UNIVERSITAT POMPEU FABRA

MASTER THESIS

Singing Vocal Enhancement for Cochlear Implant Users Based on Deep Learning Models

Author:
Tom GAJĘCKI SOMERVAIL

Supervisors:
Prof. Dr. Waldo NOGUEIRA
Dr. Jordi JANER

A thesis submitted in fulfillment of the requirements for the degree of Master in Sound and Music Computing

in the

Music Technology Group
Department of Information and Communication Technologies

2018

Declaration of Authorship

I, Tom GAJĘCKI SOMERVAIL, declare that this thesis titled, "Singing Vocal Enhancement for Cochlear Implant Users Based on Deep Learning Models " and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"If I were not a physicist, I would probably be a musician. I often think in music. I live my daydreams in music. I see my life in terms of music."

Albert Einstein

Universitat Pompeu Fabra

Abstract

Faculty of Communication

Department of Information and Communication Technologies

Master in Sound and Music Computing

Singing Vocal Enhancement for Cochlear Implant Users Based on Deep Learning Models

by Tom GAJĘCKI SOMERVAIL

Severe hearing loss problems that some people suffer from can be treated by providing them with a surgically implanted electrical device called cochlear implant (CI). These devices perform well in the context of speech intelligibility but still struggle when it comes to representing more complex audio signals such as music. However, previous studies show that CI recipients find music more enjoyable when enhancing the vocals with respect to the background music. In this thesis source separation (SS) algorithms are used to remix music multi-tracks by applying gain to the lead singing vocal. This work proposes deep convolutional auto-encoders (DCAEs), a deep recurrent neural network (DRNN), a multilayer perceptron (MLP) and non-negative matrix factorization (NMF) to be evaluated objectively and subjectively through two different perceptual experiments involving normal hearing (NH) subjects and CI recipients. The evaluation assesses the relevance of the artifacts introduced by the SS algorithms considering their degree of complexity, as this study will try to propose one of the algorithms for real-time implementation. Moreover, this work presents a benchmark which relates the measured distortions as a function of the observed preference ratings on CI subjects. Objective results based on the source to distortion ratio (SDR) and source to artifacts ratio (SAR) show that the DCAEs outperform only when presented with data similar to the one used for training, on the other hand, the MLP performs in a consistent way throughout the tested data obtaining similar performance as the DRNN while reducing algorithmic complexity. Using the benchmark, next to a MUSHRA test we propose an MLP for real-time audio SS.

***Keywords*:** cochlear implant, deep learning, neural networks, source separation.

Acknowledgements

I would first like to thank my thesis advisor Dr. Prof. W. Nogueira of the Department of Otolaryngology and Hearing4all at the Medical University of Hannover. I have been extremely lucky to have a supervisor who cared so much about my work, and who responded to my questions and queries so promptly. The door to Prof. Nogueira's office was always open whenever I ran into a trouble spot or had a question about my research or writing. He allowed this thesis to be my own work but steered me in the right the direction whenever he thought I needed it.

I would also like to thank Dr. Janer as my second supervisor. Without him, this project would never have been achieved successfully.

I would also like to acknowledge Dr. X. Sierra, Dr. J. Bonada of the department of communications at the Universitat Pompeu Fabra as the second readers of this thesis, and I am gratefully indebted to their for their very valuable comments on this thesis.

Special thanks to all the cochlear implant users from the German Hearing Center of the Medical University of Hannover that volunteered to participate in the extensive experiments.

Also thanks to all normal hearing subjects for participating in the experiments.

I would also like to thank the experts who were involved in the validation survey for this research project: Jordi Pons, Marius Miron and Pritish Chandna. Without their passionate participation and input, the validation survey could not have been successfully conducted.

I thank my fellow lab-mates Mayra Windeler, Florian Langner, Giulio Cosatti and Maria Egger in for the stimulating discussions, and for all the fun we have had during our time together. In particular, I am grateful to Mayra Windeler; thank you for making the perceptual experiments possible, for being the best office partner, for lending me the piano and most importantly, for being a great friend.

I would like to thank Dr. W. Buyens for sharing the audio excerpts used for this study. Thanks to iKala for giving us the permissions to use their dataset for this study. The TESLA k40c used in this study was provided by NVIDIA.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them.

Gracias.

This work was supported by the DFG Cluster of Excellence EXC 1077/1 "Hearing4all".

Contents

List of Figures

List of Tables

List of Abbreviations

CI	**C**ochlear **I**mplant
SS	**S**ource **S**eparation
NH	**N**ormal **H**earing
BM	**B**asilar **M**embare
HC	**H**air **C**ell
MT	**M**ulti **T**rack
NN	**N**eural **N**etwork
GD	**G**radient **D**escent
SIR	**S**ource to **I**nterferences **R**atio
BSS	**B**lind **S**ource **S**eparation
SDR	**S**ource to **D**istortion **R**atio
SAR	**S**ource to **A**rtifacts **R**atio
MIR	**M**usic **I**nformation **R**etrieval
ASR	**A**utomatic **S**peech **R**ecognition
NMF	**N**on-negative **M**atrix **F**actorization
DNN	**D**eep **N**eural **N**etwork
STFT	**S**hort **T**ime **F**ourier **T**ransform
FFNN	**F**eed **F**orward **N**eural **N**etwork
DCAE	**D**eep **C**onvolutional **A**uto **N**encoder
DRNN	**D**eep **R**ecurrent **N**eural **N**etwork
DCNN	**D**eep **C**onvolutional **N**eural **N**etwork

Chapter 1

Introduction

1.1 Motivation

A cochlear implant (CI) is a surgically implanted electronic device which provides a sense of hearing to people with a severe hearing loss by stimulating the auditory nerve [Wilson and Dorman 2008]. The CI is the most successful neural prostheses developed to date with most of the CI recipients presenting good speech understanding, however music perception remains poor [McDermott 2004; Nogueira et al. 2011]. Music perception is still a challenge because of limited pitch and timbre these devices can transmit [Burns and Viemeister 1981; Galvin, Fu, and Shannon 2009]. CI recipients struggle when tracking the lyrics of a song with complex polyphonic accompaniment, but there is an improvement when the music background is simplified [Nagathil, Weihs, and Martin 2016]. However, music perception varies a lot between implantees and depends a lot on their previous musical experience [Gfeller et al. 2015]. This research focuses on improving CI users' music perception by modifying the mixing balance between the lead singing vocals and the accompaniment.

It has been shown that CI users find music more enjoyable when enhancing the vocals contained in western pop music by 6 dB and that source separation (SS) algorithms can be used to achieve a new mix with these characteristics [Buyens et al. 2014; Pons et al. 2016]. Here we propose state-of-the-art SS algorithms targeting low distortion and low latency performance and investigating the importance of the artifacts introduced by the different algorithms.

Monaural audio SS is being paid lots of attention by many researchers. Research has been made using non-negative matrix factorization (NMF) in order to separate different sources within an audio mixture [Buyens et al. 2014; Pons et al. 2016; Duong et al. 2014]. The main problem in NMF resides in its computational complexity and latency due to its iterative nature, being difficult to adapt to real-time applications [Marxer and Janer 2013].

It is getting more common to automatically discover the higher representations from data by stacking several layers of nonlinear modules, applying the concept of deep learning [Lecun, Bengio, and Hinton 2015]. Some direct applications of deep learning architectures have been recently proposed, such as deep neural networks (DNNs) to estimate ideal binary masks to separate speech signal from noisy mixtures and to perform multi channel audio SS by using magnitude and phase information [Wang, Narayanan, and Wang 2014; Nugraha, Liutkus, and Vincent 2016; Nogueira et al.

2016]. Moreover, deep recurrent neural networks (DRNNs) are proposed to overcome algorithm complexity problems [Pons et al. 2016] and to learn timbre parameters from a single frame of the magnitude spectrogram [Huang et al. 2015]. These methods work well but do not exploit completely local time-frequency features.

Deep convolutional auto encoders (DCAE) combine the concept of denoising auto encoders (DAEs) and convolutional neural networks (CNNs) and can be used to discover robust localized low-level features from low-dimensional patterns that repeat themselves over the networks' input [Masci et al. 2011; Du et al. 2017]. A DAE is a special type of fully connected feed forward neural network (FFNN) that takes noisy input signals and outputs their denoised version [Xie, Xu, and Chen 2012; Vincent et al. 2010]. DAEs are common in deep learning, they are used to learn robust low-dimensional features even when the inputs are perturbed with some noise [Vincent et al. 2008; Hinton and Salakhutdinov 2006] and for single channel SS (SCSS) where the inputs consist of the spectral frames of the mixed signal and the outputs are the spectral frames of the target sources [Kim and Smaragdis 2015; Smaragdis and Venkataramani 2016]. Fully connected DAEs cannot capture the 2D (spectral-temporal) structures of the spectrogram of the input and output signals. Since DAEs are fully connected networks, they usually have a lot of parameters to be optimized and here is where CNNs come into play. CNNs have been used successfully in audio processing applications such as speech recognition [Qian et al. 2016], speech enhancement [Fu, Tsao, and Lu 2016], audio tagging [Xu et al. 2017], and many music related applications [Han, Kim, and Lee 2017; Choi et al. 2016; Korzeniowski and Widmer 2016] for their ability in extracting robust spectral-temporal structures from audio signals [Lee et al. 2009]. These NNs have also been implemented in previous research to address SCSS [Uhlich et al. 2017; Grais et al. 2016; Kim and Smaragdis 2015; Chandna et al. 2017]. DCAEs are proposed to take advantage of small scale features, investigating the feasibility of real-time implementation [Chandna 2016; Grais and Plumbley 2017; Goehring et al. 2017].

Motivated by the aforementioned success of deep learning on monaural SS, this thesis investigates the performance of SS algorithms, together with their potential application in the sound coding strategy of a CI to enhance the music enjoyment of its recipient. In this study, we will present the design and evaluation of two DCAEs, a DRNN, an MLP, and NMF. The way to assess their performance is divided into two main evaluation aspects:

- Objective evaluation: The quality of the SS algorithms is assessed by obtaining quantitative measurements which will reflect the amount of distortions and artifacts introduced by the SS process.
- Subjective evaluation: The different algorithms are tested on normal hearing (NH) listeners and CI recipients through two different tests. For the first test, subjects are asked to select the most enjoyable mix between two music excerpts. The second subjective test protocol relies on the principle of multi-criteria evaluation; The MUltiple Stimuli with Hidden Reference and Anchor (MUSHRA) [Emiya et al. 2011].

This thesis focuses on the separation of vocal elements from the music background. It is important to understand how susceptible the subjects are to the artifacts introduced by the SS algorithms. Therefore, this work investigates the maximum levels of acceptable signal degradation.

1.2 Goals

The aim of the study is to answer which SS algorithm will be suitable for CI users taking complexity and latency into account.

The main research questions are:

- Can CI users perceive the distortions and artifacts introduced by the SS algorithms?
- How do the physical evaluation values relate to the separation's quality for CI users?
- Is there any potential applicability to the daily live music experience?

The research questions will be answered considering data gathered from objective experiments as well as from perceptual tests conducted with CI and NH participants (see Appendix A).

1.3 Structure of the Document

The remainder of the thesis is organized as follows. Chapter 2 reviews the state-of-the-art surrounding CI music perception, SS, and deep learning. In chapter 3, we introduce the different SS techniques implemented in this study next to the proposed framework with detailed training configurations. The audio material used will be described following with the objective and subjective evaluation methods' depiction. Chapter 4 will present the results, with an analysis of the relation between the objective measures and the perceived signal degradation by the tested subjects. Finally, chapter 5 will summarize and discuss global observations of the results, contrast them with the latest related research and conclude the document.

Chapter 2

State of the Art

This chapter provides the main concepts regarding CIs and audio SS. The basic principles and theory of SS are defined in order to provide the necessary background to the reader. All the information contained in this chapter belongs to the state-of-the-art of these subjects.

2.1 Cochlear Implants

Sever hearing loss problems that some people suffer from can be treated by providing them with a surgically implanted electrical device called CI. Before going into the details of how the latter is achieved is important to understand the aspects of NH.

In NH, sound waves travel through air and reach the tympanic membrane via the ear canal, causing vibrations that move three small bones in the middle ear. This action triggers the movement of the stapes, the third bone in the chain. The "footplate" of the stapes is attached to a flexible membrane in the bony shell of the cochlea called the oval window. Inward and outward movements of this membrane induce pressure oscillations in the cochlear fluids, which in turn initiate a traveling wave of displacement along the basilar membrane (BM), a highly specialized structure that divides the cochlea along its length. This membrane has graded mechanical properties. At the base of the cochlea, near the stapes and oval window, it is narrow and stiff. At the other end, near the apex, the membrane is wide and flexible. These properties give rise to the traveling wave and to points of maximal response according to the frequency or frequencies of the pressure oscillations in the cochlear fluids. The traveling wave propagates from the base to the apex. For an oscillation with a single frequency, the magnitude of displacements increases up to a particular point along the membrane and then drops precipitously thereafter. High frequencies produce maxima near the base of the cochlea, whereas low frequencies produce maxima near the apex. Motion of the BM is sensed by the sensory hair cells in the cochlea, which are attached to the top of the BM in a matrix of cells called the organ of Corti. The cells are arranged in four rows along the length of the cochlea. The cells in the innermost row (closest to the modiolus or "core" of the cochlea) are called the inner hair cells (IHCs), and the cells in the remaining rows are called the outer hair cells (OHCs).

The principal cause of hearing loss is damage to or complete destruction of the sensory hair cells. The hair cells are fragile structures and are subject to a wide variety

of insults, including but not limited to genetic defects, infectious diseases, overexposure to loud sounds, certain drugs, and aging. In the deaf or deafened cochlea, the IHCs in particular are largely or completely absent, severing the connection between the peripheral and central auditory systems. The function of a cochlear prosthesis is to bypass the missing hair cells by directly stimulating the surviving neurons in the auditory nerve. Direct stimulation of the auditory nerve is produced by currents delivered through electrodes placed in the scala tympani (ST), one of three fluid-filled chambers along the length of the cochlea.

CIs act as a sensory neuroprosthesis by replacing the function of the outer, middle and part of the inner ear [Lehnhardt and Laszig 2009]. Therefore, a microphone and a speech processor (including a battery) are applied surrounding the outer ear. The transmitter coil is placed externally above the internal receiver coil. The stimulator and the electrode array are surgically implanted.

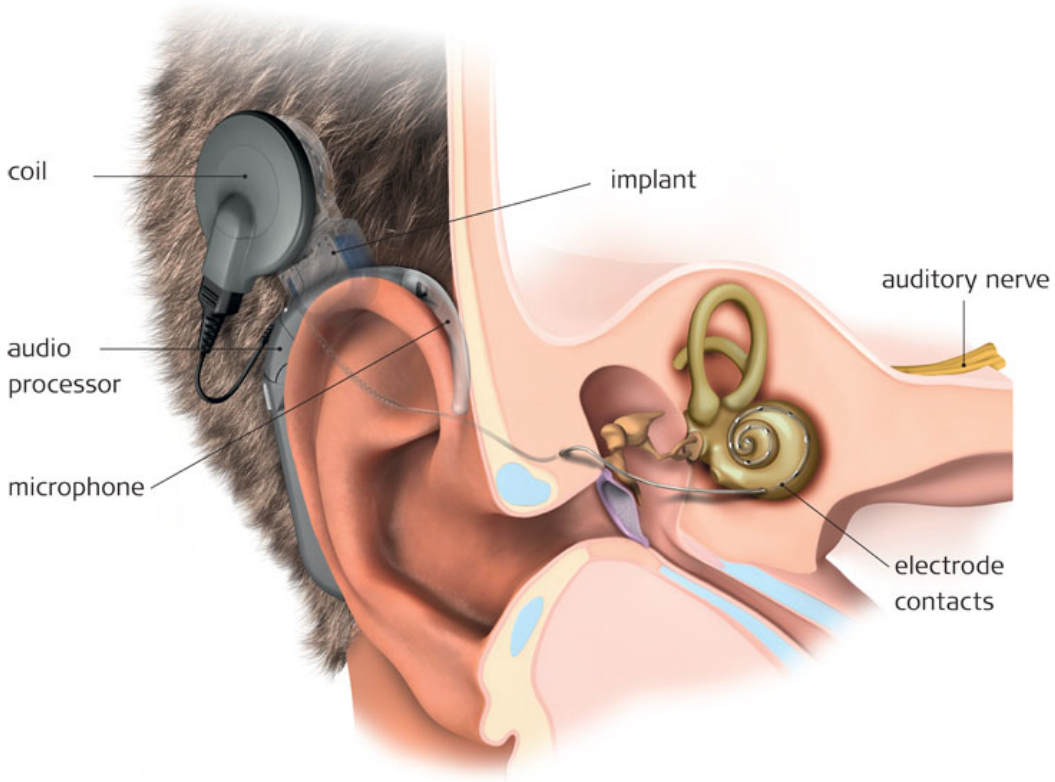

FIGURE 2.1: Ear with installed CI. (Image source: http://www.medel.com)

2.1.1 Microphone and Speech Processor

The microphone captures surrounding sounds which are further analyzed and converted into electrical signals by the speech processor. Therefore the incoming signal is band-pass filtered and compressed to the individual electrical dynamic range. The main function of the signal processor is to compartmentalize the input signal into its frequency components similar to a healthy inner ear. The speech processor combined with the battery as small as a common hearing aid and is worn around the auricle. Thereby the microphone extends into the auricle for a natural audio signal receiving due to the auricles shape.

2.1.2 Transmitter and Receiver Coil

The transmitter coil is placed posterior to the ear, above the implanted receiver coil. Energy and signals are transferred transcutaneously to the internal receiver coil which transmits the electrical signals to the electrode array. The receiver is surgically placed in a excavation drilled in the skull bone behind the auricle [Bear et al. 2001; Rau, Lenarz, and Majdani 2015].

2.1.3 Battery

To provide the CI with the required power, size 675 disposable batteries (zinc-air) or rechargeable batteries (lithium-ion) are connected. Sound coding strategies are being designed to reduce power consumption and minimize the size of the batteries and hence the speech processor [Nogueira et al. 2017; Langner et al. 2017].

2.1.4 Electrode Array

The electrode array is carefully inserted into the scala tympani along the auditory nerve. Electrode arrays differ in the number of electrodes, ranging from 8 to 22. Material properties provide different features: non-magnetics designed for use in the MRI environment or even materials with memory effect, ensuring the correct bending along the auditory nerve. Therefore, a small whole is drilled right between two facial nerve strands and passing the auditory ossicles. The surgery process takes around 4 – 5h and needs to be executed very precisely since the cochlea is positioned relatively deep and close to the brain as visible in Fig. 2.1. The electrode array activate the auditory nerve from the base towards the apex in a tonotopic arrangement: stimulation along the base evokes a perception of high frequencies, stimulation at the apex evokes low frequency sounds.

2.1.5 Sound Coding Strategies

The processor converts the received analog signal into a digital signal. Further, it processes the signal for a better intelligibility by modulating the frequencies, formants, spectral maxima and intensities. The auditory sensation is injected to the auditory nerve by an electrical charge. The stimulation can be performed either in a monopolar (Fig. 2.2 A) or bipolar mode (Fig. 2.2 B) mode. In the monopolar mode, an extra-cochlear reference electrode is commonly placed at the temporal muscle. The reference electrode for the bipolar mode accesses adjacent intra-cochlear electrodes, as shown in Fig. 2.2 B [Zhu et al. 2012].

Further, electrodes can be activated simultaneously, by activating them in sequences or at once. The parallel stimulation by two or more electrodes generates a newly combined pitch perception at a frequency level in between. This way the number of electrodes is expandable by adding virtual channels. An example of generating a virtual channel is presented in Fig. 2.3 B and C.

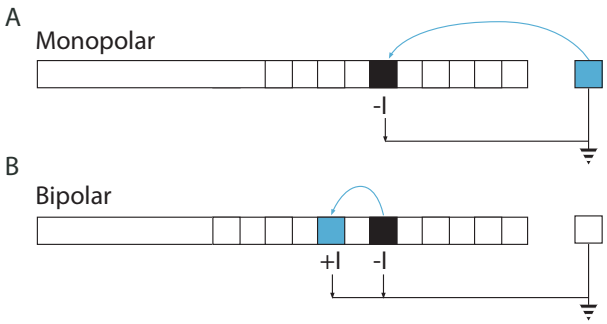

FIGURE 2.2: Monopolar and bipolar stimulation.

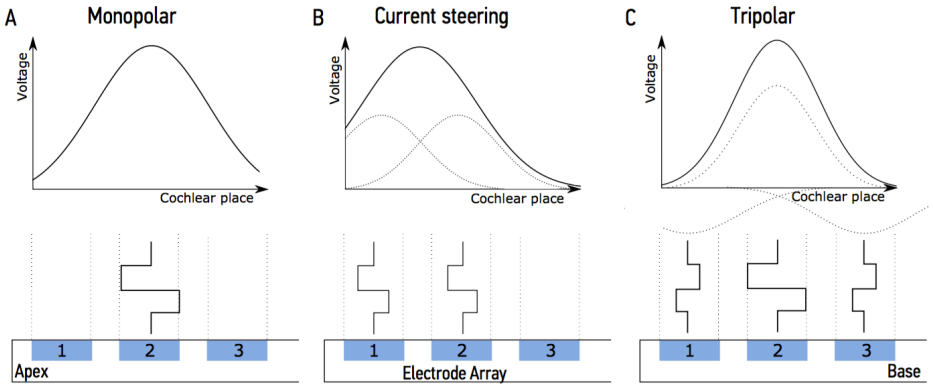

FIGURE 2.3: Electrode stimulation using virtual channels [Langner et al. 2017].

2.2 Source Separation

2.2.1 Introduction

SS problems in digital signal processing are those in which several signals have been mixed together into a combined signal and the objective is to recover the original component signals from the combined one. A good example of a SS problem in the audio context is the cocktail party effect [Bronkhorst 2000] where a number of people are talking simultaneously in a room, and a listener is trying to follow one of the discussions. The SS problem has gained some interest over the years and while techniques for music separation exist, these cannot operate real time and hence, are not suitable for applications that require low-latency processing, such as real time speech enhancement for CIs.

Humans can easily identify the elements in a music mix, but it is still a difficult task for a computer to automatically recognize them. This is mainly because music in the real world is mainly polyphonic and makes extraction of the information very challenging. Furthermore, the elements of a mix vary in many ways such as in timbre, quality, and tempo, making the problem even more complicated. In the music information retrieval (MIR) and especially in this study is highly desirable to identify the most important instrument in a music mix; the lead vocal. This can be demanding since it is been shown that CI users find western pop music more enjoyable when enhancing the lead singing vocal [Buyens et al. 2014; Pons et al. 2016].

Audio SS can be performed in various ways. In this research a deep learning framework for monaural singing voice separation is proposed, aiming at a more reliable, robust and low latency monaural audio SS targeting the lead singing voice within a western pop music track. A brief description of some of the state-of-the-art algorithms which attempt to solve the problem is provided herewith, along with links to original papers from which they have been cited. For a complete description of the algorithms, please refer to the cited papers.

2.2.2 Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) [Paatero and Tapper 1994; Lee and Seung 1999] has recently received much attention as an unsupervised learning method for finding meaningful and physically interpretable latent variable decompositions. The constraint of non-negativity is natural for a wide range of natural signals, such as pixel intensities, amplitude spectra, and occurrence counts. NMF has found widespread application in many areas and has for example been used in environmetrics [Paatero and Tapper 1994] and chemometrics [Sajda, Du, and Parra 2003] to find underlying explanatory sources in series of chemical concentration measurements; in image processing [Lee and Seung 1999] to find useful features in image databases; in text processing [Gaussier and Goutte 2005] to find groups of words that constitute latent topics in sets of documents; and in audio processing [Schmidt 2008] to separate mixtures of audio sources.

2.2.3 Neural Networks

A neural network (NN) is an information processing system inspired by the human nervous system [Grossberg 1988]. Like the nervous system, artificial NNs comprise of connections of nodes called neurons. Each neuron receives an input, processes the information in the input and gives an output defined by certain activation function (the most common activation functions are shown in Fig. 2.4). These neurons contain parameters, which must be optimized using a training set, which has a labeled ground truth. Once trained, the network can be used to input data to produce outputs for testing and for future use.

Each layer consists of neurons, which can mathematically be described in terms of its parameters. These parameters are optimized during the training phase. The training methods will be described in subsection 2.2.7.

2.2.3.1 Challenges

The main challenges one will have to address when modeling a NN are listed below:

- Data Problems: Interest in artificial NNs has evolved from their capacity to process information, which comes in data format. It is frequently necessary to carry out a pre-processing of the data before presenting it to the NN. The main data problems which may occur are the following:

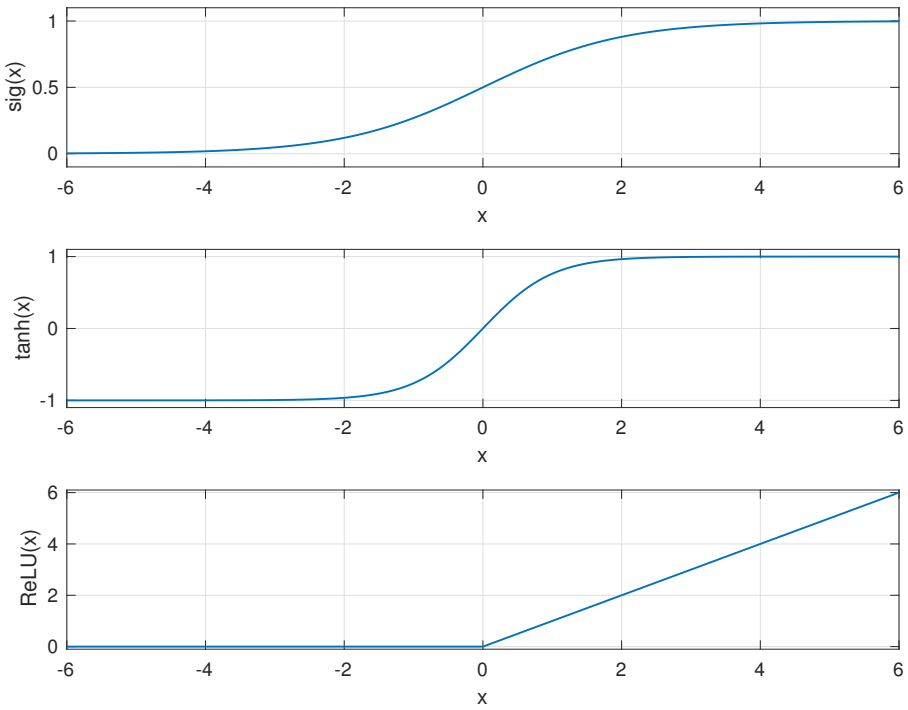

FIGURE 2.4: Most common activation functions used in NN architectures.

– Limited Data:

  When only a limited amount of data is available cross-validation techniques are commonly used based on dividing the available data into two groups, one for learning and the other to validate the behavior of the network. In order to gain a better knowledge of the network, the size and number of elements may be modified for training and evaluating the network in different situations [Geisser 1993; Kohavi 1995].

– Imbalanced Data:

  A problem which occurs in learning, usually when in a classification problem there are much more elements of some classes than others [He and Garcia 2009]. There are several techniques to solve this problem, mainly focused either at the data level (sampling methods) or at the classifier level (modifying it internally). The sampling methods in imbalanced learning applications try to modify the imbalanced data set by some mechanisms in order to provide a balanced distribution by considering the representative proportions of class examples in the distribution. The cost-sensitive learning methods target the imbalanced learning problem by using different cost matrices that describe the costs of misclassifying any particular data example [Frasca et al. 2013]. Specific kernel-based learning methods and active learning methods for imbalanced learning have also been developed.

  - Incomplete Data:

    Sometimes a collection of data to resolve a specific task is available but it has become incomplete due to being lost or because some of its variables or features are unknown. The solution to this problem is centered on approximating missing values, discovering a relationship between the known and the unknown data. Techniques based on NNs and from other perspectives, such as Multiple Kernel Learning [Kumar et al. 2013], exist to solve this problem.

- Learning Problems: Learning consists of estimating the parameters of a model of given data, and this concept is one of the most notable contributions of NNs to the field of information processing systems. However; a central problem in machine learning is how to make an algorithm that will perform well not just on the training data, but also on new inputs. Many strategies used in machine learning are explicitly designed to reduce the test error, possibly at the expense of increased training error. These strategies are known collectively as regularization.

  Regularization is "any modification we make to the learning algorithm that is intended to reduce the generalization error, but not its training error" [Goodfellow, Bengio, and Courville 2016] and the rest of this section will review the most common regularization techniques used.

  - Dataset augmentation:

    An overfitting model (NN or any other type of model) can perform better if learning algorithm processes more training data. While an existing dataset might be limited, for some machine learning problems there are relatively easy ways of creating synthetic data.

    There is no general recipe regarding how the synthetic data should be generated and it varies a lot from problem to problem. The general principle is to expand the dataset by applying operations which reflect real world variations as close as possible.

  - Early Stopping:

    Early-stopping combats overfitting interrupting the training procedure once model's performance on a validation set gets worse. A validation set is a set of examples that it is never use for gradient descent, but which is also not a part of the test set. The validation examples are considered to be representative of future test examples. Early stopping is effectively tuning the hyper-parameter number of epochs.

– Dropout:

The term "dropout" refers to dropping out units (hidden and visible) in a NN. Dropping a unit out means temporarily removing it from the network, along with all its incoming and outgoing connections. The choice of which units to drop is random. In the simplest case, each unit is retained with a fixed probability p independent of other units, where p can be chosen using a validation set or can simply be set at 0.5, which seems to be close to optimal for a wide range of networks and tasks. For the input units, however, the optimal probability of retention is usually closer to 1 than to 0.5.

Applying dropout to a NN amounts to sampling a "thinned" network from it. The thinned network consists of all the units that survived dropout. A NN with n units can be seen as a collection of 2^n possible thinned NNs. These networks all share weights so that the total number of parameters is still $O(n^2)$, or less. For each presentation of each training case, a new thinned network is sampled and trained. So training a NN with dropout can be seen as training a collection of 2^n thinned networks with extensive weight sharing, where each thinned network gets trained very rarely, if at all.

2.2.4 Deep Neural Networks

Deep neural networks (DNNs) are NNs with more than one hidden layer. As data passes through more than one layer, more abstract representations can be discovered, which might help in its classification. Each layer of the deep network has inputs and outputs and the number of inputs of each layer is dependent on the number of outputs of its predecessor.

Following the success of machine learning techniques in other fields, particularly image processing, [Krizhevsky et al., 2012a] several researchers have adopted Deep NNs to approach the SS paradigm [Huang et al. 2015; Huang et al. 2014; Grais et al. 2016; Nugraha, Liutkus, and Vincent 2016].

2.2.5 Recurrent Neural Networks

Recurrent neural networks (RNNs) are powerful learning models that achieve state-of-the-art results in a wide range of supervised and unsupervised machine learning tasks. They are suited especially well for machine perception tasks, where the raw underlying features are not individually interpretable. This success is attributed to their ability to learn hierarchical representations, unlike traditional methods that rely upon hand-engineered features [Farabet et al. 2013]. Over the past several years, storage has become more affordable, datasets have grown far larger, and the field of parallel computing has advanced considerably. In the setting of large datasets, simple linear models tend to under-fit, and often under-utilize computing resources.

Deep learning methods, in particular, those based on DNNs, which are greedily built by stacking restricted Boltzmann machines, and convolutional NNs, that exploit the local dependency of visual information, have demonstrated record-setting results on many important applications. However, despite their power, standard neural

networks have limitations. Most notably, they rely on the assumption of independence among the training and test examples. After each example (data point) is processed, the entire state of the network is lost. If each example is generated independently, this presents no problem. But if data points are related in time or space, this is unacceptable. Frames from video, snippets of audio, and words pulled from sentences represent settings where the independence assumption fails. Additionally, standard networks generally rely on examples being vectors of fixed length. Thus it is desirable to extend these powerful learning tools to model data with temporal or sequential structure and varying length inputs and outputs, especially in the many domains where NN are already the state of the art. RNNs are connectionist models with the ability to selectively pass information across sequence steps while processing sequential data one element at a time. Thus they can model input and/or output consisting of sequences of elements that are not independent. Further, RNNs can simultaneously model sequential and time dependencies on multiple scales.

Since audio is sequential by nature, RNNs have emerged as a powerful tool especially for modeling of speech [Graves, Mohamed, and Hinton 2013] and music [Böck and Schedl 2012]. In particular, they can be used for automatic speech recognition (ASR) also in noisy and reverberated environments [Geiger et al. 2013]. In this study, we introduce the RNN architecture for singing voice enhancement. Previous NN based audio SS approaches for speech were based on FFNNs, despite the context-sensitive nature of speech. Finally, we focus on low latency real-time processing, which is possible with RNNs since their output is only based on the previous time step and the state variable. NNs for blind non-linear SS have been extensively studied, e.g., in [Karhunen et al. 1997; Tan, Wang, and Zurada 2001].

2.2.6 Deep Convolutional Auto Encoders

Different types of DNNs have been used to tackle the audio single channel SS (SCSS) problem [Uhlich et al. 2017, Grais et al. 2016, Kim and Smaragdis 2015, Chandna et al. 2017]. The denoising auto-encoder (DAE) is a special type of fully connected FFNNs that takes noisy input signals and outputs their denoised version [Xie, Xu, and Chen 2012, Vincent et al. 2010]. DAEs are common in deep learning, they are used to learn robust low-dimensional features even when the inputs are perturbed with some noise [Vincent et al. 2008, Hinton and Salakhutdinov 2006]. DAEs have been used for SCSS where the inputs of the DAE are the spectral frames of the mixed signal and the outputs are the spectral frames of the target source [Kim and Smaragdis 2015, Smaragdis and Venkataramani 2016]. Fully connected DAEs can not capture the 2D (spectral-temporal) structures of the spectrogram of the input and output signals. Since DAEs are fully connected networks, they usually have a lot of parameters to be optimized.

For their ability in extracting robust spectral-temporal structures of different audio signals [Lee et al. 2009], convolutional neural networks (CNN) have been used successfully to learn useful features in many audio processing applications such as: speech recognition [Qian et al. 2016], speech enhancement [Fu, Tsao, and Lu 2016], audio tagging [Xu et al. 2017], and many music related applications [Han, Kim, and Lee 2017; Choi et al. 2016; Korzeniowski and Widmer 2016]. Deep convolutional auto-encoders (DCAEs) are also a special type of CNNs that can be used to discover robust localized low-dimensional patterns that repeat themselves over the input [Masci et al. 2011; Du et al. 2017]. DCAEs differ from conventional DAEs as

their parameters (weights) are shared which makes the DCAEs have much fewer parameters than DAEs. The ability of DCAEs to extract repeating patterns in the input makes them suitable to be used to suppress the background music signal that is embedded in speech signals to improve the quality of the speech recognition system [Zhao et al. 2015]. The results show that DCAE works better than DAE in removing the background music signals from the speech signal [Zhao et al. 2015]. Recently, a fully DCAE (all the network layers are composed of convolutional units) were used for speech enhancement [Park and Lee 2016] that maps the distorted speech signal to its clean speech signal. The results in [Park and Lee 2016] show that DCAE works as well as the fully connected FFNNs and RNNs even when the number of parameters in the DCAE is much less than the number of parameters in FFNNs and RNNs.

Motivated by the aforementioned successes of using CNNs and DCAEs in a variety of audio signals, we propose in this thesis to use DCAEs for the audio SCSS. The main idea in this thesis is to train a DCAE to extract one target source from the mixture and treats the other sources as background noise that needs to be suppressed. This means that the number of DCAEs equal the number of sources to be estimated. This is a very challenging task because each CDAE has to deal with highly non-stationary background signals/noise. Each DCAE sees the magnitude spectrograms as 2D segments which help in learning the temporal and spectral information for the audio signals. From the ability of DCAEs in learning noise robust features, in this work, we train each DCAE to learn unique temporal-spectral patterns for its corresponding target source. Each trained DCAE is then used to extract the related patterns of its corresponding target source from the mixed signal.

2.2.7 Training Algorithms

Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize NNs. At the same time, every state-of-the-art Deep Learning library contains implementations of various algorithms to optimize gradient descent (e.g. lasagne's, caffe's, and kera's documentation). These algorithms, however, are often used as black-box optimizers, as practical explanations of their strengths and weaknesses are hard to come by.

Gradient descent is a way to minimize an objective function $J(\theta)$ parameterized by a model's parameters $\theta \in \mathbb{R}^d$ (where $\mathbb{R}$ is the real space) by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ w.r.t. to the parameters. The learning rate η determines the size of the steps taken to reach a (local) minimum.

2.2.7.1 Gradient Descent Variants

There are three variants of gradient descent, which differ in how much data is used to compute the gradient of the objective function. Depending on the amount of data, making a trade-off between the accuracy of the parameter update and the time it takes to perform an update.

- Batch gradient descent:

  Vanilla gradient descent, aka batch gradient descent, computes the gradient of the cost function w.r.t. to the parameters θ for the entire training dataset:

  $$\theta = \theta - \eta \nabla_\theta J(\theta). \tag{2.1}$$

  As theres the need to calculate the gradients for the whole dataset to perform just one update, batch gradient descent can be very slow and is intractable for datasets that do not fit in memory. Batch gradient descent also does not allow us to update the model on line, i.e. with new examples on-the-fly.

  Then, the parameters are updated in the direction of the gradients with the learning rate determining how big of an update performed. Batch gradient descent is guaranteed to converge to the global minimum for convex error surfaces and to a local minimum for non-convex surfaces.

- Stochastic gradient descent:

  Stochastic gradient descent (SGD) in contrast performs a parameter update for each ith training example $x^{(i)}$ and label $y^{(i)}$:

  $$\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i)}; y^{(i)}). \tag{2.2}$$

  Batch gradient descent performs redundant computations for large datasets, as it recomputes gradients for similar examples before each parameter update. SGD does away with this redundancy by performing one update at a time. It is therefore usually much faster and can also be used to learn on line.

  While batch gradient descent converges to the minimum of the basis the parameters are placed in, SGD's fluctuation, on the one hand, enables it to jump to new and potentially better local minima. On the other hand, this ultimately complicates convergence to the exact minimum, as SGD will keep overshooting. However, it has been shown that when slowly decreasing the learning rate, SGD shows the same convergence behavior as batch gradient descent, almost certainly converging to a local or the global minimum for non-convex and convex optimization respectively.

- Mini-batch gradient descent:

  Mini-batch gradient descent finally takes the best of both worlds and performs an update for every mini-batch of n training examples:

  $$\theta = \theta - \eta \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}). \tag{2.3}$$

  This way, it a) reduces the variance of the parameter updates, which can lead to more stable convergence; and b) can make use of highly optimized matrix optimizations common to state-of-the-art deep learning libraries that make computing the gradient w.r.t. a mini-batch very efficient. Common mini-batch sizes range between 50 and 256 but can vary for different applications. Mini-batch gradient descent is typically the algorithm of choice when training a NN and the term SGD usually is employed also when mini-batches are used.

2.2.7.2 Challenges

Vanilla mini-batch gradient descent, however, does not guarantee good convergence, but offers a few challenges that need to be addressed:

- Choosing a proper learning rate can be difficult. A learning rate that is too small leads to painfully slow convergence, while a learning rate that is too large can hinder convergence and cause the loss function to fluctuate around the minimum or even to diverge.
- Learning rate schedules [Robinns and Monro 1951] try to adjust the learning rate during training by e.g. annealing, i.e. reducing the learning rate according to a predefined schedule or when the change in objective between epochs falls below a threshold. These schedules and thresholds, however, have to be defined in advance and are thus unable to adapt to a dataset's characteristics [Darken, Chang, and Moody 1992].
- Additionally, the same learning rate applies to all parameter updates. If data is sparse and the features have very different frequencies, it is maybe not a good idea to update all of them to the same extent, but perform a larger update for rarely occurring features.
- Another key challenge of minimizing highly non-convex error functions common for NNs is avoiding getting trapped in their numerous suboptimal local minima. Dauphin et al. [Dauphin et al. 2014] argue that the difficulty arises in fact not from local minima but from saddle points, i.e. points where one dimension slopes up and other slopes down. These saddle points are usually surrounded by a plateau of the same error, which makes it notoriously hard for SGD to escape, as the gradient is close to zero in all dimensions.

2.2.7.3 Gradient Descent Optimization Algorithms

In the following, some algorithms that are widely used by the deep learning community to deal with the aforementioned challenges will be presented.

- Momentum:

  SGD has trouble navigating ravines, i.e. areas where the surface curves vary much more steeply in one dimension than in another [Sutton 1986], which are common around local optima. In these scenarios, SGD oscillates across the slopes of the ravine while only making hesitant progress along the bottom towards the local optimum.

  Momentum [Qian 1999] is a method that helps accelerate SGD in the relevant direction and dampens oscillations. It does this by adding a fraction γ of the update vector of the past time step to the current update vector:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta), \tag{2.4}$$

$$\theta = \theta - v_t. \tag{2.5}$$

  Essentially, when using momentum, a ball is pushed down a hill. The ball accumulates momentum as it rolls downhill, becoming faster and faster on the

way (until it reaches its terminal velocity if there is air resistance, i.e. ($\gamma < 1$). The same thing happens to the parameter updates: The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions. As a result, faster convergence and reduced oscillation is achieved.

- Nesterov accelerated gradient:

  However, a ball that rolls down a hill, blindly following the slope, is highly unsatisfactory.

  Nesterov accelerated gradient (NAG) [Nesterov 1983] is a way to give the momentum term this kind of prescience. The momentum term γv_{t-1} is used to move the parameters θ. Computing $\theta - \gamma v_{t-1}$ thus gives us an approximation of the next position of the parameters (the gradient is missing for the full update), a rough idea where the parameters are going to be. A look ahead can be done by calculating the gradient not w.r.t. to the current parameters θ but w.r.t. the approximate future position of the parameters:

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1}), \tag{2.6}$$

$$\theta = \theta - v_t. \tag{2.7}$$

  Again, the momentum term γ is set to a value of around 0.9. While Momentum first computes the current gradient and then takes a big jump in the direction of the updated accumulated gradient, NAG first makes a big jump in the direction of the previously accumulated gradient, measures the gradient and then makes a correction, which results in the complete NAG update. This anticipatory update prevents the minimization from going too fast and results in increased responsiveness, which has significantly increased the performance of RNNs on a number of tasks [Bengio, Boulanger-Lewandowski, and Pascanu 2013].

- Adagrad:

  Adagrad [Duchi, E.Hazan, and Singer 2011] is an algorithm for gradient-based optimization that does just this: It adapts the learning rate to the parameters, performing larger updates for infrequent parameters and smaller updates for frequent parameters. For this reason, it is well-suited for dealing with sparse data. [J.Dean et al. 2012] have found that Adagrad greatly improved the robustness of SGD and used it for training large-scale neural nets at Google. Moreover, [Pennington, Socher, and Manning 2014] used Adagrad to train GloVe word embeddings, as infrequent words require much larger updates than frequent ones.

  Previously, the update was performed for all parameters θ at once as every parameter θ_i used the same learning rate η. As Adagrad uses a different learning rate for every parameter θ_i at every time step t, first, Adagrad's per-parameter update is shown, vectorizing it. For brevity, $g_{t,i}$ is set to be the gradient of the objective function w.r.t. to the parameter θ_i at time step t:

$$g_{t,i} = \nabla \theta J(\theta_i). \tag{2.8}$$

The SGD update for every parameter θ_i at each time step t then becomes:

$$\theta_{t+1,i} = \theta_{t,i} - \eta g_{t,i}. \tag{2.9}$$

In its update rule, Adagrad modifies the general learning rate η at each time step η for every parameter θ_i based on the past gradients that have been computed for θ_i:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} g_{t,i}. \tag{2.10}$$

$G_t \in \mathbb{R}^{d \times d}$ here is a diagonal matrix where each diagonal element i, i is the sum of the squares of the gradients w.r.t. θ_i up to time step i, while ϵ is a smoothing term that avoids division by zero (usually on the order of 10^{-8}). Interestingly, without the square root operation, the algorithm performs much worse.

As G_t contains the sum of the squares of the past gradients w.r.t. to all parameters θ along its diagonal, vectorizing the implementation by performing an element-wise matrix-vector multiplication $\odot$ between G_t and g_t obtaining:

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_{t,i}. \tag{2.11}$$

One of Adagrad's main benefits is that it eliminates the need to manually tune the learning rate. Most implementations use a default value of 0.01 and leave it at that.

Adagrad's main weakness is its accumulation of the squared gradients in the denominator: since every added term is positive, the accumulated sum keeps growing during training. This, in turn, causes the learning rate to shrink and eventually become infinitesimally small, at which point the algorithm is no longer able to acquire additional knowledge. The following algorithms aim to resolve this flaw.

- Adadelta:

  Adadelta [Zeiler 2012] is an extension of Adagrad that seeks to reduce it's aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size w.

  Instead of inefficiently storing w previous squared gradients, the sum of gradients is recursively defined as a decaying average of all past squared gradients. The running average $E[g^2]_t$ at time step t then depends only on the previous average and the current gradient:

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1 - \gamma) g_t^2. \tag{2.12}$$

γ is set to a similar value as the momentum term, around 0.9. The vanilla SGD update can now be expressed in terms of the parameter update vector $\Delta\theta_t$:

$$\Delta\theta = -\eta g_{t,i}, \tag{2.13}$$
$$\theta_{t+1} = \theta_t + \Delta\theta_t. \tag{2.14}$$

The previously derived parameter update vector of Adagrad thus takes the form:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t. \tag{2.15}$$

Replacing the diagonal matrix G_t with the decaying average over past squared gradients $E[g^2]_t$ provides the following expression:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t. \tag{2.16}$$

As the denominator is just the root mean squared (RMS) error criterion of the gradient, it can be replaced by the criterion short-hand:

$$\Delta\theta_t = -\frac{\eta}{\sqrt{RMS[g]_t}} g_t. \tag{2.17}$$

The authors note that the units in this update (as well as in SGD, Momentum, or Adagrad) do not match, i.e. the update should have the same hypothetical units as the parameter. To realize this, they first define another exponentially decaying average, this time not of squared gradients but of squared parameter updates:

$$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma)\Delta\theta_t^2. \tag{2.18}$$

The root mean squared error of parameter updates is thus:

$$RMS[\Delta\theta]_t = \sqrt{E[\Delta\theta^2]_t + \epsilon}. \tag{2.19}$$

Since $RMS[\Delta\theta]_t$ is unknown, it can be approximated by the RMS of parameter updates until the previous time step. Replacing the learning rate η in the previous update rule with $RMS[\Delta\theta]_{t-1}$ finally yields the Adadelta update rule:

$$\Delta\theta_t = -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t, \tag{2.20}$$

$$\theta_{t+1} = \theta_t + \Delta\theta_t. \tag{2.21}$$

With Adadelta, no default learning rate has to be set up, as it has been eliminated from the update rule.

- Adam:

  Adaptive Moment Estimation (Adam) [Kingma and Ba 2014] is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients v_t like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients m_t, similar to momentum:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{2.22}$$

$$v_t = \beta_2 vt - 1 + (1 - \beta_2) g_t^2. \tag{2.23}$$

  m_t and v_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively, hence the name of the method. As m_t and v_t are initialized as vectors of 0's, the authors of Adam observe that they are biased towards zero, especially during the initial time steps, and especially when the decay rates are small.

  They counteract these biases by computing bias-corrected first and second moment estimates:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{2.24}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{2.25}$$

  The authors propose default values of 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ϵ. They show empirically that Adam works well in practice and compares favorably to other adaptive learning-method algorithms.

- AdaMax:

  The v_t factor in the Adam update rule scales the gradient inversely proportionally to the l_2 norm of the past gradients (via the v_{t-1} term) and current gradient $|g_t|^2$:

$$v_t = \beta_2 vt - 1 + (1 - \beta_2) |g_t|^2. \tag{2.26}$$

  This update is generalized to the l_p norm. Note that [Kingma and Ba 2014] also parameterize β_2 as β_2^p:

$$v_t = \beta_2^p vt - 1 + (1 - \beta_2^p)|g_t|^p. \tag{2.27}$$

Norms for large p values generally become numerically unstable, which is why l_1 and l_2 norms are most common in practice. However, l_∞ also generally exhibits stable behavior. For this reason, the authors propose AdaMax [Kingma and Ba 2014] and show that v_t with l_∞ converges to the following more stable value. To avoid confusion with Adam, u_t is used to denote the infinity norm-constrained v_t:

$$v_t = \beta_2^\infty vt - 1 + (1 - \beta_2^\infty)|g_t|^\infty = max(\beta_2 v_{t-1}, |g_t|). \tag{2.28}$$

Plugging this into the Adam update equation by replacing$\sqrt{\hat{v}_t} + \epsilon$ with u_t the AdaMax update rule is obtained:

$$\theta_{t+1} = \theta_t - \frac{\eta}{u_t}\hat{m}_t. \tag{2.29}$$

Good default values are again η=0.002, β_1=0.9 and β_2=0.999.

- Nadam:

  Adam can be viewed as a combination of RMSprop and momentum: RMSprop contributes the exponentially decaying average of past squared gradients v_t, while momentum accounts for the exponentially decaying average of past gradients m_t. Is been shown also that Nesterov accelerated gradient (NAG) is superior to vanilla momentum.

  Nadam (Nesterov-accelerated Adaptive Moment Estimation) [Dozat 2015] thus combines Adam and NAG. In order to incorporate NAG into Adam, its momentum term m_t needs to be modified.

  Recalling the momentum update rule:

$$g_t = \nabla_{\theta_t} J(\theta_t), \tag{2.30}$$
$$m_t = \gamma m_{t-1} + \eta g_t, \tag{2.31}$$
$$\theta_{t+1} = \theta_t - m_t, \tag{2.32}$$

  where J is the objective function, γ is the momentum decay term, and η is the step size. Expanding the third equation above yields:

$$\theta_{t+1} = \theta_t - (\gamma m_{t-1} + \eta g_t). \tag{2.33}$$

This demonstrates again that momentum involves taking a step in the direction of the previous momentum vector and a step in the direction of the current gradient.

NAG then provides a more accurate step in the gradient direction by updating the parameters with the momentum step before computing the gradient. Now, only the gradient g_t needs to be modified in order to arrive at NAG:

$$g_t = \nabla_{\theta_t} J(\theta_t - \gamma m_{t-1}), \tag{2.34}$$
$$m_t = \gamma m_{t-1} + \eta g_t, \tag{2.35}$$
$$\theta_{t+1} = \theta_t - m_t, \tag{2.36}$$

[Dozat 2015] proposes to modify NAG the following way: Rather than applying the momentum step twice – one time for updating the gradient g_t and a second time for updating the parameters θ_{t-1} – a look-ahead momentum vector is directly applied to update the current parameters:

$$g_t = \nabla_{\theta_t} J(\theta_t), \tag{2.37}$$
$$m_t = \gamma m_{t-1} + \eta g_t, \tag{2.38}$$
$$\theta_{t+1} = \theta_t - (\gamma m_t + \eta g_t). \tag{2.39}$$

In order to add Nesterov momentum to Adam, the previous momentum vector can be similarly replaced with the current momentum vector. First, recall that the Adam update rule is the following:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1 g_t), \tag{2.40}$$
$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \tag{2.41}$$
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t. \tag{2.42}$$

Expanding the second equation with the definitions of m_t and m_t in turn gives:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \left(\frac{\beta_1 m_{t-1}}{1 - \beta_1^t} + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right). \tag{2.43}$$

Replacing it with $\hat{m}_{t-1}$:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \left(\beta_1 \hat{m}_{t-1} + \frac{(1 - \beta_1) g_t}{1 - \beta_1^t} \right). \tag{2.44}$$

Nesterov momentum is added by replacing the bias-corrected estimate of the momentum vector of the previous time step m_{t-1} with the bias-corrected estimate of the current momentum vector m_t, which gives the Nadam update rule:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon}\left(\beta_1 \hat{m}_t + \frac{(1-\beta_1)g_t}{1-\beta_1^t}\right). \tag{2.45}$$

Chapter 3

Methods & Materials

3.1 Source Separation Framework

The SS problem, in the signal processing context, is defined as the extraction or isolation of different components of a mixture. In the following, it is assumed a single channel music down-mix $\bar{x}[t] \in \mathbb{R}$ to be the sum of J single channel sources $\bar{y}_j[t] \in \mathbb{R}$. The mixture signal $\bar{x}[t]$ observed at discrete time t can be expressed as:

$$\bar{x}[t] = \sum_{j=1}^{J} \bar{y}_j[t]. \tag{3.1}$$

As all the algorithms in this research are implemented in the frequency domain, it is more convenient to use the frequency representation of (3.1). Let $x[f, n] \in \mathbb{C}$ and $y[f, n] \in \mathbb{C}$ denote the short-time Fourier transform (STFT) coefficients of $\bar{x}[t]$ and $\bar{y}_j[t]$ for frequency bin f and time frame n. Also, let F be the number of frequency bins and N the number of time frames. The mixture spectrum can be expressed as follows:

$$x[f, n] = \sum_{j=1}^{J} y_j[f, n]. \tag{3.2}$$

After the mixture signal has been processed by the SS algorithm, the estimated source spectrogram $\tilde{y}_j[f, n]$ is obtained. The estimated source signals are calculated by performing the inverse short time Fourier transform (ISTFT) of each estimated source spectrogram $y_j[f, n]$. The magnitude spectrogram is processed by the SS algorithm, while the phase of the original mixture is used to synthesize the estimated sources, which are used to generate a new mix based on the desired vocal to instruments ratio (VIR) as shown in Fig. 3.1. The new mix is then processed by the CI.

The following sections present the proposed SS algorithms for the multi-track estimation.

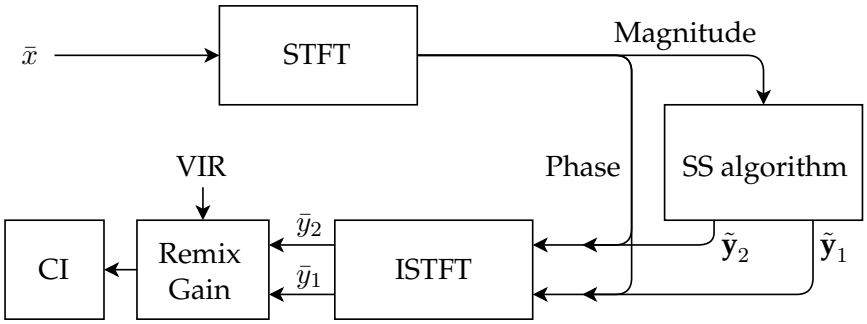

FIGURE 3.1: Block diagram showing the proposed general SS framework for CI users. The mixture spectrum is masked in order to estimate the vocal and the instruments. These are then transformed to the time domain where a vocal to instruments ratio (VIR) gain is applied before being processed by the CI.

3.1.1 NMF

Non-negative Matrix Factorization was introduced by Lee and Seung [Lee and Seung 1999] as an alternative to k-means clustering and principal component analysis (PCA) for data analysis and compression. Let $\mathbf{X} \in \mathbb{C}^{F \times N}$ be the N frame matrix representation of the input mixture spectrogram:

$$\mathbf{X} = \sum_{j}^{J} \mathbf{Y}_j. \tag{3.3}$$

NMF aims at decomposing the $F \times N$ non-negative matrix $\mathbf{X}$ as a product of two non-negative matrices $\mathbf{V}$ and $\mathbf{H}$ of dimensions $F \times K$ and $K \times N$ (where K is the number of basis vectors), respectively, such that $\mathbf{X} \approx \hat{\mathbf{X}} = \mathbf{VH}$. Here $\mathbf{V}$ contains the basis vectors and $\mathbf{H}$ the activations. The key property of NMF is that $\mathbf{W}$ and $\mathbf{H}$ are constrained to be positive matrices. This is in contrast with PCA, where there are no positivity constraints or k-means clustering where each column of $\mathbf{X}$ is constrained to be a unit vector. Subject to the positivity constraints, we seek a solution to the following minimization problem

$$(\mathbf{W}, \mathbf{H})^* = arg \min_{\mathbf{W},\mathbf{H}} D(\hat{\mathbf{X}} || \mathbf{WH}). \tag{3.4}$$

Here, the function $D(\cdot||\cdot)$ is a suitably chosen error function. One particular choice for $D(\cdot||\cdot)$, on which we will focus here, is the information (Kullback-Leibler) divergence [Kullback and Leibler 1951], which measures the similarity between two probability distribution functions $\mathbf{P}$ and $\mathbf{Q}$ as follows:

$$D(\mathbf{P}||\mathbf{Q}) = -\sum_{\nu\tau} \left(p_{\nu\tau} \log \frac{q_{\nu\tau}}{p_{\nu\tau}} - q_{\nu\tau} + p_{\nu\tau} \right). \tag{3.5}$$

Using Jensen's inequality [Bishop 2006] and concavity of $log(x)$ ($\forall\, x \in \mathbb{R}^+$), it can be shown, that $D(\cdot||\cdot)$ is non-negative and $D(\mathbf{P}||\mathbf{Q}) = 0 \iff \mathbf{P} = \mathbf{Q}$. The objective in (5) could be minimized by any suitable optimization algorithm. Lee and Seung [Lee and Seung 1999] have proposed a very efficient variational bound minimization algorithm that has attractive convergence properties and which has been successfully applied in various applications in signal analysis and SS, e.g. [Virtanen 2006]. The

estimation algorithms of Lee and Seung minimize the chosen cost function by initializing the entries of **W** and **H** with random positive values, and then by updating them iteratively using multiplicative rules. Each update decreases the value of the cost function until the algorithm converges.

The factorizations obtained are represented by Equivalent Rectangular Bandwidth (ERB) spectrograms extracted from the input spectrogram, acquired through a 0.025s Hamming window with a 0.013s hop size for a 1024 sample FFT at a 44.1 kHz sample rate.

3.1.2 MLP

A multilayer perceptron (MLP) [Gardner and Dorling 1998; Kriesel 2007] consists of a system of simple interconnected neurons (AKA units), representing a nonlinear mapping between an input vector and an output vector. The units are connected by weights and output signals which are a function of the sum of the inputs to the unit modified by a simple nonlinear, transfer, or activation function. Is the superposition of these simple nonlinear functions that enable the MLP to model complex nonlinear functions. During this study a rectifier linear unit (ReLU), defined by $\phi(x) = max(0, x),\ \forall\ x \in \mathbb{R}$ (see Fig. 2.4), is used as the activation function applied to each connection. The output of a unit is scaled by the connecting weight and fed forward to be an input to the units in the next layer of the network. This implies a direction of information processing, hence the MLP is known as a FFNN. Given an input vector **x**, the output **y** at frame n is given by:

$$\mathbf{y}[n] = \phi(\mathbf{W}\mathbf{x}[n] + \boldsymbol{\beta}), \tag{3.6}$$

where **W** and $\boldsymbol{\beta}$ are the weight matrix and bias, respectively. By selecting a suitable set of connecting weights and transfer functions, it has been shown that a multilayer perceptron can approximate any smooth, measurable function between the input and output vectors [Hornik, Stinchcombe, and White 1989].

The proposed network consists of a 16 unit hidden layer and it is been implemented with the code provided by [Huang et al. 2015] (which can be found on-line[1]). At frame n the input $\mathbf{x}[n]$ is the concatenation of features from a mixture within a window. The magnitude spectra is obtained with a 1024 analysis window for a 1024 FFT and 50% overlap. This network contains 25,666 trainable parameters.

Given the output predictions $\tilde{\mathbf{y}}_1$ and $\tilde{\mathbf{y}}_2$ from the original sources $\mathbf{y}_1$ and $\mathbf{y}_2$, is a common practice to optimize the parameters by minimizing the squared error between them [Huang et al. 2014]:

$$J_{MSE} = \|\mathbf{y}_1 - \tilde{\mathbf{y}}_1\|_2^2 + \|\mathbf{y}_2 - \tilde{\mathbf{y}}_2\|_2^2, \tag{3.7}$$

where $\|\cdot\|_2$ is the l_2 norm between the two vectors.

However, previous research [Huang et al. 2015] suggests a slightly modified version of (3.7) where a γ factor is introduced as follows:

[1] https://github.com/posenhuang/deeplearningsourceseparation

$$J_R = \|\mathbf{y}_1 - \tilde{\mathbf{y}}_1\|_2^2 - \gamma\|\mathbf{y}_1 - \tilde{\mathbf{y}}_2\|_2^2 + \|\mathbf{y}_2 - \tilde{\mathbf{y}}_2\|_2^2 - \gamma\|\mathbf{y}_2 - \tilde{\mathbf{y}}_1\|_2^2. \tag{3.8}$$

The MLP model has been optimized by back-propagating the gradients through time with respect to the objectives [Bishop 1995] for 200 iterations with a γ value of 0.05. The L-BFGS algorithm is used to train the models from normalized random initialization. Feedforward connections are initialized randomly with the initialization scheme proposed by [Hochreiter et al. 2001], with the following uniform distribution:

$$U\left(-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}}\right). \tag{3.9}$$

n_{in} refers to the number of hidden neurons in the previous layer and n_{out} refers to the number of hidden neurons in the forthcoming layer.

Bias weights are initialized to zero. This initialization is the one proposed by default in [Huang et al. 2014] code.

3.1.3 DRNN

This is a particular case of artificial NNs [Kriesel 2007] where connections between units form a direct loop, meaning that they can consider temporal context when training and testing. The general scheme of a DRNN can be defined as follows. Given an L hidden layer DRNN with the recurrent connection at layer l, the lth hidden activation at frame n is defined as:

$$\mathbf{h}^l[n] = \phi_l(\mathbf{W}^l_{rec}\mathbf{h}^l[n-1] + \mathbf{W}^l\mathbf{h}^{l-1}[n] + \boldsymbol{\beta}^l), \tag{3.10}$$

$$\mathbf{h}^1[n] = \phi_1(\mathbf{W}^1_{rec}\mathbf{h}^1[n-1] + \mathbf{W}^1\mathbf{x}[n] + \boldsymbol{\beta}^1), \tag{3.11}$$

and the output, $\mathbf{y}[n]$, can be defined as:

$$\mathbf{y}[n] = \mathbf{W}^L\mathbf{h}^{L-1}[n], \tag{3.12}$$

where $\mathbf{x}[n]$ is the vectorial representation of (3.2) containing F frequency bins of the mixture signal's magnitude spectrum at frame n, ϕ_l is an element-wise non-linear function (in this case a ReLU) and $\mathbf{W}^l$ is the weight matrix for the lth layer.

[Huang et al. 2014] obtained best results when using a deep recurrent architecture based on 3 hidden layers where the second is recurrent. Each hidden layer has 1000 units characterized by a RELU non-linear function, the output layer is a linear layer modeling two sources (see Fig. 3.2): denoted as $\hat{\mathbf{y}}_1[n]$ and $\hat{\mathbf{y}}_2[n]$.

Once the network estimated the desired target sources, a time-frequency mask was applied to the output source layer (see formulas 3.13 and 3.14). Thus imposing that the sum of the predictions is equal to the original mixture.

$$\tilde{\mathbf{y}}_1[n] = \frac{|\hat{\mathbf{y}}_1[n]|}{|\hat{\mathbf{y}}_1[n]| + |\hat{\mathbf{y}}_2[n]|} \odot \mathbf{x}[n], \tag{3.13}$$

$$\tilde{\mathbf{y}}_2[n] = \frac{|\hat{\mathbf{y}}_2[n]|}{|\hat{\mathbf{y}}_1[n]| + |\hat{\mathbf{y}}_2[n]|} \odot \mathbf{x}[n]. \tag{3.14}$$

$\mathbf{x}[n]$ refers to the input magnitude spectrum at frame n, $\odot$ denotes element-wise matrix multiplication and $|\cdot|$ is the absolute value operator.

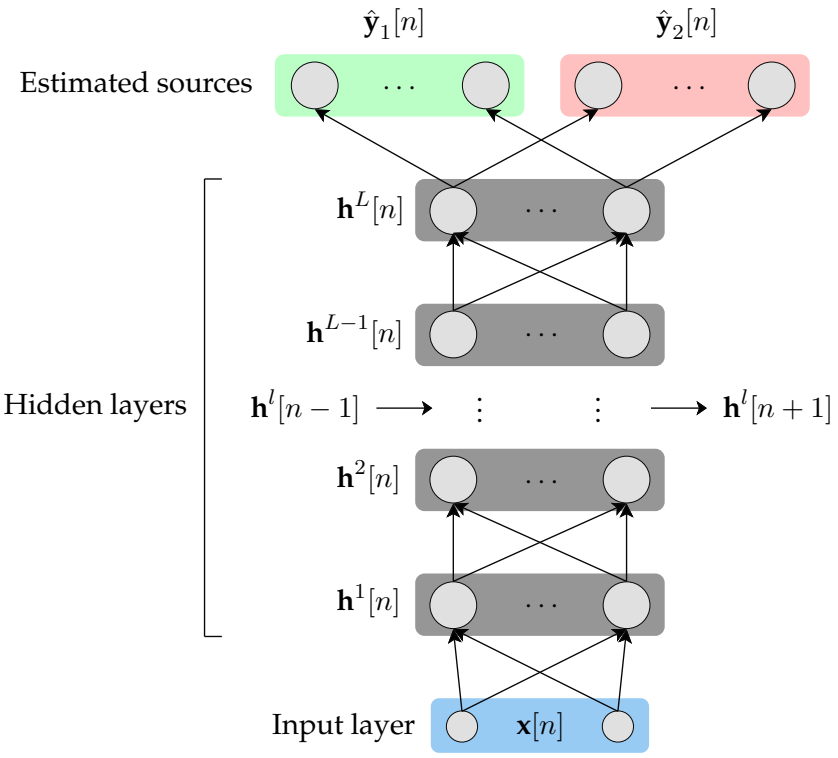

FIGURE 3.2: Baseline DRNN architecture based on the work by [Huang et al. 2014].

The implemented algorithm is based on the same code as the MLP [Huang et al. 2014]. The architecture proposed consists of 3 hidden layers with 1000 units each. This architecture presents a recurrence of three frames on the second hidden layer, also referred as a three frame context window. The network takes a 1024 point magnitude spectra as input, extracted from the audio signal through a 1024 sample raised sine window with a 50% overlap. This DRNN contains 5,569,026 trainable weights.

The training is achieved by minimizing (3.8) like in the case of the MLP for 400 iterations. In this case the recurrent connections are initialized with the initialization scheme proposed by [Hochreiter et al. 2001] but smaller:

$$U\left(-\sqrt{\frac{6}{3 \cdot n_{in}}}, \sqrt{\frac{6}{3 \cdot n_{in}}}\right). \tag{3.15}$$

3.1.4 DCAE

A DCAE is a special case of DAE which performs dimensionality reduction by taking advantage of the characteristic operations involved in CNNs. A DAE consists of two parts, a coder and a decoder, which can be referred to as functions such that:

$$\begin{aligned} \epsilon &: \mathbb{R}^F \rightarrow \mathbb{R}^{F_c} \\ \sigma &: \mathbb{R}^{F_c} \rightarrow \mathbb{R}^F \\ \epsilon, \sigma &= arg \min_{\epsilon,\sigma} \|\mathbf{x} - (\epsilon \circ \sigma)\mathbf{x}\|^2, \end{aligned} \tag{3.16}$$

where F and F_c are the length of the input and code, respectively.

Fig. 3.3 shows a schematic of a simple auto encoder, similar to a fully connected DNN with an input layer, 3 hidden layers and an output layer.

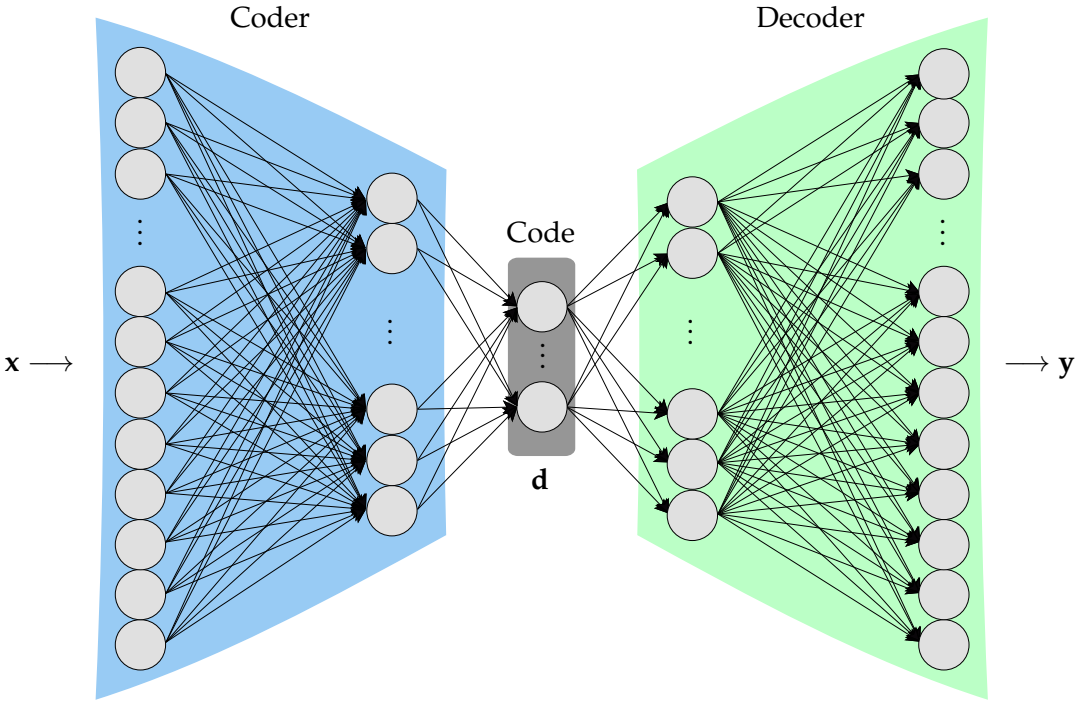

FIGURE 3.3: Schematic representation of a deep auto encoder (see eq. 3.16) with three fully connected hidden layers.

The coder generates a code for each frame n as follows:

$$\mathbf{d}[n] = \phi(\mathbf{W}\mathbf{x}[n] + \boldsymbol{\beta}), \tag{3.17}$$

where $\phi(\cdot)$ is a ReLU, $\mathbf{d}[n]$ is a vector containing the activation values of the coded input, $\mathbf{x}$[n] is an input vector, $\boldsymbol{\beta}$ is the vector containing the unit biases and $\mathbf{W}$ are the trainable weights of the encoder. Then, the decoder applies the following transformation to obtain the output of the network:

$$\mathbf{y}[n] = \phi(\mathbf{W'}\mathbf{d}[n] + \boldsymbol{\beta'}), \tag{3.18}$$

where $\boldsymbol{\beta'}$ and $\mathbf{W'}$ are the biases and trainable weights for the decoder.

The here proposed DCAEs are fed with the mixture input signal $\mathbf{X}$ containing F frequency bins and N time frames, which provide time context to the network. The input feature map will now start reducing its size in the encoder part, which is composed of repetitions of convolutional layers and pooling layers, generating a code in a fully connected layer, as shown in Fig. 3.4. The convolutional layers consist of a set of filters that extract features from their input layers, the pooling is chosen to be max-pooling [Scherer, Müller, and Behnke 2010]. Max-pooling does the downsampling of the latent representation by a constant factor by taking the maximum value within a certain scope of the mapping space and generates a new mapping space with a reduced dimension. The fully connected layer behaves like the layers found in a typical MLP. The decoder part consists of repetitions of deconvolution and unpooling layers, providing a nonlinearity layer at the output. In this work, the data, as well as the performed operations are two-dimensional. This means that the convolution of r $A \times B$ filters over an input will produce r 2D outputs. The (j, k) element obtained after the operation is given by:

$$a_{j,k} = \phi\Big(\beta + \sum_{l=1}^{A}\sum_{m=1}^{B} w_{l,m} x_{j+l,k+m}\Big). \tag{3.19}$$

In other words, the convolutional layer computes the activation of a feature of dimensions $A \times B$ across different regions of the input layer. The mapping from the input layer to the convolutional layer is often called a feature map and the shared weights and the bias unit are termed as the kernel. Since each of these kernels is detecting one feature over the input layer, the convolutional layer usually includes more than one kernel or feature map. CNNs have extensively been used for image classification including the MNIST handwritten digit set [Krizhevsky, Sutskever, and Hinton 2012]. One of the biggest advantages of using CNNs is that memory and resources required are lower than those used by a regular fully connected NN, as the weights and biases across hidden layers are shared.

The spatial distribution of the convolving operations is determined by a parameter called stride specifying how much the filter (AKA kernel) slides in space between each operation.

After convolving the whole input, pooling layers are introduced, by eliminating non-maximal values, it reduces computation for upper layers. The different layers are described using their hyper-parameters. The convolutional layers are characterized by specifying the number of kernels r, its shape $A \times B$ and its stride $S_1 \times S_2$. The pooling will only be described by its size $p_1 \times p_2$ as the pooling method will always be max-pool. With this notation, the first element of any parameters' shape will cover the frequency dimension and the second element will cover the temporal dimension. Finally, the fully connected layers are characterized by the number of units they contain. In this research two different DCAEs have been trained:

- DCAE1: This architecture consists of 2 convolutional layers, a pooling layer and a fully connected layer acting as a bottleneck. Table 3.1 shows the numeric values of each hyper parameter.

  The idea behind the shape of the first kernel is to focus only on the timbre properties of the input features, while the second convolution takes care of the dynamical behavior of the signal by introducing a time context of 20 frames after the first convolution.

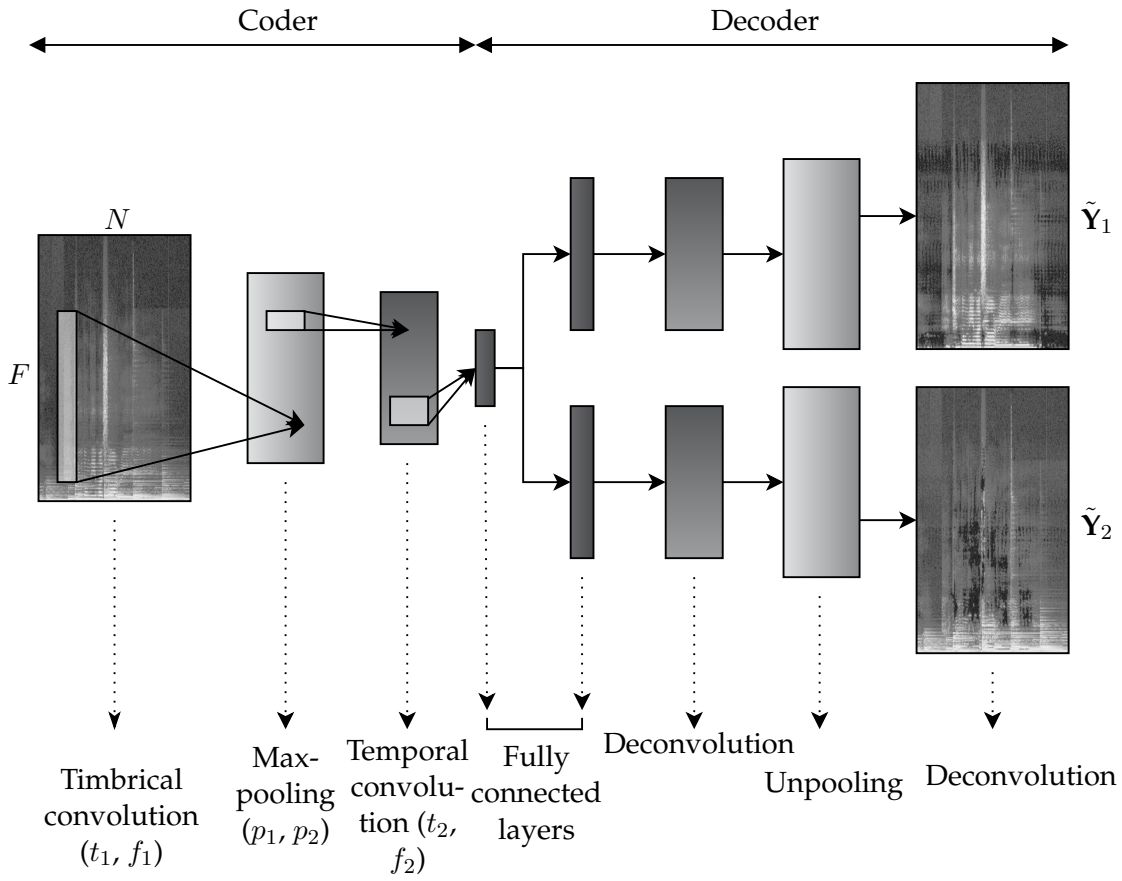

FIGURE 3.4: DCAE1 architecture for audio SS.

TABLE 3.1: DCAE1 Topology

Type	*Kernel*	*Stride*	*Number of Filters*
Conv	1×50	1×3	50
Pool	1×2	-	-
Conv	10×20	1×1	60
Fully Connected	-	-	256
Number of parameters: 206,839,778			

- DCAE2: This architecture consists of only one convolutional layer and a fully connected layer, as shown in Table 3.2. In this case, the architecture provides only one convolution, where timbre and temporal feature extraction is performed.

The input **X** of the network, given by (3.3), is obtained by stacking T frames of 1024 frequency bins with a 1024 sample window and 50% overlap. The parameter T refers to the time context in which the network is working and in this research has been set to 30 frames based on observations on the algorithms' performance.

The networks were trained by updating the weights using mini-batch gradient descent (MBSGD) with the ADADELTA algorithm [Zeiler 2012] to minimize a slightly modified version of (3.8):

$$J_C = 2\alpha L_1 + \alpha L_2 - \gamma L_{12} - \gamma L_{21}, \tag{3.20}$$

TABLE 3.2: DCAE2 Topology

Type	*Kernel*	*Stride*	*Number of Filters*
Conv	20×5	1×1	5
Fully Connected	-	-	64
Number of parameters: 20,798,346			

where,

$$
\begin{aligned}
L_1 &= \|\mathbf{Y}_1 - \tilde{\mathbf{Y}}_1\|_2^2, \\
L_2 &= \|\mathbf{Y}_2 - \tilde{\mathbf{Y}}_2\|_2^2, \\
L_{12} &= \|\mathbf{Y}_1 - \tilde{\mathbf{Y}}_2\|_2^2, \\
L_{21} &= \|\mathbf{Y}_2 - \tilde{\mathbf{Y}}_1\|_2^2.
\end{aligned} \tag{3.21}
$$

In this study $\alpha = 0.09$ and $\gamma = 0.05$. The batch size has been set to 20 training samples and iterated through 100 epochs using a NVIDIA TESLA k40 as an accelerated processor unit.

During this study, several objective experiments regarding DCAEs were conducted in order to select the appropriate architectures to be tested with subjects. The results provided by these experiments are presented in Appendix C.

3.2 Evaluation

This section presents the evaluation methods carried out during this study. These will involve objective evaluation methods based on existing standards and perceptual experiments. The evaluation aims at three different goals:

- Assess each SS algorithm's performance.
- See the relation between the obtained objective results with the subjective results.
- Evaluate the trade-off between generalization capacity and complexity for each algorithm.

The last goal has to do with a very important problem involved in deep learning techniques which is the generalization. This phenomenon is investigated in this thesis by proposing two evaluation scenarios:

- Ideal case: test set belongs to the training data set's same library.
- General case: test set will not belong to the training data set.

These two scenarios will illustrate how the different multi-track estimation strategies generalize.

3.2.1 Objective Evaluation

The SS quality is quantified by the Source-to-Distortion Ratio (SDR), the Source-to-Artifacts Ratio (SAR), and the Source-to-Interference Ratio (SIR). The computation of these values requires the assumption that the introduced degradation is a combination of errors coming from noise, artifact and interference factors:

$$e_{total}(t) = e_{noise}(t) + e_{artifacts}(t) + e_{interference}(t), \tag{3.22}$$

where $e_{interference}(t)$, $e_{noise}(t)$, $e_{artifacts}(t)$ are the interferences, noise, and artifacts error terms, respectively. Now the estimated source can decomposed as a combination of the original source $s_{target}(t)$ and the introduced errors:

$$\tilde{s}_{target}(t) = s_{target}(t) + e_{total}(t). \tag{3.23}$$

These error terms are computed by using the BSS Eval Toolbox [Vincent, Gribonval, and Févotte 2006]. The way the original signal is decomposed in these four terms is described in the following lines.

Let us denote in the following $\langle a, b\rangle = \sum_{t=0}^{T-1} a(t)\bar{b}(t)$ the inner product between two possibly complex-valued signals a and b of length T, where $\bar{b}$ is the complex conjugate of b, and $\|a\|^2 := \langle a, a\rangle$ the energy of a.

When **A** is a time-invariant instantaneous matrix and when the mixture is separated by applying a time-invariant instantaneous matrix **W**, $\hat{s}_j$ can be decomposed as:

$$\hat{s}_j = (\mathbf{WA})_{jj}s_j + \sum_{j'\neq j}(\mathbf{WA})_{jj'}s_{j'} + \sum_{i=1}^{m} W_{ji}n_i. \tag{3.24}$$

Since $(\mathbf{WA})_{jj}$ is a time-invariant gain, it seems natural to identify the three terms of this sum with s_{target}, e_{interf} and e_{noise} respectively (e_{artif} = 0 here). However, equation 3.24 cannot be used as a definition of s_{target}, e_{interf}, e_{noise} and e_{artif} since the mixing and demixing systems are unknown. Also the two first terms of 3.24 may not be perceived as separate sound objects when a nonwanted source $s_{j'}$ is highly correlated with the wanted source s_j.

Instead, the proposed decomposition is based on orthogonal projections. Let us denote $\Pi\{y_1 \cdots, y_k\}$ the orthogonal projector onto the subspace spanned by the vectors $\{y_1 \cdots, y_k\}$. The projector is a $T \times T$ matrix, where T is the length of these vectors. Considering the three orthogonal projectors

$$P_{s_j} := \Pi\{s_j\}, \tag{3.25}$$
$$P_s := \Pi\{(s_{j'})_{1\leq j'\leq n}\}, \tag{3.26}$$
$$P_{s,n} := \Pi\{(s_{j'})_{1\leq j'\leq n}, (n_i)_{1\leq i\leq m}\}. \tag{3.27}$$

And decomposing them $\hat{s}_j$ as the sum of the four terms

$$s_{target} := P_{s_j}\hat{s}_j, \tag{3.28}$$
$$e_{interf} := P_s\hat{s}_j - P_{s_j}\hat{s}_j, \tag{3.29}$$
$$e_{noise} := P_{s,n}\hat{s}_j - P_s\hat{s}_j, \tag{3.30}$$
$$e_{artif} : \hat{s}_j - P_{s,n}\hat{s}_j. \tag{3.31}$$

The computation of s_{target} is straightforward since it involves only a simple inner product: $s_{target} = \langle\hat{s}_j, s_j\rangle s_j/\|s_j\|^2$. The computation of e_{interf} is a bit more complex. If the sources are mutually orthogonal, then $e_{interf} = \sum_{j\neq j'}\langle\hat{s}_j, s_{j'}\rangle s_{j'}/\|s_{j'}\|^2$. Otherwise, using a vector c of coefficients such that $P_s\hat{s}_j = \sum_{j'=1}^{n}\bar{c}_{jj'}s_{j'} = \mathbf{c}^H s$ where $(\cdot)^H$ denotes Hermitian transposition, then $c = \mathbf{R}_{ss}^{-1}[\langle\hat{s}_j, s_{j'}\rangle, \cdots, \langle\hat{s}_j, s_n\rangle]^H$ where $\mathbf{R}_{ss}$ is the Gram matrix of the sources defined by $(\mathbf{R}_{ss})_{jj'} = \langle s_j, s_{j'}\rangle$. The computation of $P_{s,n}$ proceeds in a similar fashion, however most of the time it can be assumed that the noise signals are mutually orthogonal and orthogonal to each source, so that $P_{s,n}\hat{s}_j \approx P_s\hat{s}_j + \sum_{i=1}^{m}\langle\hat{s}_j, n_i\rangle n_i/\|n_i\|^2$.

Starting from the decomposition of s_j, one can now define numerical performance criteria by computing energy ratios as follows:

$$SDR = 10\log_{10}\frac{\|s_{target}(t)\|^2}{\|e_{total}(t)\|^2}, \tag{3.32}$$
$$SIR = 10\log_{10}\frac{\|s_{target}(t)\|^2}{\|e_{interf}(t)\|^2}, \tag{3.33}$$
$$SAR = 10\log_{10}\frac{\|s_{target}(t) + e_{inter}(t) + e_{noise}(t)\|^2}{\|e_{artifacts}\|^2}. \tag{3.34}$$

All these values are expressed in dBs and are defined as their mean value weighted by the length of the evaluated music segment. Higher SDR/SAR/SIR values will result in a better separation quality.

These four measures are inspired by the usual definition of the SNR, with a few modifications. For instance the definition of the SNR involving the term $s_{target}+e_{interf}$ at the numerator aims at making it independent of the SIR. Indeed consider the case of an instantaneous noisy 2×2 mixture where $\hat{s}_1 = \epsilon s_1 + s_2 + e_{noise}$ with $\|\epsilon s_1\| \ll \|s_2\|$ and $\|e_{noise}\| \approx \|\epsilon s_1\|$. Then $\hat{s}_1$ is perceived as dominated by the interfering signal, with the noise energy making an insignificant contribution. This is consistent with $SIR \approx -\infty$ and $SNR \approx \infty$ using the previous definitions. A SNR defined by $10log_{10}(\|s_{target}\|^2/\|s_{noise}\|^2)$ would give $SNR \approx 0$ instead. Similarly, the SAR is independent of the SIR and the SNR since the numerator in 3.34 includes the interferences and noise terms as well.

Note that the numerical precision of the measures is lower for high-performance values than for low ones. For example, a high SAR means that the denominator in 3.34 is very small, so that small constant-amplitude errors in s_{target} (due to signal quantization) result in large SDR deviations. In particular, when the signals corresponding to sound files, the precision of the results depends on the number of bits per sample.

3.2.2 Subjective Evaluation

The audio was played through a speaker (Genelec 8240 APM) while the listener was sitting at a distance of 1 m from it in an acoustically treated room. The sound pressure level was set to 60 dB(A) at that distance.

Subjects (see Appendix A) had no time limits in order to finish the different tests and were allowed to have a break whenever they desired. After the test, subjects were asked to fill a form providing information about their musical background or knowledge and music genre preference.

- *Pairwise comparison*: This experiment was designed to determine how susceptible CI recipients are to the degradation of the songs after the SS process. This was achieved by comparing processed and unprocessed music excerpts in pairs. The aim of this perceptual experiment was to provide a benchmark which related the observed algorithm preference on CI subjects as a function of the measured distortions [Pons et al. 2016]. The remix pre-set was set to a VIR of 6 dB as it has been shown that is the one preferred by CI users [Pons et al. 2016].

TABLE 3.3: Pairwise Comparison Pairs.

Pairwise Comparison		
VMR	Mixing Condition	
	Song A	Song B
6 dB	Original MT	NMF
	DCAE2	DRNN2
	DCAE1	DRNN1

Pairs shown in Table 3.3 were randomly presented with a VIR of 6 dB three times for each of the songs listed in Table 3.4 (6 songs x 3 repetitions x 3 comparisons = 54 pairs). Each excerpt has a duration of 5 seconds and can be played multiple times.

TABLE 3.4: Music tracks used in the pairwise comparison.

ID	*Song*
161V	10161_verse (iKala)
171V	10171_ verse (iKala)
174C	10174_chorus (iKala)
Hall	Hallelujah (Leonard Cohen)
Jude	Hey Jude (Excerpt 2) (The Beatles)
Mic	Michel (Anouk)

Each subject was asked to choose the most enjoyable song (A or B) through the interface shown in Fig. 3.5. When both songs were equally enjoyable, participants were asked to find any detail that would allow them to decide whether they prefer A or B. If they were not capable of finding it, they were asked to guess. A and B were the same song with the same mixing configuration but one of them was obtained by summing the two estimated sources provided by the SS algorithm while the other was obtained by mixing the original multitracks. The order of the mix assigned to song A or B was randomized. The estimated multi-tracks were obtained applying NMF as the SS algorithm.

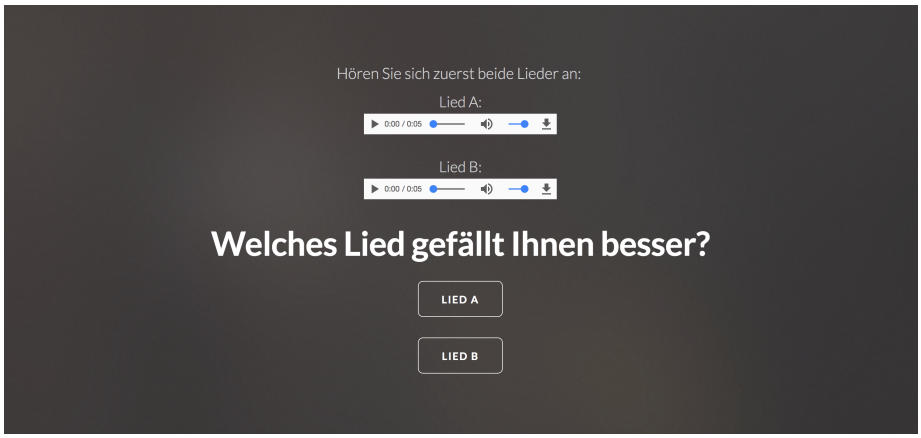

FIGURE 3.5: Pairwise comparison GUI. Subjects can listen and choose which song do they enjoy most. GUI adapted from [Pons et al. 2016].

- *MUSHRA*: This experiment was designed to obtain a relative SS separation quality score with respect to the original mix. For this test, 6 different 5-second song excerpts have been tested; 50% from the iKala database and the other 50% from the music material used in [Buyens et al. 2014]. Table 3.5 shows the used song excerpts which were chosen to cover a wide range of audio SS distortions. Each song is presented once with a VIR of 6 dB, making a total of 6 MUSHRA tests.

TABLE 3.5: Music tracks used in the MUSHRA test.

ID	*Song*
161C	10161_chorus (iKala)
171C	10171_chorus (iKala)
174V	10174_verse (iKala)
Jud	Hey Jude (Excerpt 1) (Leonard Cohen)
Jude	Hey Jude (Excerpt 2) (The Beatles)
Mic	Michel (Anouk)

In each test, the subject is presented with eight versions of an excerpt; seven to be evaluated and the reference. They were first asked to listen to the reference (target) and proceed by addressing the global quality of the other seven by moving a slider which assigned a score between 0 and 100 to each song. They were imposed to set at least one of the sliders to 100. The seven tested songs include a hidden reference and an anchor. The other five songs correspond to mixes after performing SS with each of the algorithms investigated in this thesis. The interface presented (see Fig. 3.6) to the subjects can be found online[2].

The reference corresponds to the original multi-track mixed with a 6 dB VIR. The anchor was created by low-pass filtering the target reference signal with a linear phase FIR filter. The filter was designed imposing 3.5 kHz passband edge frequency, 0.01 dB peak to peak ripple, 80 dB stop-band attenuation and setting 20% of the obtained coefficients to zero.

[2] http://c4dm.eecs.qmul.ac.uk/downloads/index.html#mushram

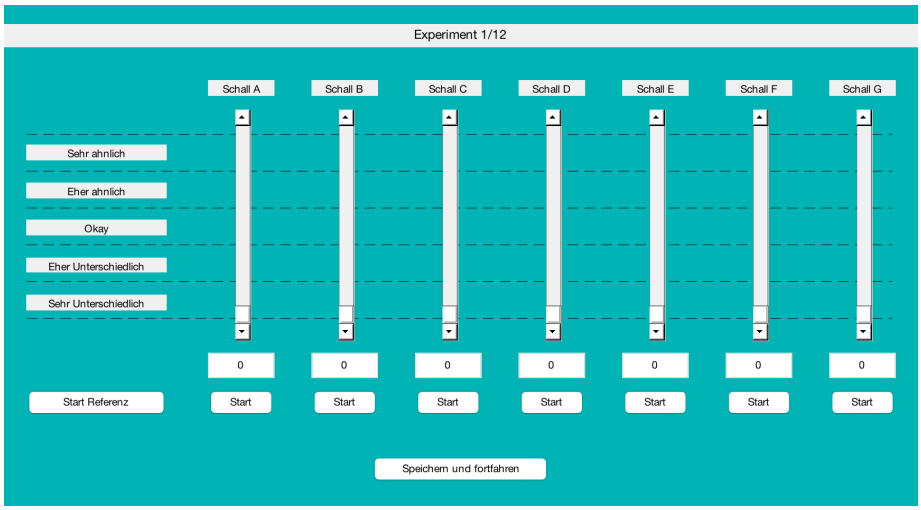

FIGURE 3.6: MUSHRA GUI. Subjects can listen and rate the quality of each sing excerpt. GUI adapted from MUSHRAM[2] (*Queen Mary University of London*).

For more details on the experiment's settings, the reader should refer to Appendix B.

3.3 Statistical Analysis

3.3.1 Chi-squared Test

The Chi-Square (χ^2) test of Independence is used to determine if there is a significant relationship between two nominal (categorical) variables. The frequency of one nominal variable is compared with different values of the second nominal variable. The data can be displayed in an $R \times C$ contingency table, where R denotes the row and C is the column.

Suppose that n observations in a random sample from a population are classified into k mutually exclusive classes with respective observed numbers x_i ($\forall\, i = 1, \cdots, k$), and a null hypothesis gives the probability p_i that an observation falls into the ith class. So we have the expected numbers $m_i = n \cdot p_i$, where

$$\sum_{i}^{k} p_i = 1, \tag{3.35}$$

$$\sum_{i}^{k} m_i = \sum_{i}^{k} x_i = n. \tag{3.36}$$

[Pearson 1900] proposed that, under the circumstance of the null hypothesis being correct, as $n \to \infty$ the limiting distribution of the quantity, which is given below, is the χ^2 distribution.

$$X^2 = \sum_{i=1}^{k} \frac{(x_i - m_i)^2}{m_i} = \sum_{i=1}^{k} \frac{x_i^2}{m_i} - n. \tag{3.37}$$

3.3.2 ANOVA

The one-way analysis of variance (ANOVA) is used to determine whether there are any statistically significant differences between the means of three or more independent (unrelated) groups.

ANOVA compares the means between the groups you are interested in and determines whether any of those means are statistically significantly different from each other. Specifically, it tests the null hypothesis:

$$H_o : \mu_1 = \mu_2 = \mu_3 = \cdots = \mu_k, \tag{3.38}$$

where μ is the group mean and k the number of groups. If, however, the one-way ANOVA returns a statistically significant result, the alternative hypothesis (H_A) is accepted, which is that there are at least two group means that are statistically significantly different from each other.

It is important to realize that the one-way ANOVA is an omnibus test statistic and cannot tell you which specific groups were statistically significantly different from each other, only that at least two groups were. To determine which specific groups differed from each other, you need to use a post hoc test.

3.3.3 Post hoc Test

Post hoc tests are designed for situations in which the researcher has already obtained a significant omnibus F-test with a factor that consists of three or more means and additional exploration of the differences among means is needed to provide specific information on which means are significantly different from each other.

During this work, the Bonferroni correction will be used as the post hoc test. The Bonferroni correction compensates for that increase by testing each individual hypothesis at a significance level of α/m, where α is the desired overall alpha level and m is the number of hypotheses.

The Bonferroni correction can be interpreted as being the correction that makes the probability of incorrectly rejecting the null hypothesis in at least one of n n independent tests to be the chosen overall significance threshold. Indeed, given independence this probability is $\alpha_{all} = 1 - (1 - \alpha_{one})^n$, which for $n \times \alpha_{one} \ll 1$ can be simplified to $\alpha_{all} \approx n \times \alpha_{one}$.

Let $H_1, \cdots, H_m$ be a family of hypotheses and $p_1, \cdots, p_m$ their corresponding p-values. Let m be the total number of null hypotheses and m_0 the number of true null hypotheses. The familywise error rate (FWER) is the probability of rejecting at least one true H_i, that is, of making at least one type I error. The Bonferroni correction rejects the null hypothesis for each $p_i \leq \alpha/m$ (where α is the error probability and in

this study is set to 0.05) , thereby controlling the FWER at $\leq \alpha$. Proof of this control follows from Boole's inequality, as follows:

$$\text{FWER} = P\left\{\bigcup_{i=1}^{m_0}\left(p_i \leq \frac{\alpha}{m}\right)\right\} \leq \sum_{i=1}^{m_0}\left\{P\left(p_i \leq \frac{\alpha}{m}\right)\right\} = m_0\frac{\alpha}{m} \leq m\frac{\alpha}{m} = \alpha. \quad (3.39)$$

This control does not require any assumptions about dependence among the p-values or about how many of the null hypotheses are true.

The Bonferroni correction can be used to adjust confidence intervals. If one establishes m confidence intervals and wishes to have an overall confidence level of $1-\alpha$, each individual confidence interval can be adjusted to the level of $1 - \alpha/m$.

3.4 Audio Material

Two sets of music data are used in this study; one set for training and testing of the SS algorithms and the other just for testing.

3.4.1 Training and Testing Data Set

The iKala Chan et al. 2015 dataset[3], which is composed of 252 30-second excerpts sampled from 206 iKala songs, is used in the training stage. The music accompaniment and the singing voice are recorded on the left and right channels respectively, a convenient format for training the network. 50% of this dataset was dedicated to the training of the network.

3.4.2 Test Data Set

The music excerpt set selected for the testing is conformed by the songs shared kindly by Buyens et al. 2014 or by the remaining 50% of the iKala dataset.

All songs were matched in loudness by applying the corresponding gain (computed with the ReplayGain[4]) leaving the original track mix intact.

[3] http://mac.citi.sinica.edu.tw/ikala/

[4] https://hydrogenaud.io/index.php/topic,85536.msg736023.html#msg736023

Chapter 4

Results

This chapter covers the quantitative results provided by the objective measurements and by the designed perceptual experiments.

4.1 Objective Results

4.1.1 Ideal Case

Fig. 4.1 presents the global SDR, SAR and SIR values averaged across the testing set from the ideal case, which is formed by 126 iKala song excerpts. Both DCAEs show the best mean values. The other three algorithms, on the other hand, show similar performance, but lower scores than the DCAEs.

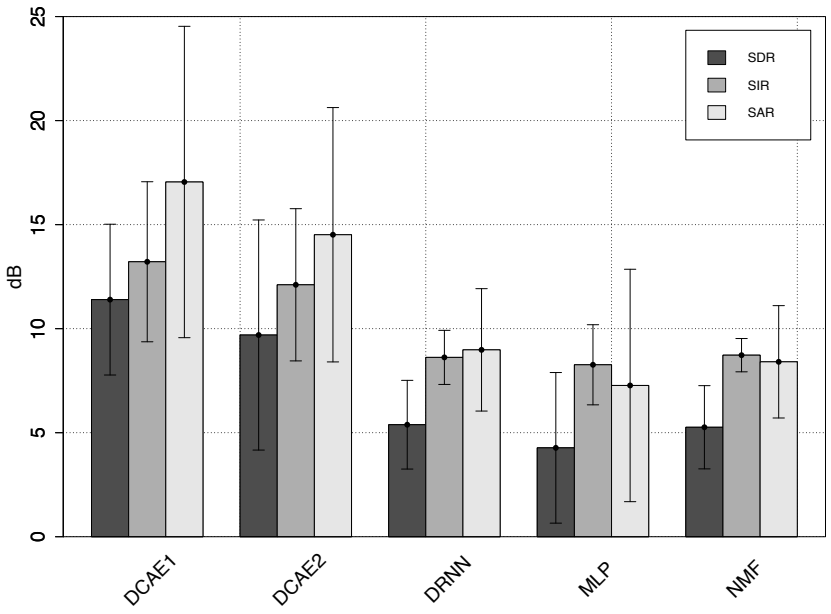

FIGURE 4.1: Mean SDR, SIR and SAR across songs for the ideal case scenario. Error bars indicate the standard deviation.

Table 4.1 shows the numerical values corresponding to the bars shown in Fig. 4.1.

TABLE 4.1: Mean and standard deviation for the SDR, SIR and SAR values expressed in dB corresponding to the implemented algorithms in the ideal case scenario.

Ideal Case			
Algorithm	SDR	SIR	SAR
DCAE1	11.39 ± 3.62	17.05 ± 4.68	13.21 ± 3.21
DCAE2	9.70 ± 4.33	14.51 ± 5.45	12.11 ± 3.53
DRNN	5.38 ± 2.13	8.98 ± 2.94	8.62 ± 1.30
MLP	4.27 ± 2.19	2.27 ± 2.62	8.26 ± 1.62
NMF	5.26 ± 2.3	8.41 ± 0.9	8.73 ± 3.5

The performance of both DCAE1 and DCAE2, as measured by the normalized SDR (NSDR), is similar to the best scores obtained in the *MIREX 2016 singing voice separation challenge*[1] [Chandna et al. 2017] (see Table 4.2).

TABLE 4.2: Mean values of Normalized SDR expressed in dB obtained from the iKala test set.

	NSDR	
Architecture	*Target (Vocals)*	*Target (Music)*
DCAE1	**6.911**	**11.133**
DCAE2	5.210	9.431
P. Chanda et al., 2016	5.2891	9.668

4.1.2 General Case

Fig. 4.2 shows the mean global SDR, SAR and SIR values for each algorithm trained with the iKala but now tested using the tracks provided by [Buyens et al. 2014]. This case shows generalization problems for the two DCAEs and NMF. However, the results of the DRNN and MLP are consistent with the ideal case.

Table 4.3 shows the numerical values corresponding to the bars shown in Fig. 4.2.

TABLE 4.3: Mean and standard deviation for the SDR, SIR and SAR values expressed in dB corresponding to the implemented algorithms in the general case scenario.

General Case			
Algorithm	SDR	SIR	SAR
DCAE1	-1.53 ± 6.27	4.43 ± 7.48	2.60 ± 3.85
DCAE2	-3.32 ± 5.53	1.09 ± 6.11	2.46 ± 3.66
DRNN	5.27 ± 4.13	9.81 ± 5.80	9.73 ± 2.01
MLP	4.43 ± 3.01	6.54 ± 3.70	9.99 ± 1.60
NMF	-5.30 ± 2.3	3.22 ± 0.9	-2.96 ± 3.5

The values achieved by the DRNN and MLP are comparable with the ones obtained in previous research [Pons et al. 2016].

[1] http://www.music-ir.org/mirex/wiki/2016:Singing_Voice_Separation_Results

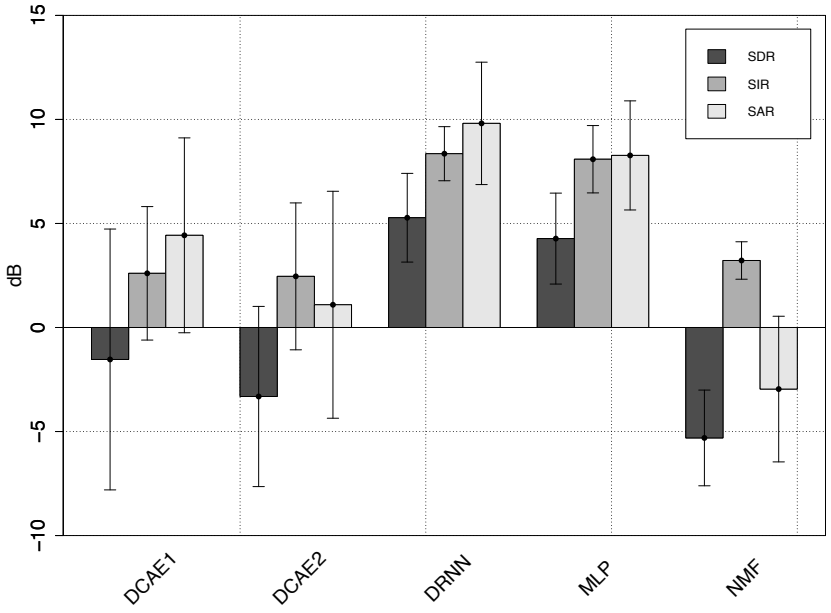

FIGURE 4.2: Mean SDR, SIR and SAR across songs for the general case scenario. Error bars indicate the standard deviation.

4.2 Subjective Results

4.2.1 Pairwise Comparison

Fig. 4.3 presents the preference rating percentage and the corresponding SDR value (see Table 4.4) of each excerpt described in Table 3.4. The green shaded area corresponds to the ratings having a $p > 0.05$ as estimated from the chi-squared test (with H_0: equal preference for both conditions). The black line corresponds to a fitted linear model which intersects with the minimum value of the non-preference area. This intersection point is indicated by the horizontal red dotted line and provides the distortion above which CI users present no significant preference between the original and the processed multi-track, i.e. SDR=0.14 dB.

For the data analysis, the χ^2 test was used to determine whether the preference for the pre-sets proposed for CI users changed when using SS or not. The test was applied as a goodness of fit test where the H_0 was set to be the preference percentage distribution of Original MT vs SS. The χ^2 test measured how well the preference percentage distribution of SS mix vs Original MT fitted the H_0 distribution. This test provides a measure of how much distortions and artifacts in the estimated SS influenced the user mixing preferences.

Fig. 4.4 shows the preference rating for SS as a function of the achieved SAR value (see Table 4.4). This figure shows the level of introduced artifacts above which CI users present no clear preference between the original multi-track and the processed one. This value corresponds to SAR=3.63 dB.

A significance test for each linear regression model was performed in order to quantify how well a linear fit represents the data. The SDR model provides a $p = 0.027$ and a $R^2 = 0.6114$ (top left of Fig. 4.3), the SAR model gives a $p = 0.018$ and a

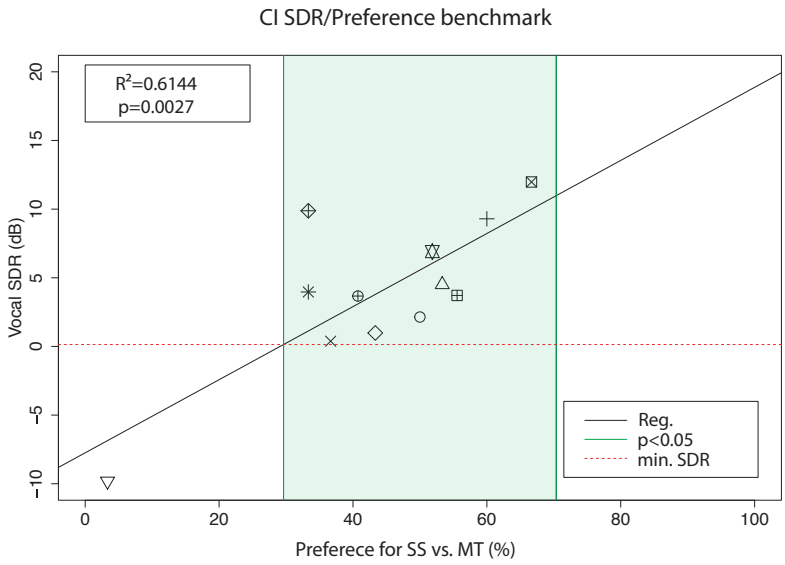

FIGURE 4.3: Results facing the SDR objective measures with the preference ratings for the first condition (see Table 4.4).

TABLE 4.4: SDR and SAR values expressed in dB corresponding to the benchmark.

ID	*SDR*	*SAR*
10161V	3.76	2.97
10171V	3.53	8.98
10174C	10.48	12.03
Jud	0.98	6.44
Jude	-9.81	-3.8
Mic	0.38	7.17
Hall [Pons et al. 2016]	11.97	12.04
Befo [Pons et al. 2016]	3.96	4.03
Mic2 [Pons et al. 2016]	9.88	9.90
Jud2 [Pons et al. 2016]	3.67	3.68
Jude [Pons et al. 2016]	6.91	7.20
Dock [Pons et al. 2016]	3.71	4.05

$R^2 = 0.6033$ (top left of Fig. 4.4). It can be observed that there is a significant correlation between the SDR/SAR and the preference ratings, complementing and extending thus the benchmark proposed by previous research [Pons et al. 2016]. Data from [Pons et al. 2016] (last 6 songs in Table 4.4) has been replotted together with the collected data during the present study. Increasing the amount of data allows us to improve the benchmark for CI users.

the SIR is disregarded, as this study is dealing with the remixing of the different sources, and therefore, the interference coming from other sources mutually combined are difficult to perceive.

Figure 4.5 shows the score preference between the mixes separated by DCAE1 vs. DRNN (Figure 4.5 a)) and by DCAE2 vs. MLP (Fig. 4.5 b)). The confidence region ($\alpha < 0.05$) is defined by the regions outside the area defined by the two green horizontal lines. This plot shows a significant preference for both DRNN (prefer-

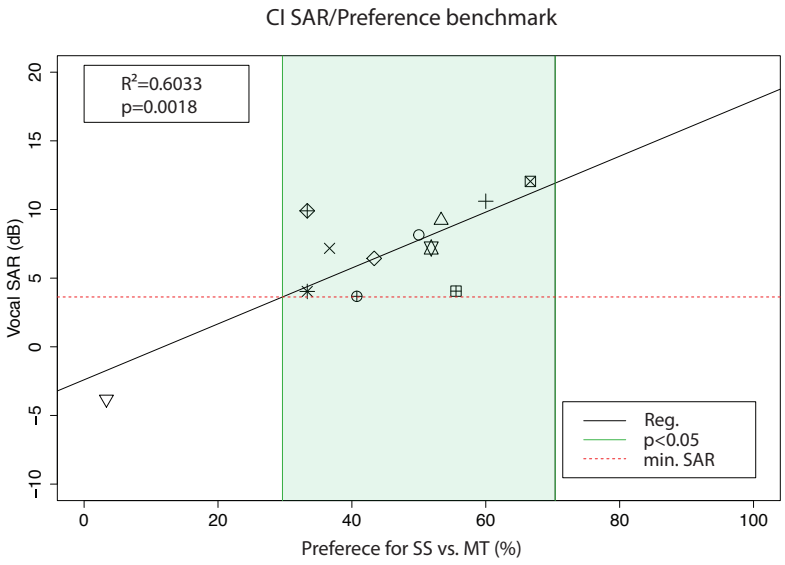

FIGURE 4.4: Results facing the SAR objective measures with the preference ratings for the first condition (see Table 4.4).

ence=74.92%) and MLP (preference=90%) for the song Jude, which is the only one presenting negative SDR and SAR for both DCAEs. It also shows significant preference for *Jud* in Fig. 4.5 a) and for *174C* in Fig. 4.5 b) for both DCAEs. These will be not discussed since the lower bound of the distortions and artifacts tolerance are of interest.

This part of the pairwise comparison serves as a confirmation of the benchmark in most of the tested cases. However, there are two situations where the benchmark fails to predict the preference scores; for the excerpt *10174C* when comparing DCAE2 vs. MLP where no clear preference should be obtained, a clear preference for the DCAE2 is observed, which can be due to the large difference between the objective values provided by the algorithms. Furthermore, for the excerpt *Jud*, a significant preference for the MLP is expected but not observed.

TABLE 4.5: SDR and SAR values expressed in dB corresponding to the DCAE1 and DRNN next to the preference ratings.

	DCAE1		DRNN		DCAE1 vs. DRNN
	SDR	SAR	SDR	SAR	% Preference
1161V	9.48	10.67	7.47	10.09	43.3% - 56.6%
1171V	7.59	9.28	2.74	6.57	56.6% - 43.3%
1174C	16.34	18.05	3.70	8.53	66.6% - 33.3%
Jud	1.60	***2.77***	5.31	6.38	73.3% - 26.6%
Jude	***-3.57***	***-2.05***	9.94	10.94	***10.0%*** - ***90.0%***
Mic	4.56	5.45	9.35	11.94	46.6% - 53.3%

Tables 4.5 and 4.6 show the preference rating next to the physical measures (SDR and SAR). The objective values which do not meet with the minimum recommended values by the benchmark are highlighted in red. The statistically significant preference scores are also highlighted in color indicating the preferred algorithm preference percentage in green and the nonpreferred algorithm percentage in red.

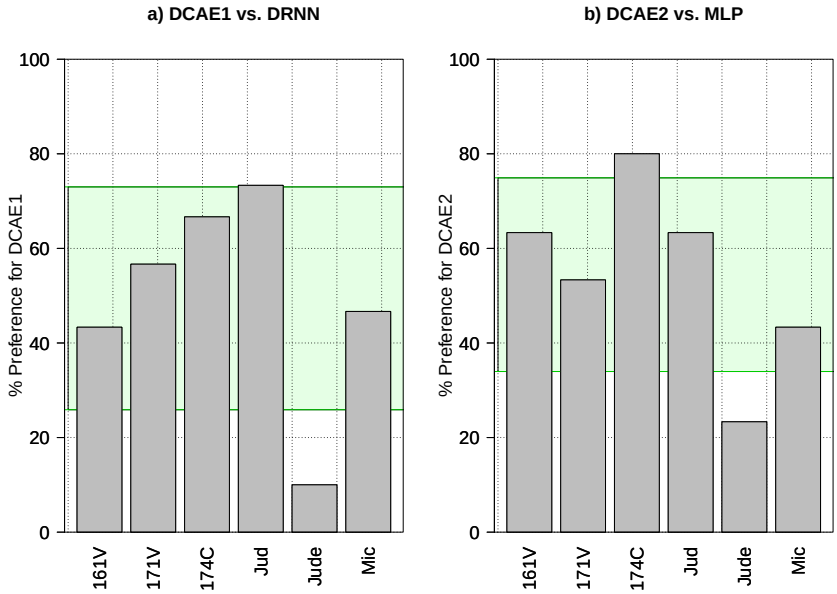

FIGURE 4.5: NN preference score for each tested song.

TABLE 4.6: SDR and SAR values expressed in dB corresponding to the DCAE2 and MLP next to the preference ratings.

	DCAE2		MLP		DCAE2 vs. MLP
	SDR	SAR	SDR	SAR	% Preference
1161V	11.07	11.88	6.53	10.29	63.3% - 36.6%
1171V	9.91	11.74	1.57	5.87	53.3% - 46.6%
1174C	17.25	19.11	2.96	8.31	***80.0%*** - ***20.0%***
Jud	***-2.20***	***0.50***	5.03	6.50	63.3% - 36.6%
Jude	***-6.63***	***-3.24***	8.30	9.27	***23.3%*** - ***76.6%***
Mic	1.80	***3.78***	7.84	10.64	43.3% - 56.6%

In summary, the benchmark provides a distortion-artifact working point threshold below which the SS degrades the signal to a point where CI users perceive it; (SDR, SAR) $\leq$ (0.14, 3.63)dB.

4.2.2 MUSHRA

For this test, a robust statistical analysis was adopted to minimize the potential effects of outliers and non-normality. Means were replaced with the more robust trimmed means, which are calculated by rank ordering the data, removing the highest and lowest values, and calculating the mean of the remaining values. In this study, the mean is trimmed by 25% due to the high variability CI users present [Aronoff et al. 2011].

The significance of the collected MUSHRA scores is determined by applying the Analysis of Variance (ANOVA) model. The result is considered significant if p is less than 0.05.

Fig. 4.6 presents the trimmed mean MUSHRA scores obtained from the evaluation of the songs belonging to the iKala dataset.

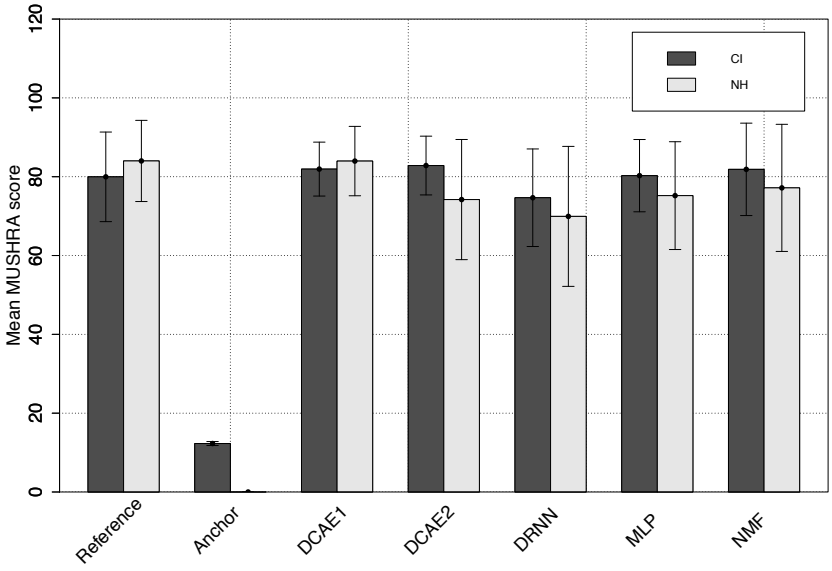

FIGURE 4.6: Trimmed mean values for the MUSHRA scores from NH and CI users for the ideal case scenario. Error bars indicate the standard deviation.

TABLE 4.7: Individual and mean SDR and SAR values expressed in dB corresponding to songs tested with the MUSHRA for the ideal case scenario.

	DCAE1		*DCAE2*		*DRNN1*		*MLP*		*NMF*	
ID	SDR	SAR	SDR	SAR	SDR	SAR	SDR	SAR	SDR	SAR
1161V	11.06	11.88	9.48	10.67	7.47	10.08	6.53	10.29	3.77	9.30
1171V	9.91	11.74	7.59	9.23	2.74	6.58	1.57	5.88	3.54	8.00
1174C	17.26	19.12	16.35	18.06	3.70	8.53	2.96	8.31	10.48	12.03
Mean	**12.74**	**14.24**	**11.14**	**12.67**	**4.64**	**8.40**	**3.69**	**8.16**	**5.93**	**9.77**

A correlation between the mean objective values (see Table 4.7) and the obtained subjective results can be observed in most of the algorithms for both CI and NH subjects. However, there is a disagreement in the case of the DRNN and the MLP.

To substantiate confidence on the results, an ANOVA test with factor "SS algorithm" was performed. R-studio [RStudio Team 2015] statistics was used with a significance level $p = 0.05$. No significant effect of algorithm on MUSHRA scores was found neither for NH ($F(1,4) = 1.788, p = 0.134$) nor for CI users ($F(1,4) = 0.483, p = 0.748$). As a result, the factor did not have a significant influence on the subjective scores, as expected from experiment 1, since all music mixes presented positive SDRs and SARs, i.e wherein the none significance region of the proposed benchmark.

Fig. 4.7 presents the trimmed mean MUSHRA scores obtained from the evaluation of the songs corresponding to the general case scenario set.

A correlation between the mean objective SDR and SAR values presented in Table 4.8 and the obtained subjective scores can be observed in the general case scenario.

In this case, the ANOVA analysis revealed a significant effect on MUSHRA scores

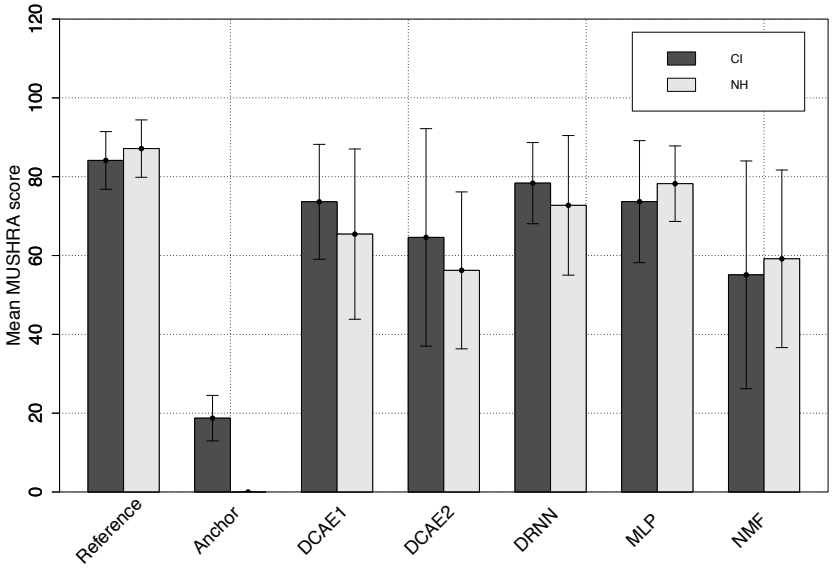

FIGURE 4.7: Trimmed mean values for MUSHRA scores from NH and CI users for the general case scenario. Error bars indicate the standard deviation.

TABLE 4.8: Individual and mean SDR and SAR values expressed in dB corresponding to songs tested with the MUSHRA for the general case scenario.

	DCAE1		*DCAE2*		*DRNN1*		*MLP*		*NMF*	
ID	SDR	SAR	SDR	SAR	SDR	SAR	SDR	SAR	SDR	SAR
Hall	2.60	7.52	0.94	7.04	4.20	8.84	2.53	9.30	-0.07	3.30
Jude	-3.56	-2.05	-6.63	-3.24	9.94	10.94	8.30	9.27	-9.81	-3.81
Mic	4.56	5.45	1.80	3.78	9.35	11.94	7.84	10.63	0.38	7.17
Mean	**0.86**	**2.05**	**-2.35**	**0.34**	**8.20**	**9.75**	**7.06**	**8.80**	**-2.82**	**3.27**

for NH subjects ($P(1,4) = 4.11, p = 0.003$), thus rejecting the null hypothesis. A post-hoc test based on Tukey's method [Tukey 1949] shows that the MLP presents significantly higher scores than the DCAE ($p = 0.007$) and than NMF ($p = 0.028$). For CI subjects an ANOVA showed a significant effect of SS algorithm on the MUSHRA scores ($F(1,4) = 2.334, p = 0.05$). However, a post-hoc test revealed that the pair DRNN-NMF ($p = 0.056$) just missed significance and that the mean differences between other pairs being compared were not significant.

It is interesting to note the differences in the scores achieved by the reference and anchor en each ideal and general case scenarios. While being subtle, is easy to see that the reference is better detected by both CI and NH subjects in the general case scenario due to the overall poor performance of the algorithms. This fact is also reflected in the anchor's score provided by the CI users in this same case, which is higher than in the ideal case. However, the anchor was perfectly detected by NH listeners in both cases which indicates the much higher sensitivity to signal degradation of the natural human hearing system.

These results indicate that NMF is the least preferred algorithm by both subject groups while the DRNN and the MLP obtain higher scores.

Chapter 5

Conclusions and Future Work

This chapter relates the data gathered during the SS experiments and the perceptual tests. The main conclusions and observations are described here following with future work ideas.

5.1 Conclusions

This work has demonstrated that any SS algorithm which achieves an SDR and SAR greater than 0.14 dB and 3.63 dB respectively, is suitable for remixing singing pop western music for CI users. Using this benchmark, next to a MUSHRA test we propose an MLP for real-time audio SS implementation.

State-of-the-art SS algorithms have been evaluated for monaural singing voice enhancement in pop music for CI users. The evaluation has been performed from an objective and subjective point of view by investigating how objective metrics (SDR, SAR) relate to the preference of remixed songs using the original multi-tracks or the estimated multi-tracks. Moreover, a MUSHRA test has been performed to assess the perceptual quality ratings of remixed songs with different SS algorithms.

Objective results show that both DCAEs perform better than the MLP, the DRNN, and NMF in the ideal case scenario. The performance achieved by these are comparable to the ones presented at the *MIREX 2016 singing voice separation challenge*[1] [Chandna et al. 2017] (see Table 4.2). However, these complex networks, as well as NMF, struggle to generalize, as shown in Fig. 4.2.

With the first perceptual experiment, a distortion artifact working point threshold above which a SS algorithm would generate a mixture with a degradation difficult to perceive for CI recipients was assessed. This working point was obtained through the extension of the benchmark proposed by [Pons et al. 2016] (see Fig. 4.3 and Fig. 4.4) and imposes the constraint of performing audio SS which achieves positive SDRs and SARs, where both DRNN and MLP work in a generalized way.

The second perceptual experiment shows a significant effect of SS algorithm on MUSHRA scores on both NH and CI users for the general case scenario. No significant effects were observed for the ideal case scenario due to the general good SS performance of the algorithms. Nonetheless, both subject groups provide a correlation between the MUSHRA scores and the mean SDR and SAR values in both ideal

[1] http://www.music-ir.org/mirex/wiki/2016:Singing_Voice_Separation_Results

and general case scenarios, following a similar trend regarding the relative difference between scores and objective values.

Results lead to assume that the DRNN and MLP could be promising candidates for general applications. In fact, considering real-time applications, the MLP could be a reasonable choice, which despite having a slightly worse objective performance than the DRNN, shows a robust behavior having thousand times less trainable parameters while working above the minimum perceivable distortion-artifact threshold for CI users.

5.1.1 Audio SS Algorithms

In this work, two different DCAEs, a DRNN, an MLP, and NMF have been implemented in order to isolate the lead singing vocal from monaural western popular music recordings.

Performance of the two DCAEs is similar to the performance achieved by state-of-the art algorithms (see Table 4.2) when tested with data similar to the training data but failed when changing the input dataset. Despite implementing regularization methods such as including dropout layers into the network, the relative performance of the DCAEs with respect to the other algorithms remained similar as shown in Tables 4.1 and 4.3. Further work has to be done regarding the generalization problems these architectures present. However, this study conducted some research regarding this generalization problems by adding a dropout layer between the last convolution and the fully connected layer (refer to Table C.7). The results show a big improvement when testing the general dataset but a big drop in performance when tested with the iKala dataset and hence this solution was not completely satisfactory. Moreover, Appendix C shows all the conducted objective experiments showing the performance of each of them next to the number of parameters contained in each of the networks. It is easy to see that the difference in performance is subtle compared to the difference in the number of trainable weights; however, it was decided to implement the simplest and the more complex architectures to assess the perceptual impact on CI users due to the high range of objective values provided by the DCAEs.

A similar problem arises when analyzing the results provided by NMF; the overall performance is worse than the case of the DCAEs and it also presents generalization problems. Due to this and its complexity, this algorithm is unlikely to be included in a future investigation related to this study.

On the other hand, the DRNN and MLP behave in a consistent way through the two test scenarios (ideal and general case) while presenting SDR and SAR values comparable to previous research [Huang et al. 2014; Pons et al. 2016] when targeting lead vocals.

The feature extraction used to construct the input of the networks was based on the STFT which produces linearly spaced frequency data, which does not correspond to human frequency perception. Moreover, big amount of frequency bins were used at a sampling frequency of 44.1 kHz. These are not realistic parameters for a CI sound processor, however; the aim of this study was to investigate the relationship between physical metrics [Vincent, Gribonval, and Févotte 2006] and the subject's

perception. The interest in using high resolution (frequency wise) input data was to obtain extreme values in these objective metrics.

These results are a motivation for further investigate these architectures in terms of real implementation by simplifying the input feature map and maybe investigating some other forms of feature extraction such as through a gammatone filterbank [Lyon, Katsiamis, and Drakakis 2010]. A specially good candidate for investigating these aspects is the MLP, which is the one presenting the less degree of complexity in terms of the number of parameters and computing operations.

5.1.2 Audio SS Perceptual Impact on CI users

The fact that that music simplification by means of re-mixing potentially improves popular music enjoyment for CI users has been previously investigated [Buyens et al. 2014; Pons et al. 2016] and confirmed by this work. It could be said that SS techniques can be used for estimating the lead singing vocal and musical accompaniment in the context of CIs.

When the subjects were asked to choose between SS vs. MT; the pairwise comparison provided the restriction of using SS algorithms that achieve a minimum SDR and SAR values of 0.14 and 3.63 dB respectively. Those are the lower boundaries estimated that determine when CI users would not perceive such errors in the vocals when attenuating the instruments 6 dB. When subjects were asked to choose between DCAE1 vs. DRNN and DCAE2 vs. MLP which presented different levels of distortions and artifacts; the test confirmed the fact that CI users have a significant preference for positive values of SDR and SAR.

When the tested subjects provided a score for each of the algorithms considering a given reference, we observed that these scores followed a similar trend as the objective values given by the objective evaluation. This relation between the objective and subjective evaluation is observed in both ideal and general case scenario; however, the general case scenario shows significant effects on scores with factor "SS algorithm" for both groups. This is likely to be because the difference in performance between the algorithms in the general case scenario is higher than in the ideal case.

After the data gathered through these perceptual experiments, we concluded that an MLP meets with the necessary characteristics to be further investigated in the CI context.

5.2 Future Work

After assessing that SS techniques are suitable for remixing popular music for CI recipients and that SS algorithmic complexity does not seem to play a big role, the next natural step would be to extend a simple MLP to tackle stereo music SS and then accomplish stereo real-time audio SS in less controlled environments e.g. live music.

The feasibility of the real-time implementation of simple NNs has been confirmed in the field of speech enhancement [Goehring et al. 2017]. Results obtained by this

previous research have to be contrasted with the ones provided by these architectures when dealing with more complex signals, such as singing music mixed with background noise.

From the perceptual point of view, this thesis has confirmed that CI users find music more enjoyable when enhancing the lead singing vocal in western pop music; however, many opportunities for extending the scope of this thesis remain.

One of the next natural steps to take regarding subjective evaluation would be to extend the proposed benchmark with more data points in order to obtain a more accurate estimation of the minimum SDR and SAR values to aim at when implementing SS algorithms. Similarly, as the MUSHRA test gave just significant results in CI users it would make sense to, again, obtain more data extending the number of tested subjects. Especially for the MUSHRA, where the number of comparisons made is big in comparison to the number of tested subjects.

Finally, a model which predicts the CI user's perceptual impact of the signal degradation introduced by the SS process would be helpful to help to design the SS algorithm. This could be achieved by relating the obtained objective and perceptual results with the predictions made by existing models, i.e. the PEASS toolkit [Emiya et al. 2011] and proposing an adaptation for CI users.

Appendix A

Tested Subjects' Profiles

TABLE A.1: Tested NH subjects.

ID	Age	Gender
NH01	19	Female
NH02	19	Female
NH03	19	Male
NH04	28	Female
NH05	27	Male
NH06	28	Female
NH07	29	Female
NH08	31	Male
NH09	28	Female
NH10	29	Male

TABLE A.2: Tested CI subjects.

ID	Age	Gender	Etiology	CI experience	Mode
CI01	68	Male	Genetic	22	Bilateral
CI02	48	Female	Unknown	unknown	Unilateral (R)
CI03	81	Male	Sudden	20	Bilateral
CI04	74	Female	Unknown	unknown	Bilateral
CI05	82	Male	Meningitis	10	Bilateral
CI06	68	Male	Genetic	22	Bilateral
CI07	48	Female	Unknown	unknown	Unilateral (R)
CI08	81	Male	Sudden	20	Bilateral
CI09	74	Female	Unknown	unknown	Bilateral
CI10	82	Male	Meningitis	10	Bilateral

Appendix B

Checklists for the Perceptual Experiments

Checklist Experiment 1

Before patient arrives

- Set the Patient ID and Name into the software.
- Calibrate the system at comfort level using a sonometer, making sure the volume is set to 60 dB(A).

After patient arrives

- Not explain too many things. Just explain: different mixes will be presented. Choose the most enjoyable.
- Prepare CI user by deactivating any special program on the device.
- Start test.
- Ask them to fill up a form to provide us with

Checklist Experiment 2

Before patient arrives

- Check with the sonometer that the volume is set to 60 dB(A) for training part (supposed comfort level).

After patient arrives

- Be careful on not explaining too many things. Just explain: score each presented music excerpt in terms of perceived quality relative to the reference, where the reference represents a full 100 point score. Explain as well that at least one of the sliders must be set to 100 as one of them is a hidden reference.
- In order for the to understand the procedure a demo is performed.
- Let them train for at least 10 minutes by listening to all of the music excerpts they will be listening to.
- Start test.
- Ask them to fill up a form to provide us with information about their musical background.

Appendix C

DCAEs Objective Experiments

DCAE Experiment 1

- Topology:

TABLE C.1: DCAE_E_1 Topology

Type	*Kernel*	*Stride*	*Number of Filters*
Conv	1×50	1×3	50
Pool	1×2	-	-
Conv	10×20	1×1	30
Fully Connected	-	-	512
Number of parameters: 113,300,034			

- Training parameters:
  - Time context = 40
  - Batch size = 30
  - Epochs: 100
  - $\alpha = 0.9$
  - $\beta = 0.05$
- Performance:

TABLE C.2: Mean and standard deviation for the SDR, SIR and SAR values expressed in dB corresponding to the the first implemented DCAE.

Ideal Case			*General Case*		
SDR	SIR	SAR	SDR	SIR	SAR
11.17 ± 3.5	17.96 ± 4.71		12.59 ± 3.14 -2.47 ± 6.12	4.19 ± 7.91	1.49 ± 3.00

DCAE Experiment 2

- Topology:

TABLE C.3: DCAE_E_2 Topology

Type	*Kernel*	*Stride*	*Number of Filters*
Conv	1×30	1×1	20
Pool	1×2	-	-
Conv	10×20	1×1	30
Fully Connected	-	-	128
Number of parameters: 116,604,050			

- Training parameters:
  - Time context = 30
  - Batch size = 20
  - Epochs: 100
  - $\alpha = 0.9$
  - $\beta = 0.05$
- Performance:

TABLE C.4: Mean and standard deviation for the SDR, SIR and SAR values expressed in dB corresponding to the the second implemented DCAE.

Ideal Case			*General Case*		
SDR	SIR	SAR	SDR	SIR	SAR
9.84 $\pm$ 4.03	13.19 $\pm$ 4.79	13.17 $\pm$ 3.16	-0.77 $\pm$ 5.86	2.19 $\pm$ 6.54	5.64 $\pm$ 2.76

DCAE Experiment 3

- Topology:

TABLE C.5: DCAE_E_3 Topology

Type	*Kernel*	*Stride*	*Number of Filters*
Conv	20×20	1×1	10
Fully Connected	-	-	64
Number of parameters: 21,472,126			

- Training parameters:
  - Time context = 30
  - Batch size = 20
  - Epochs: 100
  - $\alpha = 0.9$
  - $\beta = 0.05$
- Performance:

TABLE C.6: Mean and standard deviation for the SDR, SIR and SAR values expressed in dB corresponding to the the third implemented DCAE.

Ideal Case			*General Case*		
SDR	SIR	SAR	SDR	SIR	SAR
11.06 $\pm$ 3.74	16.68 $\pm$ 4.78	12.90 $\pm$ 3.32	-1.85 $\pm$ 6.46	3.37 $\pm$ 7.51	2.89 $\pm$ 3.84

DCAE Experiment 4

- Topology:

TABLE C.7: DCAE_E_4 Topology

Type	*Kernel*	*Stride*	*Number of Filters*	Droput
Conv	20×20	1×1	10	-
Dropout layer	5-	-	-	5%
Fully Connected	-	-	64	-
Number of parameters: 1,955,726				

- Training parameters:
  - Time context = 20
  - Batch size = 40
  - Epochs: 50
  - $\alpha = 0.005$
  - $\beta = 0.05$
- Performance:

TABLE C.8: Mean and standard deviation for the SDR, SIR and SAR values expressed in dB corresponding to the the fourth implemented DCAE.

Ideal Case			*General Case*		
SDR	SIR	SAR	SDR	SIR	SAR
1.55 ± 5.58	2.46 ± 6.17	12.83 ± 1.57	7.66 ± 5.25	9.98 ± 7.04	14.00 ± 2.53

Bibliography

Aronoff, J. M. et al. (2011). "The effect of different cochlear implant microphones on acoustic hearing individuals' binaural benefits for speech perception in noise". In: *Ear & Hearing* 32.4, pp. 468–484.

Bear, M.F. et al. (2001). *The auditory and vestibular systems. Neuroscience Exploring the brain, 2nd ed.(Katz S, Ed.), Baltimore: Lippincott Williams and Wilkins*, pp. 349–395. ISBN: 0683305964.

Bengio, Y., N. Boulanger-Lewandowski, and R. Pascanu (2013). "Advances in optimizing recurrent networks". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 8624–8628.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford University Press, Inc. ISBN: 0198538642.

– (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc. ISBN: 0387310738.

Bronkhorst, A. W. (2000). "The Cocktail Party Phenomenon: A Review of Research on Speech Intelligibility in Multiple-Talker Conditions". In: *Acta Acustica united with Acustica*.

Burns, E. M. and N. F. Viemeister (1981). "Played again SAM: Further observations on the pitch of amplitude modulated noise". In: *The Journal of the Acoustical Society of America* 70, pp. 1655–1660.

Buyens, W. et al. (2014). "Music mixing preferences of cochlear implant recipients: A pilot study". In: *International Journal of Audiology* 53.5, pp. 294–301.

Böck, S. and M. Schedl (2012). "Polyphonic piano note transcription with recurrent neural networks". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 121–124.

Chan, T. S. et al. (2015). "Vocal activity informed singing voice separation with the iKala dataset". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 718–722.

Chandna, P. (2016). "Audio Source Separation Using Deep Neural Networks". MA thesis.

Chandna, P. et al. (2017). "Monoaural Audio Source Separation Using Deep Convolutional Neural Networks". In: *13th International Conference on Latent Variable Analysis and Signal Separation (LVA ICA2017)* 10169, pp. 258–266.

Choi, K. et al. (2016). "Convolutional Recurrent Neural Networks for Music Classification". In: *Computing Research Repository* 1609.04243.

Darken, C., J. Chang, and J. Moody (1992). "Learning rate schedules for faster stochastic gradient search". In: pp. 3–12.

Dauphin, Y. et al. (2014). "Identifying and attacking the saddle point problem in high-dimensional non-convex optimization". In: *Computing Research Repository*.

Dozat, T. (2015). "Incorporating Nesterov Momentum into Adam". In:

Du, B. et al. (2017). "Stacked Convolutional Denoising Auto-Encoders for Feature Representation". In: *IEEE Transactions on Cybernetics* 47.4, pp. 1017–1027.

Duchi, J., E.Hazan, and Y. Singer (2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12, pp. 2121–2159.

Duong, N. Q. K. et al. (2014). "An interactive audio source separation framework based on non-negative matrix factorization". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1567–1571.

Emiya, V. et al. (2011). "Subjective and Objective Quality Assessment of Audio Source Separation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 19.7, pp. 2046–2057.

Farabet, C. et al. (2013). "Learning Hierarchical Features for Scene Labeling". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8, pp. 1915–1929.

Frasca, M. et al. (2013). "A neural network algorithm for semi-supervised node label learning from unbalanced data". In: *Neural Networks* 43, pp. 84 –98. URL: http://www.sciencedirect.com/science/article/pii/S0893608013000361.

Fu, S. W., Y. Tsao, and X. Lu (2016). "SNR-Aware Convolutional Neural Network Modeling for Speech Enhancement". In: *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association*, pp. 3768–3772.

Galvin, J. J., Q. Fu, and R. V. Shannon (2009). "Melodic Contour Identification and Music Perception by Cochlear Implant Users". In: *Annals of the New York Academy of Sciences* 1169, pp. 518–533.

Gardner, M. W. and S. R. Dorling (1998). "Artificial neural networks (the multilayer perceptron). A review of applications in the atmospheric sciences". In: *Atmospheric Environment* 32.14-15, pp. 2627–2636.

Gaussier, E. and C. Goutte (2005). "Relation Between PLSA and NMF and Implications". In: *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '05. Salvador, Brazil: ACM, pp. 601–602. ISBN: 1-59593-034-5.

Geiger, J.T. et al. (2013). "The TUM+TUT+KUL approach to the CHiME Challenge 2013: Multi-stream ASR exploiting BLSTM networks and sparse NMF". In: *Proc. of 2nd CHiME Workshop held in conjunction with ICASSP 2013*, pp. 25–30.

Geisser, S. (1993). *Predictive Inference*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis. ISBN: 9780412034718. URL: https://books.google.es/books?id=wfdlBZ_iwZoC.

Gfeller, K. et al. (2015). "A preliminary report of music-based training for adult cochlear implant users: rationales and development". In: *Cochlear implants international* 14.3, pp. 22–31.

Goehring, T. et al. (2017). "Speech enhancement based on neural networks improves speech intelligibility in noise for cochlear implant users". In: *Hearing Research* 344, pp. 183–194.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. http://www.deeplearningbook.org. MIT Press.

Grais, E. M. and M. D. Plumbley (2017). "Single Channel Audio Source Separation using Convolutional Denoising Autoencoders". In: *Computing Research Repository* 1703.08019.

Grais, E. M. et al. (2016). "Discriminative Enhancement for Single Channel Audio Source Separation using Deep Neural Networks". In: *Computing Research Repository* 1609.01678.

Graves, A., A. Mohamed, and G.E. Hinton (2013). "Speech Recognition with Deep Recurrent Neural Networks". In: *Computing Research Repository* abs/1303.5778.

Grossberg, S. (1988). "Nonlinear neural networks: Principles, mechanisms, and architectures". In: *Neural Networks"* 1.1, pp. 17 –61.

Han, Y., J. Kim, and K. Lee (2017). "Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.1, pp. 208–221.

He, H. and E. A. Garcia (2009). "Learning from Imbalanced Data". In: *IEEE Transactions on Knowledge and Data Engineering* 21.9, pp. 1263–1284.

Hinton, G. E. and R. R. Salakhutdinov (2006). "Reducing the dimensionality of data with neural networks". In: *Science* 313.5786, pp. 504–507.

Hochreiter, S. et al. (2001). *Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies*.

Hornik, K., M. Stinchcombe, and H. White (1989). "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural Netw.* 2.5, pp. 359–366.

Huang, P. S. et al. (2014). "Deep learning for monaural speech separation". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1562–1566.

– (2015). "Joint Optimization of Masks and Deep Recurrent Neural Networks for Monaural Source Separation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.12, pp. 2136–2147.

J.Dean et al. (2012). "Large Scale Distributed Deep Networks". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*. Curran Associates Inc., pp. 1223–1231.

Karhunen, J. et al. (1997). "A class of neural networks for independent component analysis". In: *IEEE Transactions on Neural Networks* 8.3, pp. 486–504.

Kim, M. and P. Smaragdis (2015). *Adaptive Denoising Autoencoders: A Fine-Tuning Scheme to Learn from Test Mixtures*. Cham: Springer International Publishing, pp. 100–107. ISBN: 978-3-319-22482-4.

Kingma, D. P. and J. Ba (2014). "Adam: A Method for Stochastic Optimization". In: *Computing Research Repository*.

Kohavi, R. (1995). "A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection". In: *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*. IJCAI'95. Montreal, Quebec, Canada: Morgan Kaufmann Publishers Inc., pp. 1137–1143. ISBN: 1-55860-363-8. URL: http://dl.acm.org/citation.cfm?id=1643031.1643047.

Korzeniowski, F. and G. Widmer (2016). "A Fully Convolutional Deep Auditory Model for Musical Chord Recognition". In: *Computing Research Repository* 1612.05082.

Kriesel, D. (2007). *A Brief Introduction to Neural Networks*. Available at http://www.dkriesel.com.

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Kullback, S. and R. A. Leibler (1951). "On Information and Sufficiency". In: *The Annals of Mathematical Statistics* 22.1, pp. 79–86.

Kumar, R. et al. (2013). "Multiple Kernel Completion and its application to cardiac disease discrimination". In: *2013 IEEE 10th International Symposium on Biomedical Imaging*, pp. 764–767.

Langner, F. et al. (2017). "Adding simultaneous stimulating channels to reduce power consumption in cochlear implants". In: *Hearing Research* 345, pp. 96 –107.

Lecun, Y., Y. Bengio, and G. Hinton (2015). "Deep learning". In: *Nature* 521.7553, pp. 436–444.

Lee, D. D. and H. S. Seung (1999). "Learning the parts of objects by nonnegative matrix factorization". In: *Nature* 401, pp. 788–791.

Lee, H. et al. (2009). "Unsupervised Feature Learning for Audio Classification Using Convolutional Deep Belief Networks". In: *Proceedings of the 22Nd International Conference on Neural Information Processing Systems*. NIPS'09. Vancouver, British Columbia, Canada, pp. 1096–1104. ISBN: 978-1-61567-911-9.

Lehnhardt, E. and R. Laszig (2009). *Praxis der Audiometrie*. Thieme. ISBN: 9783133690096.

Lyon, R. F., A. G. Katsiamis, and E. M. Drakakis (2010). "History and future of auditory filter models". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pp. 3809–3812.

Marxer, R. and J. Janer (2013). "Low-latency Bass Separation using Harmonic-Percussion Decomposition". In: *International Conference on Digital Audio Effects Conference (DAFx-13)*.

Masci, J. et al. (2011). "Stacked Convolutional Auto-encoders for Hierarchical Feature Extraction". In: *Proceedings of the 21th International Conference on Artificial Neural Networks - Volume Part I*. ICANN'11. Espoo, Finland: Springer-Verlag, pp. 52–59. ISBN: 978-3-642-21734-0.

McDermott, H. J. (2004). "Music Perception with Cochlear Implants: A Review". In: *Trends in Amplification* 8.2, pp. 49–82.

Nagathil, A., C. Weihs, and R. Martin (2016). "Spectral Complexity Reduction of Music Signals for Mitigating Effects of Cochlear Hearing Loss". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.3, pp. 445–458.

Nesterov, U. (1983). In: *Soviet Mathematics Doklady* 27.2, pp. 372–376.

Nogueira, W. et al. (2011). *Music Perception with Current Signal Processing Strategies for Cochlear Implants*. Barcelona, Spain: ACM, pp. 183–183. ISBN: 978-1-4503-0913-4.

Nogueira, W. et al. (2016). "Development of a Sound Coding Strategy based on a Deep Recurrent Neural Network for Monaural Source Separation in Cochlear Implants". In: *ITG Speech Communication*.

Nogueira, W.aldo et al. (2017). "Loudness and pitch perception using Dynamically Compensated Virtual Channels". In: *Hearing Research* 344, pp. 223–234.

Nugraha, A. A., A. Liutkus, and E. Vincent (2016). "Multichannel Audio Source Separation With Deep Neural Networks". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.9, pp. 1652–1664.

Paatero, P. and U. Tapper (1994). "Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values". In: *Environmetrics* 5.2, pp. 111–126.

Park, Se Rim and Jinwon Lee (2016). "A Fully Convolutional Neural Network for Speech Enhancement". In: *Computing Research Repository* abs/1609.07132.

Pearson, K. (1900). "On the Criterion that a Given System of Deviations from the Probable in the Case of a Correlated System of Variables is Such that it Can Be Reasonably Supposed to have Arisen from Random Sampling". In: 50, pp. 157–175.

Pennington, J., R. Socher, and C. D. Manning (2014). "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1532–1543.

Pons, J. et al. (2016). "Remixing music using source separation algorithms to improve the musical experience of cochlear implant users". In: *The Journal of the Acoustical Society of America* 140.6, pp. 4338–4349.

Qian, N. (1999). "On the Momentum Term in Gradient Descent Learning Algorithms". In: *Neural Networks : The Official Journal of the International Neural Network Society* 12.1, pp. 145–151.

Qian, Y. et al. (2016). "Very Deep Convolutional Neural Networks for Noise Robust Speech Recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.12, pp. 2263–2276.

Rau, T.S., T. Lenarz, and O. Majdani (2015). "Individual Optimization of the Insertion of a Preformed Cochlear Implant Electrode Array". In: *International Journal of Otolaryngology* 724703.

Robinns, H. and S. Monro (1951). "A stochastic approximation method". In: *Annals of Mathematical Statistics* 3, pp. 400–407.

RStudio Team (2015). *RStudio: Integrated Development Environment for R*. RStudio, Inc. Boston, MA.

Sajda, P., S. Du, and L.C. Parra (2003). *Recovery of constituent spectra using non-negative matrix factorization*.

Scherer, D., A. Müller, and S. Behnke (2010). "Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition". In: *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*. ICANN'10. Thessaloniki, Greece: Springer-Verlag, pp. 92–101. ISBN: 3-642-15824-2, 978-3-642-15824-7.

Schmidt, M.N. (2008). *Speech separation using non-negative features and sparse non-negative matrix factorization*.

Smaragdis, P. and S. Venkataramani (2016). "A Neural Network Alternative to Non-Negative Audio Models". In: *Computing Research Repository* 1609.03296.

Sutton, R. S. (1986). *Two Problems with Backpropagation and Other Steepest-Descent Learning Procedures for Networks*. Hillsdale, NJ: Erlbaum.

Tan, Y., J. Wang, and J. M. Zurada (2001). "Nonlinear blind source separation using a radial basis function network". In: *IEEE Transactions on Neural Networks* 12.1, pp. 124–134.

Tukey, J. W. (1949). "Comparing Individual Means in the Analysis of Variance". In: *Biometrics* 5.2, pp. 99–114.

Uhlich, S. et al. (2017). "Improving Music Source Separation based on DNNs through Data Augmentation and Network Blending". In: *IEEE SigPort*.

Vincent, E., R. Gribonval, and C. Févotte (2006). "Performance measurement in blind audio source separation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 14.4, pp. 1462–1469.

Vincent, P. et al. (2008). "Extracting and Composing Robust Features with Denoising Autoencoders". In: *Proceedings of the 25th International Conference on Machine Learning*. ICML '08. Helsinki, Finland: ACM, pp. 1096–1103. ISBN: 978-1-60558-205-4.

Vincent, P. et al. (2010). "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". In: *J. Mach. Learn. Res.* 11, pp. 3371–3408.

Virtanen, T. (2006). *Sound Source Separation in Monaural Music Signals*. Tech. rep.

Wang, Y., A. Narayanan, and D. Wang (2014). "On Training Targets for Supervised Speech Separation". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22.12, pp. 1849–1858.

Wilson, B. S. and M. F. Dorman (2008). "Cochlear implants: current designs and future possibilities". In: *Journal of rehabilitation research and development* 45.5, pp. 695–730.

Xie, J., L. Xu, and E. Chen (2012). "Image Denoising and Inpainting wPredictive Inferenceith Deep Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., pp. 341–349.

Xu, Y. et al. (2017). "Convolutional Gated Recurrent Neural Network Incorporating Spatial Features for Audio Tagging". In: *Computing Research Repository* 1702.07787.

Zeiler, M. D. (2012). "ADADELTA: An Adaptive Learning Rate Method". In: *Computing Research Repository* 1212.5701.

Zhao, M. et al. (2015). "Music removal by convolutional denoising autoencoder in speech recognition". In: pp. 338–341.

Zhu, Z. et al. (2012). "Cochlear-implant spatial selectivity with monopolar, bipolar and tripolar stimulation". In: *Hearing Research* 283.1, pp. 45 –58.