Blind Source Separation of Music Using Deep Neural Networks

Speech and Sound Group,
Sony European Technology Center,
Stuttgart, Germany

Signal Processing Laboratory 2,
Institute of Electrical Engineering,
École Polytechnique Fédérale de Lausanne

Master Thesis
March 27, 2015
Submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science
by

Ertan Kazikli

Supervisors:

Sony Supervisor: Dr. Stefan Uhlich
EPFL Supervisor: Prof. Pierre Vandergheynst

Lausanne, EPFL, 2015

Acknowledgements

I would like to thank Dr. Stefan Uhlich for giving me the opportunity to work at Sony European Technology Center. I am really grateful for his supervision, special guidance and support during my thesis work. It was a great pleasure to work with him.

I would like to thank Prof. Pierre Vandergheynst for accepting me as a master thesis student and supervising my thesis work.

I would like to express my gratitude to my family for their constant support and sincere love. I am also really grateful for their help to cope with challenges ranging from finding an accommodation in Stuttgart to getting a visa in quite short amount of time.

Finally, I would like to express my special thanks to all my friends.

Lausanne, 27 March 2015 E. K.

Abstract

Blind audio source separation refers to the extraction of audio sources from their observed mixtures. The purpose of this thesis is to investigate the source separation of single-channel instrumental music mixtures which is very challenging as we always have to deal with the underdetermined case with more sources than mixture signals. This particular case of musical source separation has such applications as upmixing and music information retrieval. In order to perform the separation, the methods considered in this work require the information that comprise the type of the instruments contained in the mixture. In particular, a deep neural network based source separation scheme is considered where the separation is performed using a time-frequency representation of the mixture signal. The investigation of deep neural network based source separation constitutes three major topics which can be listed as the network architecture being either a feedforward or a recurrent architecture, frequency bin spacing of the time frequency representation being either linear or hybrid of linear and logarithmic, and the training material being either real music or music score based music. In order to compare the results of the deep neural network based approaches, supervised nonnegative matrix factorization based source separation is investigated. The methods are applied to mixtures of three instruments so that a performance comparison of the different methods and settings can be made. The results of the conducted experiments demonstrate that the best deep neural network based method outperforms the best nonnegative matrix factorization based approach on the test dataset. The results of this empirical study show that the use of deep neural networks for source separation task is promising.

Key words: Blind source separation (BSS), single-channel audio source separation, deep neural network (DNN), nonnegative matrix factorization (NMF).

Contents

List of Figures

List of Tables

1 Introduction

Audio signals, especially music signals, are found in the form of mixtures of several sound sources. Single-channel music recordings are quite common even if the music recording contains more than a single sound source. Hence, if there are more than one source in a single-channel music recording, then the recorded signal contains the mixture of the sources in the music. In case individual sound sources are needed, the mixture signal should be processed in order to recover the sound sources.

Various applications in fact make use of recovery of sound sources from their observed mixtures. These applications include karaoke, upmixing and music information retrieval (MIR). To begin with, karaoke systems need to extract the singing voice of the player when there is also background music at the same time. For an example use where sound source separation helps a karaoke system, see [2]. A second example is the process of upmixing where one tries to increase the number of channels of a sound recording with the aim to make use of an increased number of loudspeakers in a multi-channel surround system. For example, [3] makes use of sound source extraction in order to achieve single-channel to two channels upmix. Finally, MIR where the objective is to retrieve such information as musical note transcription, singer, genre, played instruments etc. from music, and which can also make use of sound source separation techniques. For instance, [4] utilizes sound source separation methods in order to achieve musical note transcription.

The examples from above show that there is significant interest for audio source separation methods especially for music mixtures. Audio source separation is thus an active research field. Audio source separation methods are often "blind", making up the term blind audio source separation (BASS), which means that the separation is performed using only the observed mixture signals. These methods focusing not only on audio signals but on arbitrary signals are referred to as BSS methods. For a discussion of the general BSS problem and an overview of important BSS methods, see [5].

This thesis studies audio source separation of single-channel music mixtures where the music is the mixture of only instrumental sounds, such as piano, violin and horn. The objective is to separate single-channel instrumental music mixtures given the mixture signal and the information comprising the type of the instruments contained in the music mixture. In practice, if a source separation method assumes very little extra information, which in our case is the type of the instruments contained in the mixture, in addition to observed mixture signals, this method is still considered as a BSS method. In order to emphasize the additional information being used, these methods are sometimes called "informed BSS" methods.

In particular, we investigate two major approaches. The first approach and main focus of this thesis is to consider a deep neural network based framework for instrumental source extraction from music. DNNs became popular in other fields such as computer vision, natural language processing and speech recognition, and being successful in numerous applications in other fields, DNNs also attract attention in audio source separation, which we will discuss in the following section. The second approach makes use of nonnegative matrix factorization methods. NMF based source separation is investigated with the main aim to see the effectiveness of the DNN based approach by comparing the results.

1.1 Relation to Previous Work

The DNN based source separation framework in this thesis can be considered as a further investigation of the architecture proposed in [6]. In this cited work, a feedforward neural network architecture working in the short time Fourier transform (STFT) domain is trained so that the network performs the source separation task, where the training samples are generated by using a real music database. In this thesis, we investigate three major aspects. First of all, in addition to feedforward neural network architecture, a deep recurrent neural network architecture is employed. Secondly, constant Q transform (CQT) based compression of STFT magnitude frames is examined by which reduction in the DNN training time is gained. Finally, in addition to real music training material, music score database based training material generation is considered.

In order to compare the results of DNN based source separation with another method, we consider supervised NMF from [7]. There, supervised NMF is employed in order to separate two sources, namely music and speech while in our case the objective is to separate instrumental sources from their mixture. We additionally incorporate temporal continuity and sparsity constraints from [8] to see their effect on supervised separation scheme. Furthermore, although it is not investigated in this thesis, it is worthy to note that there are complex NMF [9] based methods applied to solve single-channel source separation problem. For instance, [10] uses complex NMF to separate single-channel mixtures where the separation is supervised by training material corresponding to the sources in the mixture, and further compares the separation results with the NMF based scheme.

Several other approaches exist where neural networks are used for single-channel source separation [11–15]. For instance, [11] and [12] utilize deep neural network architectures for speech separation and singing-voice separation tasks, respectively, where they consider discriminative training in order to improve the interference performance of the separation by adding a discriminative cost to the overall optimization function. Our approach, however, mainly focuses on training sample generation in order to reduce the interference from other sources simply by using many training samples.

There are also single-channel source separation methods which combine DNN and NMF. For instance, [13] uses supervised NMF to initialize the source estimates which is then used by a trained DNN to give the final source estimates. Furthermore, another approach from [14] benefit from the ability of recurrent neural network architectures capturing long-term dependencies in order to model temporal dependency in NMF based source separation. As opposed to aferomentioned approaches that combine DNN and NMF, we simply investigate DNN and NMF based separation as two different approaches, and provide with a comparison in terms of end separation results.

1.2 Notation

We give here the notation that we adopt throughout the thesis. Bold lower case letters are used to denote column vectors if not otherwise stated while bold upper case letters are used to denote matrices. Matrix operations transpose and element-wise conjugate applied to the matrix **X** are denoted by $\mathbf{X}^T$ and $\mathbf{X}^*$, respectively. Moreover, $\mathbf{X} .* \mathbf{Y}$ gives element-wise product of the matrices **X** and **Y**. Similarly, $\mathbf{X}^{.\wedge\alpha}$ applies element-wise exponentiation to the elements of **X** by the scalar α. The operation diag(**x**) where **x** is either a column or a row vector returns a square matrix whose diagonal entries are from **x**, and non-diagonal entries are zero.

Furthermore, we adopt the following definitions of gradient vectors. Gradient of the scalar α with respect to the column vector **x** and with respect to the matrix **X** are defined as

$$\frac{\partial\alpha}{\partial\mathbf{x}} = \begin{bmatrix} \frac{\partial\alpha}{\partial x_1} \\ \frac{\partial\alpha}{\partial x_2} \\ \vdots \\ \frac{\partial\alpha}{\partial x_n} \end{bmatrix}, \qquad \frac{\partial\alpha}{\partial\mathbf{X}} = \begin{bmatrix} \frac{\partial\alpha}{\partial x_{11}} & \cdots & \frac{\partial\alpha}{\partial x_{1m}} \\ \vdots & \ddots & \\ \frac{\partial\alpha}{\partial x_{n1}} & \cdots & \frac{\partial\alpha}{\partial x_{nm}} \end{bmatrix}. \tag{1.1}$$

1.3 Outline of the Thesis

The organization of this thesis is as follows:

Chapter 2 gives background information about the fundamental concepts used in the thesis.

Chapter 3 explains the blind audio source separation problem in general together with focus on single-channel source separation of instrumental music mixtures that this thesis investigates.

Chapter 4 establishes the theoretical background on deep neural networks necessary for further discussion on single-channel source separation using deep neural networks.

Chapter 5 describes extraction of instrumental sounds from their single-channel mixtures using deep neural network architectures.

Chapter 6 explains source separation applied to instrumental music mixtures using nonnegative matrix factorization methods.

Chapter 7 gives the results of the applied source separation methods and compares the results of different methods.

Chapter 8 concludes the thesis with a summary and gives an outlook on further improvements that could be made.

2 Preliminaries

Definitions regarding the fundamental concepts used in the remainder of the thesis are described here.

2.1 Discrete Fourier Transform

To analyze audio signals, transformation to frequency domain is preferred. In order to obtain frequency content of a discrete time signal, discrete Fourier transform (DFT) can be used. Given a discrete time signal $x[n]$ of length N, the DFT is defined as

$$X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi kn/N}, \quad k = 0, 1, \ldots, N-1, \tag{2.1}$$

where n and k are time and frequency indices, respectively.

In order go back into the time domain, the inverse discrete Fourier transform has to be used which is defined as

$$x[n] = \sum_{k=0}^{N-1} X[k]\, e^{j2\pi kn/N}, \quad n = 0, 1, \ldots, N-1. \tag{2.2}$$

Note that these transformations can be obtained in a computationally efficient way by using the fast Fourier transform (FFT) algorithm.

2.2 Time-frequency Representation of Discrete Time Signals

In order to see how spectral content of a time-variant signal behaves over time, a time-frequency representation is used. We consider two time-frequency representations described in the following.

2.2.1 Short time Fourier transform

A time-frequency representation commonly used in audio applications is the short time Fourier transform. The STFT of a discrete time signal $x[n]$ is defined as

$$X[k,t] = \sum_{n=0}^{N-1} x[n+tS]\ w[n]\ e^{-j2\pi kn/N}, \tag{2.3}$$

where t and k denote time frame index and discrete frequency index, respectively, S is the shift in samples between two adjacent STFT frames, $w[n]$ is the analysis window, and N is the length of the analysis window.

In fact, $X[k,t]$ is the DFT of the segment $x[n+tS]w[n]$ where n is in $[0, N-1]$. The value of S in equation (2.3) determine the overlap of these segments, namely percent of overlap between two neighboring segments can be expressed as $(1-S/N)100$. Overlapping segments are used in order to introduce redundancy in the STFT domain. Note that STFT of a signal is usually represented by constructing an STFT matrix $\mathbf{X}$ whose elements are defined as $[\mathbf{X}]_{k,t} = X[k,t]$.

In order to get the time domain signal back, an inverse STFT (ISTFT) method should be applied. One popular approach is to use overlap and add method. This method first applies a synthesis window corresponding to analysis window to each column of the STFT matrix, and then applies the inverse DFT to the windowed columns of the STFT matrix. Since STFT uses overlapping windows, ISTFT should also take overlapping windows into account. Therefore, the segments obtained by taking the inverse DFT is overlapped and added in order to construct the time-domain signal.

In this thesis, we in particular use the the least squares ISTFT method from Yang which is one of the estimators derived in [16]. The idea here is to make use of redundancy introduced while constructing the STFT representation in order to estimate the time-domain signal.

2.2.2 Constant Q transform

Constant Q transform [17] can also be used to represent spectrum of a time-variant signal. In fact, CQT can also be seen as just being a frequency representation but in the following we will give the definition when it is used as a time-frequency representation. Frequencies of musical notes in western classical music are geometrically spaced. This means that frequencies of musical notes can be written in terms of a smallest frequency $f_{\min}$ as $f_k = f_{\min}2^{1/12}$, where $k \in \mathbb{N}$ assuming a twelve-tone equal tempered scale. The idea behind the CQT is to define geometrically spaced frequency bins so that the frequency bins become consistent with the frequencies of musical notes. In addition to using geometrically spaced frequency bins, the CQT adjusts its frequency resolution depending on the frequency bin in order to have high resolution for low frequencies and low resolution for higher frequencies. Varying frequency resolution is achieved by using fixed frequency to bandwidth ratio, which is referred to as the quality factor, for each frequency bin.

First of all, we can express frequency bins of the CQT as

$$f_k = f_{\min}\, 2^{k/b}, \quad k = 0, 1, 2, \dots, \tag{2.4}$$

where $f_{\min}$ is the lowest frequency bin, and b denotes the number frequency bins per octave. In practice, number of frequency bins per octave is chosen as an integer multiple of 12 and $f_{\min}$ is chosen as a small frequency corresponding to frequency of a musical note so that the frequency bins of the CQT includes the frequencies of the musical notes. Moreover, f_k goes up to a maximum frequency of choice which is smaller than the Nyquist frequency $f_s/2$. Given the number of frequency bins per octave, we can write the constant quality factor of the CQT as

$$Q = \frac{1}{2^{1/b} - 1}. \tag{2.5}$$

In order to achieve a constant Q factor for all the frequency bins of CQT, the window size varies based on frequency bin where the window length for the frequency bin k is written as

$$N_k = \frac{Q f_s}{f_k}. \tag{2.6}$$

Having defined the necessary parameters, we can express the CQT of a discrete time signal $x[n]$ as

$$X^{\text{cqt}}[k, t] = \frac{1}{N_k} \sum_{n=0}^{N_k - 1} x[n + tS]\; w[k, n]\; e^{-j2\pi Q n / N_k}, \tag{2.7}$$

where k is the frequency bin index, $w[k,n]$ is a window of length N_k, and S is the shift in samples between neighboring CQT frames. The shift S can also be chosen as N_k resulting in a varying time resolution for each frequency bin index but in the following we assume that S is constant for all bin index k. Note also that the normalization factor N_k is included in the definition by Brown [17] because the number of terms in the summation for different frequency bin index k differ.

A computationally more efficient algorithm to obtain the CQT by making use of DFT is proposed by Brown and Puckette [18]. The idea here is that the CQT can be written as a matrix multiplication of time domain signal with a kernel. Then, in order to benefit from computational efficiency of the FFT algorithm, the multiplication is performed in the DFT domain by making use of Parseval's relation. Moreover, CQT kernel in the frequency domain becomes sparse which reduces the computational cost of the multiplication. To this end, time domain CQT kernel $\mathbf{T}$ is introduced whose elements are defined as

$$[\mathbf{T}]_{n,k} = \begin{cases} \frac{1}{N_k} w[n,k]\, e^{j2\pi Qn/N_k} & \text{if } n \leq N_k \\ 0 & \text{otherwise.} \end{cases} \tag{2.8}$$

In order to write the CQT in the form of matrix multiplication, we first construct a row vector $\mathbf{x}(t)$ of length $N_0 = Qf_s/f_{\min}$ from the discrete time signal $x[n]$ where $\mathbf{x}(t)$ contains the samples of the tth frame. Next, constant Q transform corresponding to tth frame can be written as $\mathbf{X}^{\text{cqt}}(t) = \mathbf{x}(t)\,\mathbf{T}^*$. Denoting $\mathbf{F}$ as the matrix obtained by applying DFT to each column of the time domain kernel matrix $\mathbf{T}$, we can express the CQT in the DFT domain as

$$\mathbf{X}^{\text{cqt}}(t) = \mathbf{x}(t)\,\mathbf{T}^* \tag{2.9a}$$

$$\overset{\text{Parseval}}{=} \frac{1}{N_0}\mathbf{X}(t)\,\mathbf{F}^*, \tag{2.9b}$$

where $\mathbf{X}(t)$ is a row vector obtained by taking DFT of the row vector $\mathbf{x}(t)$. Since the column vectors of $\mathbf{F}$ are obtained by taking DFT of complex exponentials, a sparse representation can be obtained by setting the elements below a threshold to zero. This way, multiplication in the DFT domain can be written as a sparse multiplication and has a much smaller computational complexity than the time domain computation.

3 Blind Audio Source Separation

This thesis investigates blind audio source separation of single-channel instrumental music mixtures using various architectures. This necessitates a proper definition of the blind source separation problem. Hence, a brief explanation of the BSS problem followed by a description of single-channel instrumental source separation problem that this thesis studies is provided.

Audio source separation aims at recovering individual sound sources from their observed mixtures. Consider for instance a classical music concert where several musical instruments are being played. Furthermore, assume that the concert is recorded by a single microphone. The signal recorded by the microphone is then the mixture of the sound sources played in the concert hall. The purpose of audio source separation is to extract individual instrumental sound sources given the recorded signal.

One can think of slight modifications to the above source separation example. First of all, the sound sources can be of different types, such as singing voice and speech. For instance, one scenario could be trying to recover individual speech sources in case of their mixture, which is referred to as the cocktail party problem [19]. In this thesis, the type of the sources are only musical instruments as in the classical music concert case. A second modification could be regarding the number of available mixtures. In the above example, there is only one microphone. We could increase the number of observations by placing several microphones. For instance, source separation of stereo recordings would also be relevant for music applications since stereo music recordings are also quite common. In this thesis, however, single-channel recordings are considered.

Furthermore, the example explained above does not make any assumption regarding the mixing process. The mixing process could be a simple superposition where the observations are weighted sum of each source. These types of mixtures are referred to as instantaneous mixtures. Another possibility would be that the mixture is a sum of linear time invariant (LTI) filtered versions of each individual source. This problem is known as source separation of convolutive mixtures and review on this specific type of BSS problem can be found at [20]. Although we consider the former mixing process in this thesis, our approach can also be used to separate convolutive mixtures if such data is contained in the training set.

3.1 Problem Formulation

In short, blind source separation refers to obtaining individual sources from their observed mixtures. Considering discrete time observations where n denotes the time index, and assuming that there are J sources where the jth source is denoted by $s_j[n]$ and I mixture observations where the ith mixture is denoted by $x_i[t]$, the $I \times J$ linear instantaneous mixing process can be described by

$$x_i[n] = \sum_{j=1}^{J} a_{ij} s_j[n], \quad i = 1, 2, \dots, I, \tag{3.1}$$

where a_{ij} is the unknown scale of contribution to the ith mixture from the jth source.

In this thesis, only single-channel BSS problem, which is a special case of the general BSS, is considered. While in the general case, there could be more than one microphones capturing mixtures of the sources, in this special case there is only one microphone recording a mixture of the sources. Hence, the mixing process reduces to

$$x[n] = \sum_{j=1}^{J} a_j s_j[n]. \tag{3.2}$$

Furthermore, the scale a_j is dropped since there is a scale ambiguity if there is no prior information regarding the energy of the source signals. That is why the scale is considered as a part of the source signal which means that the sources we are trying to recover is $a_j s_j[n]$ rather than being $s_j[n]$. Therefore, the observed mixture signal can be written as

$$x[n] = \sum_{j=1}^{J} s_j[n]. \tag{3.3}$$

In this thesis, the objective is to recover the instrumental sound sources given their single-channel instantaneous mixtures. Therefore, the mixing process can be described as in (3.3).

To illustrate, Figure 3.1 summarizes the instrumental source extraction problem that this thesis investigates. In Figure 3.1, the instrumental sounds of piano $s_1[n]$, guitar $s_2[n]$ and saxophone $s_3[n]$ are mixed yielding the mixture signal $x[n]$. The BASS system has access to the mixture

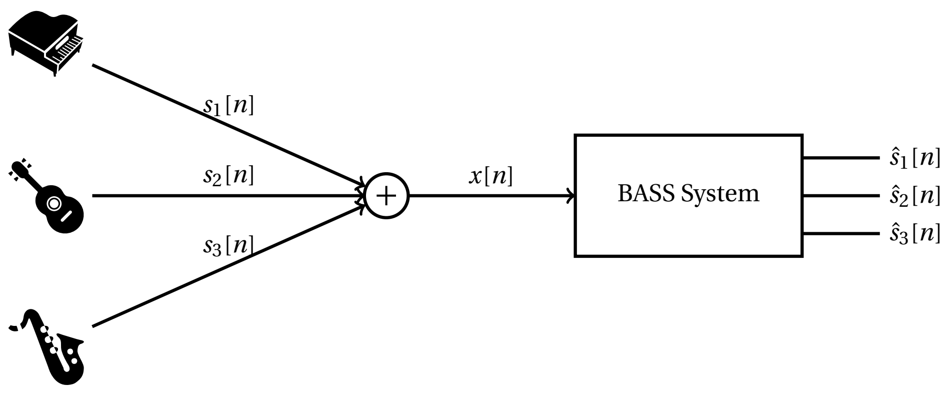


Figure 3.1: Illustration of instrumental source extraction problem

signal only but not to the sound source. The purpose is to design a BASS system such that the source estimates $\hat{s}_1[n]$, $\hat{s}_2[n]$, and $\hat{s}_3[n]$ are as close as possible to the true sources. Note that these three instruments are chosen just for illustration purposes. We in fact consider different combinations of different types of instruments. It is also worthy to note that BASS system is aware of the type of the sources contained in the mixture. This means there are in fact two inputs to the BASS system where the first input is the mixture signal and the second input is the type of the available instruments in the mixture. That is why in Figure 3.1 BASS system could also be named as piano, guitar and saxophone BASS system, which is specialized to separate these three instrumental sounds.

3.2 BASS Performance Metrics

In order to evaluate source separation performance of a BASS system, it is necessary to use a performance evaluation metric analyzing the final output of the system. This is realized by comparing the expected result with the output of the BASS system. This comparison is usually done by using the metrics that we will introduce in the following. However, it is also important to keep in mind that the output of the BASS systems are usually judged by human auditory system which is subjective.

BSS EVAL toolbox

A popular set of numerical performance evaluation criteria for BSS was suggested by Vincent et al. in [21], which we use in this thesis in order to asses the performance of the used BSS methods. In order to provide the reader with a general idea on the source separation performance measures used in the context of this thesis, we give a brief overview of the measures defined in [21]. For details, please refer to this paper. The main idea here is to define different performance measures based on the origin of error made while estimating a source in the mixture. These measures are calculated for each estimated source so that an overall performance level of the BSS system can be obtained. To this end, the estimated source $\hat{\mathbf{s}}$ is written as a sum

$$\hat{\mathbf{s}} = s_{\text{target}} + \mathbf{e}_{\text{interference}} + \mathbf{e}_{\text{artifact}} + \mathbf{e}_{\text{noise}}, \tag{3.4}$$

where $\mathbf{s}_{\text{target}}$ is due to the true source, the error term $\mathbf{e}_{\text{interference}}$ stems from interferences from other sources, $\mathbf{e}_{\text{artifact}}$ denotes the artifacts introduced by the BSS algorithm, and $\mathbf{e}_{\text{noise}}$ is the error due to sensor noise. Note that s_{target} is not exactly the true source but obtained from the true source taking into account allowed distortions on the true source. The contribution due to the true source and due these three error terms are computed by defining orthogonal projections, see [21]. Then, based on the decomposition obtained by using projections, the following performance measures are defined.

Source to Distortion Ratio (SDR) which gives an overall performance measure in estimation of the source is defined as

$$\text{SDR} = 10\log_{10}\frac{\|\mathbf{s}_{\text{target}}\|^2}{\|\mathbf{e}_{\text{interference}} + \mathbf{e}_{\text{artifact}} + \mathbf{e}_{\text{noise}}\|^2}. \tag{3.5}$$

Source to Interferences Ratio (SIR) which measures the performance in terms of the interferences due to other sources is defined as

$$\text{SIR} = 10\log_{10}\frac{\|\mathbf{s}_{\text{target}}\|^2}{\|\mathbf{e}_{\text{interference}}\|^2}. \tag{3.6}$$

Sources to Artifacts Ratio (SAR) which measures the performance considering only the artifacts is defined as

$$\text{SAR} = 10\log_{10}\frac{\|\mathbf{s}_{\text{target}} + \mathbf{e}_{\text{interference}} + \mathbf{e}_{\text{noise}}\|^2}{\|\mathbf{e}_{\text{artifact}}\|^2}. \tag{3.7}$$

Sources to Noise Ratio (SNR) which gives performance evaluation metric considering only the sensor noise is defined as

$$\text{SNR} = 10\log_{10}\frac{\|\mathbf{s}_{\text{target}} + \mathbf{e}_{\text{interference}}\|^2}{\|\mathbf{e}_{\text{noise}}\|^2}. \tag{3.8}$$

Using above numerical performance criteria, we can evaluate the performance of the used BASS algorithm. Since the measures correspond to energy ratios of the term due to true source to error terms, higher values denote a better separation performance. In order to obtain numerical results, a Matlab toolbox by Vincent [22] is used. Moreover, we do not take SNR into account which is mainly because source separation is performed on high quality music recordings for which the sensor noise is negligible.

4 Deep Neural Networks

Since main focus of this thesis is to investigate deep neural network based BASS, it is important to first provide background information regarding deep neural networks. Hence, the deep neural network structures of interest and their corresponding training methods are explained. Furthermore, definitions necessary for further discussion in the following chapters are presented.

In particular, deep feedforward neural network and deep recurrent neural network architectures are considered. First, the feedforward architecture is discussed since it can be used as a prototype for the discussion of the recurrent architecture in the section that follows.

4.1 Deep Feedforward Neural Networks

4.1.1 Network description

Feedforward neural networks, as the name suggests, propagate the input through the network only in the forward direction. This network type is a widely used neural network structure and is also referred to as multi layer perceptron (MLP).

The total network consists of several layers where the output of one layer is input to the following layer. We can describe a deep feedforward neural network architecture consisting of K layers by

$$\mathbf{y}_k = f(\mathbf{a}_k) = f(\mathbf{W}_k \mathbf{y}_{k-1} + \mathbf{b}_k), \quad k = 1, 2, \ldots, K, \tag{4.1}$$

where $\mathbf{y}_0$ stands for the vector input to the network, $\mathbf{a}_k$ and $\mathbf{y}_k$ denotes the activation and the output vectors at kth layer, respectively, $\mathbf{W}_k$ and $\mathbf{b}_k$ are the weight matrix and the bias vector at kth layer, respectively, and $f(\cdot)$ is a nonlinear activation function, which is applied element-wise. To illustrate, Figure 4.1 depicts a feedforward neural network architecture

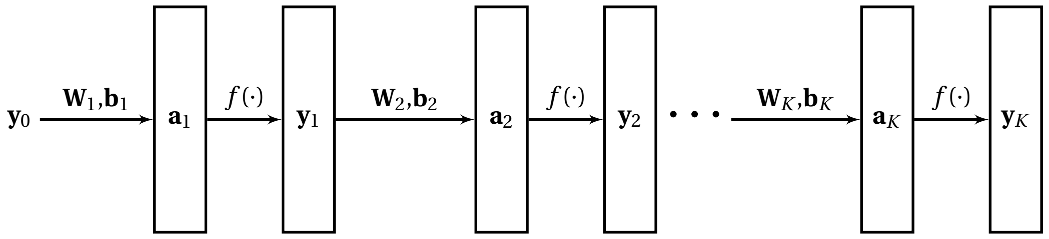


Figure 4.1: Illustration of feedforward neural network architecture with K layers

consisting of K layers.

4.1.2 Error backpropagation algorithm

Deep neural networks are used in order to perform a specific task by benefiting from its nonlinear input-output mapping which in our case is source separation. The idea is to train a network with known input-target pairs after which the expectation is that the network shows the desired behavior for arbitrary inputs. The training of the network refers to learning the parameters of the network, namely weight and bias terms. In order to learn the parameters of the network, fundamental approach is to first define a cost function, and then compute the gradients of the cost function with respect to all the parameters of the network. Then, in order to reduce the cost, the parameters are updated in negative gradient direction iteratively, which is referred to as gradient descent method. The computation of gradients in a neural network is done using the popular error backpropagation algorithm [23].

Cost function to be used for training in the context of this thesis is the sum of squared errors (SSE). Although the derivation of the error backpropagation algorithm in the following uses the SSE cost, the algorithm can be generalized to other cost functions as well.

Assume that we have M input samples where the sample at index t is denoted by $\mathbf{y}_0^t$ and the corresponding target sample is denoted by $\mathbf{d}^t$. If the input samples at index t is propagated through a network with K layers, the output we obtain is denoted by $\mathbf{y}_K^t$. With this notation, the SSE cost corresponding to M input samples can be written as

$$E = \frac{1}{M}\sum_{t=1}^{M} \left\|\mathbf{y}_K^t - \mathbf{d}^t\right\|^2/2 = \frac{1}{M}\sum_{t=1}^{M} E_t. \tag{4.2}$$

The objective is to compute the gradients of the cost with respect to the parameters of the network, namely $\frac{\partial E}{\partial \mathbf{W}_k}$ and $\frac{\partial E}{\partial \mathbf{b}_k}$, where k in $1 \le k \le K$.

First of all, we start from expanding the cost term due to the input sample $\mathbf{x}_1^t$ in the following

$$\begin{aligned} E_t &= \left\|\mathbf{y}_K^t - \mathbf{d}^t\right\|^2/2 && (4.3a)\\ &= \left\|f(\mathbf{W}_K\,\mathbf{y}_{K-1}^t + \mathbf{b}_K) - \mathbf{d}^t\right\|^2/2 && (4.3b)\\ &= \left\|f(\mathbf{W}_K f(\mathbf{W}_{K-1}\,\mathbf{y}_{K-2}^t + \mathbf{b}_{K-1}) + \mathbf{b}_K) - \mathbf{d}^t\right\|^2/2 && (4.3c)\\ &= \ldots \end{aligned}$$

Since the expansion involves only terms due to the sample index t, it can inferred that the gradients due to different sample indices do not depend on each other. Hence, the sample index is dropped for the output vectors $\mathbf{y}_k^t$, the activation vectors $\mathbf{a}_k^t$ and the desired output vector $\mathbf{d}^t$ in the following derivation.

The main idea behind the error backpropagation is the chain rule of differentiation. Therefore, by applying the chain rule of differentiation to the expansion in equation (4.3b), we can write the gradient of the cost with respect to the Kth layer bias vector $\mathbf{b}_K$ as

$$\frac{\partial E_t}{\partial \mathbf{b}_K} = \operatorname{diag}\left(f'(\mathbf{a}_K)\right)\left(\mathbf{y}_K - \mathbf{d}\right). \tag{4.4}$$

Similarly, from the expansion in equation (4.3c), we can express the gradient of the cost with respect to $\mathbf{b}_{K-1}$ as

$$\begin{aligned} \frac{\partial E_t}{\partial \mathbf{b}_{K-1}} &= \operatorname{diag}\left(f'(\mathbf{a}_{K-1})\right)\mathbf{W}_K^T \operatorname{diag}\left(f'(\mathbf{a}_K)\right)\left(\mathbf{y}_K - \mathbf{d}\right) && (4.5a)\\ &= \operatorname{diag}\left(f'(\mathbf{a}_{K-1})\right)\mathbf{W}_K^T \frac{\partial E_t}{\partial \mathbf{b}_K}, && (4.5b) \end{aligned}$$

where the second equality is due to equation (4.4). One can observe the pattern to obtain the gradient of the cost with respect to the bias vector at layer $K-1$ from the gradient of the cost with respect to the bias vector at layer K. Thus, gradient computation can be generalized by defining an error vector at kth layer as $\boldsymbol{\epsilon}_k$ being equal to the gradient of the cost with respect to $\mathbf{b}_k$. These error vectors can be computed by propagating through the network in backward direction. Hence, computation of the error vectors can be written as

$$\boldsymbol{\epsilon}_k = \begin{cases} \operatorname{diag}\left(f'(\mathbf{a}_K)\right)\left(\mathbf{y}_K - \mathbf{d}\right) & k = K \\ \operatorname{diag}\left(f'(\mathbf{a}_k)\right)\mathbf{W}_{k+1}^T\,\boldsymbol{\epsilon}_{k+1} & k < K. \end{cases} \tag{4.6}$$

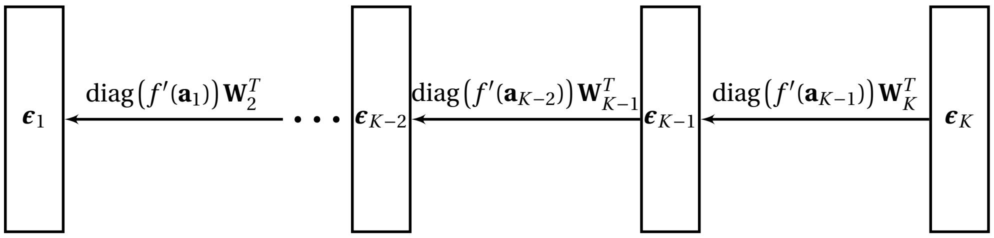

$\boldsymbol{\epsilon}_1$
$\mathrm{diag}\left(f'(\mathbf{a}_1)\right)\mathbf{W}_2^T$
$\boldsymbol{\epsilon}_{K-2}$
$\mathrm{diag}\left(f'(\mathbf{a}_{K-2})\right)\mathbf{W}_{K-1}^T$
$\boldsymbol{\epsilon}_{K-1}$
$\mathrm{diag}\left(f'(\mathbf{a}_{K-1})\right)\mathbf{W}_K^T$
$\boldsymbol{\epsilon}_K$

Figure 4.2: Backpropagation of errors in a feedforward neural network architecture with K layers

To illustrate, Figure 4.2 shows propagation of the error vectors through a feedforward network in the backward direction.

Moreover, it can be seen that the gradient of the cost with respect to the weight matrix at a specific layer can be written in terms of the error vector at the corresponding layer and the input to the corresponding layer. Hence, we can express the gradient of the cost with respect to $\mathbf{W}_k$ as

$$\frac{\partial E_t}{\partial \mathbf{W}_k} = \boldsymbol{\epsilon}_k \, \mathbf{y}_{k-1}^T. \tag{4.7}$$

All in all, computation of the gradients with respect to all parameters of the network can be done by propagating error vectors in backward direction through the network. To this end, it is necessary to store the derivatives of the activation vectors $f'(\mathbf{a}_k)$ and the input vectors $\mathbf{y}_k$ of each layer during the forward propagation of the samples through the network.

So far, the derivation considers only a single input sample. Gradients due to M samples can be computed by averaging the computed gradients over all the samples, i.e.,

$$\frac{\partial E}{\partial \mathbf{W}_k} = \frac{1}{M}\sum_{t=1}^{M}\frac{\partial E_t}{\partial \mathbf{W}_k}, \quad \frac{\partial E}{\partial \mathbf{b}_k} = \frac{1}{M}\sum_{t=1}^{M}\frac{\partial E_t}{\partial \mathbf{b}_k}. \tag{4.8}$$

4.2 Deep Recurrent Neural Networks

In a recurrent neural network, the input no longer propagates only in the forward direction opposed to the feedforward architecture. A recurrent input-output relation is realized in the network by introducing a feedback path. Several architectures can be considered for the recurrent path. In this thesis, we consider replacing a single layer of a deep feedforward neural network by a recurrent layer. The recurrent layer is constructed in such a way that activations at the recurrent layer take a recurrent input which is the previous output of the recurrent layer.

A similar recurrent architecture for instance was used for speech recognition task in [24].

4.2.1 Network description

Since the description of a feedforward layer has already been provided, we will in the following only describe the recurrent layer. Assuming that the kth layer is a recurrent layer, we can describe the recurrent layer by

$$\mathbf{a}_k^t = \mathbf{W}_k \, \mathbf{y}_{k-1}^t + \mathbf{b}_k + \mathbf{W}_{kk} \, \mathbf{y}_k^{t-1} \tag{4.9a}$$

$$\mathbf{y}_k^t = f(\mathbf{a}_k^t), \tag{4.9b}$$

where t denotes the sample index, $\mathbf{W}_k$ and $\mathbf{W}_{kk}$ are the feedforward and recurrent weight matrices, respectively. To illustrate, Figure 4.3 shows a recurrent neural network architecture consisting of K layers where the second layer is a recurrent layer.

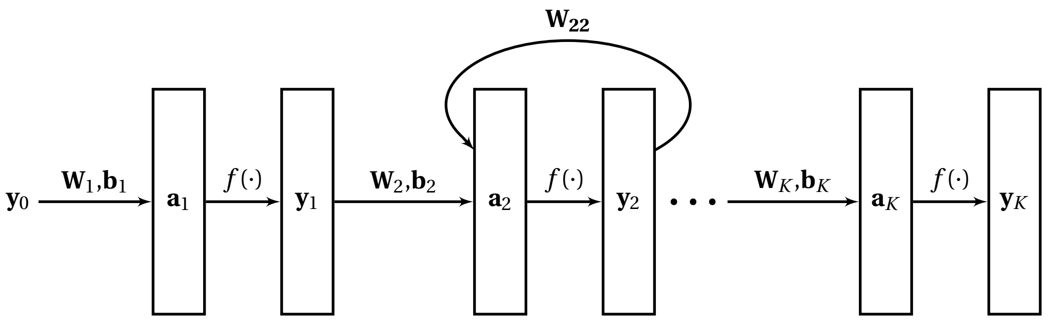


Figure 4.3: Illustration of recurrent neural network architecture with K layers where the second layer is a recurrent layer

4.2.2 Backpropagation through time algorithm

Due to the recurrent layer added to the network, the standard error backpropagation has to be modified to work with a recurrent architecture. Hence, in order to compute the gradients, we consider the backpropagation through time algorithm (BPTT) which is a generalization of the error backpropagation algorithm considering recurrent neural networks, see [25] and [26]. The idea behind the BPTT is to unfold the network in time so that the architecture looks like a feedforward architecture. Thus, the standard error backpropagation can be applied to the unfolded feedforward architecture.

The vanishing and exploding gradients problem arises in training recurrent neural networks due to the recurrent path introduced in the network [27]. Considering vanishing and exploding gradients problem, Mikolov uses a truncated BPTT approach where unfolding the network is truncated up to some maximum time steps on the order of five in the context of language modeling [28]. In this thesis, we also consider a truncated BPTT approach for training of deep recurrent neural network architecture, where we use the truncated BPTT to compute the gradients of the cost with respect to weight and bias terms up to and including the recurrent layer. Truncated BPTT was also used for instance in the context of a deep recurrent architecture where truncated BPTT is utilized for computation of the gradient of the cost with respect to recurrent layer weight matrix $\mathbf{W}_{kk}$ in [24].

We first assume that the last layer, namely the K-th layer, is the recurrent layer in the network. Note, however, that we will later generalize to the case when the recurrent layer is at an arbitrary position.

Assume that there are M training samples where the samples are provided to the network starting from sample 1 up to sample M. Similar to feedforward case we start from expanding the cost term E_t due to the output $\mathbf{y}_K^t$ by expanding the recurrent term as

$$\begin{aligned}
E_t &= \left\|\mathbf{y}_K^t - \mathbf{d}^t\right\|^2/2 && (4.10a)\\
&= \left\| f(\mathbf{W}_K\,\mathbf{y}_{K-1}^t + \mathbf{b}_K + \mathbf{W}_{KK}\,\mathbf{y}_K^{t-1}) - \mathbf{d}^t\right\|^2/2 && (4.10b)\\
&= \left\| f(\mathbf{W}_K\,\mathbf{y}_{K-1}^t + \underbrace{\mathbf{b}_K}_{\text{A}} + \mathbf{W}_{KK} f(\mathbf{W}_K\,\mathbf{y}_{K-1}^{t-1} + \underbrace{\mathbf{b}_K}_{\text{B}} + \mathbf{W}_{KK}\,\mathbf{y}_K^{t-2})) - \mathbf{d}^t\right\|^2/2 && (4.10c)\\
&= \ldots
\end{aligned}$$

Bias term $\mathbf{b}_K$ in equation (4.10c) appears twice underlined by the letters A and B. If we continue expanding the recurrent term, it can be seen that the true gradient depends on the samples 1 up to t. Hence, true gradient can be obtained by expanding the cost until the first sample. However, since we consider truncated BPTT [28], the expansion is truncated after T time steps.

In order to obtain all the gradients, we first write the gradient of the cost with respect to $\mathbf{b}_K$. To this end, we define the error vectors at the recurrent layer as the gradients due to term A as $\boldsymbol{\epsilon}_K^t(\tau = 0)$, due to term B as $\boldsymbol{\epsilon}_K^t(\tau = 1)$ and going up to $\tau = T$, where τ stands for the backward time steps. These error terms in fact corresponds to the error vectors in the unfolded network where τ corresponds to the unfolding time step. Using similar approach as in the feedforward case, these error vectors backpropagating through time can be described by

$$\boldsymbol{\epsilon}_K^t(\tau) = \begin{cases} \operatorname{diag}\big(f'(\mathbf{a}_K^t)\big)\,\big(\mathbf{y}_K^t - \mathbf{d}^t\big) & \tau = 0\\ \operatorname{diag}\big(f'(\mathbf{a}_K^{t-\tau})\big)\,\mathbf{W}_{KK}^T\,\boldsymbol{\epsilon}_K^t(\tau-1) & 0 < \tau \le T. \end{cases} \quad (4.11)$$

If instead of the last layer, the k-th layer with $k < K$ is the recurrent layer, then the $\tau = 0$ case should use the error that backpropagates from the layer $k+1$. Hence, the backpropagating through time equations should be updated in the case of recurrent layer being at kth layer as

$$\boldsymbol{\epsilon}_k^t(\tau) = \begin{cases} \operatorname{diag}\big(f'(\mathbf{a}_k^t)\big)\, \mathbf{W}_{k+1}^T\, \boldsymbol{\epsilon}_{k+1} & \tau = 0 \\ \operatorname{diag}\big(f'(\mathbf{a}_k^{t-\tau})\big)\, \mathbf{W}_{kk}^T\, \boldsymbol{\epsilon}_k^t(\tau-1) & 0 < \tau \leq T. \end{cases} \tag{4.12}$$

Moreover, the computation of gradients at layers before the recurrent is also affected by the recurrent path in the network. This can be seen from equation (4.10c) where there are two terms involving $\mathbf{y}_{K-1}$. These terms are in fact functions of $\{\mathbf{W}_1, \ldots, \mathbf{W}_{K-1}\}$ and $\{\mathbf{b}_1, \ldots, \mathbf{b}_{K-1}\}$. Therefore, layers before the recurrent layer should also consider truncated BPTT in order to compute the gradients of the cost. Hence, in order to apply truncated BPTT for the layers prior to recurrent layer, we can define the following error vectors at layers before the recurrent layer as

$$\boldsymbol{\epsilon}_{k-l}^t(\tau) = \operatorname{diag}\big(f'(\mathbf{a}_{k-l}^{t-\tau})\big)\, \mathbf{W}_{k-l+1}^T\, \boldsymbol{\epsilon}_{k-l+1}^t(\tau), \quad l = 1, 2, \ldots, \tag{4.13}$$

where kth layer is the recurrent layer. To sum up, Figure 4.4 shows how error vectors backpropagate through time steps at the recurrent layer and layers before the recurrent layer. This figure in fact summarizes (4.12) and (4.13) when $T = 2$ by showing layers $k-1$, k and $k+1$.

Using the error vector definitions, gradients of the cost with respect to parameters of the network considering truncated BPTT can be computed. Thus, gradient of the cost with respect to the bias vector at the recurrent layer or at a layer before the recurrent layer can be written as

$$\frac{\partial E_t}{\partial \mathbf{b}_k} = \sum_{\tau=0}^{T} \boldsymbol{\epsilon}_k^t(\tau). \tag{4.14}$$

In a similar manner, gradient of the cost with respect to the feedforward weight matrix at the recurrent layer or with respect to a weight matrix at a layer before the recurrent layer can be written as

$$\frac{\partial E_t}{\partial \mathbf{W}_k} = \sum_{\tau=0}^{T} \boldsymbol{\epsilon}_k^t(\tau)\, \big(\mathbf{y}_{k-1}^{t-\tau}\big)^T. \tag{4.15}$$

Finally, the gradient of the cost with respect to the recurrent weight matrix $\mathbf{W}_{kk}$ at the recurrent

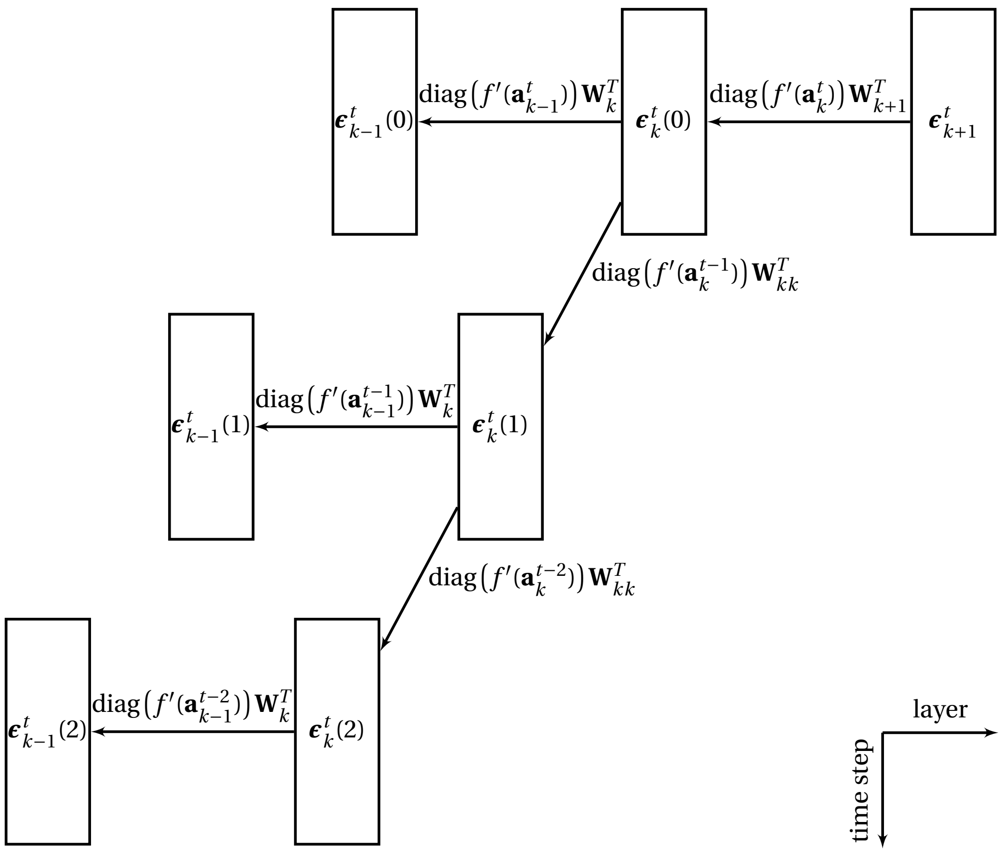

$\boldsymbol{\epsilon}_{k-1}^t(0)$
$\mathrm{diag}\big(f'(\mathbf{a}_{k-1}^t)\big)\mathbf{W}_k^T$
$\boldsymbol{\epsilon}_k^t(0)$
$\mathrm{diag}\big(f'(\mathbf{a}_k^t)\big)\mathbf{W}_{k+1}^T$
$\boldsymbol{\epsilon}_{k+1}^t$
$\mathrm{diag}\big(f'(\mathbf{a}_k^{t-1})\big)\mathbf{W}_{kk}^T$
$\boldsymbol{\epsilon}_{k-1}^t(1)$
$\mathrm{diag}\big(f'(\mathbf{a}_{k-1}^{t-1})\big)\mathbf{W}_k^T$
$\boldsymbol{\epsilon}_k^t(1)$
$\mathrm{diag}\big(f'(\mathbf{a}_k^{t-2})\big)\mathbf{W}_{kk}^T$
$\boldsymbol{\epsilon}_{k-1}^t(2)$
$\mathrm{diag}\big(f'(\mathbf{a}_{k-1}^{t-2})\big)\mathbf{W}_k^T$
$\boldsymbol{\epsilon}_k^t(2)$
layer
time step

Figure 4.4: Backpropagation of error vectors when the kth layer is a recurrent layer. Two backward time steps and the error vectors at layers $k-1$, k and $k+1$ are shown.

layer can be expressed as

$$\frac{\partial E_t}{\partial \mathbf{W}_{kk}} = \sum_{\tau=0}^{T} \boldsymbol{\epsilon}_k^t(\tau)\, \left(\mathbf{y}_k^{t-\tau-1}\right)^T. \tag{4.16}$$

So far, we derived the gradients of a single cost term E_t which is due to the error made at the output $\mathbf{y}_K^t$. In order to compute the gradient of cost E due to all M input samples, one can take the average of the gradients when t is varied between 1 to M, which can be written as

$$\frac{\partial E}{\partial \mathbf{W}_k} = \frac{1}{M}\sum_{t=1}^{M}\frac{\partial E_t}{\partial \mathbf{W}_k}, \quad \frac{\partial E}{\partial \mathbf{W}_{kk}} = \frac{1}{M}\sum_{t=1}^{M}\frac{\partial E_t}{\partial \mathbf{W}_{kk}}, \quad \frac{\partial E}{\partial \mathbf{b}_k} = \frac{1}{M}\sum_{t=1}^{M}\frac{\partial E_t}{\partial \mathbf{b}_k}. \tag{4.17}$$

Note that whenever a term in the above summations needs to access a term at a sample index not in the range $[1, M]$ which happens for instance for the first sample, the corresponding term can be set to zero.

Moreover, for the layers after the recurrent layer, the standard backpropagation as in the feedforward case can be used since the recurrent path does not affect backpropagation of errors for the layers after the recurrent layer.

Vanishing and exploding gradients problem

In short, the vanishing gradient problem refers to the fact that gradient contributions due to large unfolding time steps vanish while in the case of exploding gradient problem, those contributions explode [27]. According to [29], if the absolute largest eigenvalue of the recurrent weight matrix $\mathbf{W}_{kk}$ is smaller than some α where the α is the upper bound on absolute values of the derivatives of the activation vectors $f'(\mathbf{a}_k)$, then the vanishing gradient problem arise. They show this result by using the properties of spectral radius of matrices in [29]. It follows that in order for the exploding gradient problem to arise, the absolute largest eigenvalue of recurrent weight should be larger than α, which is a necessary condition. In our case, the activation function we use is the rectified linear units which have derivatives either 0 or 1. Therefore, for the recurrent network used in this thesis, the bound α is equal to 1.

5 DNN Based Source Separation

In order to separate single-channel music mixtures, deep neural network architectures, particularly feedforward and recurrent architectures explained earlier can be used. The investigation of DNN based source separation starts from the architecture considered in [6]. Hence, we first briefly explain this feedforward neural network based source separation framework in Sec. 5.1. Then, the next section provides the description of the aspects that this thesis investigates.

5.1 Deep Feedforward Neural Network Based Source Separation

This section summarizes the DNN based source separation framework from [6]. The explanation of the system architecture followed by the description of the DNN training are provided.

5.1.1 System architecture

In [6], a feedforward neural network is trained in such a way that, given a mixture signal containing instrumental sounds where the played instruments are known by the network, the network recovers the desired instrument. Hence, if the objective is to recover all the sources in the mixture, separate DNNs are trained so that each one outputs an estimate for the corresponding source.

The DNN based source separation works on the magnitude STFT spectrogram of the mixture signal. Note that since the mixture signals are real, it is sufficient to only use $N/2+1$ of the frequency bins for an N-point DFT based STFT assuming that N is even. If we denote the magnitude STFT spectrogram of the mixture signal $x[n]$ by $\mathbf{X}$ whose column vectors are denoted by $\mathbf{x}_i$ where i stands for the frame index, the trained DNN gives an estimate of the magnitude spectrogram $\hat{\mathbf{S}}$ for the target source whose column vectors are denoted by $\hat{\mathbf{s}}_i$.

Figure 5.1 summarizes the feedforward network based source separation architecture. The network takes the ith magnitude frame together with some contextual nonoverlapping frames before and after the ith frame of the mixture spectrogram as input. Then, the network tries to recover the ith magnitude frame of the target instrument at the output. Once the forward propagation through the network is applied for all time frame index i, we obtain a magnitude spectrogram estimate of the target source. Finally, the estimated magnitude spectrogram is combined with the phase of the original mixture spectrogram after which an ISTFT is applied to recover the time domain source signal $\hat{s}[n]$.

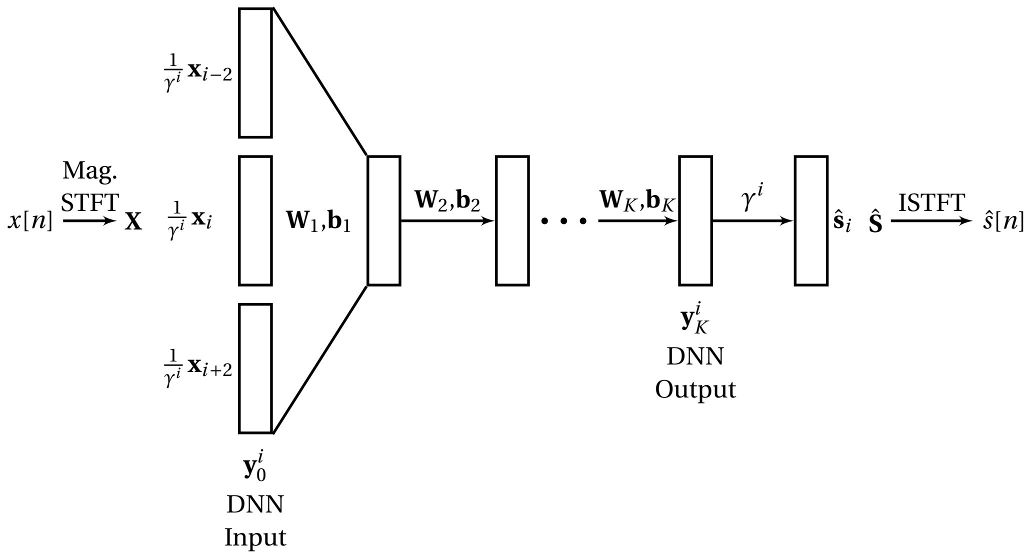

$\frac{1}{\gamma^i}\mathbf{x}_{i-2}$
Mag. STFT
$x[n]$
$\mathbf{X}$
$\frac{1}{\gamma^i}\mathbf{x}_i$
$\mathbf{W}_1,\mathbf{b}_1$
$\mathbf{W}_2,\mathbf{b}_2$
$\mathbf{W}_K,\mathbf{b}_K$
γ^i
$\hat{\mathbf{s}}_i$
$\hat{\mathbf{S}}$
ISTFT
$\hat{s}[n]$
$\frac{1}{\gamma^i}\mathbf{x}_{i+2}$
$\mathbf{y}_K^i$
DNN Output
$\mathbf{y}_0^i$
DNN Input

Figure 5.1: Feedforward neural network based source separation architecture assuming fifty percent overlap of STFT frames. The network uses one nonoverlapping contextual frame in both directions.

The feedforward network consists of K rectified linear unit layers which we can describe by

$$\mathbf{y}_k^i = \max(\mathbf{W}_k\,\mathbf{y}_{k-1}^i + \mathbf{b}_k, \mathbf{0}), \quad k = 1, 2, \ldots, K, \tag{5.1}$$

where $\mathbf{y}_0^i$ is the input to the network, $\mathbf{y}_k^i$ is the output of the kth layer, $\mathbf{0}$ is all zeros vector of the same size as $\mathbf{y}_k^i$, and $\max(\cdot)$ is the activation function returning element-wise maximum of its arguments. The superscript i denotes the frame index as previously defined. Moreover, if the number of contextual frames is denoted by $2C$ (C frames before the center frame and C frames after the center frame), the weight matrix of the first layer has the size $(N/2+1) \times (2C+1)(N/2+1)$. On the other hand, all other weight matrices are of size $(N/2+1) \times (N/2+1)$, and the bias vectors are of size $(N/2+1) \times 1$.

Note also that there is a normalization term γ^i prior to the DNN which is then used to denormalize the DNN output. The normalization term is the average Euclidean norm of the input magnitude frames at frame index i which helps the DNN to be able to separate input mixture frames with arbitrary energy.

Moreover, a common practice in single-channel source separation is to use the source estimates as spectral masks applied to the mixture spectrogram with the aim to improve the source separation performance. The use of spectral masks has also the advantage that all source estimates add up to the original mixture. The computation of the spectral mask for a specific source requires the computation of all the source estimates. Since a single DNN gives an estimate only for the target instrument we need to train separate DNNs, each one estimating a specific source in the mixture. Once all the source estimates are obtained, the spectral masks for all the sources can be calculated. In [6] and also in this thesis, Wiener spectral mask which is described in the following is considered. Assuming that there are J sources in the mixture, the Wiener spectral mask for estimating the jth source can be written as

$$\mathbf{H}_j = \frac{\hat{\mathbf{S}}_j^{.\wedge 2}}{\sum_{j'=1}^{J} \hat{\mathbf{S}}_{j'}^{.\wedge 2}}, \tag{5.2}$$

where the division is element-wise, and $()^{.\wedge 2}$ denotes element-wise squaring. Given the spectral mask $\mathbf{H}_j$ for jth source, the corresponding source estimate after Wiener filter (WF) is applied can be written as

$$\hat{s}_j[n] = \mathrm{ISTFT}\left(\mathbf{H}_j \,.\!*\, \mathrm{STFT}(x[n])\right). \tag{5.3}$$

5.1.2 DNN training

In order to train the network, input and target pairs are needed where the input is a superposition of all instruments in the mixture and the target is the contribution due to the target instrument. In order to obtain these input and target pairs, solo performances of desired instrument and background instruments are used. In order for the network to be able learn to separate arbitrary mixtures, a large database of solo performances is needed. To this end, solo performances are used from various sources giving a database of solo performances for the instruments of interest. In order to generate input-target pairs, solo music from target instrument and background instruments are randomly mixed yielding the input samples, and the contribution due to the target instrument gives the target sample.

Before explaining the training data generation approach in more details, we first define a column stacking operation V which takes a matrix $\mathbf{X}$, frame index i and number of context frames C, and returns a column vector expressed as

$$V(\mathbf{X}, i, C) = \begin{bmatrix} \mathbf{x}_{i-2C} \\ \vdots \\ \mathbf{x}_{i-2} \\ \mathbf{x}_{i} \\ \mathbf{x}_{i+2} \\ \vdots \\ \mathbf{x}_{i+2C} \end{bmatrix} \in \mathbb{R}^{(2C+1)(N/2+1)\times 1}. \tag{5.4}$$

Assuming that the magnitude spectrogram obtained by taking STFT of solo performances of the jth instrument is denoted by $\mathbf{S}_j$ where in particular $j = 1$ is the target instrument, the tth input sample is written as

$$\mathbf{y}_0^t = (1/\gamma^t)\left(\alpha_1^t\, V(\mathbf{S}_1, i_1^t, C) + \sum_{j=2}^{J} \alpha_j^t\, V(\mathbf{S}_j, i_j^t, C)\right), \tag{5.5}$$

where γ^t is the average Euclidean norm of the $2C+1$ frames in $\mathbf{y}_0^t$, the scale α_j^t for the jth instrument is taken from a uniform distribution on the interval $[0.01, 1]$ for both target instrument and background instruments, and the center frame index i_j^t for the jth instrument is chosen randomly for all instruments. By taking the amplitudes randomly for each training sample, mixture samples with different contribution scales from different instruments are obtained. The corresponding desired output of the network then becomes

$$\mathbf{d}^t = (1/\gamma^t)\alpha_1^t\, V(\mathbf{S}_1, i_1^t, 0). \tag{5.6}$$

The sample generation process is repeated with a large M giving us input-target pairs to be used in the training. The training then uses a sum of squared errors cost function that was given in chapter 4, and applies limited-memory BFGS (L-BFGS) iterations to learn the parameters of the network. Layer-wise training is applied, namely starting from the first layer one layer is added at a time after which some L-BFGS iterations are applied to learn the weight and the bias terms of the layers up to and including the layer that is added. Once the network is complete, a final training, so called fine tuning, is applied.

When a new layer is added, a least squares initialization without taking the activation function into account is used. The initialization is formulated as an optimization problem which can be expressed as

$$\{\mathbf{W}_k^{\text{init}}, \mathbf{b}_k^{\text{init}}\} = \underset{\mathbf{W}_k, \mathbf{b}_k}{\arg\min} \sum_{t=1}^{M} \left\| \mathbf{W}_k \, \mathbf{y}_{k-1}^t + \mathbf{b}_k - \mathbf{d}^t \right\|^2. \tag{5.7}$$

Since the nonlinear activation function that prevents obtaining a closed form solution is discarded, we can get a closed form solution of the above optimization problem. Thus, the least squares initialization of the kth layer weight matrix $\mathbf{W}_k$ and bias vector $\mathbf{b}_k$ can be written as

$$\mathbf{W}_k^{\text{init}} = \mathbf{C}_{dy} \, \mathbf{C}_{yy}^{-1}, \qquad \mathbf{b}_k^{\text{init}} = \bar{\mathbf{d}} - \mathbf{W}_k^{\text{init}} \, \bar{\mathbf{y}}_{k-1}, \tag{5.8}$$

where

$$\mathbf{C}_{dy} = \sum_{t=1}^{M} \left(\mathbf{d}^t - \bar{\mathbf{d}}\right) \left(\mathbf{y}_{k-1}^t - \bar{\mathbf{y}}_{k-1}\right)^T, \qquad \mathbf{C}_{yy} = \sum_{t=1}^{M} \left(\mathbf{y}_{k-1}^t - \bar{\mathbf{y}}_{k-1}\right) \left(\mathbf{y}_{k-1}^t - \bar{\mathbf{y}}_{k-1}\right)^T,$$

$\bar{\mathbf{y}}_{k-1} = \frac{1}{M} \sum_{t=1}^{M} \mathbf{y}_{k-1}^t$ corresponds to the mean vector of $\mathbf{y}_{k-1}^t$, and $\bar{\mathbf{d}} = \sum_{t=1}^{M} \sum_1^M \mathbf{d}^t$ corresponds to the mean vector of $\mathbf{d}^t$. For the derivation of this least squares initialization solution, please refer to appendix A.

5.2 Investigation Topics

We give here the topics that are investigated in this thesis. First of all, a time-frequency representation that combines STFT and CQT is proposed in order to reduce the computational complexity of the network. Secondly, a recurrent neural network based source separation scheme is considered. Finally, a music score based music generation is investigated such that the obtained music can be used for the DNN training.

5.2.1 CQT based compression of STFT magnitude frames

This investigation topic focuses on reducing the computational complexity of the DNN based source separation using the CQT to compress the STFT magnitude frames. The CQT where the frequency bins are geometrically spaced is in fact suitable for instrumental source separation since frequencies of musical notes are also geometrically spaced in Western classical music. The CQT is thus utilized in various audio source separation frameworks [30–33]. Note, however, that our approach mainly focuses on compression of the STFT magnitude frames benefiting from the ideas in the CQT.

We propose using a time-frequency representation that has linearly spaced frequency bins for low frequencies while it has geometrically spaced frequency bins for high frequencies. The goal here is to reduce the size of input and output feature vectors of the DNN with the aim to reduce the computational complexity. This way, DNN training time can be reduced while other parameters such as the number of training samples M and the number of context frames C are kept the same. This representation can be considered as a hybrid of STFT and CQT because linearly spaced frequency bins correspond to the STFT bins while geometrically spaced frequency bins correspond to the CQT bins. The value of the transition frequency between linearly and geometrically spaced bins becomes a design choice that will be discussed in the following. We call this representation hybrid STFT/CQT being hybrid of the CQT and the STFT.

Hybrid STFT/CQT

Since the DNN works in magnitude STFT domain, it is sufficient to use a time-frequency representation that takes only the magnitude components into account. In order to obtain the hybrid STFT/CQT representation of a discrete time signal, first the STFT of the signal is computed. Given the discrete time signal $x[n]$, we first compute the N-point DFT based STFT from which we keep only the magnitude information represented by the matrix $\mathbf{X}$. The magnitude STFT matrix can be compactly represented by $N/2+1$ frequency bins assuming that N is even since the time domain signal is real. Then, the magnitude STFT spectrogram $\mathbf{X}$ is multiplied by a kernel matrix $\mathbf{C}$ which is defined as

$$\mathbf{C} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{M} \end{bmatrix}, \tag{5.9}$$

where $\mathbf{I}$ is an identity matrix that keeps the linearly spaced frequency bins due to the STFT, and $\mathbf{M}$ is a nonnegative matrix obtained from the CQT frequency domain kernel that applies compression to high frequency bins of the STFT. The computation of the matrix $\mathbf{M}$ involves the steps described in the following pseudocode.

Algorithm 1 Pseudocode to obtain the hybrid STFT/CQT compression matrix $\mathbf{M}$

```
Choose a number of frequency bins per octave $b$ which is an integer multiple of 12
$Q \leftarrow 1/(2^{1/b}-1)$
$f_{\min} \leftarrow$ floorToNearestNoteFreq($Qf_s/N$)
$f_{\max} \leftarrow$ floorToNearestNoteFreq($f_s/2$)
Compute the CQT frequency domain kernel $\mathbf{F}$ with above $Q$, $f_{\min}$ and $f_{\max}$
Keep only the rows of $\mathbf{F}$ between $\lfloor f_{\min}N/f_s \rfloor$ and $(N/2+1)$
$\mathbf{M} \leftarrow N|\mathbf{F}|^T$
```

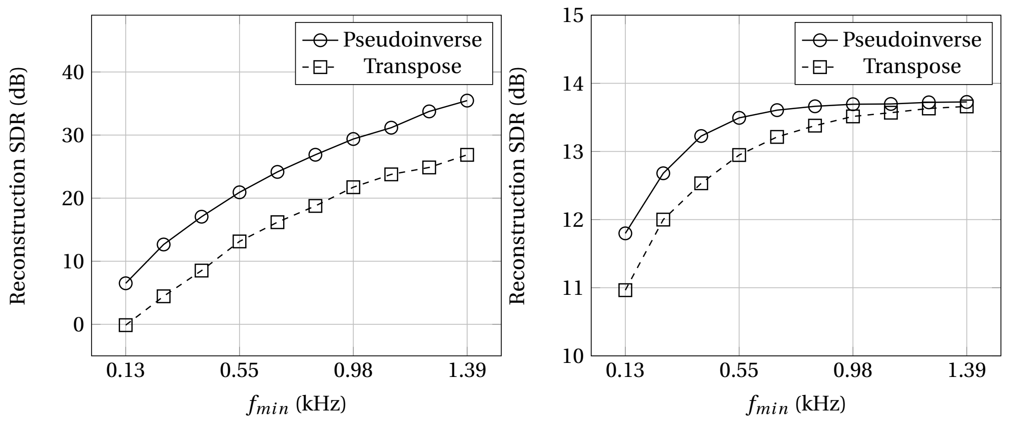
Given an initial choice of frequency bins per octave, the computation of matrix $\mathbf{M}$ can be done as described in algorithm 1. The choice of b sets the transition frequency $f_{\min}$ and the Q factor of the CQT frequency domain kernel. By this choice of the Q factor, the transition between the linearly spaced part and geometrically spaced part becomes smooth. The function floorToNearestNoteFreq(·) returns a frequency corresponding to a musical note frequency lower than its argument so that frequency bins of CQT kernel includes the musical note frequencies. After we compute the frequency domain CQT kernel, we truncate the rows of this kernel matrix where the lower bound is due to the fact that low frequency bins are already represented by the STFT bins, and higher bound is due to the fact that the magnitude STFT matrix has only $N/2+1$ bins. Next, we compute the element-wise magnitude values of this truncated kernel, which is then multiplied by N. The multiplication with N is added in order to make entries comparable in terms of its magnitude with the identity matrix. The computation of the CQT frequency domain kernel involves two normalization terms where the first normalization is due to definition and the second is due to Parseval relation. Since these two normalization results in a kernel with small magnitude values, we added the multiplicative N term. Finally, we take the transpose of this magnitude matrix to obtain the matrix $\mathbf{M}$ so that the hybrid STFT/CQT can be written as multiplication in the form $\mathbf{X}^{\text{hcqt}} = \mathbf{C}\,\mathbf{X}$.

In order to obtain the hybrid STFT/CQT kernel, we first need to set the number of frequency bins per octave parameter. This in fact determines the transition frequency $f_{\min}$ which is a design choice. When this transition frequency is too large, we end up low compression but high DNN computational complexity. On the other hand, if this frequency is too small, reconstruction quality would be low because of high compression. In order to determine a transition frequency which results in a reasonable compression rate without causing high reconstruction error, we performed the following experiment.

We consider different $f_{\min}$ values corresponding to number of frequency bins per octave b in the set $(12, 24, 36, \ldots, 96)$. We choose the STFT window size N equal to 4096 and the system sampling rate f_s equal to 32 kHz. For reconstruction of magnitude STFT spectrogram from the hybrid STFT/CQT, we use two approaches. The first one uses transpose of matrix $\mathbf{M}$ and the second one uses the pseudoinverse of $\mathbf{M}$. The transpose idea is taken from [34] where they consider the conjugate transpose for the inverse CQT. Conjugate transpose in our case corresponds to transpose since we take only the magnitude into account. We additionally consider the pseudoinverse and decide on the better approach based on the results in the following. We can describe these two inverse hybrid STFT/CQT matrices by

$$\mathbf{C}^{\text{inv}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}^T \end{bmatrix}, \qquad \mathbf{C}^{\text{inv}} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}^p \end{bmatrix}. \tag{5.10}$$

For different $f_{\min}$ values, we obtain the hybrid STFT/CQT kernel and the inverse hybrid STFT/CQT kernels based on the description above. Then, we apply the hybrid STFT/CQT followed by inverse hybrid STFT/CQT to the sources in the TRIOS dataset [1]. This dataset includes four mixtures together with the true source signals. Assuming that the mixture signal is denoted by $x[n]$, and the true sources are denoted by $s_1[n]$, $s_2[n]$, $s_3[n]$, the reconstruction gives $\tilde{s}_1[n]$, $\tilde{s}_2[n]$ and $\tilde{s}_3[n]$. We compute the reconstruction SDR values for all the sources in the TRIOS dataset and plot the mean of the reconstruction SDR for different values of f_{min}. From this dataset, we can also use the magnitude STFT spectrograms of $\tilde{s}_1[n]$, $\tilde{s}_2[n]$ and $\tilde{s}_3[n]$ as spectral mask applied to the mixture. Thus, we also plot the mean SDR values after a Wiener spectral mask is applied. Since in our source separation experiments, we also use Wiener filters with the aim to enhance the source separation performance, this reconstruction plot gives us an idea about how much we lose for source separation when we apply the hybrid STFT/CQT compression. We plot the results for both transpose and pseudoinverse inverses to see their reconstruction performance.



(a) Raw reconstruction

(b) Reconstruction used as Wiener spectral mask applied to the mixture

Figure 5.2: Average reconstruction SDR values for the TRIOS dataset sources [1] (f_{min} corresponds to the transition frequency from the STFT bins to the CQT bins)

From Figure 5.2, one can see that the pseudoinverse results in higher reconstruction SDR compared to transpose for the same f_{min}. Therefore, we choose the pseudoinverse for the inverse hybrid STFT/CQT. Moreover, when we look at the Wiener filter plot, we can observe a saturation after which SDR values do not improve even if the compression is reduced. This saturation is in fact expected because even if the true sources are used to construct the spectral masks, we cannot obtain perfect reconstruction of the true sources by applying the spectral masks. This saturation corresponds to best that can be achieved with this specific type of spectral mask when the true sources are used to construct the spectral masks. Noting that saturation starts around 1 kHz for the pseudoinverse case, we use f_{min} as 830 Hz and the corresponding number of frequency bins per octave b as 72 in our source separation experiments. We think that as a result of the performed experiment above, this choice of transition frequency is a good compromise between a reasonable data size reduction and low reconstruction error.

System architecture

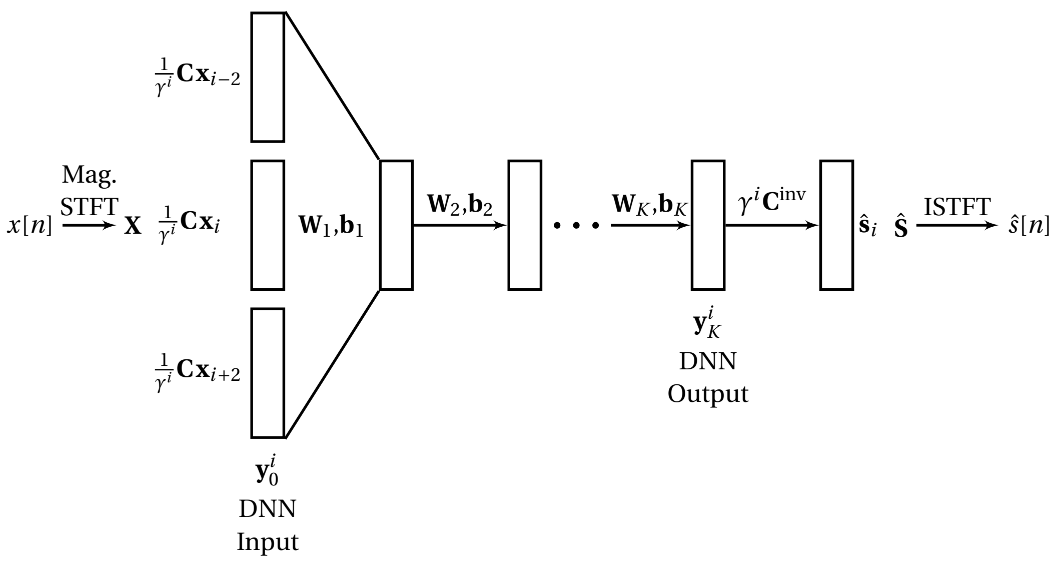

$\frac{1}{\gamma^i}\mathbf{C}\mathbf{x}_{i-2}$
Mag. STFT
$x[n]$
$\mathbf{X}$
$\frac{1}{\gamma^i}\mathbf{C}\mathbf{x}_i$
$\mathbf{W}_1,\mathbf{b}_1$
$\mathbf{W}_2,\mathbf{b}_2$
$\mathbf{W}_K,\mathbf{b}_K$
$\gamma^i\mathbf{C}^{\text{inv}}$
$\hat{\mathbf{s}}_i$
$\hat{\mathbf{S}}$
ISTFT
$\hat{s}[n]$
$\frac{1}{\gamma^i}\mathbf{C}\mathbf{x}_{i+2}$
$\mathbf{y}_K^i$ DNN Output
$\mathbf{y}_0^i$ DNN Input

Figure 5.3: Feedforward neural network based source separation when the STFT magnitude frames are compressed by using the hybrid STFT/CQT (assuming fifty percent overlap of STFT frames)

Figure 5.3 shows the DNN based source separation architecture when the hybrid STFT/CQT based compression is applied. The main difference with the previously described DNN based separation is that magnitude frames are compressed with a hybrid STFT/CQT kernel matrix $\mathbf{C}$ at the input of the DNN. The output of the DNN is then multiplied by $\mathbf{C}^{\text{inv}}$ in order to obtain magnitude STFT spectrogram estimate of the target source. Once the magnitude spectrogram estimate is obtained, it is combined with the original mixture phase after which an ISTFT method is applied to recover the target source.

The described complexity reduction approach can be utilized in different ways. First of all, an idea would be to reduce the complexity without increasing the number of training samples which in turn reduces the training time. A second idea would be to reduce the input feature vector size and increase the number of training samples, which in turn gives us the flexibility of using more training samples. Finally, one can increase the frame duration which favors the large frequency resolution for low frequencies then apply the compression in order to make the DNN training feasible. In this thesis, we use a setting that both reduces the training time and have large frame duration favoring high frequency resolution for low frequencies.

5.2.2 Deep recurrent neural network architecture

In this investigation topic, we replace the feedforward network architecture with a recurrent architecture to solve the source separation problem. For the recurrent network, we consider the architecture that was described in chapter 4. In the feedforward network based architecture, contextual frames are employed in order to capture the temporal structure of the spectrogram. In the case of recurrent neural network architecture, we replace the idea of contextual frames by using a recurrent network to represent temporal dependencies of magnitude spectrogram frames knowing that recurrent networks are good at capturing temporal dependencies. We can describe the recurrent network by

$$\mathbf{y}_k^i = \begin{cases} \max(\mathbf{W}_k \mathbf{y}_{k-1}^i + \mathbf{b}_k, \mathbf{0}), & \text{if } k\text{th layer is a feedforward layer} \\ \max(\mathbf{W}_k \mathbf{y}_{k-1}^i + \mathbf{W}_{kk} \mathbf{y}_k^{i-1} + \mathbf{b}_k, \mathbf{0}), & \text{if } k\text{th layer is a recurrent layer.} \end{cases} \tag{5.11}$$

Note that there is a single recurrent layer in the network and the remaining layers are feedforward layers.

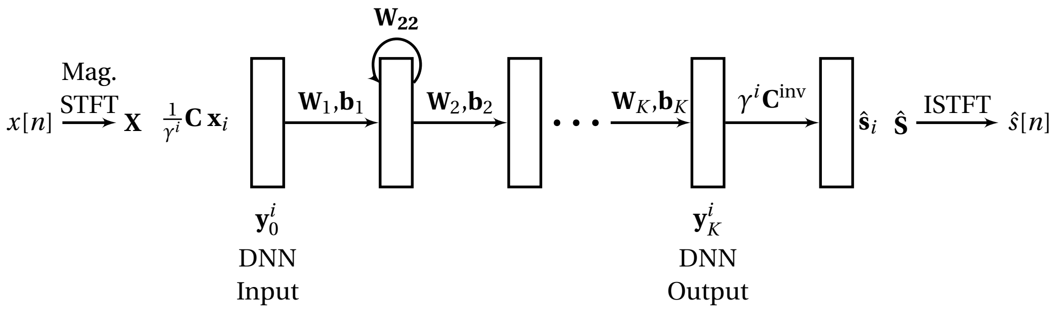

$\mathbf{W_{22}}$
Mag. STFT
$x[n]$
$\mathbf{X}$
$\frac{1}{\gamma^i}\mathbf{C}\,\mathbf{x}_i$
$\mathbf{W}_1,\mathbf{b}_1$
$\mathbf{W}_2,\mathbf{b}_2$
$\mathbf{W}_K,\mathbf{b}_K$
$\gamma^i\mathbf{C}^{\text{inv}}$
$\hat{\mathbf{s}}_i$
$\hat{\mathbf{S}}$
ISTFT
$\hat{s}[n]$
$\mathbf{y}_0^i$
DNN Input
$\mathbf{y}_K^i$
DNN Output

Figure 5.4: Recurrent neural network based source separation where the STFT magnitude frames are compressed by using the hybrid STFT/CQT and second layer is the recurrent layer

To illustrate, Figure 5.4 shows the recurrent network based source separation scheme. Since we noticed improvement with the hybrid STFT/CQT based compression approach, which we will discuss in chapter 7, we decided to use this approach in the case of recurrent network based source separation.

In the feedforward neural network based training, while generating training samples for each training sample, a random center frame index and random amplitude for both target and background instruments are picked. However, since the recurrent architecture is used to learn the temporal structure of spectral sequences, neighboring samples should reflect the spectral sequence structure contained in the instrument database. That is why rather than picking a random amplitude and a random center frame index for each and every sample, we pick those random values for window of samples. Within these sample windows, frame indices for an instrument are linearly spaced and the amplitudes for an instrument are kept the same. Assuming that the sample window size is denoted by W, this training data generation process more conveniently can be described by algorithm 2.

In algorithm 2, $\mathrm{U}[0.01, 1]$ returns an amplitude from a uniform distribution between 0.01 and 1. Moreover, the functions randi$(\cdot)$ and size$(\cdot,\cdot)$ are adopted from Matlab notation where randi$(\cdot)$

Algorithm 2 Pseudocode for recurrent network training data generation

```
for $t = 1$ to $M$ do
  if mod $(t, W) = 1$ then
    for $j = 1$ to $J$ do
      $\alpha_j \leftarrow \mathrm{U}[0.01, 1]$
      $i_j \leftarrow \mathrm{randi}(\mathrm{size}(\mathbf{S}_j, 2) - W)$
    end for
  end if
  $\mathbf{y}_0^t = \sum_{j=1}^{J} \alpha_j V(\mathbf{S}_j, i_j + \mod(t, W) - 1, 0)/\gamma^t$
  $\mathbf{d}^t = \alpha_1 V(\mathbf{S}_1, i_1 + \mod(t, W) - 1, 0)/\gamma^t$
end for
```

returns a random integer between 1 and its integer argument, and size$(\cdot, 2)$ returns the number of column of its first argument.

In the feedforward case, the layer-wise training uses least squares initialization when a new layer is added. In the case of recurrent network, the same approach can be used for the feedforward layers. However, when the recurrent layer is added, least squares initialization cannot be obtained because of the recurrent path. Hence, we decided to use least squares initialization for $\mathbf{W}_k$ and $\mathbf{b}_k$ as if the layer is a feedforward layer. Then, the recurrent weight $\mathbf{W}_{kk}$ is simply initialized to an all zeros matrix. This initialization point in fact corresponds to feedforward least squares initialization point. Once the network training is started, through L-FBGS iterations both feedforward weights and recurrent weights gets updated, which results in a recurrent input-output relation.

5.2.3 MIDI based training material

As mentioned earlier, in order to generate training data for the DNN training we need a database of solo performances for the instruments contained in the mixture. In [6], real music from various sources are used to form the database. In this thesis, in addition to using real music as training material, we consider artificial music generation using a music score transcription database. In order to compare the separation performance, we use the real music and the music score based music in separate source separation experiments and give the corresponding results. Please note that the real music database used in this thesis is the same as the one in [6] (also additional music files were added to this music database at the time of this thesis work). Therefore, results are comparable.

Figure 5.5 illustrates the MIDI based solo performance generation. We first download music scores from the music score database [35] in MIDI file format. The MIDI file format includes the musical note transcription information together with some additional metadata information. From the MIDI file, we use the note transcription and the instrument information. The files in the music score database in MIDI format is first converted to comma separated values using the tool from [36]. Then, we use these comma separated values to generate

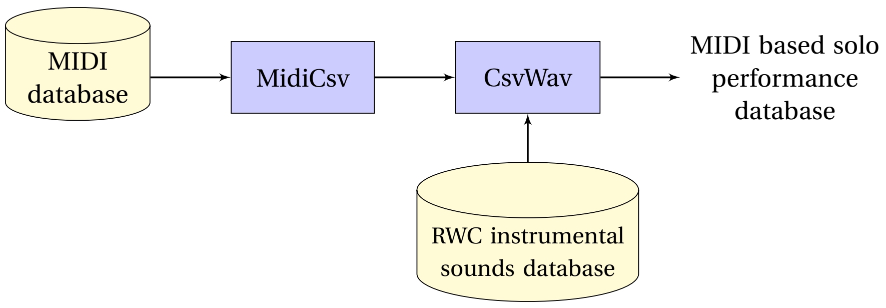

MIDI database
MidiCsv
CsvWav
MIDI based solo performance database
RWC instrumental sounds database

Figure 5.5: MIDI based solo performance generation

the waveforms in Matlab. The waveform generation block uses the waveforms from RWC database [37] for single notes while generating the music.

Algorithm 3 Pseudocode for generating MIDI based music

```
for All composers in the MIDI database do
    for All MIDI files for the corresponding composer do
        if MIDI metadata contains the corresponding instrument then
            Randomly pick a file from RWC database (sets the playing style and the manufacturer of the instrument)
            Randomly pick a resampling rate from the set {-1.5%, -0.75%, 0%, 0.75%, 1.5%}
            Apply the previously determined resampling rate to the RWC file
            Randomly pick a decay length from the uniform distribution in the interval [0 ms,200 ms]
            Generate the waveform given the MIDI note specification and the RWC file
        end if
    end for
end for
```

In order to generate the solo waveforms database, we go through the MIDI database and look for the MIDI transcriptions available for the corresponding instrument. If there is a MIDI file that contains a piece for the corresponding instrument, we start generating the waveform considering only the contribution for this instrument. While going through the database, we restrict to have pieces from a single composer to be less than one hour duration in total by looking at the metadata. This way, the generated waveforms do not focus on a single composer in case more material exists for a specific composer in the MIDI database. While generating a piece, we use the single note waveforms from RWC database by picking a random file for the corresponding instrument by which we set the playing style and the manufacturer. Then, we apply a random resampling factor to the single note waveforms, which we will discuss in the following. We also initially specify a decay length which is used when a note duration in the MIDI specification is smaller than the duration of the corresponding note in the RWC file. If that is the case, we apply a window that is constant in the beginning and decaying at the end to the single note waveforms from RWC file. This window is used in order to avoid “clic” sounds at the end of the notes because of discontinuities. Finally, we generate the music piece given the MIDI specification. All the generated waveforms are concatenated which gives us a database of solo performances for the corresponding instrument. By repeating this process for all instruments of interest, we obtain an artificially generated music database.

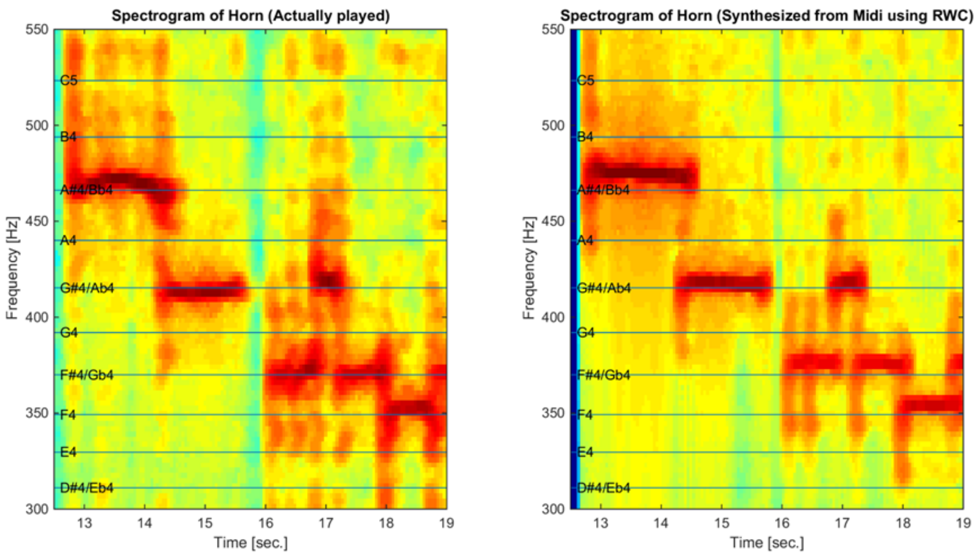


Figure 5.6: Spectrograms of an horn excerpt where the first spectrogram is the spectrogram of the actual piece while the second spectrogram is the spectrogram of the MIDI based piece

As mentioned earlier, we apply a random resampling rate to the single note waveforms whenever a new piece is generated. The purpose here is to help the DNN to be able to capture different center frequency tuning choices of the corresponding instrument. For instance, Figure 5.6 gives spectrogram of an horn excerpt taken from an actually played music piece together with the spectrogram of the same piece when generated using the MIDI based approach. We can see that for some notes the frequencies of the MIDI based music is shifted compared to actually played music. This shift is in fact related to small difference in tuning frequency choice of the artist. Since we notice that the single note waveforms in the RWC database do not cover various tuning frequency choices, we apply a random resampling factor so that the network can see different variations in terms of different tuning frequency choices.

Table 5.1: Training material durations for real music and MIDI based music

Instrument	**Real music**	**MIDI based music**
Bassoon	3.05 hours	1.12 hours
Cello	4.07 hours	6.00 hours
Horn	1.63 hours	0.75 hours
Piano	14.49 hours	7.12 hours
Trumpet	0.75 hours	0.54 hours
Violin	12.26 hours	7.33 hours

For comparison, Table 5.1 shows the durations of training material for different instruments used in the source separation experiments in this thesis. Having long durations is desired since it would help the DNN to be able to see large variations so that the DNN can better generalize to new mixtures.

6 Supervised NMF Based Source Separation

A popular method to solve the single-channel BASS problem is to make use of nonnegative matrix factorization. Nonnegative matrix factorization is applied to the magnitude of the time-frequency representation of the mixture signal in order to obtain a compact representation. The STFT is a commonly used time-frequency representation to perform NMF based source separation. Once NMF is applied to the magnitude STFT spectrogram of the mixture signal, there are different methods that can be used to perform the source separation. This thesis in particular focuses on supervised NMF based source separation. Before going into details of source separation using supervised NMF, first NMF is introduced.

6.1 Nonnegative Matrix Factorization

Assuming that the matrix $\mathbf{X} \in \mathbb{R}^{m \times n} \geq 0$ is a nonnegative matrix, the purpose of NMF is to find a factorization in the form

$$\mathbf{X} \approx \mathbf{BG} \quad \text{s.t.} \quad \mathbf{B} \in \mathbb{R}^{m \times k} \geq 0, \quad \mathbf{G} \in \mathbb{R}^{k \times n} \geq 0, \quad k \ll \min(m, n), \tag{6.1}$$

where $\mathbf{B}$ and $\mathbf{G}$ are nonnegative factorization matrices whose all elements are greater than or equal to zero. In the context of audio source separation, the matrix $\mathbf{X}$ is the magnitude spectrogram obtained by taking STFT of the mixture signal. The factorization simply tries to approximate the matrix $\mathbf{X}$ by the factorization matrices with rank much lower than the original mixture matrix rank with the aim to ease source separation. In the context of nonnegative matrix factorization for audio signals, the matrices $\mathbf{B}$ and $\mathbf{G}$ have special meanings. The matrix $\mathbf{B}$ is referred to as the basis matrix since its columns represent the frequency basis vectors corresponding to the mixture. The matrix $\mathbf{G}$ is identified as the gain matrix since its rows represent the gains of the corresponding frequency basis vector at each time index.

In order to solve the nonnegative matrix factorization problem, the main idea is to introduce a cost function to be minimized so that the decomposition approximates the original nonnegative matrix **X**. There are several cost functions such as the generalized Kullback–Leibler (KL) divergence and the Itakura-Saito (IS) divergence used in the context of audio source separation. Based on the statistical interpretation provided in [38], the KL divergence is preferred when the magnitude spectrogram is used while the IS divergence is preferred when the power spectrogram is considered. We in particular consider here the KL divergence cost function since we use the magnitude spectrogram in this thesis. The iterative update rules when the KL divergence cost function is taken into account was given by Lee and Seung in [39]. The cost function is defined as the divergence from **X** to **BG** where the divergence is expressed as

$$D(\mathbf{X}\,||\,\mathbf{BG}) = \sum_{ij} [\mathbf{X}]_{ij} \log \frac{[\mathbf{X}]_{ij}}{[\mathbf{BG}]_{ij}} - [\mathbf{X}]_{ij} + [\mathbf{BG}]_{ij}. \tag{6.2}$$

Hence, the optimization objective is to minimize the above cost function with the constraints that **B** and **G** are nonnegative matrices. In [39], multiplicative update rules for **B** and **G** are provided which can be written in the matrix form as

$$\mathbf{B} \leftarrow \mathbf{B} .* \frac{\frac{\mathbf{M}}{\mathbf{BG}}\mathbf{G}^T}{\mathbf{1}\mathbf{G}^T}, \qquad \mathbf{G} \leftarrow \mathbf{G} .* \frac{\mathbf{B}^T\frac{\mathbf{M}}{\mathbf{BG}}}{\mathbf{B}^T\mathbf{1}}, \tag{6.3}$$

where all divisions are element-wise, and **1** denotes an all ones matrix of the same size as **X**. The basis and gain matrices are randomly initialized to nonnegative matrices so that the method is guaranteed to give a nonnegative decomposition because of the multiplicative updates. Once the matrices are initialized, the update rules for the matrices **B** and **G** are applied in an alternating fashion. Alternating updates are applied up to some maximum number of iterations after which the factorization is obtained.

6.2 Supervised NMF

In this thesis, we examine supervised NMF based source separation so that the results can be compared with the DNN based source separation. In [7], a supervised NMF scheme is used in order to separate mixtures of speech and music signals while we use it for the instrumental source separation purpose. The essential idea in supervised NMF is to use a pretrained frequency basis matrix for the factorization of the magnitude spectrogram of the original mixture.

Assuming that the task is to separate a discrete time single-channel mixture observation denoted as $x[n]$ and that the corresponding magnitude spectrogram obtained by taking STFT is **X**, the standard NMF decomposes the magnitude matrix **X** into the frequency basis matrix

B and the gain matrix **G**. On the other hand, supervised NMF uses a pretrained frequency basis matrix. Determination of this basis matrix involves separate pretraining phases for each instrument using a database of audio material for the corresponding instrument. For each instrument, NMF is applied to the magnitude spectrogram of the solo performance of the corresponding instrument which gives a basis and a gain matrix. From this pretraining phase, only the basis matrix is kept to be used for the NMF on the mixture. The frequency basis matrix for the actual NMF on the mixture is initialized to the concatenation of the frequency basis matrices for each instrument, i.e. it can be expressed as

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{B}_J \end{bmatrix}, \tag{6.4}$$

where $\mathbf{B}_j$ corresponds to the frequency basis matrix obtained in the pretraining phase for the jth instrument in the mixture. The actual NMF on the mixture involves only the determination of the gain matrix. After NMF is applied to the mixture signal magnitude spectrogram, the factorization we obtain can be written as

$$\mathbf{X} \approx \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_2 & \cdots & \mathbf{B}_J \end{bmatrix} \begin{bmatrix} \mathbf{G}_1 \\ \mathbf{G}_2 \\ \vdots \\ \mathbf{G}_J \end{bmatrix}, \tag{6.5}$$

where $\mathbf{G}_j$ denotes the activation matrix of the jth instrument. Once the factorization is obtained, magnitude spectrogram estimate of the jth source becomes $\mathbf{B}_j\mathbf{G}_j$. We can use these magnitude spectrogram estimates to obtain the Wiener spectral masks as in the DNN based separation case. Thus, applying spectral masks to the original mixture spectrogram, we can get the spectrogram estimates of the sources from which we can obtain the time domain source estimates by applying an ISTFT method.

In order to determine the basis components in the pretraining phase, the KL divergence cost function described earlier is used as in [7]. In order to learn the gain matrix at the separation phase, we adopt the following two approaches. The first approach is to again use the KL divergence as in [7]. In the second approach, we use the constrained NMF from [8] where temporal continuity and sparsity constraints and the corresponding iterative update rules were proposed. These constraints are added as additional cost terms which depend only on the gain matrix. Since the last step of the supervised NMF is used to learn the gain matrix, we can apply the constrained NMF at the last step. This way it is possible to see the effect of these additional constraints on supervised NMF scheme. Note that one could also apply the constrained NMF to obtain the basis components in the pretraining phase but we do not consider this approach in this thesis. Using the same notation as in [8], the cost function they introduce is given by

$$c(\mathbf{B},\mathbf{G}) = c_r(\mathbf{B},\mathbf{G}) + \alpha c_t(\mathbf{G}) + \beta c_s(\mathbf{G}), \tag{6.6}$$

where $c_r(\mathbf{B},\mathbf{G})$ is the KL divergence reconstruction cost term, $c_t(\mathbf{G})$ and $c_s(\mathbf{G})$ are the temporal continuity and sparsity cost terms, respectively, and α and β are the weights applied to the corresponding cost terms which gives us the flexibility to put more emphasis on the one constraint more than the other.

7 Results and Discussion

In order to evaluate the performance of the different source separation methods described earlier, we apply them to three mixtures from the TRIOS dataset [1]. This dataset includes mixtures of three instruments from different composers together with the true source signals. The mixtures used in this thesis are a piece from "Brahms" containing a mixture of horn, piano and violin, a piece from "Lussier" containing a mixture of bassoon, piano and trumpet, and a piece from "Schubert" containing a mixture of cello, piano and violin.

7.1 Organization of the Results

The source separation results of each mixture are given in separate sections after which an overall summary of the results is provided. The following gives the structure of how the source separation results of a mixture are presented.

First of all, we provide the results of the feedforward network based source separation. These results include the outcome of the source separation experiments with the two time-frequency representations where the first one is the standard STFT and the second one is the hybrid STFT/CQT that we described in Sec. 5.2.1. Moreover, we perform the experiments for both time-frequency representations when the training dataset is real music and MIDI based music, which in total gives four different settings. These settings in fact comprise the two investigation topics of the DNN based separation in this thesis.

In a second step, we analyze the performance of the recurrent network based separation. For the recurrent network based case, we use the data from the real music database as the feedforward network performed the best with this dataset on all three mixtures. For the recurrent network, we consider two different configurations which are the configuration with the first layer being the recurrent layer and the configuration with the second layer being the recurrent layer. This way we can see the effect of the recurrent layer position on the separation performance since other parameters remain the same.

In order to see the effectiveness of the DNN based separation, supervised NMF based separation is investigated. Here, we first report the results when the update rules take only the KL divergence cost function into account, both in the pretraining and in the separation phases. Since the number of basis components affect the separation performance, we vary the number of components per instrument and then pick experimentally the best setting in terms of average SDR performance for the corresponding mixture. We also perform supervised NMF with temporal continuity and sparsity constrains where those constraints are used at the separation stage. This method uses the best number of basis components for the corresponding mixture from plain supervised NMF. The plain method and the constrained method based experiments are repeated for both training datasets.

Finally, we give an overall summary of the results for the corresponding mixture. The numerical results in the summary contain the results after a Wiener filter is applied since it improves the separation in most cases. Moreover, the results after fine tuning are reported in the summary for the DNN based methods. The summary consists of the following results:

- *Globally optimal NMF*: Supervised NMF where the number of basis components is fixed experimentally to the best setting in terms of average SDR over all the mixtures.
- *Locally optimal NMF*: Supervised NMF where the number of basis components setting is empirically the best in terms of average SDR performance for the corresponding mixture.
- *Constrained NMF*: Supervised NMF with temporal continuity and sparsity constraints where experimentally the best α and β configuration taking average SDR performance into account for the corresponding mixture is used.
- *Feedforward DNN with STFT*: Feedforward architecture that uses the STFT as time-frequency representation.
- *Feedforward DNN with hybrid STFT/CQT*: Feedforward architecture that uses the hybrid STFT/CQT as time-frequency representation.
- *Recurrent DNN with hybrid STFT/CQT*: Recurrent architecture that uses the hybrid STFT/CQT as time-frequency representation. The best configuration in terms of the recurrent layer position in mean SDR sense is reported.

The term “optimal” refers to the fact that the configuration is empirically optimal among the tried settings. Moreover, the term global means that the setting is optimal when all the mixtures are taken into account while the term local means that the setting is optimal for the corresponding mixture. The distinction between the globally optimal NMF and the locally optimal NMF methods is made because the latter approach requires the user to choose the number of basis components parameter based on the mixture. Since in the summary part, the reported results for the DNN based approaches do not require any parameter to be set depending on the mixture (except for the recurrent layer position in the recurrent architecture), it is interesting to compare the DNN based approaches with an NMF setting that does not require mixture specific settings, i.e. to compare with the globally optimal NMF.

7.2 Experimental Settings

First of all, we use a system sampling rate of 32 kHz for all the experiments. The purpose of using this sampling rate is to reduce the DNN training time. Since the mixtures and the training material are at a higher sampling rate, we downsample all the audio material to this system sampling rate for all the experiments. This means that the evaluation of the separation performances is also done at this system sampling rate while computing the BSS Eval measures.

The feedforward architecture with STFT based magnitude frames uses STFT with window size 1024 and a 50 percent overlap, which gives 32 ms frame duration and 16 ms frame overlap. In order to capture the temporal dependencies, the feedforward network uses $C = 3$ contextual frames in both directions. The network consists of $K = 5$ rectified linear unit layers. In order to train the network, $M = 5 \times 10^5$ training samples are generated with the training data generation approach that was described in Sec. 5.1.2. For the layer-wise training, 600 L-BFGS iterations are applied whenever a new layer is added while 1000 L-BFGS iterations are applied in the fine tuning of the network phase, which gives in total 4000 iterations for the training of the network.

The feedforward architecture with hybrid STFT/CQT based magnitude frames uses a slightly different setting from the STFT based case in order to improve the source separation results. This means that we do not aim to see the sole effect of using the hybrid STFT/CQT representation on the source separation performance when other parameters are the same. In the hybrid STFT/CQT case, we first compute the STFT with window size 4096 and a 50 percent overlap, which results in 128 ms frame duration and 64 ms frame overlap. Then, using this computed STFT, we obtain the hybrid STFT/CQT representation by using a hybrid STFT/CQT kernel with the number of frequency bins per octave $b = 72$ which induces the transition frequency being $f_{\text{min}} = 830$ Hz. This gives a hybrid STFT/CQT kernel matrix $\mathbf{C}$ of size 414×2049. The inverse hybrid STFT/CQT uses the pseudoinverse because of its better reconstruction performance as discussed in Sec. 5.2.1. We use $C = 2$ contextual frames in both directions since the frame duration is larger in this case. Other parameters such as the number of training samples and the number of L-BFGS iterations are the same as the experiments with the STFT based magnitude frames.

For the recurrent network based separation, the time-frequency representation is the same as the one in the feedforward network with the hybrid STFT/CQT case. In the recurrent network case, there are no contextual frames as the recurrent network uses its recurrent input-output relationship to model the temporal dependencies. As mentioned earlier, the training data generation method for the recurrent network requires a sample window size W within which the contribution to the training samples reflect the spectral sequence structure contained in the training data in terms of the amplitudes and the frame indices. Hence, we choose a window size $W = 100$ that corresponds to chunks with approximately 6.4 seconds duration. We then have 5×10^3 many chunks giving in total $M = 5 \times 10^5$ training samples provided to the DNN, which is the same as the previous feedforward cases. Moreover, in oder to train the network, truncated BPTT with $T = 3$ backward time steps is used. Whenever a layer is added 600 L-BFGS iterations are applied after which 1000 fine tuning iterations are applied to obtain the final DNN that will be used for the extraction of the target instrument.

We only consider the STFT for the supervised NMF based separation. The window size of the STFT in this case is 4096 that corresponds to 128 ms frame duration. The first step of the supervised NMF based separation is to obtain the basis components corresponding to the instruments contained in the mixture. To this end, we use the solo performance databases for the corresponding instruments. Since we have both real music and MIDI based music databases, we can use those databases in separate source separation experiments. This way we can obtain supervised NMF based source separation results that can be compared with the DNN based separation when the training material is the same. As described in Sec. 6.2, we consider the supervised NMF scheme from [7]. The first step is to apply NMF to the magnitude STFT spectrogram obtained by concatenating solo performances of a single instrument in the instrument database. The NMFs in this pretraining phase use 500 iterations to obtain the factorization from which we keep the basis matrices. The next step of supervised NMF is the separation phase where the NMF is applied to the magnitude STFT spectrogram of the mixture signal to learn the gain matrix. To this end, first STFT of the mixture signal with window size 4096 and 75 percent overlap is computed. Then, applying 500 iterations gives the nonnegative factorization from which the separation can be obtained.

For the constrained NMF, we use the found optimal number of components for the corresponding mixture. This way we have the chance of improving the best result of the plain NMF case. In order to show how the weight of the temporal continuity cost term α and the weight of the sparsity cost term β affect the separation performance, we plot average BSS Eval measures over all the source estimates of the corresponding mixture when those weights are varied.

7.3 Experimental Results

7.3.1 Source separation results of the Brahms mixture

Table 7.1 details the feedforward neural network based source separation results. When we look at the results of the setting with STFT as time-frequency representation and real music as training material, we can see that having a deep architecture helps the network to achieve a better separation performance as the addition of each layer improves the separation performance during the layer-wise training. Note also that the improvement slows down for the addition of layers after the third layer which is mainly because the SSE cost that the network tries to minimize does not go down that fast compared to the addition of the initial layers.

Since the separation improves during the layer-wise training but at the same time towards the addition of the last layers the improvement is not that high, using a deep architecture with five layers seems to be a good compromise between the training time and the separation performance. Moreover, looking at the results after fine tuning, we can observe significantly better separation performance when Wiener filter is applied compared to the raw separation results, namely Wiener filter results in more than 1 dB higher SDR and more than 2 dB higher SIR for all source estimates compared to raw separation. This is mainly because when all the source estimates are well separated by the DNN, the Wiener spectral mask can better make use of the information from the separation of other sources which in turn results in better separation performance for all the source estimates.

When we further evaluate the separation performance of the configuration with the hybrid STFT/CQT time-frequency representation and real music training data, we can observe that the final separation performance after fine tuning is better for all the source estimates compared to the previously discussed STFT case. This in fact shows that the use of the hybrid STFT/CQT representation for the source separation of this mixture is advantageous since we obtain a better separation performance and at the same time the DNN training time is less because of the reduced computational complexity. Furthermore, contrasting the two time-frequency representations, the hybrid STFT/CQT has 128 ms frame duration while the STFT has 32 ms frame duration in the experiments. This in turn gives higher frequency resolution for low frequency bins of the hybrid STFT/CQT case. Higher frequency resolution for low frequencies helps the network to achieve better separation performance. On the other hand, the main drawback of the hybrid STFT/CQT is that it introduces a reconstruction error for its high frequency bins. This can be seen by the fact that the least raw separation SDR improvement is for violin which has the most high frequency components. However, the separation after Wiener filtering improves because the effect of the reconstruction error introduced by the inverse hybrid STFT/CQT is reduced when the spectral mask is applied.

If we analyze the experiment that uses STFT based magnitude frames and MIDI based music training data, we can see that there is substantial degradation in extraction of piano and horn compared to the experiments with real music training data. Although the raw separation of violin in fact gets better, the result after Wiener filtering gets worse because other raw source estimates are significantly worse. The reason for the poor separation performance for piano and horn extraction could be either related to our solo performance generation method or the MIDI database since other parameters remain the same compared to the setting with STFT and real music. In our opinion, the poor separation performance of the horn is due to the training data that comes mostly from a single composer for this instrument. Since the network sees training samples that come from a single composer, the network is not good at extracting the horn signal from a new horn mixture. On the other hand, the piano separation is more affected by the MIDI based solo performance generation method. We think that when the instrument contains notes with short durations, which is the case for piano, our music generation method introduces more distortions because of the windows that we applied to the waveforms of a single note. The network learns to extract mixtures with distorted contributions from piano. Since the music mixture we use for evaluation does not have those distortions, the separation performance becomes significantly poorer on this mixture.

When we further look at the experiment with the hybrid STFT/CQT and MIDI based music setting, we can see that the raw separation results of horn and piano do not change much compared to the STFT and MIDI based music setting. However, the raw separation results for violin becomes significantly poorer in this case. In our opinion, this poor performance is related to difference in input frame durations of the DNNs. Since the total DNN input frame duration including the contextual frames increases from 224 ms to 640 ms, more training samples include the portions distorted by the applied windows to the single note waveforms during the music generation. Hence, those distortions cause the network to show poor separation performance in the hybrid STFT/CQT case.

Table 7.1: Feedforward network based source separation results of the Brahms mixture (setting corresponds to used time-frequency representation and audio material used for training, all values are given in dB)

Setting	Instrument	Measure	After 1st layer		After 2nd layer		After 3rd layer		After 4th layer		After 5th layer		After fine tuning	
			Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF
STFT & Real music	Horn	SDR	3.23	3.92	4.90	6.28	5.00	6.44	5.06	6.55	5.19	6.68	5.25	**6.73**
		SIR	4.49	5.26	7.99	9.60	8.23	10.26	8.40	10.61	8.73	10.97	8.96	11.25
		SAR	10.52	10.81	8.48	9.45	8.41	9.15	8.34	9.07	8.28	9.03	8.18	8.94
	Piano	SDR	0.93	2.66	2.35	4.12	2.75	4.26	2.89	4.31	2.95	4.39	3.08	**4.41**
		SIR	2.07	4.65	4.45	7.50	5.22	8.15	5.37	8.30	5.57	8.48	5.86	8.73
		SAR	9.40	8.28	7.86	7.50	7.51	7.16	7.59	7.13	7.46	7.11	7.34	6.96
	Violin	SDR	-0.80	3.56	2.57	5.50	2.86	5.75	2.94	5.88	2.97	**5.90**	3.01	5.79
		SIR	1.02	8.07	8.01	13.66	8.37	14.47	8.60	14.84	8.70	15.01	9.06	15.07
		SAR	6.38	6.09	4.68	6.40	4.88	6.53	4.88	6.61	4.87	6.60	4.76	6.46
Hybrid STFT/CQT & Real music	Horn	SDR	4.83	5.76	5.56	6.91	5.68	7.05	5.74	7.08	5.81	7.11	5.85	**7.11**
		SIR	8.29	9.90	11.99	14.48	12.18	14.91	12.27	15.03	12.39	15.21	12.50	15.25
		SAR	8.03	8.29	6.95	7.90	7.04	7.97	7.08	7.97	7.13	7.97	7.14	7.96
	Piano	SDR	2.06	3.94	3.46	5.03	3.97	5.17	4.05	5.22	4.09	**5.25**	4.05	5.24
		SIR	4.37	7.23	6.78	9.12	7.67	9.78	7.82	9.91	7.93	9.97	7.93	10.05
		SAR	7.25	7.43	7.01	7.68	7.06	7.45	7.07	7.45	7.04	7.46	6.99	7.39
	Violin	SDR	-1.53	4.10	1.20	5.48	2.85	6.16	2.87	6.24	3.05	6.36	3.10	**6.39**
		SIR	2.15	10.93	7.34	13.69	10.38	16.03	10.47	16.21	10.88	16.46	11.23	16.44
		SAR	2.97	5.45	3.14	6.37	4.08	6.74	4.07	6.81	4.17	6.90	4.14	6.94
STFT & MIDI based music	Horn	SDR	1.61	2.10	1.37	2.98	2.04	**3.00**	2.17	2.97	2.25	2.93	2.15	2.89
		SIR	3.09	3.58	3.26	5.40	4.90	6.92	4.95	7.09	5.21	7.16	5.08	7.17
		SAR	8.74	9.06	7.58	7.77	6.43	6.06	6.62	5.86	6.45	5.75	6.41	5.69
	Piano	SDR	-0.09	0.70	0.74	0.85	1.11	0.97	1.16	0.92	**1.19**	0.88	1.15	0.82
		SIR	0.87	2.02	2.08	2.29	2.86	2.84	3.02	2.73	3.09	2.70	3.07	2.66
		SAR	9.57	8.63	8.58	8.35	7.69	7.36	7.49	7.45	7.41	7.41	7.37	7.33
	Violin	SDR	-1.07	3.24	1.36	**3.53**	2.53	3.11	2.80	2.82	3.23	2.86	3.52	3.17
		SIR	0.69	6.97	5.53	9.26	8.01	10.35	8.74	10.43	9.49	10.71	10.28	11.49
		SAR	6.38	6.43	4.52	5.36	4.61	4.41	4.62	4.03	4.87	3.99	4.93	4.16
Hybrid STFT/CQT & MIDI based music	Horn	SDR	1.50	2.95	1.81	3.49	1.98	3.85	2.08	3.96	2.23	4.03	2.24	**4.06**
		SIR	4.29	7.13	4.52	7.81	4.90	8.56	5.15	8.91	5.40	9.08	5.49	9.18
		SAR	6.12	5.80	6.45	6.17	6.30	6.21	6.19	6.16	6.19	6.17	6.09	6.15
	Piano	SDR	-0.43	0.83	-0.17	**0.90**	-0.04	0.88	-0.04	0.88	-0.04	0.89	-0.04	0.86
		SIR	0.53	1.85	0.68	1.86	0.83	1.84	0.86	1.84	0.89	1.87	0.88	1.82
		SAR	9.37	9.80	10.04	10.14	9.97	10.08	9.82	10.09	9.73	10.05	9.74	10.07
	Violin	SDR	-5.71	-1.22	-5.10	-0.72	-4.45	-0.11	-4.20	0.02	-3.91	0.05	-3.56	**0.35**
		SIR	-4.18	1.12	-3.40	1.78	-2.67	2.64	-2.38	2.87	-2.03	2.96	-1.59	3.40
		SAR	5.15	5.06	4.84	5.07	4.82	5.05	4.80	4.99	4.76	4.94	4.70	4.96

Table 7.2: Recurrent network based source separation results of the Brahms mixture (setting corresponds to the position of the recurrent layer, all values are given in dB)

Setting	**Instrument**	**Measure**	**After 1st layer**		**After 2nd layer**		**After 3rd layer**		**After 4th layer**		**After 5th layer**		**After fine tuning**	
			Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF
1st layer	Horn	SDR	3.58	**5.41**	4.22	5.17	4.28	5.19	4.53	5.21	4.76	5.37	4.70	5.39
		SIR	7.79	9.00	8.11	9.88	8.10	9.71	8.89	10.17	9.57	10.79	9.44	10.83
		SAR	6.33	8.42	7.13	7.39	7.22	7.53	7.04	7.28	6.95	7.18	6.95	7.19
	Piano	SDR	1.49	**3.68**	2.65	3.05	2.78	3.17	2.73	3.13	2.73	3.12	2.82	3.21
		SIR	5.51	8.33	6.41	7.70	6.64	8.06	6.79	8.13	6.96	8.26	6.91	8.29
		SAR	4.76	6.10	5.92	5.55	5.92	5.50	5.72	5.41	5.59	5.31	5.77	5.42
	Violin	SDR	0.04	5.37	1.28	5.28	1.10	5.19	1.73	5.47	3.28	6.43	3.45	**6.57**
		SIR	5.93	13.81	6.51	13.90	6.17	13.47	7.92	14.81	11.64	17.23	12.12	17.63
		SAR	2.32	6.22	3.71	6.09	3.66	6.08	3.57	6.15	4.25	6.88	4.35	6.99
2nd layer	Horn	SDR	3.78	4.53	5.15	5.84	5.73	6.47	5.68	6.57	5.77	6.63	5.76	**6.63**
		SIR	6.18	7.26	8.27	9.71	9.76	11.78	9.79	12.16	9.88	12.28	9.88	12.32
		SAR	8.43	8.58	8.65	8.58	8.35	8.26	8.25	8.23	8.33	8.26	8.31	8.24
	Piano	SDR	1.72	3.14	2.10	3.91	2.95	4.55	3.21	4.59	3.25	**4.64**	3.25	4.60
		SIR	3.50	6.22	4.16	7.40	6.48	9.30	6.91	9.57	7.00	9.69	7.01	9.59
		SAR	8.06	7.01	7.74	7.21	6.38	6.80	6.43	6.70	6.41	6.71	6.42	6.71
	Violin	SDR	-0.87	5.56	2.45	6.55	3.86	7.14	4.32	7.46	4.79	7.68	4.97	**7.75**
		SIR	1.55	12.17	9.34	16.33	12.80	17.30	13.42	17.93	14.13	18.34	15.06	18.55
		SAR	5.12	6.88	3.92	7.14	4.68	7.66	5.09	7.94	5.49	8.14	5.55	8.18

Table 7.2 gives the recurrent network based source separation results of the Brahms mixture. When we contrast the configurations in terms of the recurrent layer position, we can observe that the configuration where the second layer is the recurrent layer outperforms. This can be seen by the fact that the setting with second layer being the recurrent layer results in around 1 dB higher SDR for all instruments if the best values are considered. This shows that the recurrent layer position is an important factor that affects the separation performance. Therefore, further investigation on this topic would potentially give better separation performance with the recurrent architecture.

Since the settings of the recurrent network based case are comparable with the feedforward architecture that uses the hybrid STFT/CQT time-frequency representation and real music training data, we contrast their separation performance. Since the configuration where the second layer is the recurrent layer outperforms, we compare this configuration with the feedforward architecture. The comparison shows that while piano and horn are better separated in the case of feedforward architecture, violin is better separated in the case of recurrent architecture. The improvement in the separation of violin suggest that the recurrent network can better capture temporal dependencies in some cases since sound of violin has spectral content that is continuous over time. Moreover, we can also see that performance degradation of piano and horn is less than 1 dB in SDR measure. More than 1 dB SDR improvement in violin estimate could be desirable in some cases at the cost of less then 1 dB SDR degradation for other instruments.

Figure 7.1 illustrates the supervised NMF based source separation results of the Brahms

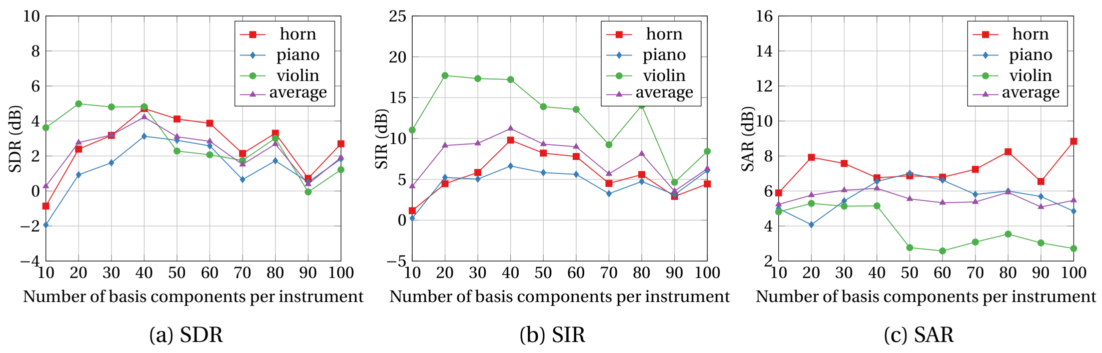


Figure 7.1: Supervised NMF based source separation results of the Brahms mixture when the training material is real music

mixture where the separation is supervised by real music training data. The plots show that the best separation performance in terms of mean SDR over all instruments is achieved when the number of basis components is 40. This in fact corresponds to the globally optimal setting if all the mixtures are taken into account. Moreover, this setting not only gives the best average performance but also results in close to best separation quality for all instruments. We can also observe that the separation performance depends a lot on the number of basis components as can be seen by the fluctuations of the evaluation measures. There is approximately 4 dB average SDR difference between the lowest and highest values while it is around 5 dB for the SIR measure. This is mainly because the basis components can easily become correlated for slight change in the number of basis components. Therefore, we can infer that the number of basis components should be optimized in order to achieve a reasonable separation performance.

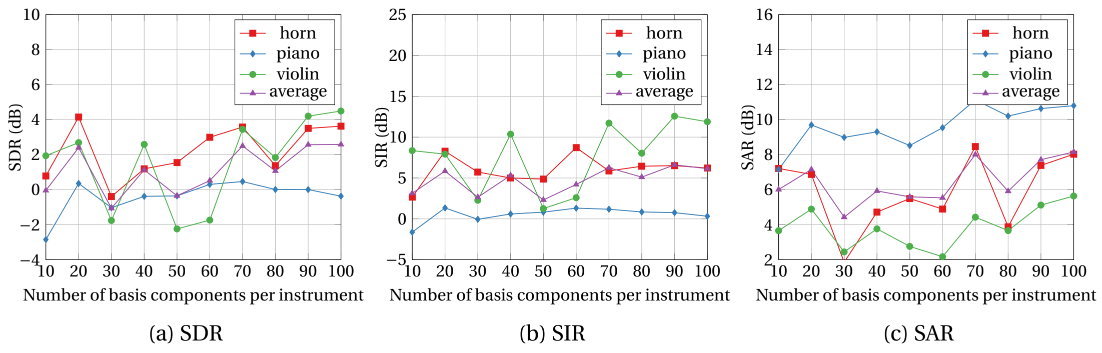
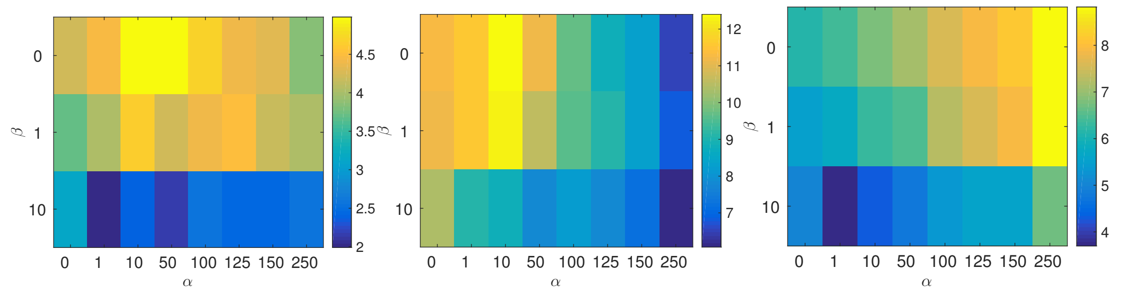


Figure 7.2: Supervised NMF based source separation results of the Brahms mixture when the training material is MIDI based music

Figure 7.2 demonstrates the separation performance of the supervised NMF scheme when the training material is MIDI based music. In order to achieve the best average SDR over all mixtures with MIDI based music training data, we need to use 20 basis components per instrument. On the other hand, the best average performance is achieved for this mixture when we use 100 basis components for each instrument. These two settings in fact show comparable separation performance. Furthermore, if we compare the best setting of this MIDI based separation with the best setting of real music based separation, we can observe that the separation gets worse in this case, which can be seen by the fact that there is around 2 dB and 5 dB decrease in average SDR and average SIR measures, respectively. This also explains why the DNNs with MIDI based music training material do not perform well for this mixture. Therefore, both MIDI based data generation approach and the MIDI database used for the instruments contained in this mixture should be carefully analyzed to improve the separation performance of both the DNN approaches and the NMF approaches when the training data is MIDI based music.

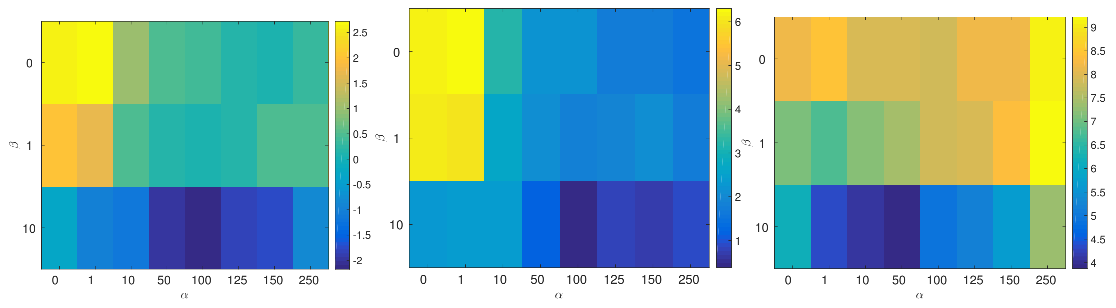


(a) SDR (b) SIR (c) SAR

Figure 7.3: Supervised NMF with temporal continuity and sparsity constraints based source separation results of the Brahms mixture when the training material is real music. α and β correspond to weights of temporal continuity and sparsity costs, respectively. The shown values are average over all instruments given in dB.

Figure 7.3 depicts average BSS Eval measures over the instruments considering constrained NMF based separation as discussed in Sec. 6.2 when the weights of the cost terms are varied. From the plots, we can see that temporal continuity constraint improves the overall separation quality while the sparsity constraint gives poorer separation quality. If we enforce the temporal continuity constraint, the overall separation performance improves until $\alpha = 10$. Then, increasing the continuity weight further degrades the performance. This phenomenon can be explained by the fact that the factorization will try to enforce longer continuities beyond the average note duration which causes the separation to get worse.

Figure 7.4 depicts the overall performance of the constrained NMF when the training data is MIDI based music. From the plots, we can see that temporal continuity constraint also improves the average separation performance but this time maximum average SDR is obtained when the α is equal to 1. We notice that this overall improvement is due to better separation of violin. This in fact is understandable since note durations of violin are longer and enforcing continuity in supervised NMF results in better estimate for violin.



(a) SDR (b) SIR (c) SAR

Figure 7.4: Supervised NMF with temporal continuity and sparsity constraints based source separation results of the Brahms mixture when the training material is MIDI based music. α and β correspond to weights of temporal continuity and sparsity costs, respectively. The shown values are average over all instruments given in dB.

All in all, Table 7.3 gives an overall summary of the results for the Brahms mixture. We can see that the best DNN outperforms the best NMF. Even if the NMF based approach is further improved by introducing temporal continuity constraint in the separation phase, the DNN methods still show better separation performance considering the real music training data. This shows that the use of DNN for source separation of horn, piano and violin is promising. For instance, if we analyze the real music based separation, using the setting with the hybrid STFT/CQT results in better separation quality for all the source estimates compared to the STFT setting. Moreover, the use of recurrent architecture results in further improvement of violin separation. Furthermore, when we look the effect of training material, we can see that using real music results in significantly better separation performance. Therefore, the use of real music as training data is more suitable for the extraction of the sources from this mixture.

Table 7.3: Summary of the source separation results for the Brahms mixture (setting corresponds to training material used for training, all values are given in dB)

Setting	Instrument	Measure	Globally optimal NMF	Locally optimal NMF	Constrained NMF	Feedforward DNN with STFT	Feedforward DNN with hybrid STFT/CQT	Recurrent DNN with hybrid STFT/CQT
Real music	Horn	SDR	4.71	4.71	5.42	6.73	**7.11**	6.63
		SIR	9.79	9.79	11.37	11.25	15.25	12.32
		SAR	6.75	6.75	7.00	8.94	7.96	8.24
	Piano	SDR	3.12	3.12	3.92	4.41	**5.25**	4.60
		SIR	6.63	6.63	7.29	8.73	10.05	9.59
		SAR	6.54	6.54	7.34	6.96	7.39	6.71
	Violin	SDR	4.81	4.81	5.60	5.79	6.39	**7.75**
		SIR	17.21	17.21	18.53	15.07	16.44	18.55
		SAR	5.15	5.15	5.89	6.46	6.94	8.18
MIDI based music	Horn	SDR	**4.14**	3.62	3.45	2.89	4.06	n/a
		SIR	8.27	6.22	5.83	7.17	9.18	n/a
		SAR	6.87	8.02	8.21	5.69	6.15	n/a
	Piano	SDR	0.35	-0.37	-0.46	0.82	**0.86**	n/a
		SIR	1.33	0.31	0.19	2.66	1.82	n/a
		SAR	9.69	10.79	10.93	7.33	10.07	n/a
	Violin	SDR	2.68	4.49	**5.17**	3.17	0.35	n/a
		SIR	7.91	11.88	12.89	11.49	3.40	n/a
		SAR	4.88	5.63	6.20	4.16	4.96	n/a

7.3.2 Source separation results of the Lussier mixture

Table 7.4 shows the feedforward network based separation results with different settings after addition of each layer together with the results after fine tuning. When we examine the configuration with STFT as the time-frequency representation and real music training material, we can observe that after addition of third layer for bassoon and piano, and after addition of fourth layer for trumpet the network gives the best separation performance. If we consider the DNN training time, it would be a better idea to stop the training after addition of third layer. Although the separation starts to get worse after addition of the following layers, still it does not degrade significantly. This brings us the advantage that the final results that we use for comparison are still close to the best results achieved by smaller networks. Another observation in this configuration is that although the raw separation of the piano is poor, it can benefit from other source estimates when the Wiener spectral mask is applied.

If we further look at the numerical evaluation results of the separation when the network uses the setting with the hybrid STFT/CQT as time-frequency representation and real music as training material, we can see that the separation performance slightly improves for piano and trumpet while there is small degradation in estimation of bassoon considering the SDR values. We can also observe that after addition of the second layer the network gives the best SDR performance for bassoon and piano while it is the third layer for trumpet. This observation together with a similar trend in the STFT case suggest that using a network with around three layers is preferable for this mixture when the training dataset is real music.

When we consider the setting with STFT and MIDI based music, we can observe that the separation of trumpet is comparable with the real music based settings even if there is slight performance degradation. On the other hand, raw estimates of piano and bassoon gets significantly worse compared to real music based training. This observation suggests that further investigation on MIDI based music generation for instruments bassoon and piano has to be made in order to get better separation quality of those instruments when MIDI based music is used for training of the DNNs. Similarly, the DNN that uses the setting with the hybrid STFT/CQT and MIDI based music also suffers from the training dataset being not suitable for the instruments contained in this mixture.

Table 7.4: Feedforward network based source separation results of the Lussier mixture (setting corresponds to used time-frequency representation and audio material used for training, all values are given in dB)

Setting	Instrument	Measure	After 1st layer		After 2nd layer		After 3rd layer		After 4th layer		After 5th layer		After fine tuning	
			Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF
STFT & Real music	Bassoon	SDR	2.95	4.25	3.26	4.57	3.60	**4.70**	3.65	4.54	3.69	4.45	3.67	4.43
		SIR	5.09	6.99	5.98	8.14	6.83	8.82	6.89	8.71	7.12	8.81	7.11	8.80
		SAR	8.23	8.35	7.56	7.71	7.21	7.37	7.26	7.17	7.10	6.96	7.06	6.94
	Piano	SDR	1.60	3.32	1.57	3.90	1.57	**3.90**	1.68	3.78	1.62	3.61	1.62	3.59
		SIR	3.29	6.34	3.31	7.36	3.26	7.11	3.46	7.00	3.46	6.87	3.46	6.85
		SAR	8.18	7.22	8.04	7.23	8.17	7.48	8.04	7.38	7.85	7.19	7.85	7.17
	Trumpet	SDR	5.02	6.12	6.35	7.04	6.77	7.62	6.71	**7.69**	6.69	7.67	6.63	7.63
		SIR	9.71	10.63	11.71	12.71	12.13	13.88	12.05	13.99	12.07	13.96	12.07	13.94
		SAR	7.27	8.38	8.12	8.64	8.52	8.97	8.47	9.02	8.43	9.01	8.36	8.96
Hybrid STFT/CQT & Real music	Bassoon	SDR	2.78	4.31	3.39	**4.56**	3.62	4.55	3.76	4.54	3.75	4.49	3.73	4.48
		SIR	6.04	8.08	6.25	8.70	7.08	9.61	7.34	9.85	7.41	9.89	7.46	9.93
		SAR	6.53	7.30	7.49	7.22	7.00	6.63	7.01	6.48	6.92	6.39	6.83	6.35
	Piano	SDR	2.29	3.95	2.36	**4.03**	2.51	3.86	2.51	3.82	2.47	3.79	2.50	3.78
		SIR	5.12	7.64	5.13	7.63	5.29	7.47	5.30	7.42	5.28	7.42	5.35	7.45
		SAR	6.66	7.07	6.80	7.21	6.90	7.06	6.88	7.03	6.81	6.99	6.79	6.94
	Trumpet	SDR	4.72	7.32	4.91	7.89	5.58	**7.91**	5.63	7.91	5.65	7.86	5.63	7.86
		SIR	12.22	13.47	12.79	14.42	13.83	15.45	13.86	15.52	13.89	15.47	13.90	15.48
		SAR	5.82	8.71	5.90	9.13	6.46	8.87	6.51	8.85	6.52	8.81	6.50	8.80
STFT & MIDI based music	Bassoon	SDR	1.59	**3.22**	1.30	2.55	1.39	2.68	1.49	2.62	1.59	2.68	1.63	2.69
		SIR	4.00	5.80	4.56	6.20	4.47	6.24	4.51	6.22	4.63	6.34	4.65	6.32
		SAR	6.75	7.72	5.37	5.94	5.66	6.12	5.80	6.05	5.86	6.03	5.91	6.07
	Piano	SDR	-0.79	0.62	-0.21	0.70	-0.29	0.73	-0.29	0.71	-0.26	**0.73**	-0.25	0.71
		SIR	1.04	3.19	1.67	3.04	1.48	3.11	1.53	3.12	1.52	3.10	1.54	3.07
		SAR	6.36	5.81	6.60	6.26	6.77	6.20	6.67	6.14	6.79	6.23	6.77	6.23
	Trumpet	SDR	2.23	4.40	4.29	5.97	4.95	6.28	5.15	6.37	5.23	6.33	5.32	**6.39**
		SIR	6.38	6.98	8.69	9.43	9.17	10.18	9.47	10.49	9.54	10.56	9.73	10.69
		SAR	5.24	8.69	6.79	9.04	7.51	8.95	7.62	8.87	7.69	8.76	7.72	8.77
Hybrid STFT/CQT & MIDI based music	Bassoon	SDR	2.36	3.35	3.08	**3.47**	3.11	3.32	3.29	3.27	3.35	3.29	3.25	3.21
		SIR	7.14	8.01	7.20	8.40	8.30	9.03	8.56	9.31	8.80	9.58	8.95	9.67
		SAR	4.88	5.80	5.97	5.74	5.28	5.20	5.39	4.99	5.35	4.90	5.13	4.76
	Piano	SDR	-0.35	1.38	-0.13	**1.40**	-0.21	1.37	-0.24	1.34	-0.20	1.34	-0.15	1.32
		SIR	1.71	2.95	1.66	2.95	1.61	2.94	1.57	2.91	1.59	2.91	1.64	2.89
		SAR	6.12	8.35	6.83	8.40	6.74	8.31	6.73	8.33	6.79	8.34	6.83	8.30
	Trumpet	SDR	3.19	5.79	3.36	6.05	3.48	6.19	3.55	**6.22**	3.58	6.18	3.57	6.13
		SIR	9.54	9.58	9.37	9.79	9.73	10.12	9.92	10.22	10.02	10.23	10.00	10.16
		SAR	4.79	8.59	5.09	8.88	5.09	8.84	5.12	8.82	5.11	8.74	5.10	8.72

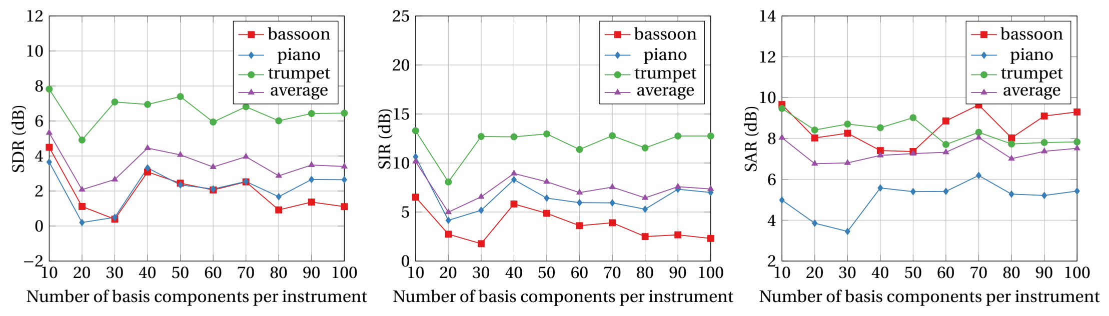
Table 7.5: Recurrent network based source separation results of the Lussier mixture (setting corresponds to the position of the recurrent layer, all values are given in dB)

Setting	Instrument	Measure	After 1st layer		After 2nd layer		After 3rd layer		After 4th layer		After 5th layer		After fine tuning	
			Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF
1st layer	Bassoon	SDR	2.55	5.04	4.32	5.65	4.39	**5.68**	4.40	5.63	4.47	5.66	4.31	5.52
		SIR	5.98	8.42	7.04	9.99	7.31	10.27	7.42	10.34	7.54	10.38	7.66	10.50
		SAR	6.15	8.29	8.43	8.06	8.23	7.92	8.14	7.80	8.13	7.83	7.70	7.54
	Piano	SDR	2.34	4.79	3.26	5.10	3.36	**5.12**	3.32	5.09	3.34	5.07	3.34	4.96
		SIR	5.98	9.22	5.94	9.03	5.96	9.05	5.99	9.11	6.04	9.07	6.12	8.95
		SAR	5.78	7.22	7.60	7.86	7.80	7.88	7.67	7.79	7.65	7.79	7.55	7.69
	Trumpet	SDR	4.40	7.98	4.84	7.89	4.92	7.86	5.22	8.01	5.55	**8.09**	5.50	7.79
		SIR	11.93	14.30	11.99	14.84	12.12	15.01	12.59	15.32	13.34	15.81	13.58	15.77
		SAR	5.52	9.30	6.03	9.01	6.10	8.92	6.33	9.03	6.54	9.01	6.42	8.66
2nd layer	Bassoon	SDR	3.04	4.57	3.64	4.47	3.53	4.48	3.92	4.69	4.03	4.75	4.13	**4.83**
		SIR	6.19	8.03	6.51	8.63	7.17	9.38	7.64	9.98	7.72	10.01	7.78	9.93
		SAR	6.85	7.81	7.67	7.14	6.75	6.65	7.01	6.63	7.14	6.70	7.25	6.85
	Piano	SDR	2.62	**4.33**	2.30	4.12	2.70	4.06	2.73	4.11	2.77	4.17	2.72	4.16
		SIR	5.50	8.80	5.47	8.43	6.07	8.04	6.09	8.14	6.16	8.16	6.12	8.12
		SAR	6.83	6.78	6.25	6.71	6.33	6.91	6.37	6.92	6.39	6.99	6.32	7.01
	Trumpet	SDR	4.86	7.68	5.10	7.95	5.56	7.96	5.59	**8.02**	5.58	7.95	5.50	7.93
		SIR	11.55	13.61	12.69	14.45	13.34	15.40	13.28	15.65	13.34	15.70	13.39	15.78
		SAR	6.20	9.15	6.16	9.20	6.55	8.95	6.60	8.97	6.57	8.87	6.47	8.82

Table 7.5 shows the source extraction performance of two different recurrent network architectures. If we compare the two configurations at the layers where they achieve their best separation performance for individual instruments, we can see that the first configuration outperforms for all the instruments. The same observation is made for the final results except for some small performance degradation in extraction of trumpet. The performance difference between the two architectures imply that by trying different configurations, one could potentially get better separation performance. Moreover, one could also use the best configuration for each instrument to enhance the separation after Wiener filtering.

If we further contrast the best recurrent architecture with the feedforward architecture that has similar settings, better separation quality for the recurrent architecture can be observed. While there is around 1 dB SDR improvement for bassoon and piano with recurrent architecture, there is no notable difference between the two in terms of trumpet separation. Noting better separation quality of bassoon and piano compared to feedforward architecture, the use of recurrent architecture for extraction of the sources from the Lussier mixture is worth further investigation.

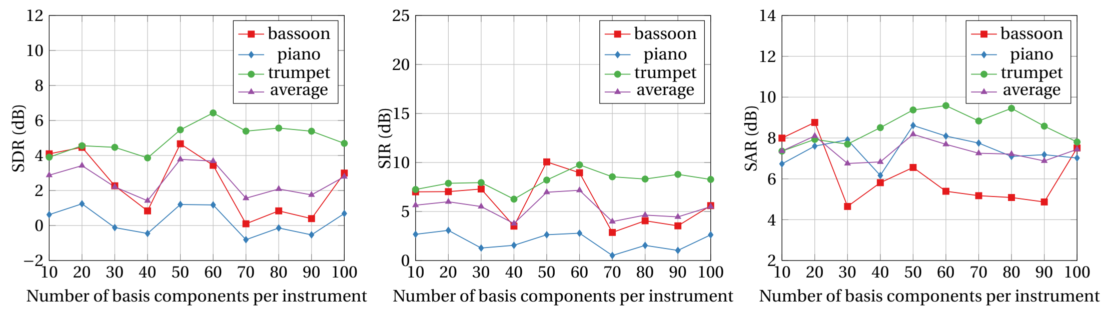
Figure 7.5 illustrates supervised NMF results for the separation of the Lussier mixture when the training data is real music. As it can be seen from the plots, the best average SDR setting for this mixture is obtained when the number of basis components per instrument is 10. As already mentioned, globally the best setting for the real music database based separation is obtained when the number of basis components is 40. Although using globally the best setting degrades the separation performance, the difference is not that high because globally the



(a) SDR (b) SIR (c) SAR

Figure 7.5: Supervised NMF based source separation results of the Lussier mixture when the training material is real music

best setting does not coincide with the worst setting for this mixture. However, in order to obtain acceptable separation performance, it is crucial to find the best setting considering the mixture as the separation performance highly depends on the number of basis components setting. Moreover, the overall trend in terms of the performances of individual instruments is similar to the DNN cases where the performance of the source separation is the best for trumpet, and piano and bassoon show comparable separation quality.



(a) SDR (b) SIR (c) SAR

Figure 7.6: Supervised NMF based source separation results of the Lussier mixture when the training material is MIDI based music

Figure 7.6 shows the supervised NMF separation results of the Lussier mixture when the training dataset is MIDI based music. When we compare the results with the real music database based separation, we can observe that the real music based case outperforms. Therefore, we can conclude that the MIDI based training data is not suitable also for the NMF based approach for this mixture. We can also see that the best settings in terms of the number of basis components for the two training material are not the same. Therefore, we can infer that it is necessary to optimize the number of basis components considering also the dataset in addition to the mixture.

Figure 7.7 shows the source separation performance of the supervised NMF scheme when

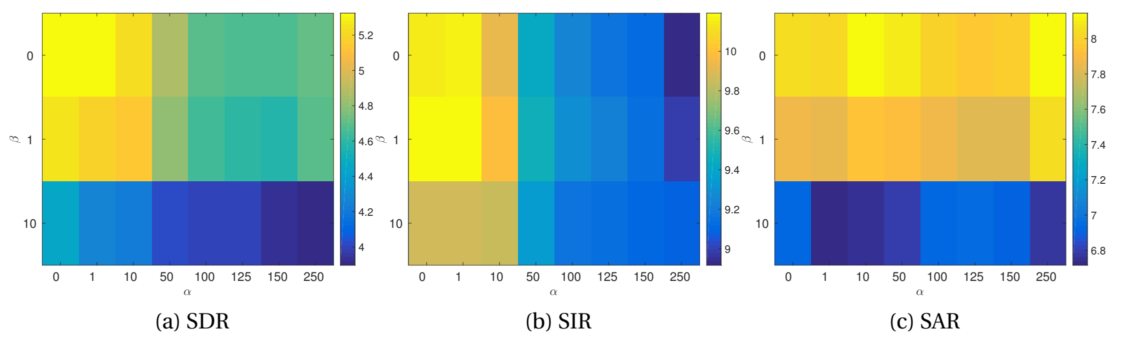


Figure 7.7: Supervised NMF with temporal continuity and sparsity constraints based source separation results of the Lussier mixture when the training material is real music. α and β correspond to weights of temporal continuity and sparsity costs, respectively. The shown values are average over all instruments given in dB.

additional constraints are incorporated at the separation stage. By looking at the plots, we can conclude that both of the constraints result in poorer average separation performance as can be seen by the SDR plot. We can infer that the best configuration in terms of the weights of the cost terms also depend on the type of the instruments contained in the mixture. While temporal continuity helps the separation of the Brahms mixture, the separation performance is degraded for the Lussier mixture when temporal continuity is enforced.

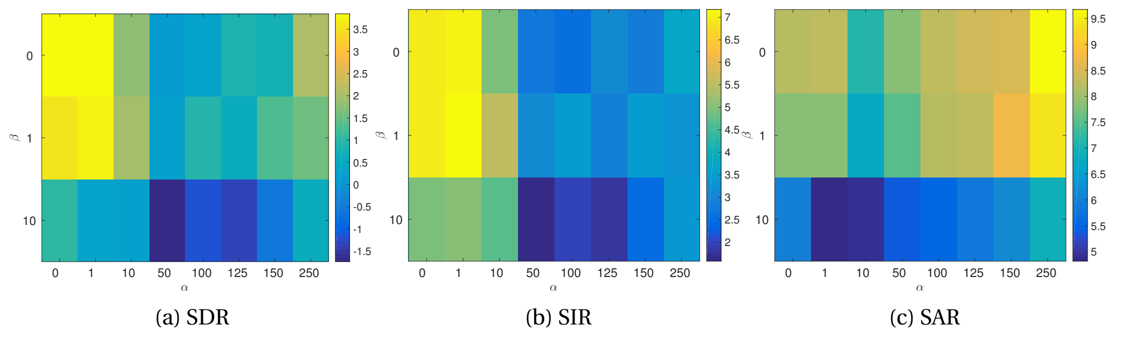


Figure 7.8: Supervised NMF with temporal continuity and sparsity constraints based source separation results of the Lussier mixture when the training material is MIDI based music. α and β correspond to weights of temporal continuity and sparsity costs, respectively. The shown values are average over all instruments given in dB.

Figure 7.8 shows the source separation performance of constrained NMF with varying weights of additional constraints when the training dataset is MIDI based music. We can observe slight improvement when the weight of the temporal continuity cost term is 1. However, the overall trend is close to the real music based case. Therefore, we can infer that both of the constraints do not help the separation of the Lussier mixture much.

To conclude, Table 7.6 summarizes how well different methods and different settings can sep-

Table 7.6: Summary of the source separation results for the Lussier mixture (setting corresponds to training material used for training, all values are given in dB)

Setting	Instrument	Measure	Globally optimal NMF	Locally optimal NMF	Constrained NMF	Feedforward DNN with STFT	Feedforward DNN with hybrid STFT/CQT	Recurrent DNN with hybrid STFT/CQT
Real music	Bassoon	SDR	3.09	4.49	4.49	4.43	4.48	**5.52**
		SIR	5.82	6.51	6.51	8.80	9.93	10.50
		SAR	7.40	9.66	9.66	6.94	6.35	7.54
	Piano	SDR	3.32	3.65	3.65	3.59	3.78	**4.96**
		SIR	8.30	10.63	10.63	6.85	7.45	8.95
		SAR	5.57	4.98	4.98	7.17	6.94	7.69
	Trumpet	SDR	6.94	7.82	7.82	7.63	**7.86**	7.79
		SIR	12.66	13.29	13.29	13.94	15.48	15.77
		SAR	8.52	9.47	9.47	8.96	8.80	8.66
MIDI based music	Bassoon	SDR	4.46	4.66	**4.92**	2.69	3.21	n/a
		SIR	7.02	10.06	10.29	6.32	9.67	n/a
		SAR	8.76	6.55	6.80	6.07	4.76	n/a
	Piano	SDR	1.24	1.20	1.23	0.71	**1.32**	n/a
		SIR	3.07	2.63	2.68	3.07	2.89	n/a
		SAR	7.60	8.61	8.59	6.23	8.30	n/a
	Trumpet	SDR	4.55	5.46	5.37	**6.39**	6.13	n/a
		SIR	7.88	8.20	8.05	10.69	10.16	n/a
		SAR	7.93	9.37	9.37	8.77	8.72	n/a

arate the Lussier mixture. In fact, a similar trend of DNNs performing better as in the Brahms mixture is observed when we compare the best NMF and the best DNN approaches. Thus, deep neural network based separation of Lussier mixture is also worth further investigation. We can also see that the best separation quality is obtained with the recurrent architecture for bassoon and piano. This shows that the recurrent architecture can show better separation quality in some cases. When different training material is contrasted, we again conclude that the use of real music for separation of this mixture is more suitable as in the Brahms mixture case.

7.3.3 Source separation results of the Schubert mixture

Table 7.7 provides the source separation performance of the feedforward architecture with four different settings. When we first analyze the setting with STFT and real music, we can see that cello and violin are poorly separated. Since sounds of cello and violin are similar, the separation of those instruments becomes difficult. In this setting, especially the violin estimate contains the contributions from the cello which causes the poor separation performance. On the other hand, piano being dissimilar to those instruments, it can be separated more easily. Since the separation performance of other instruments are poor, the Wiener spectral mask of the piano cannot gain from other source estimates. Therefore, the raw separation and the WF separation results of piano are comparable.

When we look at the source separation outcomes of the experiment with hybrid STFT/CQT and real music, we can observe improvement over the STFT and real music case. This in fact confirms the overall trend of the setting with hybrid STFT/CQT outperforming when the training data is real music. Although the separation of cello and violin also get better, their separation performances are still not satisfactory.

If we consider the setting with STFT and MIDI based music, we can observe that the separation quality of piano is close to the setting with STFT and real music. Similarly, the DNN gives poor estimates for cello and violin as in the real music case. Although in the previous experiments for other mixtures, we observed that the piano was not well separated in the case of MIDI based training material, in this case the extraction of piano gets better. The reason why the piano estimate in this case has better quality is that the characteristic of the sources start to dominate for this mixture rather than the characteristics of the training data. Piano having properties dissimilar to other sources, the DNN can extract the piano sound even if the training material does not represent arbitrary piano sounds well. If we also look at the result with hybrid STFT/CQT and MIDI based music, we can observe a similar trend.

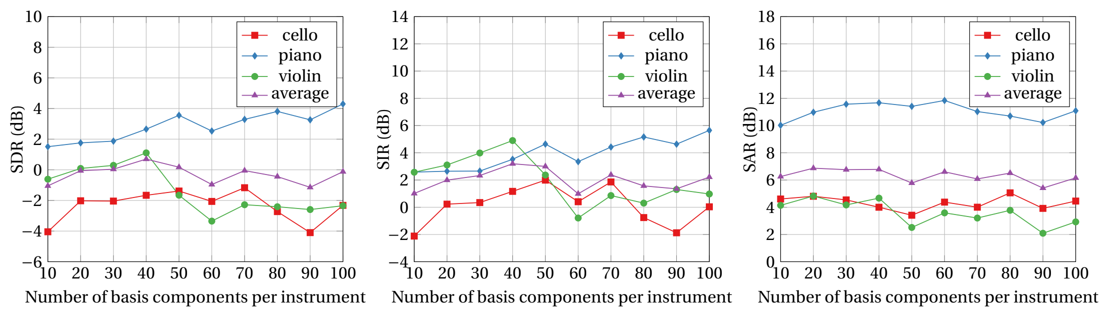
Table 7.7: Feedforward network based source separation results of the Schubert mixture (setting corresponds to used time-frequency representation and audio material used for training, all values are given in dB)

Setting	Instrument	Measure	After 1st layer		After 2nd layer		After 3rd layer		After 4th layer		After 5th layer		After fine tuning	
			Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF
STFT & Real music	Cello	SDR	-0.20	**0.96**	-0.34	0.23	-0.67	0.33	-0.69	0.39	-0.75	0.33	-0.66	0.41
		SIR	1.59	4.18	3.41	5.60	3.40	6.30	3.52	6.57	3.77	6.61	3.92	6.71
		SAR	6.81	5.17	3.67	2.78	3.12	2.52	2.98	2.45	2.66	2.35	2.68	2.41
	Piano	SDR	2.62	2.51	3.38	3.78	4.20	4.40	4.39	4.51	4.43	4.54	4.49	**4.64**
		SIR	3.73	3.42	4.71	5.20	6.01	6.17	6.20	6.38	6.33	6.49	6.49	6.68
		SAR	10.63	11.36	10.42	10.48	9.85	10.08	9.98	9.97	9.83	9.82	9.71	9.74
	Violin	SDR	-2.64	-1.40	-0.73	**-0.26**	-0.38	-0.44	-0.42	-0.45	-0.40	-0.48	-0.38	-0.47
		SIR	-1.68	0.25	1.27	2.28	2.02	2.16	2.04	2.16	2.10	2.15	2.25	2.20
		SAR	8.33	6.47	6.04	5.30	5.48	5.09	5.33	5.07	5.28	5.02	5.08	4.95
Hybrid STFT/CQT & Real music	Cello	SDR	-0.35	0.16	0.10	0.79	0.94	1.41	1.04	1.43	1.16	1.47	1.26	**1.52**
		SIR	2.87	4.22	4.16	5.82	5.35	7.37	5.58	7.53	5.91	7.75	6.23	7.93
		SAR	4.27	3.72	3.68	3.44	4.00	3.41	3.99	3.36	3.92	3.30	3.85	3.29
	Piano	SDR	4.17	4.08	4.62	4.57	4.85	5.13	4.92	5.23	4.98	5.33	5.04	**5.42**
		SIR	5.85	5.38	6.25	5.94	6.55	6.61	6.67	6.76	6.76	6.88	6.87	7.03
		SAR	10.13	11.06	10.61	11.24	10.62	11.37	10.57	11.35	10.55	11.35	10.48	11.30
	Violin	SDR	-2.12	-0.39	-1.53	0.31	-0.09	1.19	0.01	1.25	0.04	1.29	0.07	**1.31**
		SIR	-0.18	1.98	0.99	2.84	2.64	3.71	2.74	3.79	2.78	3.82	2.84	3.86
		SAR	5.42	5.49	4.58	5.68	5.11	6.30	5.16	6.31	5.17	6.33	5.15	6.33
STFT & MIDI based music	Cello	SDR	-1.51	-0.87	-1.22	-0.97	-0.94	-0.89	-0.90	-0.91	**-0.78**	-0.92	-0.83	-0.93
		SIR	-0.08	0.90	0.96	1.21	1.40	1.38	1.46	1.39	1.63	1.41	1.58	1.39
		SAR	7.04	6.49	5.37	5.53	5.25	5.38	5.22	5.32	5.19	5.28	5.19	5.28
	Piano	SDR	2.53	3.07	2.97	4.09	3.89	4.50	3.92	4.53	4.00	4.56	4.05	**4.59**
		SIR	3.61	4.25	4.35	5.81	5.60	6.45	5.75	6.54	5.85	6.63	5.93	6.67
		SAR	10.65	10.72	9.99	9.95	9.83	9.79	9.59	9.71	9.59	9.63	9.58	9.62
	Violin	SDR	-2.26	-2.25	-0.90	-1.25	-0.29	-0.36	**-0.25**	-0.35	-0.29	-0.32	-0.26	-0.28
		SIR	-1.22	-1.13	1.03	0.59	2.27	2.30	2.28	2.35	2.22	2.42	2.27	2.46
		SAR	8.15	7.77	6.06	6.11	5.24	5.03	5.31	4.99	5.34	4.95	5.32	4.96
Hybrid STFT/CQT & MIDI based music	Cello	SDR	-0.91	-0.24	0.00	**0.37**	-0.01	0.13	0.16	0.15	0.16	0.08	0.27	0.16
		SIR	2.06	2.61	2.95	3.26	2.99	3.03	3.34	3.18	3.41	3.16	3.66	3.37
		SAR	4.24	4.84	4.85	5.19	4.78	5.00	4.65	4.83	4.59	4.75	4.49	4.63
	Piano	SDR	3.41	3.68	3.77	4.18	3.94	4.36	3.94	4.43	3.95	**4.47**	3.95	4.46
		SIR	5.26	5.40	5.53	5.91	5.72	6.20	5.77	6.30	5.77	6.36	5.77	6.36
		SAR	9.14	9.63	9.61	9.99	9.69	9.92	9.61	9.90	9.61	9.88	9.61	9.88
	Violin	SDR	-3.08	-1.76	-2.62	-1.18	-1.99	-0.74	-1.81	-0.58	-1.74	-0.51	-1.69	**-0.46**
		SIR	-1.64	-0.20	-0.98	0.43	-0.30	1.12	-0.10	1.35	0.00	1.45	0.09	1.53
		SAR	6.30	6.56	5.94	6.70	6.09	6.34	6.13	6.26	6.07	6.23	6.00	6.19

Table 7.8: Recurrent network based separation results of the Schubert mixture (setting corresponds to the position of the recurrent layer)

Setting	Instrument	Measure	After 1st layer		After 2nd layer		After 3rd layer		After 4th layer		After 5th layer		After fine tuning	
			Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF	Raw	WF
1st layer	Cello	SDR	-1.34	0.47	0.04	0.44	0.15	0.44	0.09	0.58	0.43	**0.73**	0.35	0.68
		SIR	3.28	5.38	4.44	6.14	5.28	6.75	5.43	7.67	5.79	7.76	5.67	7.73
		SAR	2.17	3.27	3.34	2.75	2.86	2.43	2.69	2.21	2.94	2.36	2.91	2.31
	Piano	SDR	3.59	5.05	4.20	4.81	4.34	4.93	4.58	5.59	4.68	5.72	4.69	**5.73**
		SIR	5.82	6.68	5.67	6.32	5.95	6.61	6.45	7.59	6.63	7.71	6.69	7.77
		SAR	8.57	10.94	10.66	11.03	10.41	10.71	10.04	10.61	9.96	10.76	9.86	10.66
	Violin	SDR	-1.55	0.37	-1.01	0.57	-1.09	0.62	-0.47	0.79	-0.29	**0.80**	-0.34	0.77
		SIR	0.83	2.44	1.23	2.90	1.14	2.91	1.98	3.03	2.09	3.03	1.97	2.95
		SAR	4.82	6.54	5.37	6.18	5.36	6.28	5.32	6.50	5.54	6.52	5.64	6.60
2nd layer	Cello	SDR	-0.90	0.13	-0.12	0.83	0.30	1.18	0.48	1.25	0.57	1.24	0.68	**1.26**
		SIR	1.83	3.87	4.14	6.06	4.98	7.23	5.16	7.39	5.42	7.43	5.64	7.53
		SAR	4.61	4.01	3.35	3.34	3.31	3.17	3.44	3.18	3.38	3.15	3.39	3.14
	Piano	SDR	4.19	4.00	4.25	4.73	4.50	5.40	4.67	5.54	4.75	5.61	4.79	**5.64**
		SIR	5.72	5.19	5.59	6.10	6.22	7.12	6.50	7.33	6.62	7.46	6.67	7.52
		SAR	10.49	11.33	11.07	11.35	10.28	11.04	10.20	10.98	10.17	10.91	10.17	10.90
	Violin	SDR	-2.00	-0.20	-1.52	0.66	-0.80	1.01	-0.73	**1.06**	-0.73	1.05	-0.73	1.03
		SIR	-0.24	2.21	0.63	3.12	1.61	3.52	1.63	3.57	1.64	3.54	1.63	3.53
		SAR	5.90	5.54	5.27	6.02	5.19	6.18	5.31	6.22	5.30	6.24	5.31	6.21

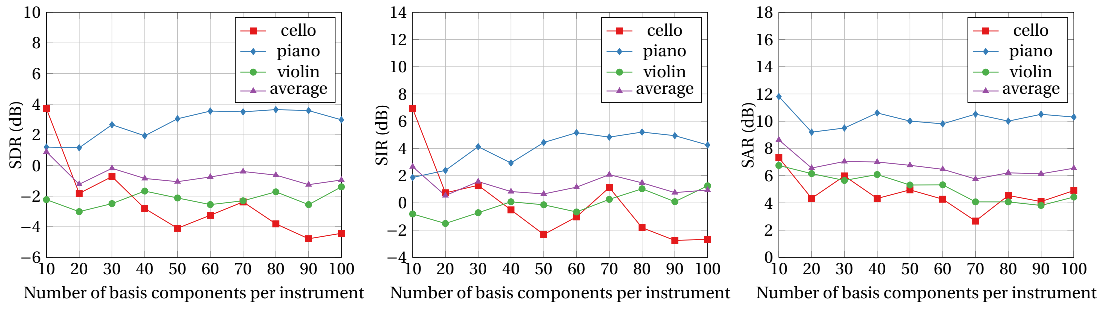
Table 7.8 details the separation results when the recurrent network used for the source separation of the Schubert mixture. Since we use the hybrid STFT/CQT and real music training data with the recurrent network, the comparison should be made with the feedforward network with the same setting. By looking at the results, we can see that the piano slightly gets better with the recurrent network with both configurations in terms of the recurrent layer position compared to feedforward network. For the separation of cello and violin being poor in all three cases, further investigation on extraction of those instruments is required.



(a) SDR (b) SIR (c) SAR

Figure 7.9: Supervised NMF based source separation results of the Schubert mixture when the training material is real music

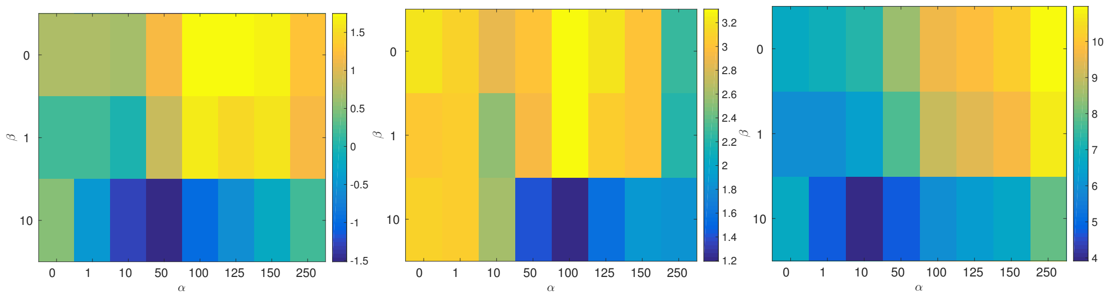
Figure 7.9 shows the supervised NMF based separation results when the training dataset is real music. These results also confirm that the separation of cello and violin is difficult. As it can be seen from the plots, increasing the number of basis components make the basis components of cello and violin become correlated which results in poorer separation performance. Although the best configuration in terms of average SDR performance occur when the number of basis components is 40, it could also be desirable to use the setting when it is 100 since the piano separation gets better. Since separation performance of cello and violin is poor and slight change is not very important, the focus could be more on the piano.



(a) SDR (b) SIR (c) SAR

Figure 7.10: Supervised NMF based source separation results of the Schubert mixture when the training material is MIDI based music

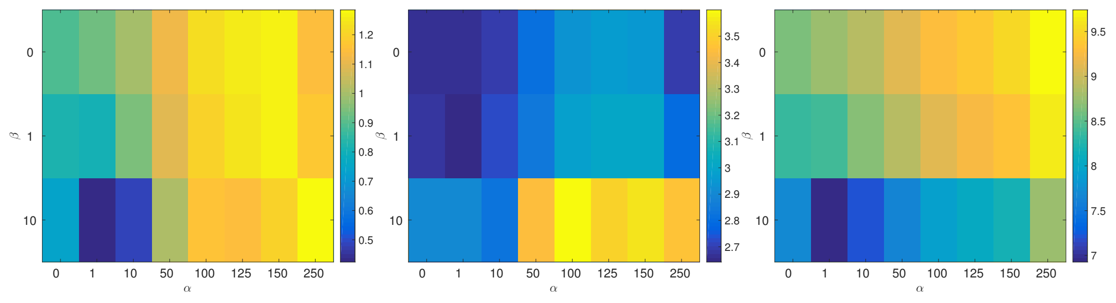
Figure 7.10 shows the separation of the Schubert mixture where the the supervised NMF computes the basis components using the MIDI based music database. Although similar trend with real music based case can be seen when the number of basis components increase, the fluctuations occur differently. We also notice that the best piano separation performance is poorer compared to the real music case, which suggests that the use of real music is more appropriate for separation of this mixture.



(a) SDR (b) SIR (c) SAR

Figure 7.11: Supervised NMF with temporal continuity and sparsity constraints based source separation results of the Schubert mixture when the training material is real music. α and β correspond to weights of temporal continuity and sparsity costs, respectively. The shown values are average over all instruments given in dB.

Figure 7.11 demonstrates average BSS Eval measures of the source estimates obtained by constrained NMF where the weight of the additional constraints are varied. The plots illustrate that the average SDR measure over all the measures gets better when the temporal continuity weight increases where the best result is achieved when $\alpha = 125$. The improvement in the overall performance is due to the fact that violin and cello SDR measures get better. This could be explained by the fact that those instruments have continuous spectral properties so that enforcing temporal continuity helps the separation.



(a) SDR (b) SIR (c) SAR

Figure 7.12: Supervised NMF with temporal continuity and sparsity constraints based source separation results of the Schubert mixture when the training material is MIDI based music. α and β correspond to weights of temporal continuity and sparsity costs, respectively. The shown values are average over all instruments given in dB.

Figure 7.12 shows the constrained NMF results of the Schubert mixture when the dataset is MIDI based music. We can observe that the best average SDR performance is achieved when $\alpha = 250$ and $\beta = 10$. Although the average difference is not that high compared to the case without the constraints, the constrained method increases the SDR value of cello estimate a lot, at the expense of degrading the SDR performance of violin.

Table 7.9 summarizes the source extraction performance of the different methods and settings for the Schubert mixture. First of all, while the DNN methods can show some satisfactory degree of separation of piano, they cannot separate other instruments. On the other hand, NMF based methods can favor one instrument at the expense of degrading performance of other instruments. By looking at the summary of the results, we can infer that main difficulty in separation of the Schubert mixture is the similarity of cello and violin affecting the overall performance of all the methods significantly. Furthermore, the effect of the MIDI based music on separation performance is not very apparent since the characteristics of the sources dominate for this mixture.

Table 7.9: Summary of the source separation results for the Schubert mixture (setting corresponds to training material used for training, all values are given in dB)

Setting	Instrument	Measure	Globally optimal NMF	Locally optimal NMF	Constrained NMF	Feedforward DNN with STFT	Feedforward DNN with hybrid STFT/CQT	Recurrent DNN with hybrid STFT/CQT
Real music	Cello	SDR	-1.66	-1.66	0.33	0.41	**1.52**	1.26
		SIR	1.16	1.16	1.75	6.71	7.93	7.53
		SAR	4.00	4.00	8.09	2.41	3.29	3.14
	Piano	SDR	2.65	2.65	2.42	4.64	5.42	**5.64**
		SIR	3.52	3.52	2.89	6.68	7.03	7.52
		SAR	11.67	11.67	14.16	9.74	11.30	10.90
	Violin	SDR	1.10	1.10	**2.48**	-0.47	1.31	1.03
		SIR	4.89	4.89	4.95	2.20	3.86	3.53
		SAR	4.66	4.66	7.30	4.95	6.33	6.21
MIDI based music	Cello	SDR	-1.82	-4.04	**5.06**	-0.93	0.16	n/a
		SIR	0.74	-2.11	9.48	1.39	3.37	n/a
		SAR	4.33	4.60	7.47	5.28	4.63	n/a
	Piano	SDR	1.15	1.50	1.11	**4.59**	4.46	n/a
		SIR	2.38	2.58	1.69	6.67	6.36	n/a
		SAR	9.19	10.01	12.36	9.62	9.88	n/a
	Violin	SDR	-3.01	-0.61	-2.32	**-0.28**	-0.46	n/a
		SIR	-1.51	2.57	-0.81	2.46	1.53	n/a
		SAR	6.15	4.13	6.42	4.96	6.19	n/a

7.3.4 Summary of the source separation results

Figure 7.13 provides an overall summary of the applied methods by showing their average separation performance when the training dataset is real music. The results show that the deep neural network based approaches show better separation performance over nonnegative matrix factorization based approaches on average for the considered mixtures, e.g. the best DNN outperforms on average the best supervised NMF by 1 dB in SDR. Although the plot shows the average performance, we also know from the previous discussion that the best separation performance in most cases (i.e., except for the violin estimate of the Schubert mixture where constrained NMF outperforms, for all other source estimates the best results are given by the DNNs) is obtained with the DNN based approaches. Moreover, we can observe the setting that uses the hybrid STFT/CQT improves over the setting that uses the STFT, i.e. there is on average 0.66 dB improvement in SDR. With this hybrid STFT/CQT setting, instead of using the feedforward architecture, using the recurrent architecture, slight improvement in average separation quality is gained.

The DNN methods except for the recurrent architecture which uses its best configuration in terms of recurrent layer position do not use mixture specific settings. On the other hand, the locally optimal NMF and the constrained NMF uses mixture specific settings. Even if these two NMF methods gain from their mixture specific settings, still the DNNs outperform. In fact, the DNNs could also be tuned to the mixture if we use the best raw separation results out of the three different DNN approaches for each instrument while constructing the Wiener spectral

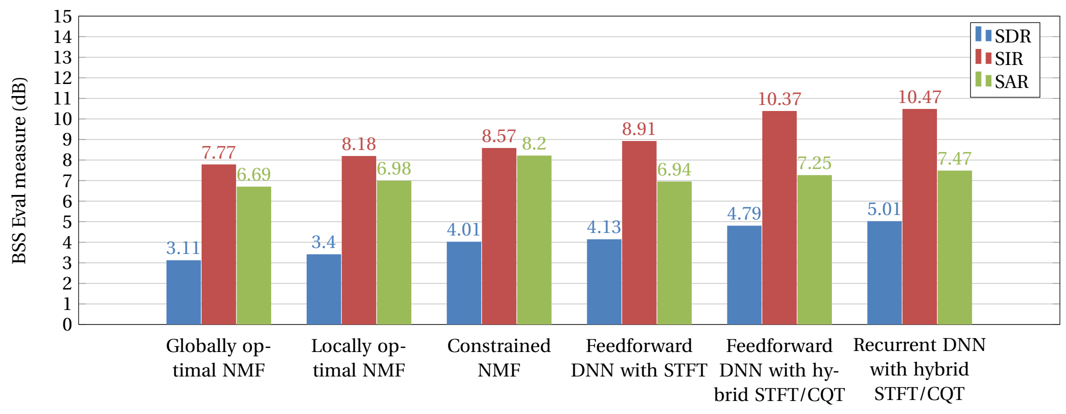


Figure 7.13: Average SDR, SIR and SAR values over all the instruments and all the mixtures when the training material is real music

masks. This way further improvement using a mixture specific setting could be obtained.

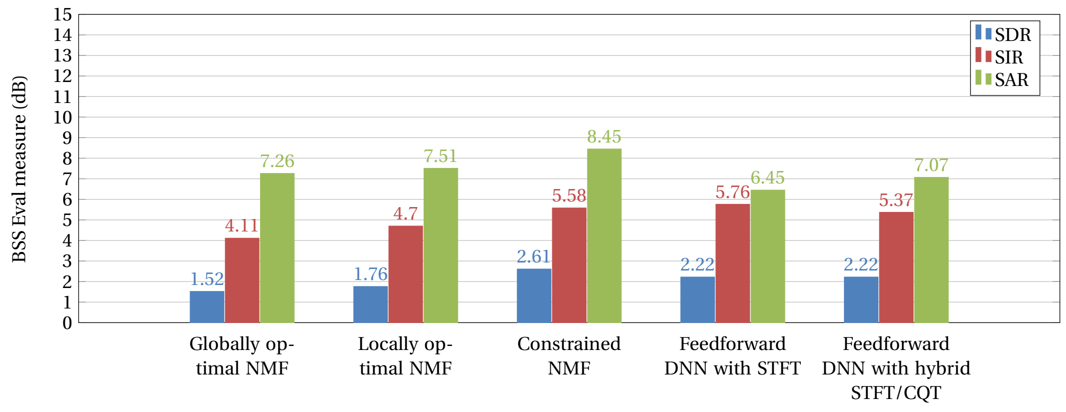


Figure 7.14: Average SDR, SIR and SAR values over all the instruments and all the mixtures when the training material is MIDI based music

Figure 7.14 demonstrate the overall performance of the applied methods when the training dataset is MIDI based music. By contrasting the results with the real music based approaches, we can notice significantly poorer performance with MIDI based approach. In particular, we observe that the DNN based approaches are more sensitive to the solo performance generation method. Therefore, we can conclude that in order for the DNNs with MIDI based training data to result in better overall separation quality, further examination of the MIDI based solo performance generation method should be made.

8 Conclusion and Outlook

All in all, this thesis investigates deep neural network based source separation of instrumental music mixtures. To this end, feedforward and recurrent neural network architectures are employed. The standard STFT and the hybrid STFT/CQT that combines the STFT and the CQT are used in order to perform the separation in a time-frequency domain. The parameters of the network architectures are learned during a layer-wise training phase where the training samples are generated by using an instrument database. In addition to a real music database, a music score database based synthesis of a training dataset is considered. Evaluation of different network architectures and settings is made by performing source separation experiments on three different music mixtures. In order to contrast the results with another method, supervised NMF based source separation experiments are conducted on the same test dataset.

The results of the conducted experiments show that the best DNN based approach outperforms the best supervised NMF based approach for all mixtures. The simulations also show that by using the hybrid STFT/CQT representation, better separation performance over the STFT representation can be obtained. With the recurrent architecture working in the hybrid STFT/CQT domain, the separation performance was further improved for some instruments. On the other hand, when the synthesized music is used to generate the training samples, the DNNs show poor separation performance compared to the case when real music is used.

Future work involves the investigation on the following major topics. First of all, further investigation on the parameters of the recurrent architecture ranging from backward time steps used by the BPTT to the recurrent layer position could potentially improve the source separation performance of the recurrent architecture. Secondly, different settings of the hybrid STFT/CQT representation could be tried to see their effect on quality of the source estimates. Finally, further investigation on music score based training data generation is required in order to achieve separation performance that is comparable with the real music based approach. In particular, diversity of the music score database, variations contained in the single note instrumental sounds database and the proposed music generation approach should be analyzed to find out the cause of poor separation performance with this approach.

A Least Squares Initialization of Network Weights

As mentioned in Sec. 5.1.2, least squares initialization is used when a new layer is added during the layer-wise training. We give here the derivation of this initialization for the kth layer weight matrix $\mathbf{W}_k$ and bias vector $\mathbf{b}_k$ when the corresponding layer is added to the network. The optimization problem is formulated as

$$\{\mathbf{W}_k^{\text{init}}, \mathbf{b}_k^{\text{init}}\} = \underset{\mathbf{W}_k, \mathbf{b}_k}{\arg\min} \sum_{t=1}^{M} \left\| \mathbf{W}_k \, \mathbf{y}_{k-1}^t + \mathbf{b}_k - \mathbf{d}^t \right\|^2, \tag{A.1}$$

where $\mathbf{y}_{k-1}^t$ is the output of previous layer for the tth training sample and $\mathbf{d}^t$ is the corresponding target.

We first write the gradient of the total cost with respect to the bias vector, which is given by

$$\begin{aligned}
\frac{\partial E}{\partial \mathbf{b}_k} &= 2 \sum_{t=1}^{M} \left(\mathbf{W}_k \, \mathbf{y}_{k-1}^t + \mathbf{b}_k - \mathbf{d}^t \right) \\
&= 2\mathbf{W}_k \left(\sum_{t=1}^{M} \mathbf{y}_{k-1}^t \right) + 2M\mathbf{b}_k - 2 \sum_{t=1}^{M} \mathbf{d}^t \\
&= 2M\mathbf{W}_k \bar{\mathbf{y}}_{k-1} + 2M\mathbf{b}_k - 2M\bar{\mathbf{d}},
\end{aligned}$$

where $\bar{\mathbf{d}}$ and $\bar{\mathbf{y}}_{k-1}$ are mean vectors over the M samples of $\{\mathbf{d}^t\}$ and $\{\mathbf{y}_{k-1}^t\}$, respectively. Then, equating the gradient to zero gives

$$\mathbf{b}_k^{\text{init}} = \bar{\mathbf{d}} - \mathbf{W}_k^{\text{init}} \, \bar{\mathbf{y}}_{k-1}. \tag{A.2}$$

In a second step, we write the gradient of the total cost with respect to the weight matrix

$$
\begin{aligned}
\frac{\partial E}{\partial \mathbf{W}_k} &= 2\sum_{t=1}^{M} \left(\mathbf{W}_k\,\mathbf{y}_{k-1}^{t} + \mathbf{b}_k - \mathbf{d}^{t}\right)\left(\mathbf{y}_{k-1}^{t}\right)^{T} \\
&= 2\sum_{t=1}^{M} \left(\mathbf{W}_k\,\mathbf{y}_{k-1}^{t} + \bar{\mathbf{d}} - \mathbf{W}_k\,\bar{\mathbf{y}}_{k-1} - \mathbf{d}^{t}\right)\left(\mathbf{y}_{k-1}^{t}\right)^{T} \\
&= 2\sum_{t=1}^{M} \left(\mathbf{W}_k\,\mathbf{y}_{k-1}^{t} - \mathbf{W}_k\,\bar{\mathbf{y}}_{k-1} - \mathbf{d}^{t} + \bar{\mathbf{d}}\right)\left(\mathbf{y}_{k-1}^{t}\right)^{T} \\
&= 2\mathbf{W}_k\sum_{t=1}^{M} \left(\mathbf{y}_{k-1}^{t} - \bar{\mathbf{y}}_{k-1}\right)\left(\mathbf{y}_{k-1}^{t}\right)^{T} - 2\sum_{t=1}^{M} \left(\mathbf{d}^{t} - \bar{\mathbf{d}}\right)\left(\mathbf{y}_{k-1}^{t}\right)^{T} \\
&= 2\mathbf{W}_k\sum_{t=1}^{M} \left(\mathbf{y}_{k-1}^{t} - \bar{\mathbf{y}}_{k-1}\right)\left(\mathbf{y}_{k-1}^{t}\right)^{T} - 2\mathbf{W}_k\sum_{t=1}^{M} \left(\mathbf{y}_{k-1}^{t}\right)\left(\bar{\mathbf{y}}_{k-1}\right)^{T} + 2\mathbf{W}_k\sum_{t=1}^{M} \left(\bar{\mathbf{y}}_{k-1}\right)\left(\bar{\mathbf{y}}_{k-1}\right)^{T} \\
&\quad - 2\sum_{t=1}^{M} \left(\mathbf{d}^{t} - \bar{\mathbf{d}}\right)\left(\mathbf{y}_{k-1}^{t}\right)^{T} + 2\sum_{t=1}^{M} \left(\mathbf{d}^{t}\right)\left(\bar{\mathbf{y}}_{k-1}\right)^{T} - 2\sum_{t=1}^{M} \left(\bar{\mathbf{d}}\right)\left(\bar{\mathbf{y}}_{k-1}\right)^{T} \\
&= 2\mathbf{W}_k\sum_{t=1}^{M} \left(\mathbf{y}_{k-1}^{t} - \bar{\mathbf{y}}_{k-1}\right)\left(\mathbf{y}_{k-1}^{t} - \bar{\mathbf{y}}_{k-1}\right)^{T} - 2\sum_{t=1}^{M} \left(\mathbf{d}^{t} - \bar{\mathbf{d}}\right)\left(\mathbf{y}_{k-1}^{t} - \bar{\mathbf{y}}_{k-1}\right)^{T}.
\end{aligned}
$$

Then, equating the gradient to zero gives

$$
\mathbf{W}_k^{\text{init}} = \mathbf{C}_{dy}\,\mathbf{C}_{yy}^{-1}, \tag{A.3}
$$

where

$$
\mathbf{C}_{dy} = \sum_{t=1}^{M} \left(\mathbf{d}^{t} - \bar{\mathbf{d}}\right)\left(\mathbf{y}_{k-1}^{t} - \bar{\mathbf{y}}_{k-1}\right)^{T}, \qquad \mathbf{C}_{yy} = \sum_{t=1}^{M} \left(\mathbf{y}_{k-1}^{t} - \bar{\mathbf{y}}_{k-1}\right)\left(\mathbf{y}_{k-1}^{t} - \bar{\mathbf{y}}_{k-1}\right)^{T}.
$$

Bibliography

[1] J. Fritsch, “High quality musical audio source separation,” Master’s thesis, UPMC / IRCAM / Telecom ParisTech, 2012.

[2] M. Ryynanen, T. Virtanen, J. Paulus, and A. Klapuri, “Accompaniment separation and karaoke application based on automatic melody transcription,” in *Proc. IEEE International Conference on Multimedia and Expo*, pp. 1417–1420, June 2008.

[3] D. FitzGerald, “Upmixing from mono - a source separation approach,” in *Proc. 17th International Conference on Digital Signal Processing*, pp. 1–7, July 2011.

[4] O. Gillet and G. Richard, “Transcription and separation of drum signals from polyphonic music,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, pp. 529–540, March 2008.

[5] C. Jutten and P. Comon, *Handbook of Blind Source Separation.* Academic Press, 2010.

[6] S. Uhlich, F. Giron, and Y. Mitsufuji, “Deep neural network based instrument extraction from music,” in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, accepted.

[7] E. Grais and H. Erdogan, “Single channel speech music separation using nonnegative matrix factorization and spectral masks,” in *Proc. 17th International Conference on Digital Signal Processing*, pp. 1–6, July 2011.

[8] T. Virtanen, “Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 3, pp. 1066–1074, 2007.

[9] H. Kameoka, N. Ono, K. Kashino, and S. Sagayama, “Complex NMF: A new sparse representation for acoustic signals,” in *IEEE International Conference on Acoustics, Speech and Signal Processing, 2009 (ICASSP)*, pp. 3437–3440, April 2009.

[10] B. King and L. Atlas, “Single-channel source separation using complex matrix factorization,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, pp. 2591–2597, Nov 2011.

[11] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Deep learning for monaural speech separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1562–1566, 2014.

[12] P.-S. Huang, M. Kim, M. Hasegawa-Johnson, and P. Smaragdis, "Singing-voice separation from monaural recordings using deep recurrent neural networks," in *International Society for Music Information Retrieval (ISMIR)*, 2014.

[13] E. Grais, M. Sen, and H. Erdogan, "Deep neural networks for single channel source separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3734–3738, May 2014.

[14] N. Boulanger-Lewandowski, G. Mysore, and M. Hoffman, "Exploiting long-term temporal dependencies in NMF using recurrent neural networks with application to source separation," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6969–6973, May 2014.

[15] F. Weninger, J. Hershey, J. Le Roux, and B. Schuller, "Discriminatively trained recurrent neural networks for single-channel speech separation," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 577–581, Dec 2014.

[16] B. Yang, "A study of inverse short-time Fourier transform," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3541–3544, March 2008.

[17] J. C. Brown, "Calculation of a constant Q spectral transform," *Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[18] J. C. Brown and M. S. Puckette, "An efficient algorithm for the calculation of a constant Q transform," *Journal of the Acoustical Society of America*, vol. 92, no. 5, pp. 2698–2701, 1992.

[19] S. Haykin and Z. Chen, "The cocktail party problem," *Neural Computation*, vol. 17, pp. 1875–1902, September 2005.

[20] M. S. Pedersen, J. Larsen, U. Kjems, and L. C. Parra, "A survey of convolutive blind source separation methods," in *Springer Handbook of Speech Processing*, Springer Press, Nov 2007.

[21] E. Vincent, R. Gribonval, and C. Fevotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 1462–1469, July 2006.

[22] E. Vincent, "BSS Eval Toolbox 3.0." http://bass-db.gforge.inria.fr/bss_eval/. [Online, accessed on 04-02-2015].

[23] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, Oct. 1986.

[24] C. Weng, D. Yu, S. Watanabe, and B.-H. Juang, “Recurrent deep neural networks for robust speech recognition,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5532–5536, May 2014.

[25] M. C. Mozer, “A focused back-propagation algorithm for temporal pattern recognition,” *Complex systems*, vol. 3, no. 4, pp. 349–381, 1989.

[26] M. Gori, Y. Bengio, and R. De Mori, “BPS: a learning algorithm for capturing the dynamic nature of speech,” in *International Joint Conference on Neural Networks (IJCNN)*, vol. 2, pp. 417–423, 1989.

[27] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5, pp. 157–166, Mar 1994.

[28] T. Mikolov, *Statistical language models based on neural networks.* PhD thesis, Brno University of Technology, 2012.

[29] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” in *Proc. International Conference on Machine Learning (ICML)*, 2013.

[30] J. Ganseman, P. Scheunders, and S. Dixon, “Improving PLCA-based score-informed source separation with invertible constant-Q transforms,” in *Proc. 20th European Signal Processing Conference (EUSIPCO)*, pp. 2634–2638, Aug 2012.

[31] R. Jaiswal, D. Fitzgerald, E. Coyle, and S. Rickard, “Towards shifted NMF for improved monaural separation,” in *Signals and Systems Conference (ISSC)*, pp. 1–7, June 2013.

[32] B. Fuentes, R. Badeau, and G. Richard, “Blind harmonic adaptive decomposition applied to supervised source separation,” in *Proc. 20th European Signal Processing Conference (EUSIPCO)*, pp. 2654–2658, Aug 2012.

[33] P. Sprechmann, J. Bruna, and Y. LeCun, “Audio source separation with discriminative scattering networks,” *ICLR*, 2015, submitted.

[34] C. Schörkhuber and A. Klapuri, “Constant-Q transform toolbox for music processing,” in *7th Sound and Music Computing Conference, Barcelona, Spain*, pp. 3–64, 2010.

[35] C. S. Sapp, “Online database of scores in the humdrum file format.,” in *ISMIR*, pp. 664–665, 2005.

[36] Fourmilab, “MidiCsv Tool.” http://www.fourmilab.ch/webtools/midicsv/. [Online, accessed on 16-02-2015].

[37] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka, “RWC music database: Music genre database and musical instrument sound database,” in *ISMIR*, vol. 3, pp. 229–230, 2003.

[38] C. Févotte, N. Bertin, and J. Durrieu, “Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis,” *Neural Computation*, vol. 21, pp. 793–830, March 2009.

[39] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems*, vol. 13, pp. 556–562, 2001.