
- [A18] J. Abeßer, M. Götze, T. Clauß, D. Zapf, C. Kühn, H. Lukashevich, S. Kühnlenz, and **S. I. Mimitakis**, “Urban Noise Monitoring in the Stadtlärm Project - A Field Report”, in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, 2019, pp. 1–4.
- [A19] M. Taenzer, J. Abeßer, **S. I. Mimitakis**, C. Weiss, M. Müller, and H. Lukashevich, “Investigating CNN-Based Instrument Family Recognition For Western Classical Music Recordings”, in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.

- [A20] **S. I. Mimitakis**, N. J. Bryan, and P. Smaragdis, “One-Shot Parametric Audio Production Style Transfer with Application to Frequency Equalization”, in *Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, 2020, pp. 256–260.
- [A21] **S. I. Mimitakis**, K. Drossos, and G. Schuller, “Unsupervised Interpretable Representation Learning for Singing Voice Separation”, in *Proceedings of the 27th European Signal Processing Conference (EUSIPCO 2020)*, 2020.
- [A22] **S. I. Mimitakis**, K. Drossos, E. Cano, and G. Schuller, “Examining the Mapping Functions of Denoising Autoencoders in Singing Voice Separation”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 266–278, 2020.
- [A23] K. Drossos, **S. I. Mimitakis**, S. Gharib, Y. Li, and T. Virtanen, “Sound Event Detection with Depthwise Separable and Dilated Convolutions”, in *Proceedings of the 2020 IEEE International Joint Conference on Neural Networks (IJCNN)*, 2020.
- [A24] **S. I. Mimitakis**, K. Drossos, and G. Schuller, *Revisiting Representation Learning for Singing Voice Separation with Sinkhorn Distances*, 2020. arXiv: 2007.02780 [cs.SD].
- [A25] P. Pyykkönen, **S. I. Mimitakis**, K. Drossos, and T. Virtanen, “Depthwise Separable Convolutions Versus Recurrent Neural Networks for Monaural Singing Voice Separation”, in *Proceedings of the 22nd IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2020.
- [A26] K. Drossos, **S. I. Mimitakis**, and T. Virtanen, *Conditioned Time-Dilated Convolutions for Sound Event Detection*, 2020. arXiv: 2007.05183 [cs.SD].

## References by Other Authors

- [1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [2] T. M. Mitchell, *Machine Learning*, 1st ed. McGraw-Hill, Inc., 1997.
- [3] E. Cano, D. FitzGerald, A. Liutkus, M. D. Plumbley, and F. Stöter, “Musical Source Separation: An Introduction”, *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 31–40, 2019.
- [4] G. R. Naik and W. Wang, *Blind Source Separation*. Springer Verlag Berlin Heidelberg, 2014.
- [5] C. Duxbury, M. Davies, and M. Sandler, “Separation of Transient Information in Musical Audio Using Multiresolution Analysis Techniques”, in *Proceedings of the 4th International Conference on Digital Audio Effects (DAFx 2001)*, 2001.
- [6] R. Füg, A. Niedermeier, J. Driedger, S. Disch, and M. Müller, “Harmonic-Percussive-Residual Sound Separation Using the Structure Tensor on Spectrograms”, in *Proceedings of the 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2016)*, 2016, pp. 445–449.

- [7] X. Serra, “A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition”, PhD thesis, Stanford University, 1989.
- [8] A. Liutkus, D. Fitzgerald, Z. Rafii, B. Pardo, and L. Daudet, “Kernel Additive Models for Source Separation”, *IEEE Transactions on Signal Processing*, vol. 62, no. 16, pp. 4298–4310, 2014.
- [9] E. Vincent, N. Bertin, R. Gribonval, and F. Bimbot, “From Blind to Guided Audio Source Separation: How Models and Side Information can Improve the Separation of Sound”, *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 107–115, 2014.
- [10] A. Liutkus, J. Pinel, R. Badeau, L. Girin, and G. Richard, “Informed Source Separation Through Spectrogram Coding and Data Embedding”, *Signal Processing*, vol. 92, 2012.
- [11] E. Cano, C. Dittmar, and G. Schuller, “Re-Thinking Sound Separation: Prior Information and Additivity Constraints in Separation Algorithms”, in *16th International Conference on Digital Audio Effects*, 2013.
- [12] G. Ballou, *Electroacoustic Devices*. Focal Press, 2009.
- [13] B. C. J. Moore, *An Introduction to the Psychology of Hearing*, Sixth. Emerald Group Publishing, 2012.
- [14] Y. C. Eldar, *Sampling Theory: Beyond Bandlimited Systems*, 1st. Cambridge University Press, 2015.
- [15] S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*, 2nd. Academic Press, Inc., 2015.
- [16] M. Vetterli, J. Kovačević, and V. K. Goyal, *Foundations of Signal Processing*, 3rd. Cambridge University Press, 2014.
- [17] O. Yilmaz and S. Rickard, “Blind Separation of Speech Mixtures via Time-Frequency Masking”, *IEEE Transactions on Signal Processing*, vol. 52, no. 7, pp. 1830–1847, 2004.
- [18] J. Smith, *Spectral Audio Signal Processing*. <http://ccrma.stanford.edu/~jos/sas>, 2011, Online book, Accessed: June 2020.
- [19] G. Heinzel, A. Rüdiger, and R. Schilling, *Spectrum and Spectral Density Estimation by the Discrete Fourier Transform (DFT), Including a Comprehensive List of Window Functions and Some New At-Top Windows*, Online technical report, Accessed: December 2020, 2002.
- [20] M. D. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. E. Davies, “Sparse Representations in Audio and Music: From Coding to Source Separation”, *Proceedings of the IEEE*, vol. 98, no. 6, pp. 995–1005, 2010.
- [21] Y. LeCun, S. Chopra, R. Hadsell, M. A. Ranzato, and F. Jie, “Energy-Based Models”, in *Predicting Structure Data*, MIT Press, 2006.
- [22] S. S. Chen, D. L. Donoho, and M. A. Saunders, “Atomic Decomposition by Basis Pursuit”, *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.

- [23] I. Daubechies, M. Defrise, and C. De Mol, “An Iterative Thresholding Algorithm for Linear Inverse Problems with a Sparsity Constraint”, *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, pp. 1413–1457, 2004.
- [24] S. G. Mallat and Zhifeng Zhang, “Matching Pursuits With Time-Frequency Dictionaries”, *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.
- [25] B. L. Sturm, J. J. Shynk, L. Daudet, and C. Roads, “Dark Energy in Sparse Atomic Estimations”, *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 3, pp. 671–676, 2008.
- [26] L. Le Magoarou and R. Gribonval, “Chasing Butterflies: In Search of Efficient Dictionaries”, in *Proceedings of the 40th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2015)*, 2015, pp. 3287–3291.
- [27] V. Pappas, Y. Romano, and M. Elad, “Convolutional Neural Networks Analyzed via Convolutional Sparse Coding”, *Journal of Machine Learning Research*, vol. 18, no. 83, pp. 1–52, 2017.
- [28] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online Dictionary Learning for Sparse Coding”, in *Proceedings of the 26th International Conference on Machine Learning (ICML)*, Association for Computing Machinery, 2009, pp. 689–696.
- [29] D. FitzGerald and R. Jaiswal, “On the Use of Masking Filters in Sound Source Separation”, in *15th International Conference on Digital Audio Effects (DAFx-12)*, 2012.
- [30] D. Giannoulis, D. Barchiesi, A. Klapuri, and M. D. Plumbley, “On the Disjointness of Sources in Music Using Different Time-Frequency Representations”, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2011)*, 2011, pp. 261–264.
- [31] G. D. T. Schuller, Bin Yu, Dawei Huang, and B. Edler, “Perceptual Audio Coding Using Adaptive Pre- and Post-Filters and Lossless Compression”, *IEEE Transactions on Speech and Audio Processing*, vol. 10, no. 6, pp. 379–390, 2002.
- [32] G. D. T. Schuller and M. J. T. Smith, “New Framework for Modulated Perfect Reconstruction Filter Banks”, *IEEE Transactions on Signal Processing*, vol. 44, no. 8, pp. 1941–1954, 1996.
- [33] P. Smaragdis, B. Raj, and M. Shashanka, “A Probabilistic Latent Variable Model for Acoustic Modeling”, in *Proceedings of the 21st International Conference on Neural Information Processing Systems: Workshop on Advances in Models for Acoustic Processing*, 2006.
- [34] D. FitzGerald, “Harmonic/Percussive Separation Using Median Filtering”, in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, 2010, pp. 246–253.
- [35] Z. Rafii and B. Pardo, “REpeating Pattern Extraction Technique (REPET): A Simple Method for Music/Voice Separation”, *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 21, no. 1, pp. 73–84, 2013.



- [36] M. Hamidi and J. Pearl, “Comparison of the Cosine and Fourier Transforms of Markov-1 Signals”, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 5, pp. 428–429, 1976.
- [37] V. Sanchez, P. Garcia, A. M. Peinado, J. C. Segura, and A. J. Rubio, “Diagonalizing Properties of the Discrete Cosine Transforms”, *IEEE Transactions on Signal Processing*, vol. 43, no. 11, pp. 2631–2641, 1995.
- [38] F. Rosenblatt, “The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain”, *Psychological Review*, pp. 65–386, 1958.
- [39] —, *Principles of Neurodynamics*. Spartan Books, 1963.
- [40] A. Ivakhnenko and V. Lapa, *Cybernetic Predicting Devices*, ser. Jprs report. CCM Information Corporation, 1973. [Online]. Available: <https://books.google.de/books?id=FhwVNQAACAAJ>.
- [41] A. G. Ivakhnenko, “The Group Method of Data Handling-A rival of the Method of Stochastic Approximation”, *Soviet Automatic Control*, no. 3, pp. 43–55, 1968.
- [42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Internal Representations by Error Propagation”, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. MIT Press, 1986, pp. 318–362.
- [43] K. Fukushima, “Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position”, *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.
- [44] K. Cho, B. van Merriënboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”, *CoRR*, vol. abs/1406.1078, 2014. arXiv: 1406.1078.
- [45] K. Hornik, M. Stinchcombe, and H. White, “Multilayer Feedforward Networks Are Universal Approximators”, *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.
- [46] V. Nair and G. E. Hinton, “Rectified Linear Units Improve Restricted Boltzmann Machines”, in *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML)*, Omnipress, 2010, pp. 807–814.
- [47] L. Wonyeol, Y. Hangeol, R. Xavier, and Y. Hongseok, “On Correctness of Automatic Differentiation for Non-Differentiable Functions”, in *Proceedings of the 34th Neural Information Processing Systems (NeurIPS)*, 2020.
- [48] S. Linnainmaa, “Taylor Expansion of the Accumulated Rounding Error”, *BIT Numerical Mathematics*, vol. 16, no. 2, pp. 146–160, 1976.
- [49] A. Paszke, S. Gross, F. Massa, *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library”, in *Proceedings of the 32nd International Conference Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 8024–8035.
- [50] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization”, in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

- [51] S. J. Reddi, S. Kale, and S. Kumar, “On the Convergence of Adam and Beyond”, in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.
- [52] A. Graves, “Supervised Sequence Labelling with Recurrent Neural Networks”, PhD thesis, Technical University Munich, 2008.
- [53] I. Sutskever, “Training Recurrent Neural Networks”, PhD thesis, University of Toronto, 2013.
- [54] P. J. Werbos, “Backpropagation Through Time: What it Does and How to Do it”, *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
- [55] Y. Bengio, P. Frasconi, and P. Simard, “The Problem of Learning Long-term Dependencies in Recurrent networks”, in *Proceedings of the IEEE International Conference on Neural Networks*, 1993, 1183–1188 vol.3.
- [56] R. Pascanu, T. Mikolov, and Y. Bengio, “On the Difficulty of Training Recurrent Neural Networks”, in *Proceedings of the 30th International Conference on International Conference on Machine Learning (ICML)*, 2013, pp. 1310–1318.
- [57] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory”, *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [58] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”, in *Proceedings of the 27th Neural Information Processing Systems (NeurIPS): Workshop on Deep Learning*, 2014.
- [59] M. Schuster and K. K. Paliwal, “Bidirectional Recurrent Neural Networks”, *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.
- [60] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation Applied to Handwritten Zip Code Recognition”, *Neural Computation*, vol. 1, pp. 541–551, 1989.
- [61] Y. Fisher and K. Vladlen, “Multi-Scale Context Aggregation by Dilated Convolutions”, in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.
- [62] A. M. Saxe, J. L. McClelland, and S. Ganguli, “Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks”, *CoRR*, vol. abs/1312.6120, 2013. arXiv: 1312.6120.
- [63] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”, in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV ’15)*, 2015, pp. 1026–1034.
- [64] X. Glorot and Y. Bengio, “Understanding the Difficulty of Training Deep Feed-forward Neural Networks”, in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS’10)*, 2010, pp. 249–256.
- [65] R. Izhaki, *Mixing Audio: Concepts, Practices and Tools*, ser. Electronics & Electrical. Focal Press, 2008.

- [66] J. Chua, G. Wang, and W. B. Kleijn, “Convolutional Blind Source Separation With Low Latency”, in *Proceedings of 14th International Workshop on Acoustic Signal Enhancement (IWAENC)*, 2016, pp. 1–5.
- [67] E. Vincent, H. Sawada, P. Bofill, S. Makino, and J. P. Rosca, “First Stereo Audio Source Separation Evaluation Campaign: Data, Algorithms and Results”, in *Proceedings of the 7th International Conference on Independent Component Analysis and Signal Separation (ICA 2007)*, Springer Berlin Heidelberg, 2007, pp. 552–559.
- [68] U. Zölzer, *DAFX: Digital Audio Effects, Second Edition*. John Wiley & Sons, Ltd, 2011.
- [69] B. Katz and R. A. Katz, *Mastering Audio: The Art and the Science*. Butterworth-Heinemann, 2003.
- [70] A. Liutkus and R. Badeau, “Generalized Wiener Filtering with Fractional Power Spectrograms”, in *Proceedings of the 40th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2015)*, 2015, pp. 266–270.
- [71] S. Voran, “Exploration of the Additivity Approximation for Spectral Magnitudes”, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2015)*, 2015, pp. 1–5.
- [72] H. Erdogan, J. R. Hershey, S. Watanabe, and J. L. Roux, “Phase-Sensitive and Recognition-Boosted Speech Separation Using Deep Recurrent Neural Networks”, in *40th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2015)*, 2015, pp. 708–712.
- [73] C. Févotte, N. Bertin, and J.-L. Durrieu, “Non-negative Matrix Factorization with the Itakura-Saito Divergence: With Application to Music Analysis”, *Neural Computation*, vol. 21, no. 3, pp. 793–830, 2009.
- [74] S. Voran, “The Selection of Spectral Magnitude Exponents for Separating Two Sources is Dominated by Phase Distribution Not Magnitude Distribution”, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2017)*, 2017, pp. 279–283.
- [75] N. Q. K. Duong, E. Vincent, and R. Gribonval, “Under-Determined Reverberant Audio Source Separation Using a Full-Rank Spatial Covariance Model”, *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1830–1840, 2010.
- [76] A. A. Nugraha, A. Liutkus, and E. Vincent, “Multichannel Music Separation with Deep Neural Networks”, in *Proceedings of the 24th European Signal Processing Conference (EUSIPCO 2016)*, 2016, pp. 1748–1752.
- [77] K. Brandenburg and T. Sporer, “NMR and Masking Flag: Evaluation of Quality Using Perceptual Criteria”, in *Audio Engineering Society Conference: 11th International Conference: Test & Measurement*, 1992.
- [78] V. Emiya, E. Vincent, N. Harlander, and V. Hohmann, “Subjective and Objective Quality Assessment of Audio Source Separation”, *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2046–2057, 2011.

- [79] E. Cano, D. FitzGerald, and K. Brandenburg, “Evaluation of Quality of Sound Source Separation Algorithms: Human Perception vs Quantitative Metrics”, in *Proceedings of the 24th European Signal Processing Conference (EUSIPCO 2016)*, 2016, pp. 1758–1762.
- [80] E. Vincent, R. Gribonval, and C. Févotte, “Performance Measurement in Blind audio Source Separation”, *IEEE Transactions on Audio, Speech and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.
- [81] F.-R. Stöter, A. Liutkus, and N. Ito, “The 2018 Signal Separation Evaluation Campaign”, in *Proceedings of the Latent Variable Analysis and Signal Separation: 14th International Conference on Latent Variable Analysis and Signal Separation*, 2018, pp. 293–305.
- [82] J. L. Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, “SDR – Half-baked or Well Done?”, in *Proceedings of the 44th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*, 2019, pp. 626–630.
- [83] G. Roma, E. M. Grais, A. J. R. Simpson, and M. D. Plumbley, “Music Remixing and Upmixing using Source Separation”, in *Proceedings of the 2nd Audio Engineering Society Workshop on Intelligent Music Production*, 2016.
- [84] ITU, *Recommendation ITU-R BS.1534-3: Method for the Subjective Assessment of Intermediate quality Level of Audio Systems*, 2014.
- [85] M. Schoeffler, S. Bartoschek, F.-R. Stöter, M. Roess, S. Westphal, B. Edler, and J. Herre, “webMUSHRA – A Comprehensive Framework for Web-based Listening Tests”, *Journal of Open Research Software*, vol. 6, no. 1, 2018.
- [86] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and Composing Robust Features with Denoising Autoencoders”, in *Proceedings of the 25th International Conference on Machine Learning (ICML)*, ACM, 2008, pp. 1096–1103.
- [87] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion”, *Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.
- [88] A. Liutkus, F.-R. Stöter, Z. Rafii, D. Kitamura, B. Rivet, N. Ito, N. Ono, and J. Fontecave, “The 2016 Signal Separation Evaluation Campaign”, in *Proceedings of the 13th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA 2017)*, 2017, pp. 323–332.
- [89] Y. Luo, Z. Chen, J. R. Hershey, J. Le Roux, and N. Mesgarani, “Deep Clustering and Conventional Networks for Music Separation: Stronger Together”, in *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*, 2017, pp. 61–65.
- [90] E. M. Grais, G. Roma, A. J. R. Simpson, and M. D. Plumbley, “Two-Stage Single-Channel Audio Source Separation Using Deep Neural Networks”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 9, pp. 1773–1783, 2017.

- [91] D. Williamson, Y. Wang, and D. Wang, “Complex Ratio Masking for Monaural Speech Separation”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 3, pp. 483–492, 2016.
- [92] Y. Wang, A. Narayanan, and D. Wang, “On Training Targets for Supervised Speech Separation”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 12, pp. 1849–1858, 2014.
- [93] S. Uhlich, F. Giron, and Y. Mitsufuji, “Deep Neural Network Based Instrument Extraction from Music”, in *Proceedings of the 40th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2015)*, 2015, pp. 2135–2139.
- [94] F. Weninger, J. R. Hershey, J. L. Roux, and B. Schuller, “Discriminatively Trained Recurrent Neural Networks for Single-Channel Speech Separation”, in *Proceedings of the 2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2014, pp. 577–581.
- [95] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, “Joint Optimization of Masks and Deep Recurrent Neural Networks for Monaural Source Separation”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 12, pp. 2136–2147, 2015.
- [96] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, “Singing Voice Separation with Deep U-Net Convolutional Networks”, in *Proceedings of the 18th International Society for Music Information Retrieval Conference*, 2017.
- [97] J. Lee, J. Skoglund, T. Shabestary, and H. Kang, “Phase-Sensitive Joint Learning Algorithms for Deep Learning-Based Speech Enhancement”, *IEEE Signal Processing Letters*, vol. 25, no. 8, pp. 1276–1280, 2018.
- [98] S. Uhlich, M. Porcu, F. Giron, M. Elenkl, T. Kemp, N. Takahashi, and Y. Mitsufuji, “Improving Music Source Separation Based On Deep Neural Networks Through Data Augmentation and Network Blending”, in *Proceedings of the 42nd International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*, 2017, pp. 261–265.
- [99] N. Takahashi and Y. Mitsufuji, “Multi-Scale Multi-Band DenseNets for Audio Source Separation”, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2017)*, 2017.
- [100] R. Hennequin, A. Khelif, F. Voituret, and M. Moussallam, *Spleeter: A Fast And State-of-the Art Music Source Separation Tool With Pre-trained Models*, Late-Breaking/Demo ISMIR 2019, Deezer Research, 2019.
- [101] D.-J. Im, M. I. Belghazi, and R. Memisevic, “Conservativeness of Untied Auto-Encoders”, in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016.
- [102] J. Särelä and H. Valpola, “Denoising Source Separation”, *Journal of Machine Learning Research*, vol. 6, pp. 233–272, 2005.

- [103] Y. Bengio, L. Yao, G. Alain, and P. Vincent, “Generalized Denoising Auto-Encoders as Generative Models”, in *Proceedings of the 26th International Conference Advances in Neural Information Processing Systems (NeurIPS)*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., 2013, pp. 899–907.
- [104] P. Vincent, “A Connection Between Score Matching and Denoising Autoencoders”, *Neural Computation*, vol. 23, no. 7, pp. 1661–1674, 2011.
- [105] W. K. Pratt, “Generalized Wiener Filtering Computation Techniques”, *IEEE Transactions on Computers*, vol. C-21, no. 7, pp. 636–641, 1972.
- [106] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Training Very Deep Networks”, in *Proceedings of the 28th International Conference Advances in Neural Information Processing Systems (NeurIPS)*, 2015, pp. 2377–2385.
- [107] G. Montavon, W. Samek, and K.-R. Müller, “Methods for Interpreting and Understanding Deep Neural Networks”, *Digital Signal Processing*, vol. 73, pp. 1–15, 2018.
- [108] M. T. Ribeiro, S. Singh, and C. Guestrin, “"Why Should I Trust You? ": Explaining the Predictions of Any Classifier”, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1135–1144.
- [109] S. Mishra, E. Benetos, B. L. T. Sturm, and S. Dixon, “Reliable Local Explanations for Machine Listening”, in *Proceedings of the 2020 IEEE International Joint Conference on Neural Networks (IJCNN)*, 2020, pp. 1–8.
- [110] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network”, in *Proceedings of the 28th International Conference on Neural Information Processing Systems (NeurIPS), Workshop on Deep Learning and Representation Learning*, 2015.
- [111] R. Flamary, C. F  votte, N. Courty, and V. Emiya, “Optimal Spectral Transportation with Application to Music Transcription”, in *Proceedings of the 29th International Conference on Neural Information Processing Systems (NeurIPS)*, 2016.
- [112] S. Kolouri, S. R. Park, M. Thorpe, D. Slepcev, and G. K. Rohde, “Optimal Mass Transport: Signal Processing and Machine-Learning Applications”, *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 43–59, 2017.
- [113] S. Wang, A.-R. Mohamed, R. Caruana, J. Bilmes, M. Plilipose, M. Richardson, K. Geras, G. Urban, and O. Aslan, “Analysis of Deep Neural Networks with the Extended Data Jacobian Matrix”, in *Proceedings of the 33rd International Conference on Machine Learning*, vol. 48, 2016, pp. 718–727.
- [114] H. Zhou, J. Lan, R. Liu, and J. Yosinski, “Deconstructing Lottery Tickets: Zeros, Signs, and the Supermask”, in *Proceedings of the 32nd International Conference on Advances in Neural Information Processing Systems (NeurIPS)*, 2019, pp. 3597–3607.

- [115] R. Pascanu, G. Montufar, and Y. Bengio, “On the Number of Response Regions of Deep Feedforward Networks with Piecewise Linear Activations”, in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.
- [116] P. Baldi and K. Hornik, “Neural Networks and Principal Component Analysis: Learning from Examples without Local Minima”, *Neural Netw.*, vol. 2, no. 1, pp. 53–58, 1989.
- [117] G. Hinton and R. Salakhutdinov, “Reducing the Dimensionality of Data with Neural Networks”, *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
- [118] J. Santos and T. Falk, “Investigating the Effect of Residual and Highway Connections in Speech Enhancement Models”, in *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS), Workshop on Interpretability and Robustness in Audio, Speech, and Language*, 2018.
- [119] S. Dieleman and B. Schrauwen, “End-to-End Learning for Music Audio”, in *Proceedings of the 39th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)*, 2014, pp. 6964–6968.
- [120] Y. Luo and N. Mesgarani, “Conv-TasNet: Surpassing Ideal Time–Frequency Magnitude Masking for Speech Separation”, *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, pp. 1256–1266, 2019.
- [121] M. Kim and P. Smaragdis, “Adaptive Denoising Autoencoders: A Fine-Tuning Scheme to Learn from Test Mixtures”, in *Proceedings of the 12th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA 2015)*, 2015.
- [122] P. Chandna, M. Miron, J. Janer, and E. Gómez, “Monoaural Audio Source Separation Using Deep Convolutional Neural Networks”, in *Proceedings of the 13th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA 2017)*, 2017, pp. 258–266.
- [123] A. J. R. Simpson, G. Roma, and M. D. Plumbley, “Deep Karaoke: Extracting Vocals from Musical Mixtures Using a Convolutional Deep Neural Network”, in *Proceedings of the 12th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA 2015)*, 2015.
- [124] E. M. Grais, G. Roma, A. J. R. Simpson, and M. D. Plumbley, “Single Channel Audio Source Separation using Deep Neural Network Ensembles”, in *Proceedings of the 140th Audio Engineering Society Convention*, 2016.
- [125] —, “Discriminative Enhancement for Single Channel Audio Source Separation using Deep Neural Networks”, in *Proceedings of the 13th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA 2017)*, 2017.
- [126] F. R. Bach and M. I. Jordan, “Learning Spectral Clustering”, in *Proceedings of the 16th International Conference on Advances in Neural Information Processing Systems (NeurIPS)*, MIT Press, 2004, pp. 305–312.
- [127] D. Ward, R. D. Mason, C. Kim, F. R. Stöter, A. Liutkus, and M. D. Plumbley, “SiSEC 2018: State of the Art in Musical Audio Source Separation - Subjective Selection of the Best Algorithm”, in *Proceedings of the 4th Audio Engineering Society Workshop on Intelligent Music Production (WIMP)*, 2018.

- [128] M. Michelashvili, S. Benaïm, and L. Wolf, “Semi-Supervised Monaural Singing Voice Separation with a Masking Network Trained on Synthetic Mixtures”, in *Proceedings of the 44th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*, 2019, pp. 291–295.
- [129] L. Pr  tet, R. Hennequin, J. Royo-Letelier, and A. Vaglio, “Singing Voice Separation: A Study on Training Data”, in *Proceedings of the 44th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2019)*, 2019, pp. 506–510.
- [130] F.-R. St  ter, S. Uhlich, A. Liutkus, and Y. Mitsufuji, “Open-Unmix - A Reference Implementation for Music Source Separation”, *Journal of Open Source Software*, 2019. [Online]. Available: <https://doi.org/10.21105/joss.01667>.
- [131] Y. Wu, M. Schuster, Z. Chen, *et al.*, “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”, *CoRR*, vol. abs/1609.08144, 2016.
- [132] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, “Deep Networks with Stochastic Depth”, *CoRR*, vol. abs/1603.09382, 2016.
- [133] D. Serdyuk, N. R. Ke, A. Sordoni, A. Trischler, C. Pal, and Y. Bengio, “Twin Networks: Matching the Future for Sequence Generation”, in *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.
- [134] C. Lordelo, E. Benetos, S. Dixon, and S. Ahlback, “Investigating Kernel Shapes and Skip Connections for Deep Learning-Based Harmonic-Percussive Separation”, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2019)*, 2019, pp. 40–44.
- [135] E. Cano, J. Liebetrau, D. Fitzgerald, and K. Brandenburg, “The Dimensions of Perceptual Quality of Sound Source Separation”, in *Proceedings of the 43rd International Conference on Acoustics, Speech and Signal Processing (ICASSP 2018)*, 2018, pp. 601–605.
- [136] A. D  fossez, N. Usunier, L. Bottou, and F. Bach, “Music Source Separation in the Waveform Domain”, HAL, Tech. Rep. 02379796v1, 2019.
- [137] D. Samuel, A. Ganeshan, and J. Naradowsky, “Meta-Learning Extractors for Music Source Separation”, in *Proceedings of the 45th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, 2020.
- [138] Y. Bengio, A. Courville, and P. Vincent, “Representation Learning: A Review and New Perspectives”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [139] J. Engel, L. H. Hantrakul, C. Gu, and A. Roberts, “DDSP: Differentiable Digital Signal Processing”, in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.
- [140] M. Ravanelli and Y. Bengio, “Interpretable Convolutional Filters with SincNet”, in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS): Workshop on Interpretability and Robustness for Audio, Speech and Language*, 2018.



- [141] M. Pariente, S. Cornell, A. Deleforge, and E. Vincent, “Filterbank Design for End-to-end Speech Separation”, in *Proceedings of the 45th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, 2020.
- [142] S. Venkataramani, J. Casebeer, and P. Smaragdis, “End-to-End Source Separation With Adaptive Front-Ends”, in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 684–688.
- [143] E. Tzinis and S. Venkataramani and Z. Wang and C. Subakan and P. Smaragdis, “Two-Step Sound Source Separation: Training on Learned Latent Targets”, in *Proceedings of the 45th International Conference on Acoustics, Speech and Signal Processing (ICASSP 2020)*, 2020.
- [144] Z. C. Lipton, “The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability is Both Important and Slippery”, *Queue*, vol. 16, no. 3, pp. 31–57, 2018.
- [145] P. Smaragdis and S. Venkataramani, “A Neural Network Alternative to Non-Negative Audio Models”, in *Proceedings of the 42nd IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017)*, 2017, pp. 86–90.
- [146] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear Total Variation Based Noise Removal Algorithms”, in *Proceedings of the Eleventh Annual International Conference of the Center for Nonlinear Studies on Experimental Mathematics: Computational Issues in Nonlinear Science: Computational Issues in Nonlinear Science*, 1992, pp. 259–268.
- [147] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein Generative Adversarial Networks”, in *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, 2017, pp. 214–223.
- [148] M. Cuturi, “Sinkhorn Distances: Lightspeed Computation of Optimal Transport”, in *Proceedings of the 26th International Conference Advances in Neural Information Processing Systems (NeurIPS)*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2013, pp. 2292–2300.
- [149] R. Sinkhorn, “Diagonal Equivalence to Matrices with Prescribed Row and Column Sums”, *The American Mathematical Monthly*, vol. 74, no. 4, pp. 402–405, 1967.
- [150] I. Kavalierov, S. Wisdom, H. Erdogan, B. Patton, K. Wilson, J. Le Roux, and J. R. Hershey, “Universal Sound Separation”, in *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2019)*, 2019, pp. 175–179.
- [151] P. Huang, S. D. Chen, P. Smaragdis, and M. Hasegawa-Johnson, “Singing-Voice Separation from Monaural Recordings Using Robust Principal Component Analysis”, in *Proceedings of the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2012)*, 2012, pp. 57–60.
- [152] M. Michelashvili and L. Wolf, *Speech Denoising by Accumulating Per-Frequency Modeling Fluctuations*, 2019. arXiv: 1904.07612 [cs.SD].

- [153] Z. Zhang, Y. Wang, C. Gan, J. Wu, J. B. Tenenbaum, A. Torralba, and W. T. Freeman, “Deep Audio Priors Emerge From Harmonic Convolutional Networks”, in *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

# Index

- Adam algorithm, 122
- back-propagation, 24
- back-propagation through time, 26
- bi-directional recurrent neural networks, 28
- binary mask, 33
- compositional strategy, 50
- compositional strategy derivatives, 51
- convolutional neural networks, 28
- convolutive mixing model, 31
- Denoiser, 76
- Denoiser's effect in separation performance, 84
- denoising autoencoder, 42
- denoising source separation, 45
- discrete Fourier transform, 18
- empirical loss minimization, 17
- energy-based learning, 16
- entropy function, 99
- feed-forward neural networks, 22
- fractional power spectrograms, 35
- gated recurrent unit(s), 27
- generalized Wiener filtering, 34
- gradients of the student strategy, 49
- hyperbolic tangent function, 23
- ideal amplitude mask, 34
- instantaneous (linear) mixing model, 30
- interpretability of the representation, 112
- inverse discrete Fourier transform, 19
- Kullback-Leibler divergence between matrices, 78
- linear composition, 48
- listening tests, 37
- logistic sigmoid function, 23
- MaD TwinNet, 79
- magnitude, 19
- Masker, 72
- Masker-and-Denoiser training objective, 77
- mean squared error, 45
- mean squared error between matrices, 78
- mixing function, 30
- network distillation, 48
- neural couplings algorithm, 47
- neural couplings illustrations, 60
- neural network depth, 23
- overlap-add, 14
- phase, 19
- probabilistic graphical models, 43
- re-parameterization with modulated cosines, 101
- rectified linear unit (algebraic expression), 48
- rectified linear unit (function), 23
- recurrent inference, 75
- recurrent inference performance, 85
- recurrent neural networks, 25
- relaxed mixing model, 32
- separation by masking, 33
- short-time Fourier transform, 19
- signal-to-artifacts ratio (SAR), 37
- signal-to-distortion ratio (SDR), 37
- signal-to-interference ratio (SIR), 37
- signal-to-noise ratio, 37
- Sinkhorn distances, 99
- Sinkhorn-Knopp algorithm, 100
- skip-filtering connections, 46
- soft mask, 34
- supervised music source separation, 42
- time-frame segmentation, 14
- total-variation denoising, 98
- trace-to-off-diagonal-ratio (TOD-R), 54