END-TO-END NON-NEGATIVE AUTO-ENCODERS: A DEEP NEURAL ALTERNATIVE TO NON-NEGATIVE AUDIO MODELING

BY

SHRIKANT VENKATARAMANI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2020

Urbana, Illinois

Doctoral Committee:

Associate Professor Paris Smaragdis, Chair
Professor Andrew Singer
Professor Mark Hasegawa-Johnson
Assistant Professor Minje Kim, University of Indiana

ABSTRACT

Over the last decade, non-negative matrix factorization (NMF) has emerged as one of the most popular approaches to modeling audio signals. NMF allows us to factorize the magnitude spectrogram to learn representative spectral bases that can be used for a wide range of applications. With the recent advances in deep learning, neural networks (NNs) have surpassed NMF in terms of performance. However, these NNs are trained discriminatively and lack several key characteristics like re-usability and robustness, compared to NMF.

In this dissertation, we develop and investigate the idea of end-to-end non-negative autoencoders (NAEs) as an updated deep learning based alternative framework to non-negative audio modeling. We show that end-to-end NAEs combine the modeling advantages of non-negative matrix factorization and the generalizability of neural networks while delivering significant improvements in performance.

To this end, we first interpret NMF as a NAE and show that the two approaches are equivalent semantically and in terms of source separation performance. We exploit the availability of sophisticated neural network architectures to propose several extensions to NAEs. We also demonstrate that these modeling improvements significantly boost the performance of NAEs.

In audio processing applications, the short-time fourier transform (STFT) is used as a universal first step and we design algorithms and neural networks to operate on the magnitude spectrograms. We interpret the sequence of steps involved in computing the STFT as additional neural network layers. This enables us to propose end-to-end processing pipelines that operate directly on the raw waveforms. In the context of source separation, we show

that end-to-end processing gives a significant improvement in performance compared to existing spectrogram based methods. Furthermore, to train these end-to-end models, we investigate the use of cost functions that are derived from objective evaluation metrics as measured on waveforms. We present subjective listening test results that reveal insights into the performance of these cost functions for end-to-end source separation.

Combining the adaptive front-end layers with NAEs, we propose end-to-end NAEs and show how they can be used for end-to-end generative source separation. Our experiments indicate that these models deliver separation performance comparable to that of discriminative NNs, while retaining the modularity of NMF and the modeling flexibility of neural networks. Finally, we present an approach to train these end-to-end NAEs using mixtures only, without access to clean training examples.

ACKNOWLEDGMENTS

As we tread through the grind of daily life and work, it is often easy to lose ourselves in all the seeming clutter and apparent chaos that surrounds us. It is only when we look back on our journey retrospectively that the real magic reveals itself. It is only during these moments that we realize what we've gained, how far we've traveled and how much we owe to the people in our life: our advisors, our parents, our friends and our teachers. I take this opportunity to thank you all; my achievements would not be possible without each and every one of you.

First and foremost, I am extremely grateful to my *guru* Dr. Paris Smaragdis. Right from the time I moved here, he has been a friend, a philosopher and a guide in the truest sense of the terms. I realize now that he has led by example every step of the way while allowing me to develop my own philosophies and pursue my own interests with all the freedom I could have. Discussing exciting ideas, projects and research in the office, facing heartbreaks over rejected papers and lost opportunities, learning how to teach and guide students, understanding how important it is to balance life and work, giddily running to get green tea in Honolulu before the shops close for the day ... I'm really gonna miss you and all of that, Paris.

I'd also like to thank the other members of my committee – Prof. Andrew Singer, Prof. Mark Hasegawa-Johnson and my friend Dr. Minje Kim – for all their time, effort and constructive feedback.

I've also had a great time and learned a great deal over four amazing internships. Thank you Dr. Gautham Mysore and Dr. Juan-Pablo Caceres for being amazing mentors. Your feedback has always been extremely insightful and productive and I've learned a great deal from you. I also really appre-

ciate the efforts of Dr. Jonathan Le Roux and Dr. Gordon Wichern for all their guidance and insights during my time at MERL. I'd like to really thank my dear friend Dr. Prem Seetharaman with whom I've shared thrilling discussions, amazing beers and a couple of desks at Adobe and at MERL. Last, but not the least, I really appreciate the efforts of my mentors and soon-to-be colleagues Dr. Umut Isik, Dr. Ritwik Giri and Dr. Arvindh Krishnaswamy. Thank you for your trust and for the opportunity you've afforded me.

In my time here, I've also forged some deep friendships that I will cherish for the entirety of my lifetime. I owe a great deal to my dearest lab-mates Jonah, Thymios, Cem, Anny, Zhepei, Ryley, Ramin, Minje, Joh and Jeffrey (Yu-Che). We've had exciting discussions, written some papers under gruelling deadlines and been through times when we wanted to possibly throttle each other. Thank you guys, you have been my family and I will fondly remember the times we've spent together for the rest of my life.

Through my TA-ships, I've had the pleasure of interacting with some brilliant students here. Thank you all for your patience, your time and your questions. I'd also like to express my sincere gratitude to all my teachers here. All of you have made me a better person.

My forays into audio and signal processing would not have been possible without my advisors at IIT Bombay. I sincerely thank Dr. Preeti Rao and Dr. Rajbabu Velmurugan for their guidance and efforts. I'm also indebted to my friends and lab-mates Mayur, Amrita, Prateek, Hitesh, Kaustuv, Vedhas, Nachiket, Parthe, Gaurang, Venkhat, Kumar and Suhas. You have challenged me at every step and always made me better.

I owe the most to my parents – thank you for all your motivation, encouragement and everything you've given me in the 29 years of my life so far. I also owe a tremendous amount to everyone in my family for their support. Finally, to my dearest friends Rama, Aditya, Varun, Tanmay, Chintan, Darshan, Sohan and Tiara, thank you for being there for me when I needed you all. You guys are the best!

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

With the current surge of deep learning (DL) and the recent advancements in neural networks (NNs), NNs are being predominantly used to solve a wide variety of tasks in different fields. These applications range from common computer science topics like natural language processing, computer vision, and computer audition to fields like astronomy and biomedical and civil engineering. In addition, NNs have also been driving the state-of-the-art in research with their impressive performances.

Even in the case of audio signals, NNs have been used for several applications. Some of these examples include source separation (SS) and speech enhancement [1], speech recognition [2], audio classification [3], speech syndissertation [4], music processing [5] etc. These NNs are typically trained in a supervised or discriminative manner. The training set consists of large amounts of input examples and their corresponding ideal targets. The input examples are fed into the NN and the NN is trained to produce an output the resembles the ideal target. This is done by training the network to minimize a suitable measure of discrepancy between the output produced by the network and the ideal target using back-propagation. For example, in the context of speech denoising, the input to the NNs consists of noisy or degraded mixtures and the NNs are trained to produce their corresponding clean versions at the output. In the context of bandwidth extension, a narrowband or filtered signal is given as the input and the NN is trained to generate the corresponding wideband version. In the areas of polyphonic music transcription or speech recognition, the speech/music samples are fed as inputs and the corresponding music/text symbols are expected to be produced at the output of the network.

There are some significant drawbacks when training NNs in a discrimina-

tive manner.

- Training NNs discriminatively requires huge amounts of training data and this is often performed by enormous amounts of data-augmentation on the training set. As an illustrative example, in the context of speech denoising, to have a practically usable network, we need to consider input mixtures with a wide variety of possible interfering noises and mixtures at varying signal-to-noise ratios (SNRs).

- The NN models we learn using discriminative training are often task specific and not reusable. For example, speech denoising models trained for extracting human speech from mixtures recorded on the street cannot be used to extract human speech from mixtures recorded in a different ambient environment like the living room of a house. Likewise, these models also cannot be extended to work on other problems like bandwidth extension of speech signals.

- Finally, the discriminatively trained NNs are single-pass methods. In other words, the test examples pass through the network only once at inference time and the NNs are expected to produce the desired outputs. Thus, there is no scope for fitting these models on slightly varying test sounds. This lack of a fitting process at inference time restricts the models from being extended to other related applications.

In contrast, a classical approach to modeling audio signals has been the idea of non-negative matrix factorization (NMF). In the case of audio signals, we apply NMF on the magnitude spectrogram of the audio signal to learn representative spectral bases. Typically, we use these factorizations on clean examples to get generative spectral bases for the sounds. An illustration of applying NMF and the nature of these spectral bases are given in Section 2.1. Also generative NMF models do not suffer from the drawbacks of discriminatively trained NNs. The advantages of NMF models over discriminative models can be itemized as follows:

- Generative NMF models trained on clean audio signals do not require any data-augmentation when using them for various tasks.

- NMF models are also reusable in nature. For example, NMF models for human speech can be used to extract human speech from any mixture irrespective of the nature of the interfering sounds.

- Being a multi-pass method, NMF models are fitted iteratively on the test examples at inference. This allows us to fit NMF models on related but unseen test scenarios and leads to several novel applications like querying and learning from mixtures [6].

At the same time, NMF suffers from a few drawbacks of its own.

- It is very difficult to generalize NMF models to propose more sophisticated extensions. Generalizing NMF models requires a significant effort in re-working the model and deriving the update equations.

- Traditional NMF approaches use fixed audio representations like the magnitude spectrogram or the Mel spectrogram which also completely discard the underlying phase information. Learning a representation optimal to the data and the task can be beneficial in improving the performance of these models.

- To train NMF based models, we have been restricted to using simple cost functions that allow us to derive suitable updates for the factorization. Thus, complicated perceptual metrics like Short-term Objective Intelligibility [7] and waveform based metrics like the BSS_eval metrics [8] have not been explored as cost functions.

- NMF based models only deliver a modest performance when compared to neural networks and significantly lag behind state-of-the-art results.

Thus, the goal of this dissertation is introduce a new framework to model audio signals such that we retain the advantages of NMF and NNs, while simultaneously alleviating their drawbacks. We do so by interpreting NMF as a NN and then generalizing the architecture further. The availability of automatic differentiation tools and NN toolboxes would then allow us to propose and explore the use of meaningful metrics as cost functions themselves. Being generatively trained, the resulting models are multi-purpose in nature and can be reused for different test conditions and tasks.

The contributions and the outline of this dissertation are as follows:

1. By interpreting NMF as a non-negative autoencoder (NAE), we first provide a suitable NN based alternative to modeling audio signals. We then show how we can generalize and deploy these NAE models to extract the sources from a mixture. These ideas are explored in Chapter 2.

2. To relax the constraints involved and learn a trainable representation for the audio signals, we interpret the short-time fourier transform (STFT) as a NN. These NN based transforms allow us to operate directly on the waveforms and use representations that are optimal for the data and task at hand. We also show how we can interpret perceptual and waveform based metrics as cost functions to train NNs. This allows us to propose several new previously infeasible cost functions that boost the performance of our networks. Chapter 3 deals with these explorations.

3. Combining these ideas, we can update and improve upon the modeling capabilities of NAE models to propose end-to-end NAEs. We show that these end-to-end NAEs can compete with discriminatively trained NNs in the context of single-channel source separation. NMF models can be easily tweaked and adapted to perform querying and learn directly from mixtures without access to clean training examples. We demonstrate capabilities similar to those of our end-to-end NAEs and show how they can be easily adjusted to learn without clean examples. We discuss these in Chapter 4

This dissertation primarily uses single-channel source separation as the application of choice. However, the ideas developed are general and can be used as a potential replacement to NMF in several allied applications like dereverberation, bandwidth expansion, and declipping that use similar algorithms.

CHAPTER 2

NEURAL NETWORK ALTERNATIVES TO NON-NEGATIVE AUDIO MODELS

Over the last decade, non-negative modeling of audio signals and non-negative matrix factorization (NMF) [9] has emerged as one of the most popular tools to perform audio source separation. More recently, with the advent of neural networks (NN) and deep learning, NN based approaches to source separation have steadily grown in popularity and have resulted in state-of-the-art source separation performance. A primary contributing factor is the ability of neural networks to generalize and learn complex multi-layer models by exploiting the modeling flexibility that comes with NNs. However, unlike generative modeling approaches like NMF, these models are often trained as discriminative models. Thus, the resulting models are not transferable and cannot be reused when there is a mismatch between the training and test conditions. The goal of this chapter is to develop a neural network that can be used as an equivalent to NMF based generative models and show how they can be used for single-channel source separation.

2.1 Non-Negative Matrix Factorization (NMF)

Given a matrix $\mathbf{X}$, the goal of NMF is to approximate the input matrix $\mathbf{X}$ as a product of two low-rank matrices given as

$$\mathbf{X} \approx \mathbf{W} \cdot \mathbf{H} \tag{2.1}$$

Here, the input matrix $\mathbf{X} \in \mathbb{R}_{\geq 0}^{M \times N}$ is a matrix of non-negative real numbers of size $M \times N$. The matrices $\mathbf{W} \in \mathbb{R}_{\geq 0}^{M \times K}$ and $\mathbf{H} \in \mathbb{R}_{\geq 0}^{K \times N}$ represent the two low-rank non-negative factor matrices and K denotes the rank of the decomposition. In the case of audio signals, we apply NMF on the magnitude spectrogram. In such a setting, the matrix $\mathbf{W}$ is known as the basis matrix while the matrix $\mathbf{H}$ is known as the activation matrix. The reason for such a

naming convention becomes clear when we consider the application of NMF on a simple audio signal consisting of a sequence of piano notes.

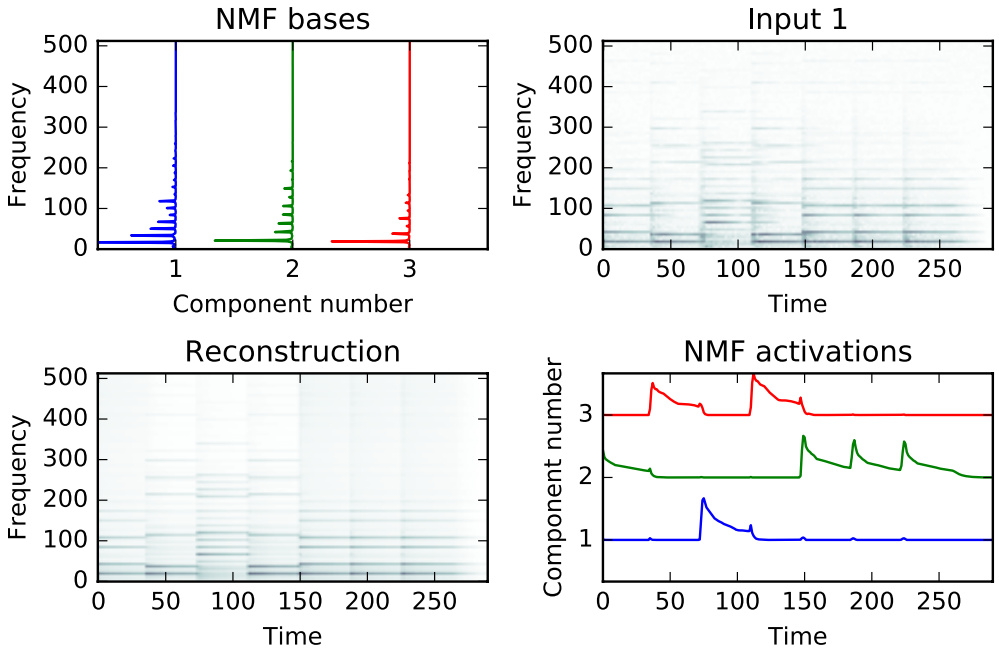

Figure 2.1: The top-right image represents the magnitude spectrogram of a sequence of piano notes. We see that the given audio snippet consists of 3 distinct piano notes. We thus perform a rank 3 NMF decomposition. The columns of $\mathbf{W}$ and the corresponding rows of $\mathbf{H}$ are shown as the NMF bases (top-left) and the NMF activations (bottom-right). The resulting approximation of the input spectrogram is also shown as the reconstruction (bottom-left) in the figure.

We minimize the Kullback-Leibler divergence between the input $\mathbf{X}$ and its factorization $\mathbf{Y} = \mathbf{W} \cdot \mathbf{H}$,

$$KL(\mathbf{X}||\mathbf{Y}) = \mathbf{X} \odot \log\left(\frac{\mathbf{X}}{\mathbf{Y}}\right) - \mathbf{X} + \mathbf{Y} \tag{2.2}$$

Here, $\odot$ represents an element-wise multiplication operation and the division is also element-wise. To solve the optimization problem, we particularly prefer the use of multiplicative updates as described in [10]. The main advantage of using multiplicative updates is that once we set a non-negative initialization for the $\mathbf{W}$ and $\mathbf{H}$, the non-negativity of the successive updates

are automatically guaranteed.

Figure 2.1 shows the NMF decomposition of the magnitude spectrogram of a music signal comprising a sequence of 3 distinct piano notes. As shown in the figure, the columns of **W** begin to look like the individual magnitude-spectra of the different notes of the piano. For example, the third column of **W** is representative of the first and third piano notes played in the given audio signal. Similarly, component 2 represented by the second column of **W** indicates the last three piano notes in the given signal. Thus, the columns of **W** act as representative bases for the different notes of the piano itself and the matrix is consequently known as the "basis" matrix. The rows of **H** begin to indicate when the corresponding bases are activated over time in the input spectrogram and hence **H** is also known as the "activation" matrix. Such a factorization technique has been the core idea behind several source separation approaches over the past few years [11, 12].

The idea of NMF has been extensively used for a wide variety of audio applications. Popular examples of these applications in the domain of audio signals are single-channel [13] and multi-channel [14] source separation, speech enhancement [15], speech dereverberation [16], audio inpainting [17], audio compression [18], polyphonic music transcription [9] and many more. NMF has also found several applications outside the audio domain. In video processing, NMF has been used for video compression [19], video fingerprinting [20], action recognition [21], video summarization [22] and several others. NMF has also found wide applicability in the case of EEG data [23], text mining [24], bio-informatics [25] and many others.

2.2 Non-negative Autoencoder (NAE)

Now that we understand the application of NMF in the context of modeling audio signals, we can begin to consider the problem of interpreting NMF as a neural network. As shown in [26], we can generalize the idea of NMF by interpreting NMF as a neural network in the following way:

$$\begin{aligned} 1^{\text{st}} \text{ layer:} \quad & \mathbf{H} = \mathbf{W}^{\ddagger} \cdot \mathbf{X} \\ 2^{\text{nd}} \text{ layer:} \quad & \hat{\mathbf{X}} = \mathbf{W} \cdot \mathbf{H} \end{aligned} \tag{2.3}$$

such that $\mathbf{W}, \mathbf{H} \geq 0$. We have thus interpreted NMF as a linear, two-layered neural network. The first layer linearly transforms the input magnitude spectrogram $\mathbf{X}$ to get the activation matrix $\mathbf{H}$. The second layer applies a linear transformation $\mathbf{W}$ on the activation matrix to give $\hat{\mathbf{X}}$ which is an approximation of the $\mathbf{X}$. The weights of the first layer given by $\mathbf{W}^{\ddagger}$ act as a version of the pseudo-inverse of $\mathbf{W}$ to estimate a non-negative $\mathbf{H}$. This formulation does not functionally improve upon the existing NMF model and is also not in agreement with classical neural network architectures. We can relax and generalize this model into a neural-network like formulation by incorporating a non-linearity $g(.)$ into the neural network.

$$\begin{aligned} 1^{\text{st}} \text{ layer:} \quad & \mathbf{H} = g\left(\mathbf{W}^{\ddagger} \cdot \mathbf{X}\right) \\ 2^{\text{nd}} \text{ layer:} \quad & \hat{\mathbf{X}} = g\left(\mathbf{W} \cdot \mathbf{H}\right) \end{aligned} \tag{2.4}$$

Here $g : \mathbb{R}^{M \times N} \mapsto \mathbb{R}_{\geq 0}^{M \times N}$ denotes an element-wise function that maps from the domain of real numbers to the domain of non-negative real numbers. Popular examples of such functions in the neural network literature include the rectified linear unit $g(x) = \max(x, 0)$, the softplus function $g(x) = \log(1+e^x)$ or even the absolute value function $g(x) = |x|$. Incorporating such a non-linearity into the neural network automatically ensures the non-negativity of the activation matrix $\mathbf{H}$ and the reconstruction $\hat{\mathbf{X}}$. We use the softplus non-linearity for all our plots and experiments.

To train the autoencoder, we use a suitable cost function that measures the discrepancy between the network output $\hat{\mathbf{X}}$ and the input $\mathbf{X}$ and seek to minimize it through back-propagation. In the case of NMF, several such cost functions have been proposed and used to learn NMF decompositions in the case of audio signals. Popular cost functions include the Frobenius norm [10], the Kullback-Leibler (KL) divergence [10] and the Itakura-Saito (IS) divergence [27]. Due to its overwhelming popularity, we will use KL divergence as our preferred cost function for all our experiments and figures.

In the case of NAEs, we also relax the constraint of the basis matrix $\mathbf{W}$

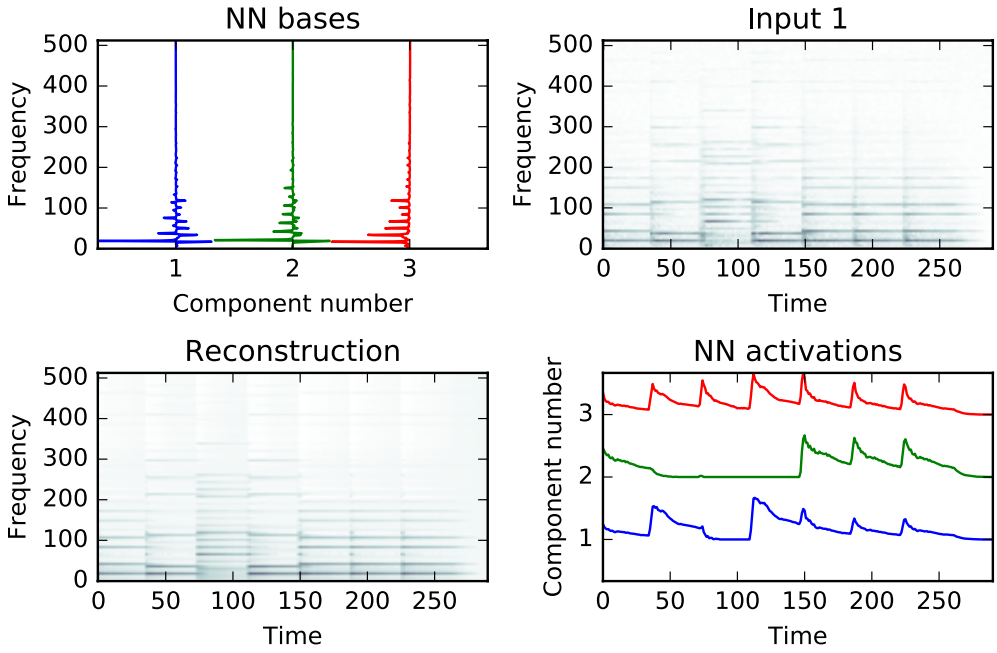

Figure 2.2: The learned basis and activation matrices for a non-negative autoencoder. As in the case of NMF, we minimize the KL divergence between the input and its reconstruction, $KL(\mathbf{X}||\hat{\mathbf{X}})$. We see that the columns of the basis matrix acquire negative values and cross-cancel unlike the case of NMF.

compared to NMF. In fact, as shown in Figure 2.2, the weights of the decoder can indeed take up negative values and this allows for cross-cancellations between the bases when representing the input spectrograms. Consequently, the activation matrix $\mathbf{H}$ indicates that multiple bases are often activated in trying to best approximate the input spectrogram. This can be alleviated by introducing a sparsity constraint into the cost function in the form of an L-1 loss on the activation matrix. As shown in Figure 2.3, we see that the basis and activation matrices are now qualitatively similar to NMF. Like NMF, we can now use this modeling strategy to learn suitable generative models for the sources in a mixture and use them for source separation.

2.2.1 Fitting a model on unseen inputs

We now turn our attention to the problem of fitting a trained NMF or NAE model on an unseen example. Figure 2.4 shows the block diagram of the

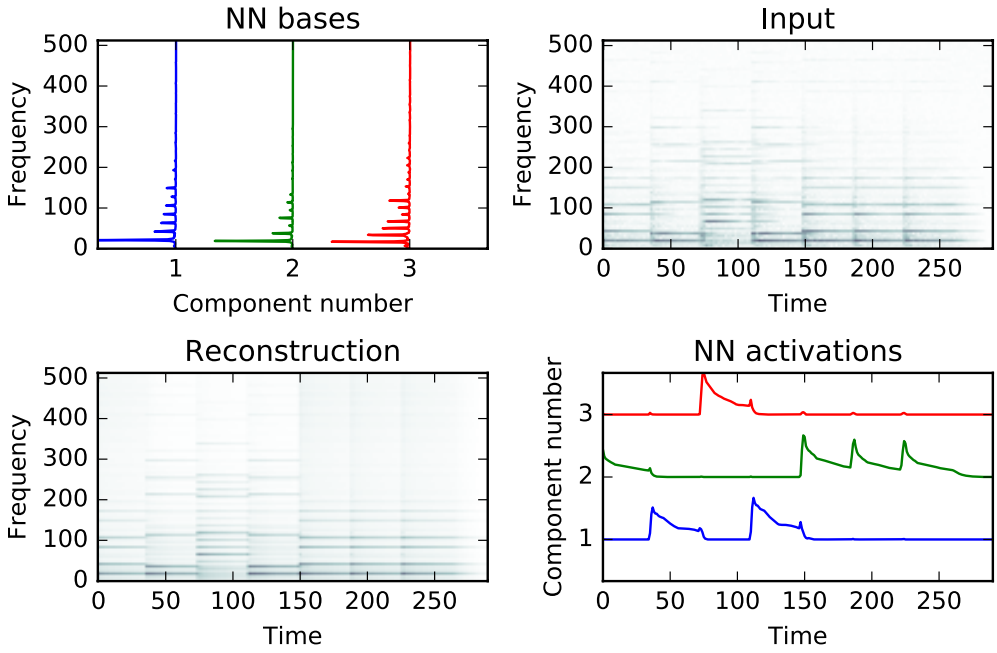

Figure 2.3: The learned basis and activation matrices for a non-negative autoencoder. As in the case of NMF, we minimize the KL divergence between the input and its reconstruction. In addition, we also incorporate a sparsity constraint on the activation matrix and incorporate it into the cost function. Thus, we minimize the overall cost function given by $KL(\mathbf{X}||\hat{\mathbf{X}}) + ||\mathbf{H}||_1$. We see that the columns of the basis and the rows of the activation matrix are now qualitatively similar to NMF.

fitting process in the case of NMF models. As shown in the figure, the first step is to use the available training data to learn suitable models for the audio signals. Thus, given the training spectrogram $\mathbf{S_1}$, we learn an NMF decomposition of the input training spectrogram to learn the basis matrix $\mathbf{W_1}$ and the activation matrix $\mathbf{H_1}$. The columns of $\mathbf{W_1}$ act as the representative bases for the sounds. Thus, we discard the activation matrix $\mathbf{H_1}$ and reuse the basis matrix $\mathbf{W_1}$ in the fitting step. In the fitting step, given an unseen spectrogram $\mathbf{X}$ and the model $\mathbf{W_1}$, the goal is to estimate a suitable activation matrix $\mathbf{H_{x_1}}$ such that the unseen spectrogram is approximated as

$$\mathbf{X} \approx \mathbf{W_1} \cdot \mathbf{H_{x_1}} \tag{2.5}$$

In terms of the optimization procedure, this is a simplified problem where we use similar multiplicative updates only for the activation matrix of NMF,

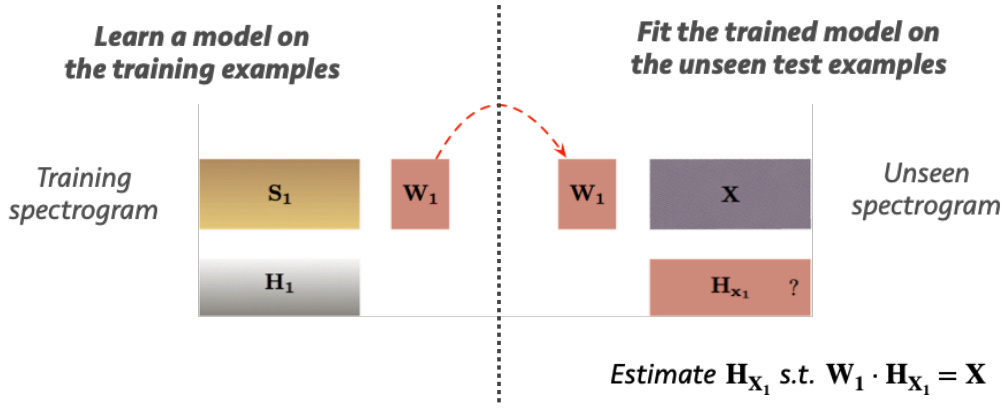

Figure 2.4: Illustration of how we can fit a pre-trained NMF model on an unseen test example. Fitting the NMF model amounts to estimating the activation matrix, while keeping the basis matrix fixed.

while keeping the basis matrix fixed.

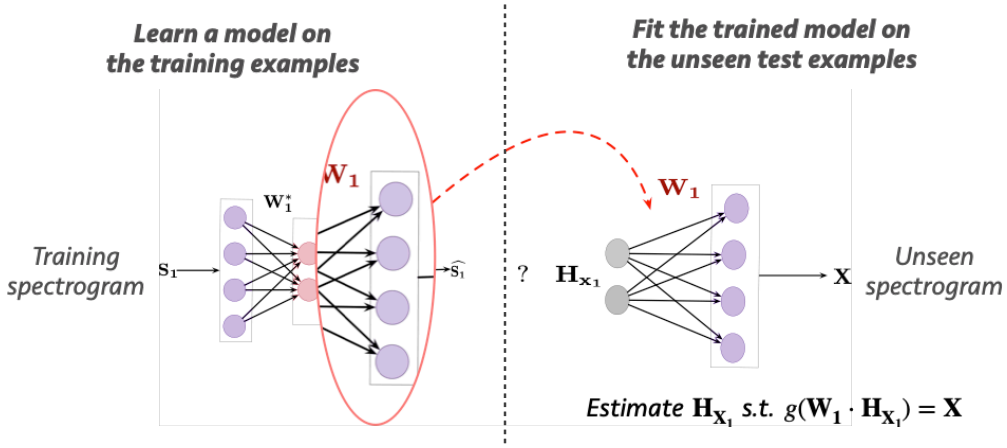

Figure 2.5: Illustration of how we can fit a pre-trained NAE on an unseen test example. Fitting the NAE amounts to training for the right input to the NN while keeping the weights fixed.

To avail such a fitting procedure for our NAE models, we re-interpret this problem in an NAE setting. Figure 2.5 shows the block diagram of the fitting process in the case of NAE models. As before, the first step is to use the available training spectrogram to learn a suitable NAE model. We first train a NAE on the input training spectrogram. In this case, the weights of the decoder act as the model for the sounds in the input audio signal. In the fitting step, we reuse this trained decoder to construct a new fitting NN. Given the pre-trained decoder weights, the goal is to estimate a suitable

activation matrix $\mathbf{H_{x_1}}$ such that the unseen spectrogram $\mathbf{X}$ is approximated as

$$\mathbf{X} \approx g(\mathbf{W_1} \cdot \mathbf{H_{x_1}}) \tag{2.6}$$

In other words, instead of training for the weights of a neural network, we now train to find the appropriate input to the neural network so as to approximate the given unseen spectrogram. In terms of coding complexities, this is as straightforward as a regular neural network and only requires switching the inputs to be trainable symbolic variables.

2.3 Single-channel Source Separation

We now consider some applications of our NAE models. Given the popularity of NMF for source separation, we now show how we can use NAEs as an alternative to NMF for supervised single-channel source separation. Given a mixture of concurrently active sounds, the goal of single-channel source separation is to extract the underlying sounds in the mixture. To do so, we assume that we have clean training examples available for each of the sources in the mixture. We use these training examples to build suitable models for our sources.

We now show how we can use our NAE models to perform supervised single-channel source separation. Figure 2.6 briefly describes the separation procedure. As shown in the figure, the source separation problem uses a modified version of the fitting problem described in Section 2.2.1. As an example, we consider that the given mixture consists of two sources and the goal is to isolate the individual sources from the mixture. Like NMF [13, 28], source separation using NAEs is a two-step procedure. The first step is to learn suitable NAE models for the sources in the mixture. We use the clean training examples for the sources and train NAEs for each of the sources. The decoders of these NAEs now act as representative models for the sources. The second-step of the procedure involves constructing a new separation-NN using these pretrained decoders. The goal now is to estimate the unknown activation matrices $\mathbf{H_{x_1}}$ and $\mathbf{H_{x_2}}$ such that the unseen mixture spectrogram

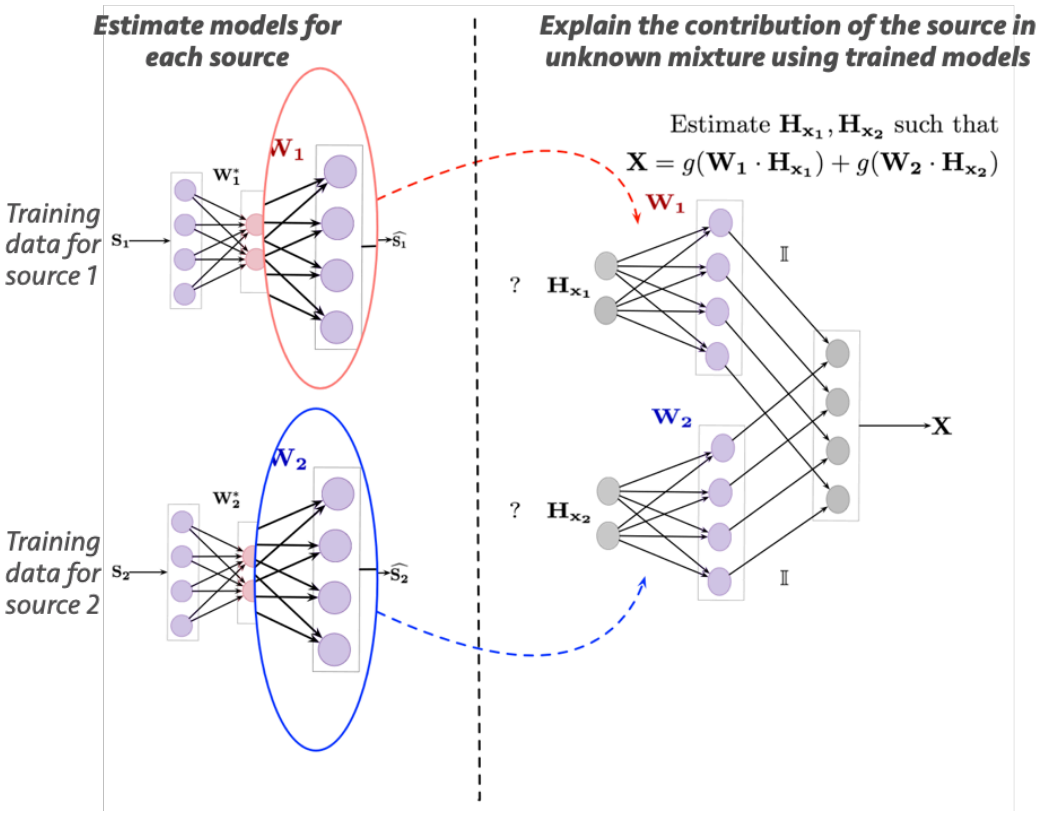

Figure 2.6: Single-channel source separation using NAEs.

$\mathbf{X}$ can be approximated as

$$\mathbf{X} \approx g(\mathbf{W_1} \cdot \mathbf{H_{x_1}}) + g(\mathbf{W_2} \cdot \mathbf{H_{x_2}}) \tag{2.7}$$

Generalizing this idea, the goal of the separation procedure is to estimate the activation matrices (decoder inputs) $\mathbf{H_{x_1}}$ and $\mathbf{H_{x_2}}$ such that the decoder outputs add up to approximate the spectrogram of the unseen mixture. In order to evaluate our NAE models and see how they compare to NMF models, we evaluate the separation performance of NAE and NMF models on a source separation experiment. The details of our experiments and their results are discussed in Section 2.6.

2.4 Advantages of NAEs

We now consider the advantages NAE models offer over discriminative NN models and NMF models, particularly in the context of source separation. However, we note that these advantages are general and hold for other applications as well.

In the case of NAEs, we primarily learn generative models using clean training examples for the sounds. Unlike discriminative separation models, these models do not require any mixing or additional data augmentation for their training. The availability of a trainable inference (fitting) step allows the model to automatically adapt to mixtures at different SNRs. In addition, NAEs are neural network interpretations of NMF and they retain the "modularity" and "reusability" of NMF models. Once we have the models for the sources, these models can be used to extract the sources from any mixture containing the sources, irrespective of the interfering sources. This ability of NAE models to adapt to unseen mixing conditions is evaluated in Section 4.3.3.

NAEs also offer a very significant advantage over NMF based generative models. We can exploit the available diversity of sophisticated neural network architectures to propose multi-layered, convolutional and recurrent extensions to our NAEs. Proposing such extensions in the case of NMF requires a significant re-working of the model and the corresponding update equations [29]. We discuss these extensions further in Section 2.5.

2.5 NAE Extensions

To further explore the generalizations of our NAE models, we consider the following extensions.

2.5.1 Multi-layer NAEs

The first extension to the single-layer NAEs described in Section 2.2 is to replace the encoders and decoder by multi-layered neural networks. We im-

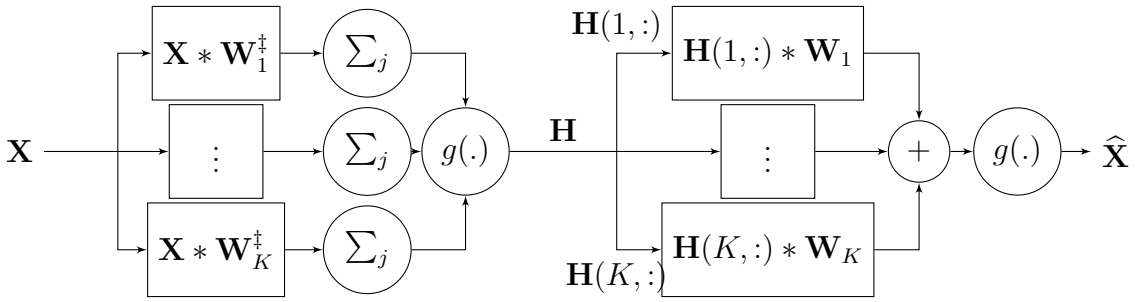

Figure 2.7: Block diagram of CNN-CNN autoencoder.

plement the multi-layered NAE as follows:

$$\begin{aligned} \mathbf{Y}_0 &= \mathbf{X} \\ \mathbf{Y}_i &= g(\mathbf{W}_i \cdot \mathbf{Y}_{i-1}), i = 1, 2, ..., 2L \\ \mathbf{H} &= \mathbf{Y}_L \\ \hat{\mathbf{X}} &= \mathbf{Y}_{2L} \end{aligned} \tag{2.8}$$

Thus, the multi-layered NAE uses a total of $2L$ layers with L layers each for the encoder and the decoder. In this formulation, the layers are enforced to be symmetric about the latent representation. In other words, if the i^{th} layer of the encoder $\mathbf{W}_i \in \mathbb{R}^{M \times N}$, then the $(2L+1-i)^{th}$ layer of the decoder $\mathbf{W}_{2L+1-i} \in \mathbb{R}^{N \times M}$. In this case, the output the encoder, i.e., the output of the L^{th} layer, is regarded as the latent representation. As before, the goal of the decoder is to produce an approximation of the input $\hat{\mathbf{X}}$. To train the network, we minimize the discrepancy between the network's output $\hat{\mathbf{X}}$ and input $\mathbf{X}$, and train using the same methods as before. We compare the separation performance of multi-layer NAEs to single-layer NAEs in our experiments in Section 2.6.

2.5.2 Convolutional NAEs

The single- and multi-layer NAE models do not consider the spectro-temporal relationships present in an audio spectrogram. To do so, we can replace the dense layers in our NAE by convolutional layers as follows:

$$
\begin{aligned}
1^{\text{st}} \text{ layer: } \mathbf{H}(i,t) &= g\left(\sum_{j=0}^{M-1}\sum_{k=0}^{T-1}\mathbf{W}_i^{\ddagger}(j,k)\mathbf{X}(j,t-k)\right) \\
2^{\text{nd}} \text{ layer: } \hat{\mathbf{X}}(f,t) &= g\left(\sum_{i=1}^{K}\sum_{k=0}^{T-1}\mathbf{W}_i(k,f)\cdot\mathbf{H}(i,t-k)\right)
\end{aligned} \tag{2.9}
$$

Here, barring the non-linearity $g(.)$, each column of the latent representation is now given as a linear combination of T previous columns of the input spectrogram. These cross-frame relationships now allow the model to learn spectro-temporal bases and activations. Effectively, we have now replaced the column vectors of $\mathbf{W}$ by matrices of size $M \times T$, where T represents the depth of the convolution and M denotes the height of the input matrix $\mathbf{X}$. $\mathbf{W}_i$ and $\mathbf{H}$ correspond to the i^{th} basis matrix and the activation matrix respectively. As before, the filters of the encoder convolutional neural network (CNN) act as inverse filters. We will refer to this NAE as the CNN-CNN autoencoder (CCAE) in the following sections. Figure 2.7 shows the block diagram of our CCAE.

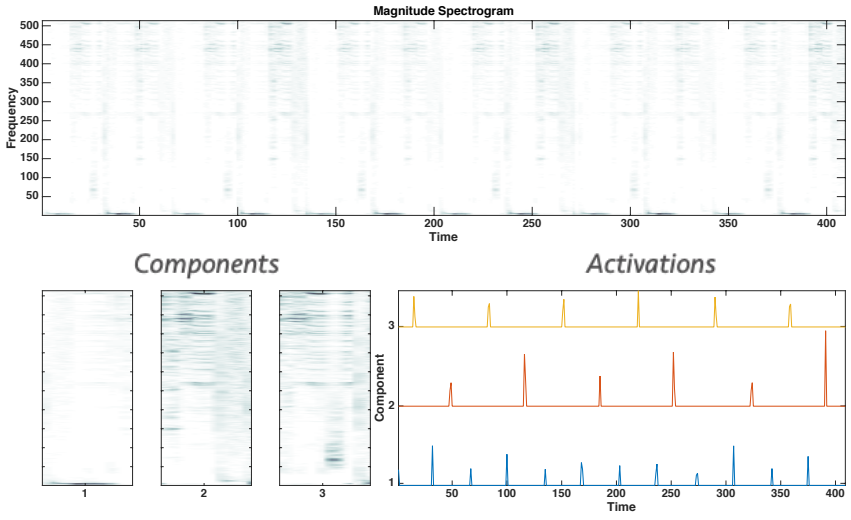

Figure 2.8: CCAE trained on a magnitude spectrogram of a sequence of drum sounds. We see that the decoder filters capture spectro-temporal bases that resemble the different drums in the audio signal.

The ability of our CCAE to learn spectro-temporal basis functions can be understood better by training a CCAE on the magnitude spectrogram of an

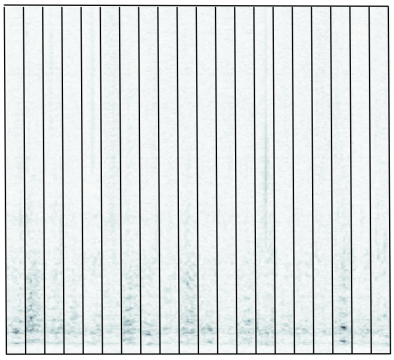

Figure 2.9: A subset of decoder filters obtained by training the CCAE on magnitude-spectrograms of the utterances of a male speaker. The decomposition uses a rank of 80 and a filter width of 8 frames. We see that the filters approximately resemble snippets of a speech spectrogram and capture harmonic information.

audio signal consisting of drum sounds. Drum sounds are characterized by an impulsive burst followed by a temporal decay. Depending on the resonances of the drum, some frequencies are sustained longer than the others. The spectrogram of the audio example is shown in Figure 2.8. As before, we perform a rank-3 NMF decomposition. The 3 decoder filters are shown in the figure as components and their corresponding activations are also shown alongside. To train the autoencoder we use the KL divergence with an added sparsity regularizer on the activations. We see that the filters learn spectro-temporal bases that look like the magnitude spectra of the drum sounds. For example, the mid-range portion of component-3 clearly indicates the snare drum. Likewise, the sustained low-frequency part in component-1 indicates a bass drum. As shown in Figure 2.9, training on speech spectrograms also allows the CCAEs to learn spectro-temporal filters that resemble snippets of the spectrogram of speech itself.

2.5.3 Recurrent NAEs

The third extension we consider is the use of a hybrid NAE where the encoder consists of a recurrent NN and the decoder consists of a convolutional NN. From a signal processing perspective, the inverse of a finite impulse response filter has an infinitely long impulse response. Since the filters of the encoder

act as inverse filters to the decoder, the use of (theoretically) infinitely long temporal dependencies can be beneficial to the NAE. Thus, we consider the recurrent-convolutional autoencoder (RCAE) next.

The block diagram of the recurrent encoder is shown in Figure 2.10. The goal of this construction is to have K independent recursive channels where K is the rank of the NAE latent representation. The k^{th} recursion in the encoder is given as

$$\mathbf{Z}(k_1, t, k) = \tanh\left(\sum_{k_2=1}^{K_{in}} \mathbf{W}^{\ddagger^k}(k_1, k_2)\mathbf{Z}(k_2, t-1, k) + \sum_{l} \mathbf{U}^{\ddagger^k}(k_1, l)\mathbf{X}(l, t)\right),\ k \in \{1, \ldots, K\}, \tag{2.10}$$

Here, $\mathbf{Z}(:, t, k) \in \mathbf{R}^{K_{in}}$ denotes the hidden vector of the k^{th} RNN at frame t. The dimensionality of this hidden state is denoted by K_{in}. The recurrent and projection matrices of the k^{th} RNN are given by $\mathbf{W}^{\ddagger^k}$, and $\mathbf{U}^{\ddagger^k}$ respectively. Although these equations indicate the use of a vanilla-recursion, there is no limitation on the type of recursion we could use. In our experiments, we use the LSTM architecture [30, 31]. The encoder output $\mathbf{H}(i, t)$ is then obtained by adding the outputs of the recurrent network over the latent dimension (first dimension).

$$\mathbf{H}(i, t) = \sum_{k_1=1}^{K_{in}} \mathbf{Z}(k_1, t, i) \tag{2.11}$$

More recently, probabilistic variants of these autoencoders obtained by using variational autoencoders (VAEs) have also been considered [32].

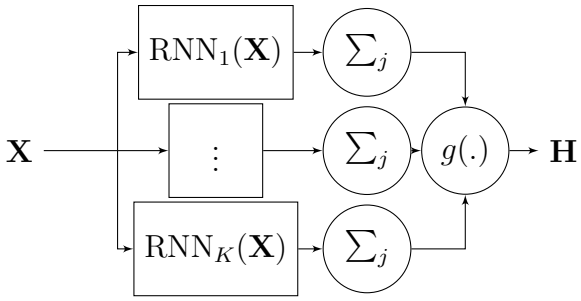

Figure 2.10: Block Diagram of RNN Encoder

2.6 Experiments

We now present some experiments and results to evaluate the performance of our NAE models in the context of source separation. We primarily discuss the results of three of our experiments. The first experiment is aimed at comparing the separation performance of one-layer and two-layer NAEs to NMF. Here, the numbers represent the number of layers in both the encoder and decoder. The second experiment aims to compare the separation performance of one-layer and two-layer NAEs for varying decomposition ranks (size of the latent representation). The final experiment compares CCAE and RCAE models to one-layer NAEs. We begin with a description of the experimental setup used.

2.6.1 Experimental setup

For our experiments, we use speaker utterances from the TIMIT database [33]. The database consists of 10 clean utterances per speaker. We use a randomly selected subset of 9 of these sentences for training and the remaining utterance is used for testing. For the evaluation, we use 32 such mixture pairs of randomly selected users from the database. The test utterances are mixed at a SNR of 0 dB for these experiments. To compute the magnitude spectrogram of these utterances, we use a 512-pt DFT, a square-root Hann window, and a hop size of 25%. The training examples are used to train NMF, NAE, CCAE and RCAE models for the sources. To transform the separated magnitude spectrograms back into their waveform representations, we use the overlap-and-add inverse STFT operation as follows:

$$s_i(t) = \text{STFT}^{-1}\left(\frac{\hat{\mathbf{X}}_i}{\sum_i \hat{\mathbf{X}}_i} \odot \mathbf{X} \odot e^{i\mathbf{\Phi}}\right) \tag{2.12}$$

Here, $\mathbf{X_i}$ denotes the estimated magnitude spectrogram of the i^{th} source, $\mathbf{X}$ represents the magnitude spectrogram of the mixture and $\mathbf{\Phi}$ represents the matrix containing the mixture phase. As before, the operator $\odot$ denotes element-wise multiplication and the division is also element-wise.

The separation performance is measured in terms of the BSS_eval metrics [8] viz., signal-to-distortion ratio (SDR), signal-to-interference ratio (SIR) and signal-to-artifacts ratio (SAR). The results are shown in the form of a box-plot. The solid line at the center indicates the median value and the extremities of the box indicate the inter-quartile range (25 percentile and the 75 percentile points).

2.6.2 Experiment 1: NMF vs shallow NAE vs multi-layer NAE

The first experiment aims to compare the separation performance of single-layer ($L = 1$) and multi-layer NAEs ($L = 2$) with that of NMF. Figure 2.11 shows the results of the experiments. The top figure compares the performance of the NAEs with NMF for a decomposition rank $K = 20$. We see that both these versions are comparable to NMF in terms of source separation performance. The bottom figures show the same comparison for a decomposition rank of $K = 100$. Here, the multi-layer NAE significantly outperforms NMF and single-layer NAEs.

2.6.3 Experiment 2: NAE performance for varying decomposition ranks

Figure 2.12 compares the performance of single and multi-layer NAEs over varying decomposition ranks. The points show the median value of the separation performance over all the 32 test mixtures. We see that the multi-layer

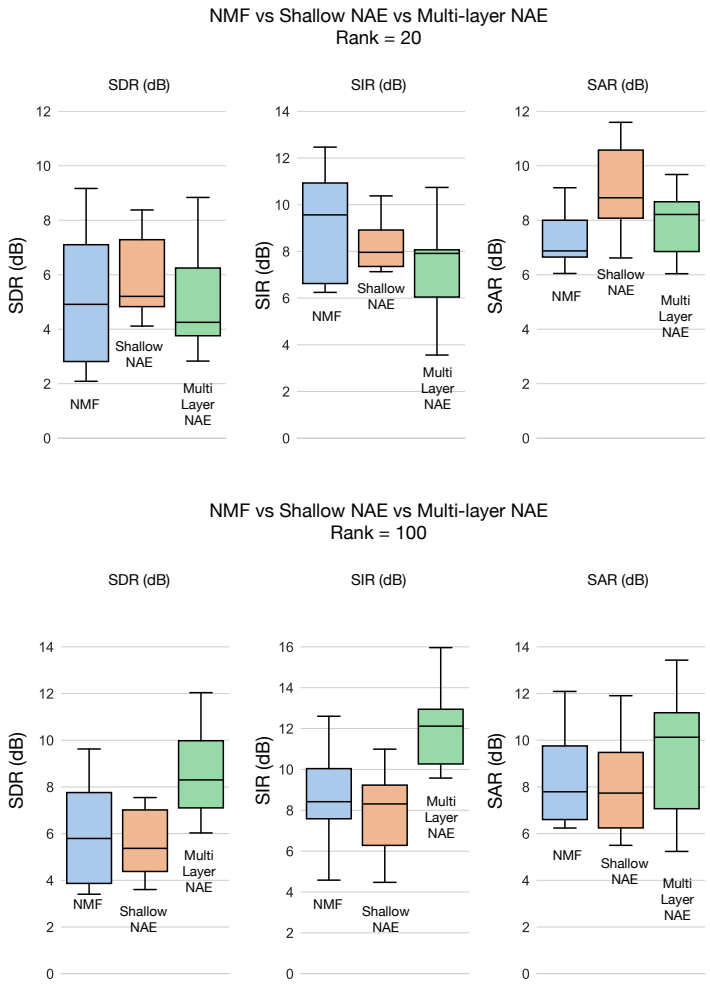

Figure 2.11: Comparison of source separation performance on speech/speech mixtures between NMF, shallow NAEs ($L = 1$) and multi-layer NAEs ($L = 2$). The top figure shows this comparison for a decomposition rank of 20 (top). The bottom figure shows the same comparison for a decomposition rank of 100.

NAE consistently outperforms single-layer NAEs and the separation performance improves with higher decomposition ranks. These generalizations and improvements are not easily possible in the case of NMF models.

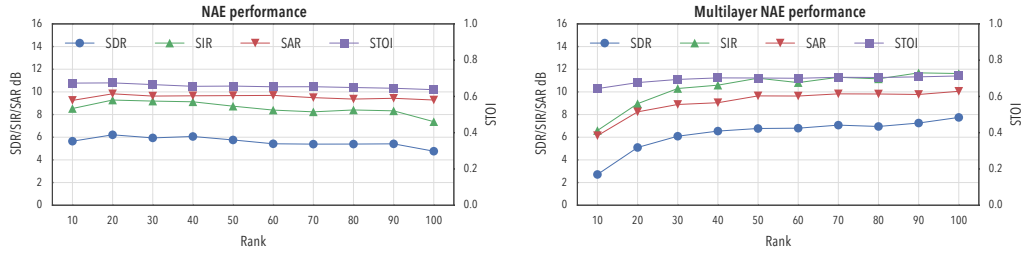

Figure 2.12: Comparison of single-layer NAEs (left) and multi-layer NAEs (right) over varying decomposition ranks. We also include a comparison in terms of Short-term Objective Intelligibility [7] in addition to the BSS_Eval metrics [8].

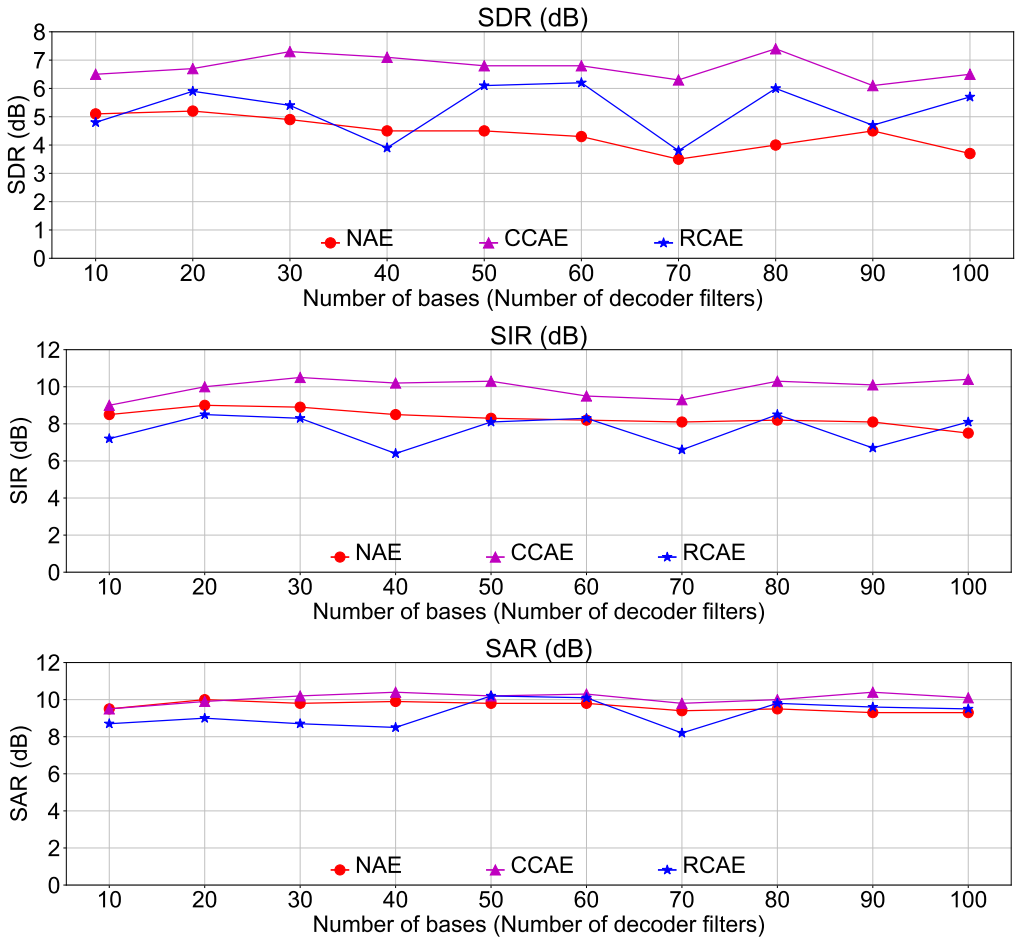

Figure 2.13: Comparison of separation performance of NAE, CCAE and RCAE models for varying decomposition ranks. The SAR values for all the models seem to be comparable. However, the suppression of interfering sources improves significantly in the case of CCAE models (SIR). This gives a significant boost in overall separation performance (SDR).

2.6.4 Experiment 3: NAE vs CCAE vs RCAE

The final experiment compares the separation performance of CCAEs and RCAEs with single-layer NAEs and the results are shown in Figure 2.13. For clarity and interpretability reasons, we show a plot of the median values of the metrics for varying decomposition ranks (number of filters in the decoder). Here again, we see that there is a significant improvement in separation performance using sophisticated neural networks over single-layer NAEs. Of all the three models, the CCAE model gives the best results in terms of overall separation performance across all the metrics. The RCAE also gives a significant improvement over NAEs. This improvement is not as pronounced as the CCAE model. All the models are comparable in terms of their SAR values. These experiments indicate the importance of generalizability to NAE performance, particularly in the context of single-channel source separation.

CHAPTER 3

END-TO-END MODELS FOR AUDIO SOURCE SEPARATION

In Chapter 2, we have introduced the idea of non-negative audio modeling and non-negative autoencoders (NAEs). We have also seen that these NAE models offer significant modeling advantages compared to discriminatively trained neural networks (NNs) and non-negative matrix factorization (NMF) models.

However, all of these models continue to operate on magnitude spectrograms of the audio signal. For example, in several discriminative source separation models, the magnitude spectrogram of the mixture is given as an input to the neural network and the neural network is trained to estimate the magnitude spectrograms of the clean sounds. Even in the case of generative models like NMF and NAEs, we continue to model the magnitude spectrograms of the clean sounds as a low-rank factorization or an autoencoder respectively. In doing so, we are completely neglecting the phase of the audio signal in all the approaches.

In the case of audio signals, the short-time Fourier transform (STFT) is used as a universal front-end transformation for almost all the applications. Consequently, the representation of the audio signal may not necessarily be optimal for all the tasks. In other allied audio applications like speech recognition [34] and music information retrieval (MIR) [35], the use of data-driven representations has been shown to improve performance. These data-driven representations can be trained with the rest of the model to learn representations optimal to the dataset and the application. More recently, even in the case of NMF, the use of a trainable audio representation has been shown to significantly boost modeling and source separation performance [36].

Audio applications like speech recognition and MIR operate on the audio

signal to learn underlying information. Thus, these applications do not require a strategy to invert the data-driven representations of the audio signal back into the waveform domain. In the applications primarily of our interest (source separation, de-reverberation, audio enhancement, bandwidth extension etc.), we aim to reconstruct the enhanced version of the audio signal from the latent representation. Consequently, these applications demand the availability of trainable forward and inverse transforms in order to learn data-driven front-ends.

To this end, we interpret the operations involved in computing the STFT as a neural network and use the resulting model as a trainable transform layer. This allows us to develop models that operate directly on the audio waveforms and learn invertible optimal representations. Several evaluation metrics are often defined on the domain of waveforms. With the availability of such end-to-end networks, these evaluation metrics can also be interpreted as suitable cost functions to train our networks. Modeling the STFT as a neural network also allows us to propose interesting generalizations to the STFT that outperform existing front-end representations. We primarily explore these ideas in the current chapter. In particular, we explore these topics in the context of discriminative end-to-end source separation. The extendable ideas that can be used to improve NAE models are summarized in Section 3.5

3.1 From STFTs to Adaptive Front-ends

3.1.1 Complex valued STFT representation

The STFT converts the audio signal into a complex-valued time-frequency (TF) representation. Figure 3.1 denotes the block diagram of the STFT-based source separation network. Given the audio waveform x, the generalized short-time transform of the audio signal can be written as

$$\mathbf{X}(n,k) = \sum_{t=0}^{N-1} x(nh+t) \cdot w(t) \cdot b(k,t) \tag{3.1}$$

Here, $\mathbf{X}(n,k)$ represents the frequency domain coefficient corresponding to the k^{th} component in the n^{th} time frame of x, N represents the size of the window function w, and h denotes the hop size used. $b(k,t)$ represents the basis functions used for the representation.

For the STFT representation, the basis functions $b(k,t)$ are complex exponentials defined by the complex DFT operation. These basis functions can be written as

$$b(k,t) = \exp\left(j\frac{2\pi \cdot k \cdot t}{N}\right) \tag{3.2}$$

where integers k, t denote the frequency and time indices such that, $0 \leq k, t \leq N-1$. Thus,

$$\mathbf{X}_{\text{STFT}}(n,k) = \sum_{t=0}^{N-1} x(nh+t) \cdot w(t) \cdot \exp\left(j\frac{2\pi \cdot k \cdot t}{N}\right)$$

From the complex valued STFT representation, we compute the magnitude and phase components as follows:

$$\mathbf{M}_{\text{STFT}}(n,k) = |\mathbf{X}_{\text{STFT}}(n,k)|$$
$$\mathbf{P}_{\text{STFT}}(n,k) = \exp\left(j \cdot \angle\mathbf{X}_{\text{STFT}}(n,k)\right)$$

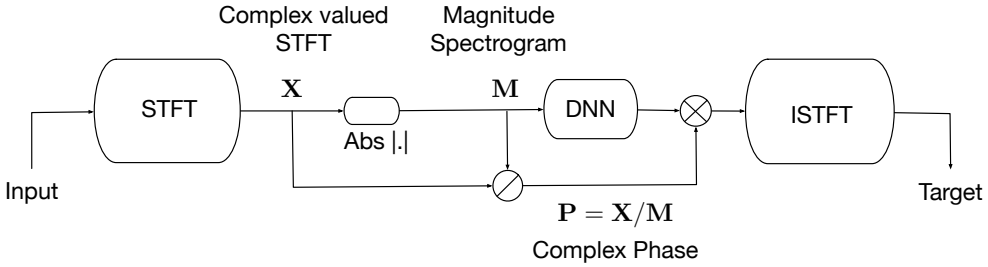

Figure 3.1: Source separation using the STFT. Magnitude and phase are initially separated, then a neural network operates on the magnitudes, and phase is used to modulate the network's output in order to produce the resulting source waveform.

As shown in Figure 3.1, the magnitude component ($\mathbf{M}_{\text{STFT}}$) of the STFT representation is now fed as the input to the neural network. The network is trained to estimate the magnitude spectrogram of the source given the mixture spectrogram. The mixture phase is multiplied element-wise to the estimated source spectrogram. We then transform the separated source to the time domain using the inverse STFT operation.

3.1.2 Smoothed STFT representation

Until recently, the optimization of deep complex-valued neural networks had not been explored in detail [37]. In the case of audio signals, this would mean that the phase component would have to be truncated, resulting in significant loss of information. In addition, the lack of availability of automatic-differentiation toolboxes and coding support meant that developing and training these networks from scratch required an enormous amount of coding. To circumvent these numerical challenges, we can retain all the information in the STFT by stacking the real and imaginary coefficients into a real-valued vector of size $2N$, for every frame. As shown in Eq. (3.1), the STFT coefficient $\mathbf{X}_{nk}$ can be written as a convolution between the current frame $[x_{nh}, x_{nh+1}, \ldots, x_{nh+N-1}]$ and the k^{th} basis, weighted by the window function. Thus, the real and imaginary parts of the STFT coefficients can be written as the output of a convolutional layer. The convolutional layer filters are set to $\mathbf{B}$ which is obtained by stacking the real and imaginary parts of the STFT bases as follows:

$$\mathbf{B} = \begin{bmatrix} \cos\left(\frac{2\pi \cdot k \cdot t}{N}\right)_{0 \leq k,t \leq N-1} \\ \sin\left(\frac{2\pi \cdot k \cdot t}{N}\right)_{0 \leq k,t \leq N-1} \end{bmatrix}$$

Instead of computing the magnitudes of the STFT representation, we could directly train the neural network to operate on these real-valued coefficients. We note here that the representation now contains $2N$ coefficients as opposed to the N STFT magnitude coefficients. To obtain a representation akin to STFT magnitudes, we use an abs$(\cdot)$ non-linearity at the output of the convolutional layer.

In order to get an intuitive understanding of the real-valued representation,

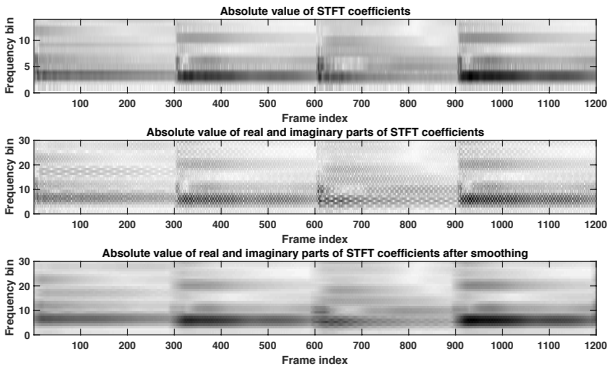

Figure 3.2: (a) Absolute value of the first 15 STFT coefficients for a sequence of piano notes. (b) Modulus of equivalent real and imaginary parts of STFT coefficients. Note that we now deal with the first 30 coefficients instead. The unsmoothed coefficients rapidly vary and have to be smoothed across time to resemble what we would expect as a magnitude spectrogram. (c) Modulus of real and imaginary parts of STFT coefficients after smoothing by a rectangular filter of length 5.

we compare the absolute value of the output of the front-end convolutional layer ($|\mathbf{X}|$) with the STFT magnitudes for a simple audio signal consisting of a sequence of piano notes. The plots of this comparison are shown in Figure 3.2. Unlike the STFT magnitudes, the outputs of the convolutional layer exhibit rapid variability as a consequence of incorporating the phase into the representation. These variations could potentially be dependent on the frequency of the STFT bases, the frame-size and hop-size of the front-end transform. To obtain a smooth representation that resembles the STFT magnitudes, we apply a temporal smoothing operation on $|\mathbf{X}|$. The effect of smoothing the real and imaginary STFT coefficients using a rectangular window of duration 5 samples is shown in Figure 3.2. We see that the smoothed coefficients are now similar to the STFT magnitudes.

The temporal smoothing operation can also be implemented as a convolution operation across time and hence as a convolutional layer.

$$\mathbf{M} = |\mathbf{X}| * p \tag{3.3}$$

Here, $|\cdot|$ represents the element-wise modulus operator, p represents a filterbank of smoothing filters and $*$ denotes the one-dimensional convolution

operation which operates only along the time axis. Replacing this smoothing operation by a convolutional layer would allow us to learn smoothing filters tailored to each frequency component. We follow the smoothing layer with a softplus non-linearity to obtain a carrier/modulator representation of the audio signal, similar to the STFT coefficients. The output of the smoothing layer is denoted as the modulation component (M) and captures the smooth aspects of the STFT representation. The carrier component captures the information lost by the abs(·) and temporal smoothing operation and is given as

$$\mathbf{C} = \frac{\mathbf{X}}{\mathbf{M}} \tag{3.4}$$

where the division is element-wise.

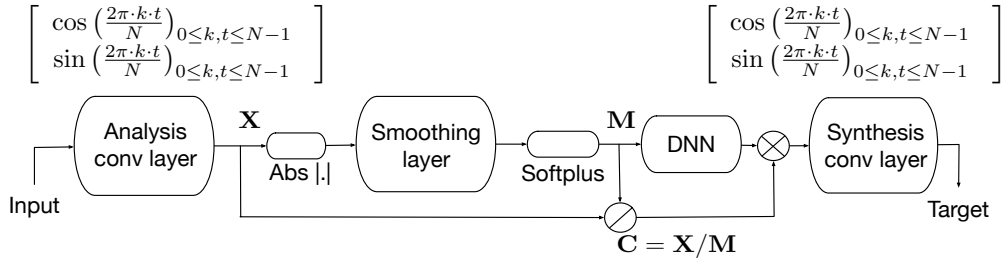

Figure 3.3: Taking the model in Figure 3.1, we implement it as a convolutional network with a skip connection. The analysis layer implements a real-valued version of the STFT by computing its real and imaginary components as separate dimensions. Subsequent steps extract their corresponding amplitude, process it, and recombine it with the original phase. The final layer implements the inverse transform that produces the output waveform. We additionally use a smoothing layer to compensate for the complementary modulations between the sine and cosine coefficients.

Figure 3.3 gives the block diagram of an end-to-end source separation network using the smoothed STFT front-end. Similar to the STFT setting, the modulation component is fed to the network. The carrier component of the mixture is multiplied with the estimated source modulation and inverted into time using a transposed convolution layer. We initialize the transposed convolutional layer as the front-end to get perfect reconstruction. For the current model, these layers are held fixed during training. From Figure 3.6, we see that the smoothed STFT model is comparable to STFT magnitudes in terms of source separation performance. This indicates that there is possibly no loss of information in using a smoothed STFT representation over the

magnitude STFT.

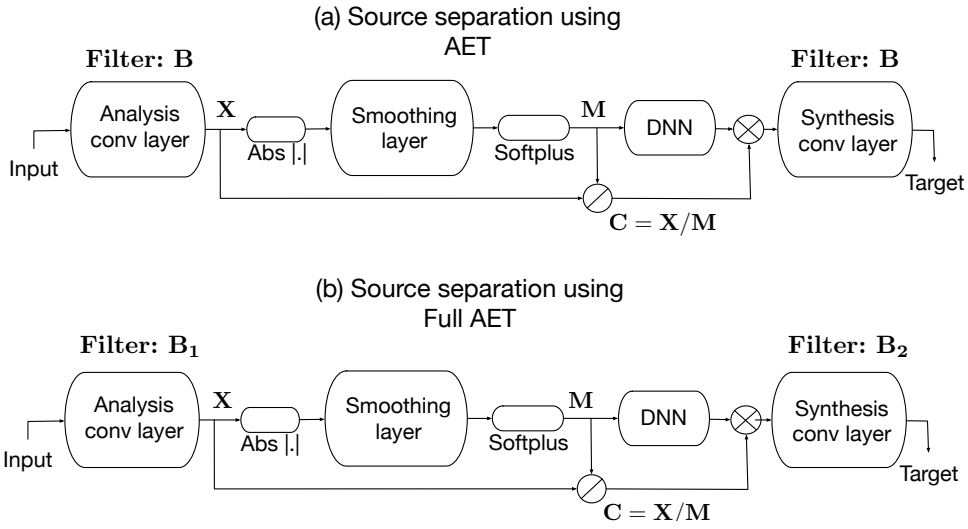

Figure 3.4: Block diagrams of end-to-end source separation networks using the AET (a), and the full-AET (b). In contrast to before, these models have a trainable front-end as opposed to using a fixed Fourier-related filterbank. In the case of the full-AET, the final layer has an independent trainable filterbank, whereas the AET is using the transpose of the learned analysis filterbank.

3.1.3 Adaptive Front Ends

We can now make the convolutional and smoothing layers learnable to learn a fully trainable TF representation for the audio signals. Making the front-end and smoothing layers adaptive allows the network to learn basis and smoothing functions directly from the raw waveform of the signal. Thus, these functions would be optimal for the separation task. The transposed convolutional layer acts as an adaptive reconstruction step that transforms the audio signal back into the waveform domain. We refer to this as the autoencoder transform (AET) front-end. This allows us the possibility to explore two possible configurations of the AET networks. In the first configuration, the front-end and synthesis convolutional layers share the same filters. The second configuration is to allow the front-end and synthesis convolutional layers to be independently trainable. We refer to these configurations as the AET and full-AET respectively. Figure 3.4 gives the block

diagrams of the AET and full-AET architectures.

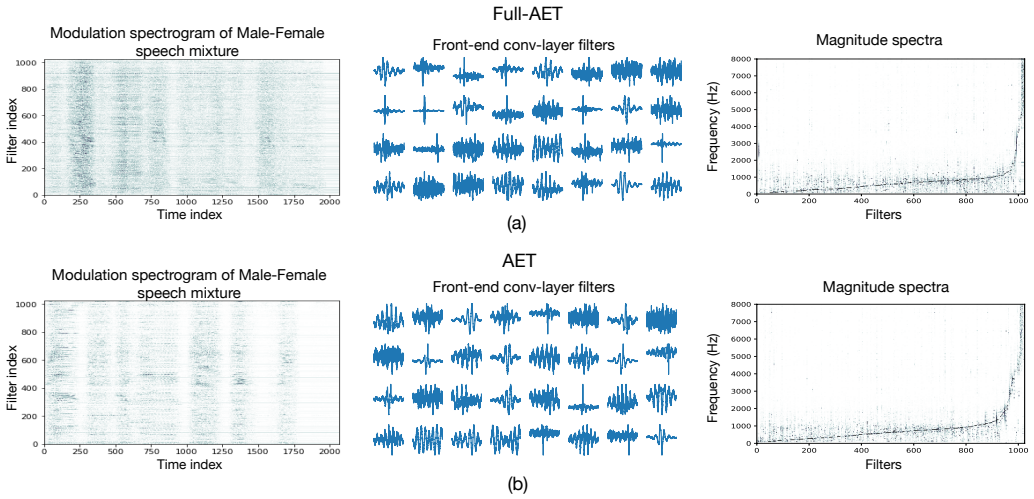

Figure 3.5: (a) A modulation spectrogram obtained of a speech mixture consisting of a male and female speaker, front-end convolutional layer filters, and their corresponding normalized magnitude spectra for the full-AET model (top) (b) A modulation spectrogram obtained for a speech mixture consisting of a male and female speaker, front-end convolutional layer filters, and their corresponding normalized magnitude spectra for the AET model (bottom). The front-end and synthesis layers share the filters in the AET model while the full-AET model allows the front-end and synthesis layers to have independent weights. The filters are ordered according to their dominant frequency component (from low to high). In the middle subplots, we show the waveforms for a subset of the first 32 filters.

Figure 3.5 shows the AET and full-AET front-end filters learned on male-female mixtures. We plot these filters in time and frequency. We see that the filters are frequency selective like the STFT. Also, the filters are concentrated towards the lower frequencies and spread out at the higher frequencies, similar to the Mel filter bank. Unlike the STFT, the filters of the front-end and synthesis convolutional layers are not restricted to be orthogonal or inverse versions of each other. In addition, the stride of the convolutional layers also plays a role in developing trainable architectures. The effect of striding on AET and full-AET models is shown in Figure 3.7.

3.2 Cost functions for End-to-end Speech Separation

Previously, magnitude spectrograms have been interpreted as probability density functions of random variables with varying characteristics. This has led to the use of divergence-based cost functions for single-channel source separation. Some of these examples include mean squared error (MSE) [38], KL divergence [26, 39] and IS divergence [39]. However these are often used as proxies to the performance metrics we ultimately measure, which are almost always waveform based. For end-to-end architectures, statistical metrics like mean squared error [40, 41] and l-1 loss [42, 43] have been tried. Cross-entropy and its modified versions have also been tried out [44]. In source separation, however, we most often evaluate the performance of source separation algorithms using BSS_Eval metrics viz., SDR, SIR, SAR [8] and Short-term Objective Intelligibility (STOI)[45] metrics. Thus, a logical step is to interpret these metrics as cost functions themselves. In these cost functions, we denote the network output waveform as $\mathbf{x}$. This output should ideally match the source waveform $\mathbf{y}$ and suppress the interference $\mathbf{z}$. Thus, $\mathbf{y}$ and $\mathbf{z}$ are fixed constants with respect to the optimization.

3.2.1 BSS_Eval cost functions

The distortions in the network output $\mathbf{x}$ can result from the effects of the interfering source $\mathbf{z}$ or from the artifacts introduced by the processing algorithm. SDR incorporates the overall effect of distortions in the network output. The effects of the interfering sources are observed in the SIR score. The processing artifacts are measured by SAR. Maximizing SDR with respect to $\mathbf{x}$ can be given as

$$\max \mathrm{SDR}(\mathbf{x}, \mathbf{y}) = \max \frac{\langle \mathbf{xy} \rangle^2}{\langle \mathbf{yy} \rangle \langle \mathbf{xx} \rangle - \langle \mathbf{xy} \rangle^2} \tag{3.5}$$

$$\equiv \min \frac{\langle \mathbf{yy} \rangle \langle \mathbf{xx} \rangle - \langle \mathbf{xy} \rangle^2}{\langle \mathbf{xy} \rangle^2} \tag{3.6}$$

$$= \min \frac{\langle \mathbf{yy} \rangle \langle \mathbf{xx} \rangle}{\langle \mathbf{xy} \rangle^2} - \frac{\langle \mathbf{xy} \rangle^2}{\langle \mathbf{xy} \rangle^2} \tag{3.7}$$

$$\propto \min \frac{\langle \mathbf{xx} \rangle}{\langle \mathbf{xy} \rangle^2} \tag{3.8}$$

Thus, maximizing the SDR maximizes the correlation between $\mathbf{x}$ and $\mathbf{y}$ and simultaneously minimizes the energy of $\mathbf{x}$ to produce a minimum energy solution.

Maximizing the SIR cost function can be given as

$$\max \mathrm{SIR}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \max \frac{\langle \mathbf{zz} \rangle^2 \langle \mathbf{xy} \rangle^2}{\langle \mathbf{yy} \rangle^2 \langle \mathbf{xz} \rangle^2} \equiv \min \frac{\langle \mathbf{xz} \rangle^2}{\langle \mathbf{xy} \rangle^2} \tag{3.9}$$

Thus, maximizing the SIR maximizes the correlation of the network output $\mathbf{x}$ with the desired source $\mathbf{y}$ and minimizes the correlation between $\mathbf{x}$ and the interference $\mathbf{z}$. Removing the minimum energy constraint makes the network focus predominantly on time-frequency bins that are dominated by $\mathbf{y}$ and do not contain $\mathbf{z}$. Thus, SIR as a cost function needs to be supported by SAR.

For the SAR cost function, we assume that the clean target source $\mathbf{y}$ and the clean interference $\mathbf{z}$ are orthogonal in waveform domain. Thus, the corresponding inner-products zero out and we can then derive the following simplification for maximizing SAR:

$$\max \mathrm{SAR}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \max \frac{\| \frac{\langle \mathbf{xy} \rangle}{\langle \mathbf{yy} \rangle} \mathbf{y} + \frac{\langle \mathbf{xz} \rangle}{\langle \mathbf{zz} \rangle} \mathbf{z} \|^2}{\| \mathbf{x} - \frac{\langle \mathbf{xy} \rangle}{\langle \mathbf{yy} \rangle} \mathbf{y} - \frac{\langle \mathbf{xz} \rangle}{\langle \mathbf{zz} \rangle} \mathbf{z} \|^2} \tag{3.10}$$

$$\equiv \min \frac{\| \mathbf{x} - \frac{\langle \mathbf{xy} \rangle}{\langle \mathbf{yy} \rangle} \mathbf{y} - \frac{\langle \mathbf{xz} \rangle}{\langle \mathbf{zz} \rangle} \mathbf{z} \|^2}{\| \frac{\langle \mathbf{xy} \rangle}{\langle \mathbf{yy} \rangle} \mathbf{y} + \frac{\langle \mathbf{xz} \rangle}{\langle \mathbf{zz} \rangle} \mathbf{z} \|^2} \tag{3.11}$$

$$= \min \frac{\langle \mathbf{xx} \rangle - \frac{\langle \mathbf{xy} \rangle^2}{\langle \mathbf{yy} \rangle} - \frac{\langle \mathbf{xz} \rangle^2}{\langle \mathbf{zz} \rangle}}{\frac{\langle \mathbf{xy} \rangle^2}{\langle \mathbf{yy} \rangle} + \frac{\langle \mathbf{xz} \rangle^2}{\langle \mathbf{zz} \rangle}} \tag{3.12}$$

$$\propto \min \frac{\langle \mathbf{xx} \rangle}{\frac{\langle \mathbf{xy} \rangle^2}{\langle \mathbf{yy} \rangle} + \frac{\langle \mathbf{xz} \rangle^2}{\langle \mathbf{zz} \rangle}} \tag{3.13}$$

The SAR cost function does not distinguish between $\mathbf{y}$ and $\mathbf{z}$ and cannot be used to train a network to separate the source from the interferences. However, in combination with SIR, it can be used for end-to-end separation.

3.2.2 Short-term Objective Intelligibility (STOI)

The drawback of BSS_Eval metrics is that they do not necessarily reflect the amount of intelligibility of the resulting output. STOI is a popular metric that correlates with subjective speech intelligibility. The first step of computing the STOI metric is to transform the audio signals into the time-frequency domain using a 512-point STFT with a "Hann" window size of 256 samples and a hop of 128 samples. The STFT representations are transformed into an octave band representation using 15 (1/3)rd octave bands that extend up to $10,000$ Hz. These steps are applied on the network output $\mathbf{x}$ and the source of interest $\mathbf{y}$ to obtain $\hat{\mathbf{X}}$ and $\hat{\mathbf{Y}}$ respectively. Here, $\hat{\mathbf{X}}_{j,m}$ corresponds to the energy of $\mathbf{x}$ in the j -th one-third octave band at the m -th time frame.

Given $\hat{\mathbf{X}}$ and $\hat{\mathbf{Y}}$, the intermediate STOI measure for one bin, denoted by $d_{j,m}$, depends on a neighborhood of N previous bins. To do so, we construct new vectors $\mathbf{X}_{j,m}$ and $\mathbf{Y}_{j,m}$ consisting of $N = 30$ previous frames before the m -th time frame as follows:

$$\mathbf{X}_{j,m} = [\hat{\mathbf{X}}_{j,m-N+1}, \hat{\mathbf{X}}_{j,m-N+2}, ..., \hat{\mathbf{X}}_{j,m}]^T$$

We scale and clip $\mathbf{X}_{j,m}$ to construct $\bar{\mathbf{X}}_{j,m}$ as follows:

$$\bar{\mathbf{X}}_{j,m}(n) = \min\left(\frac{\|\mathbf{Y}_{j,m}\|}{\|\mathbf{X}_{j,m}\|}\mathbf{X}_{j,m}(n), (1 + 10^{-\beta/20})\mathbf{Y}_{j,m}(n)\right)$$

In this equation, $\mathbf{X}_{j,m}(n)$ denotes the n^{th} value of $\mathbf{X}_{j,m}$ and β is set to -15 dB. The intermediate intelligibility matrix, $d_{j,m}$ can be calculated as the correlation between $\bar{\mathbf{X}}_{j,m}$ and $\mathbf{Y}_{j,m}$. This can be written as

$$d_{j,m} = \frac{(\bar{\mathbf{X}}_{j,m} - \mu_{\bar{\mathbf{X}}_{j,m}})^T(\mathbf{Y}_{j,m} - \mu_{\mathbf{Y}_{j,m}})}{\|\bar{\mathbf{X}}_{j,m} - \mu_{\bar{\mathbf{X}}_{j,m}}\| \cdot \|\mathbf{Y}_{j,m} - \mu_{\mathbf{Y}_{j,m}}\|} \tag{3.14}$$

The overall STOI metric is given as the average of $d_{j,m}$ over all time-frames and all octave bands. Thus, maximizing STOI as a cost function maximizes the correlation between the octave band representations of $\mathbf{x}$ and $\mathbf{y}$.

3.3 Experiments on Adaptive Front-Ends

We now describe the experiments used for our evaluations. The goal of the experiments in this chapter is two-fold: (i) To evaluate the separation performance of adaptive front-end based separation models and compare them with models that use fixed front-ends [41]. (ii) To compare the various performance-based cost functions and evaluate them on end-to-end source separation performance [46]. Considering the broad scope of these experiments, we divide this section into two parts and deal with each of them individually.

We begin with evaluating our adaptive front-ends in the context of end-to-end source separation and compare its performance with the fixed front-ends for the same task. The experimental setup is described next. In particular, we compare the separation performance of STFT, smoothed STFT, AET and full-AET models.

3.3.1 Experimental setup

For our experiments, we used the training folders (si_tr_s) of the Wallstreet Journal 0 (WSJ0) [47] corpus. For training our networks, 25 male and 25 female speakers were selected at random. For the evaluations, we selected a different set of 10 male and 10 female speakers. Thus, the evaluation is truly speaker independent because the evaluation speakers and mixtures were not a part of the training set. For training, we randomly selected a pair of male and female speakers from the training set speakers. An audio utterance was selected at random for each speaker and a 2-sec snippet was drawn at random from each utterance. These snippets were mixed at 0 dB to generate the mixture waveform. The female utterance was selected as the source of interest and all the networks were trained to separate the female speech from the mixture. For the evaluations, we constructed similar male-female speaker mixtures at 0 dB from speakers selected for testing.

We now describe the parameters of our network architecture. For the STFT front-end, the coefficients were computed using a Hann window of du-

ration 1024 samples at a hop of 16 samples. For the smoothed STFT version, the front-end consisted of 1024 coefficients with 512 real and 512 imaginary components each. For these coefficients, a smoothing operation of duration 5 samples resulted in a smooth modulation spectrogram as shown in Figure 3.2. Thus, smoothing layer filters were set to a length of 5 so as to average over 5 previous frames. For the AET networks, a front-end convolutional layer was set to have a stride of 16 samples so as to enable a fair comparison with the fixed front-end networks. The separation network consisted of a cascade of 3 dense layers, each followed by a softplus non-linearity. Each of these hidden layers were selected to have a depth of 512 neurons. For these experiments, all the networks were trained using SDR as the cost function. For the evaluation, we compare the separation performance of 4 models described in this dissertation. These models are briefly summarized below.

(i) STFT: uses STFT magnitudes for source separation (Figure 3.1).

(ii) STFT smoothed: uses smoothed real and imaginary STFT coefficients for source separation (Figure 3.3).

(iii) AET: uses autoencoder like convolutional layers to learn adaptive bases. The front-end and synthesis layers share the weights (Figure 3.4(a)).

(iv) Full-AET: uses autoencoder like convolutional layers to learn adaptive bases. The front-end and synthesis layers have independent weights (Figure 3.4(b)).

These networks are trained to separate the female speaker from a 0 dB mixture consisting of a male and a female speaker.

The first experiment aims to compare the separation performance of the proposed end-to-end architectures. We construct a training dataset of 200 minute duration. All the networks were trained on this dataset and evaluated on a testset of duration 20 minutes. We compare the models in terms of the BSS_Eval metrics: SDR, SIR and SAR computed on the test set. Figure 3.6 gives the results of the experiment in the form of a box plot that shows the

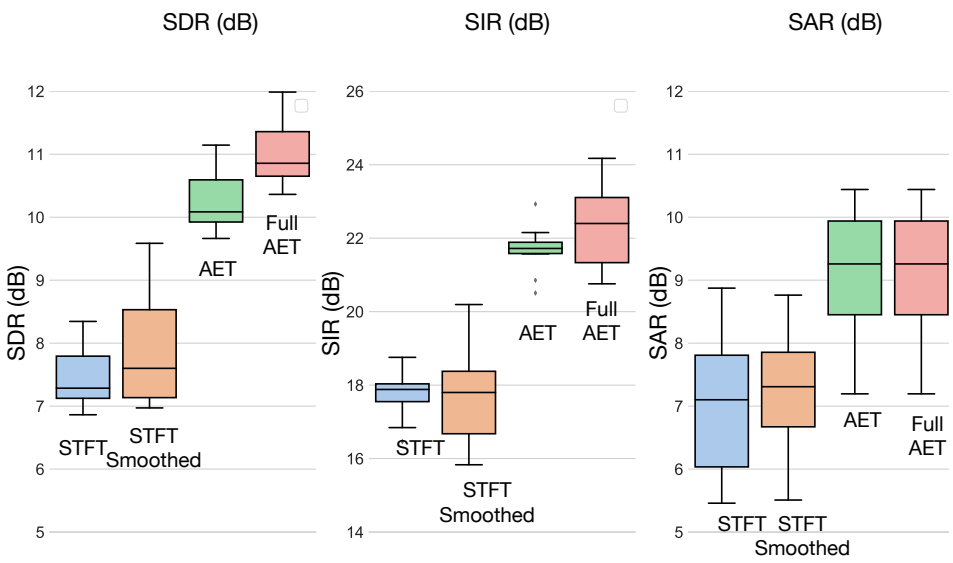

Figure 3.6: Comparison of the separation performance of the proposed models in terms of BSS_Eval metrics: SDR, SIR and SAR. The use of end-to-end neural networks to learn adaptive front-ends also boosts separation performance.

median (solid line in the middle) and the 25^{th} percentile and the 75^{th} percentile points as the box extremities.

We observe that the smoothed STFT model is comparable to the STFT model in terms of median separation performance. However, it results in a higher variance compared to the STFT front-end. The results also demonstrate a consistent improvement in the separation performance as we move towards a completely trainable network with adaptive front-ends, with the full-AET network significantly outperforming the rest.

3.3.2 Effect of stride

One of the advantages of the STFT is that STFTs allow us to perfectly reconstruct the audio signal from a highly sub-sampled time-frequency representation. Perfect reconstruction is achieved by using overlapping windows and specific values of hop [48]. In case of the adaptive front-ends the STFT hop has been replaced by the stride of the front-end convolutional layers.

We now consider the effect of strides on source separation performance. As before, we plot the variation in median SDR for different values of stride for STFT, STFT smoothed, AET and full-AET models. To have a fair comparison with the STFT model, we show the plots for hop/stride values that give perfect reconstruction for a window-size of 1024 samples. The advantage of using strided convolutional layers is that it reduces the number of computations required.

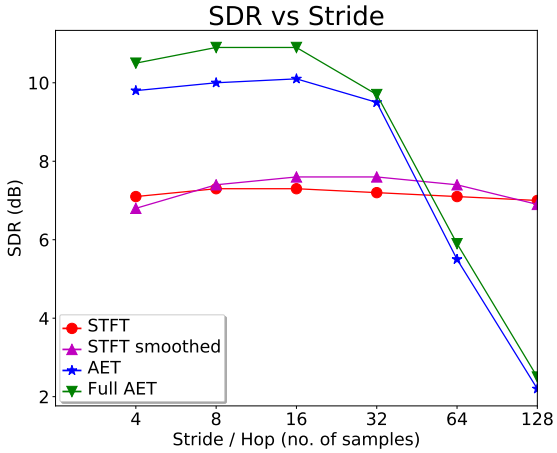

Figure 3.7: Plot of separation performance vs convolutional layer stride/hop. In case of the STFT versions, the separation performance remains consistent as it continues to have perfect reconstruction. In case of these end-to-end models, the separation performance rolls off significantly if we increase the stride beyond 32 samples. Thus, the adaptive latent representation cannot be as highly subsampled as the STFT.

Figure 3.7 shows the variation in SDR for varying values of stride. There are a few key differences in strided convolutions as compared to STFT front-ends. (i) STFTs allow us to perfectly reconstruct the audio signal in time only for specific values of the hop size [48]. In the case of end-to-end networks, strided convolutional layers can be trained to reconstruct the waveforms for all values of stride up to 32 samples. The window shape and bases are adaptively learned to suit the stride without restrictions. (ii) There is no significant improvement or deterioration in separation performance for strides up to 16 samples. Increasing the stride beyond 32 samples results in a sharp fall in separation performance. However, the STFT continues to perfectly

reconstruct the audio signal if the hop size is appropriately chosen.

3.4 Experiments on Cost Functions

The second part of our experimentation deals with comparing the various waveform-based cost functions for source separation. The contents of this section have been published in [46]. Since we deal with interpreting source separation metrics as a cost function, it is not reasonable to reuse the same metrics for evaluation. Consequently, we use subjective listening tests targeted at evaluating the separation performance, artifacts and intelligibility of the separation results to compare the different loss functions. We use the crowd-sourced audio quality evaluation (CAQE) toolkit [49] to set up the listening tests over Amazon Mechanical Turk (AMT). The details and results of our experiments follow.

3.4.1 Experimental setup

For our experiments, we use the end-to-end network shown in Figure 3.4(b). The separation was performed with a 1024 dimensional AET representation computed at a stride of 16 samples. A smoothing of 5 samples was applied by the smoothing convolutional layer. The separation network consisted of 2 dense layers each followed by a softplus non-linearity. This network was trained using different proposed cost functions and their combinations. We compare the cost functions by evaluating their performance on isolating the female speaker from a mixture comprising a male speaker and a female speaker, using the above end-to-end network.

To train the network, we randomly selected 15 male-female speaker pairs from the TIMIT database [33]. Ten (10) pairs were used for training and the remaining 5 pairs were used for testing. Each speaker has 10 recorded sentences in the database. For each pair, the recordings were mixed at 0 dB. Thus, the training data consisted of 100 mixtures. The trained networks were compared on their separation performance on the 50 test sentences. Clearly, the test speakers were not a part of the training data to ensure that the network learns to separate female speech from a mixture of male and female

speakers and does not memorize the speakers themselves.

In the subjective listening tests we compare the performance of end-to-end source separation under the following cost functions:

(i) Mean squared error
(ii) SDR
(iii) $0.75 \times SDR + 0.25 \times STOI$
(iv) $0.5 \times SDR + 0.5 \times STOI$
(v) $0.75 \times SIR + 0.25 \times SAR$
(vi) $0.5 \times SIR + 0.5 \times SAR$
(vii) $0.25 \times SIR + 0.75 \times SAR$.

These combinations were selected to understand the effects of individual cost functions on separation performance. We scale the value of each cost function to unity before starting the training procedure. This was done to control the weighting of terms in case of composite cost functions.

3.4.2 Evaluation

Using CAQE over a web environment like AMT has been shown to give consistent results on listening tests performed in controlled lab environments [49]. Thus, we use the same approach for our listening tests. The details are briefly described below.

Recruiting Listeners

For the listening tasks, we recruited listeners on Amazon Mechanical Turk that were over the age of 18 and had no previous history of hearing impairment. Each listener had to pass a brief hearing test that consisted of identifying the number of sinusoidal tones within two segments of audio. If the listener failed to identify the correct number of tones within the audio clip in two attempts, the listener's response was rejected. For the listening tests, we recruited a total of 180 participants over AMT.

Subjective Listening Tests

We assigned each of the accepted listeners to one of four evaluation tasks. Each task asked listeners to rate the quality of separation based on one of four perceptual metrics: preservation of source, suppression of interference, absence of additional artifacts, and speech intelligibility. The perceptual metrics such as preservation of source, suppression of interference, absence of additional artifacts, and speech intelligibility directly correspond to objective metrics such as SDR, SIR, SAR, and STOI respectively.

Accepted listeners were given the option to submit multiple evaluations for each of the different tasks. For each task, we trained listeners by giving each listener an audio sample of the isolated target source as well as a mixture of the source and interfering speech. We also provided 1-3 audio separation examples of poor quality and 1-3 audio examples of high quality according to the perceptual metric assigned to the listener. The audio files used to train the listener all had exceptionally high or low objective metrics (SDR, SIR, SAR, STOI) with respect to the pertaining task so that listeners could base their ratings in comparison to the best or worst separation examples.

After training, the listeners were then asked to rate eight unlabelled, randomly ordered, separation samples from 0 to 100 based on the metric assigned. The isolated target source was included in the listener evaluation as a baseline. The other seven audio samples correspond to separation examples output by a neural network trained with different cost functions enlisted in section 3.4.1.

3.4.3 Results and discussion

Figure 3.8 gives the results of the subjective listening tests performed through AMT for each of the four tasks. The results are shown in the form of a bar-plot that shows the median value (solid line in the middle) and the 25-percentile and 75-percentile points (box boundaries). The vertical axis gives the distribution of listener-scores over the range (0-100) obtained from the tests. The horizontal axis shows the different cost functions used for evaluation, as listed in section 3.4.1. This also helps us to understand the nature of

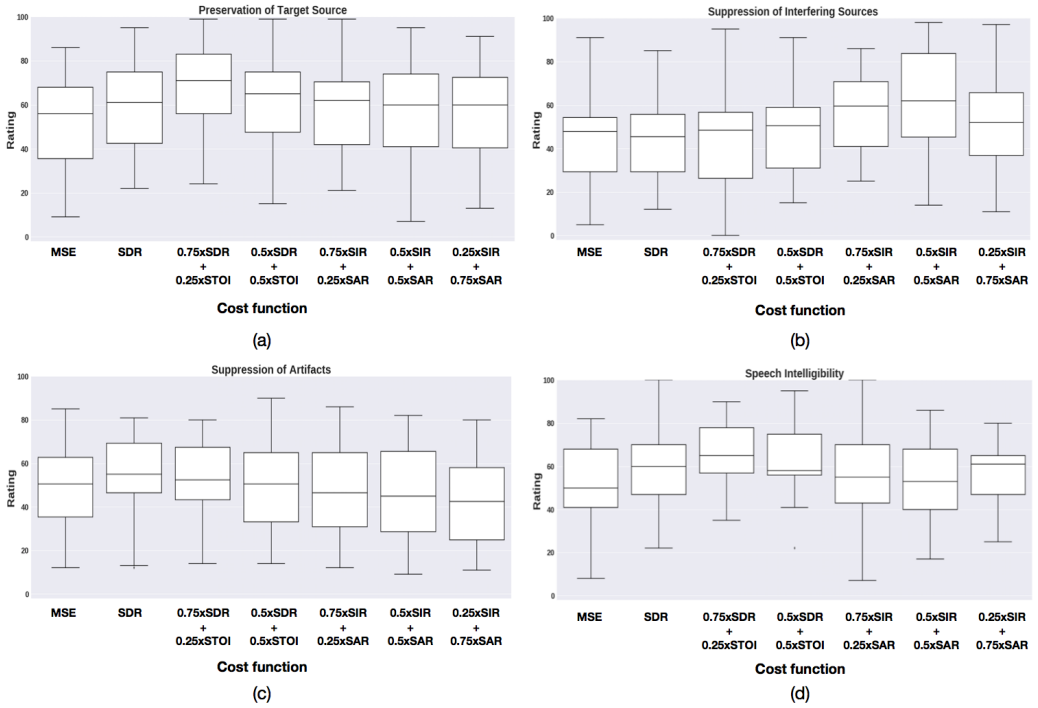

Figure 3.8: Listening test scores for different tasks. (a) Preservation of target source. (b) Suppression of interfering sources. (c) Suppression of artifacts. (d) Speech intelligibility over different cost functions. The distribution of scores is presented in the form of a box-plot where, the solid line in the middle gives the median value and the extremities of the box give the 25^{th} and 75^{th} percentile values.

the proposed cost functions. For example, Figure 3.8(b) (bars 5,6,7) shows that incorporating the SIR term into the cost function explicitly, helps the network to suppress the interfering sources better. Similarly, the addition of a STOI term into the cost function improves the results in terms of speech intelligibility as seen in Figure 3.8(d). It is also observed that adding STOI to the SDR cost function helps in preserving the target source better (Figure 3.8(a), bars 2,3 and 4). One possible reason for this could be that increasing the intelligibility of the separation results in a perceptual notion of preserving the target source better. The BSS_Eval cost functions appear to be comparable in terms of preserving the target source (Figure 3.8(a), bars 2,5,6,7) and slightly better than MSE. In terms of artifacts in the separated source, SDR outperforms all the cost functions, all of which seem to introduce comparable levels of artifacts into the separation results (Figure 3.8(c)). The use of SAR in the cost function does not seem to have favorable or adverse effects on the

perception of artifacts on the separation results.

3.5 Lessons for NAE Models

After our proposed AET models, several extended versions and neural networks have been proposed for discriminative end-to-end source separation. Some popular extensions include Tasnet [50], ConvTasnet [51], time-dilated convolutional networks (TDCNNs) [52] and two-step source separation [53, 54]. These extensions have led to state-of-the-art separation performance in the case of multi-talker and multi-source source separation. A common thread across these models is that the front-end transform is replaced by convolutional layer and the inverse transform is replaced by a transposed convolutional layer. In terms of the cost functions, SDR has become the cost function of choice when it comes to models that operate directly in the waveform domain, particularly for source separation. We can incorporate these ideas into our proposed NAE models to develop NAE models with end-to-end generative modeling capabilities.

CHAPTER 4

END-TO-END NON-NEGATIVE AUTOENCODERS

As described in Chapter 1, neural networks (NNs) are typically discriminatively trained and these networks require large amounts of training data and data-augmentation to be practically useful. In addition, in the event of a mismatch between the training and test conditions, these networks are unusable. Unlike these methods, generative approaches like non-negative matrix factorization (NMF) and non-negative autoencoders (NAEs) learn models from clean training examples. The availability of a trainable inference step allows us to fit these pre-trained models on different test examples.

Replacing NMF by NAEs allows us to learn and easily generalize NMF models. In addition, the availability of several NN toolboxes allows us to easily crunch and learn from larger amounts of training data. In the case of audio signals, we can combine NAEs to work with adaptive front-ends to learn end-to-end NAE models that directly operate on the waveform and further boost NAE performance. We explore this particular idea in this chapter and demonstrate that these models are comparable to discriminative NNs for source separation. Then, we also show that these models also retain the modularity of NMF models. Finally, we explore the possibility of learning these models directly from mixtures of sounds without clean training examples, similar to NMF models.

4.1 End-to-end Non-negative Autoencoder

As described in Section 2.2, we can generalize NMF models by interpreting them as a neural network. In the case of NMF, we can replace it by a two-layer NAE given by

$$1^{\text{st}} \text{ layer: } \text{(Encoder) } \mathbf{H} = g(\mathbf{W}^{\ddagger} \cdot \mathbf{X})$$
$$2^{\text{nd}} \text{ layer: } \text{(Decoder) } \hat{\mathbf{X}} = g(\mathbf{W} \cdot \mathbf{H})$$

In this equation, $\mathbf{X}$ represents the input spectrogram, the decoder weights $\mathbf{W}$ give the equivalent of NMF bases and the encoder output $\mathbf{H}$ gives the equivalent of the NMF activations. The weights of the encoder $\mathbf{W}^{\ddagger}$ represent a form of a pseudo-inverse matrix that can be applied to the input spectrogram to get the activations. The non-linearity $g(.) : \mathbf{R} \rightarrow \mathbf{R}_{\geq 0}$ operates element-wise and maps a real number to the space of positive-real numbers. This allows the network to learn a non-negative $\mathbf{H}$ and a non-negative reconstruction $\hat{\mathbf{X}}$ of the input $\mathbf{X}$. Unlike NMF, the decoder weights need not be strictly non-negative. But, under suitable sparsity constraints on the activations $\mathbf{H}$, they can be shown to be non-negative like NMF bases.

Although we have defined our NAE to have a single dense layer in the encoder and the decoder, we are not necessarily restricted by this formulation. We can take advantage of the modeling flexibility of neural networks and develop complex encoder and decoder architectures that adhere to the above format. These extensions are covered in Section 2.5. As before, in the generalized version, the weights of the decoder act as a representative model for the source. The output of the encoder indicates the corresponding activations.

4.1.1 End-to-end processing

To introduce end-to-end processing capabilities into our NAE, we replace the front-end transform step by a 1D-convolutional layer. To get a non-negative representation like the magnitude spectrogram, we use a softplus non-linearity for the layer. To transform back into the waveform from the latent representation, we use a transposed 1D-convolutional layer. The use of these adaptive front-ends has been explored in detail in the context of discriminative source separation in Section 3.1.3.

These modifications allow the network to accept a waveform and learn a

trainable latent representation that is optimal for representing a particular source. As we will show in Section 4.2, this also enables operating simultaneously on multiple customized latent representations corresponding to the different sources in the mixture, to extract the sources. Figure 4.1b shows the block diagram of our end-to-end NAE.

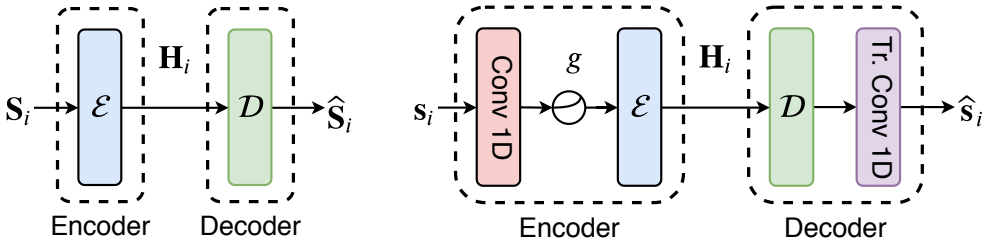

Figure 4.1: Block diagrams for: (a) a non-negative autoencoder (NAE), (b) an end-to-end NAE. Here, $\mathcal{E}$ and $\mathcal{D}$ represent the encoder and decoder of the NAE respectively. We append a 1D-convolutional layer as a front-end and back-end to enable the network operate to on waveforms directly. Thus, the end-to-end NAE encoder consists of the front-end layer and the NAE encoder $\mathcal{E}$. Similarly, end-to-end NAE decoder is made up of the NAE decoder $\mathcal{D}$ and the back-end layer. In the training step, we build and train an end-to-end NAE for every source we hope to encounter. The trained model is then used in the inference step for separating the sources.

4.2 Supervised Source Separation

Having developed the end-to-end NAE architecture, we now show how we can use it for end-to-end source separation. Like NMF, source separation using end-to-end NAEs is a two-step process:

Step 1: Learn suitable end-to-end NAE models for all the sources we expect to encounter in the mixture. We refer to this as the "training" step.
Step 2: Given an unseen mixture, fit the trained models to explain the contributions of the individual sources in the mixture. We refer to this as the "inference" step.

To extract the sources, we use the pre-trained end-to-end NAEs from the training step to construct an inference network. In this chapter, we consider

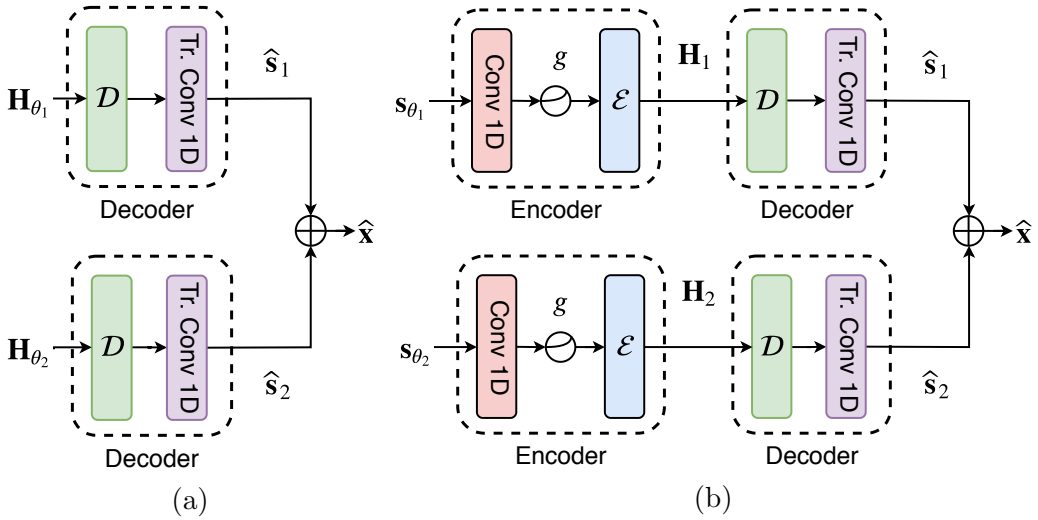

Figure 4.2: Block diagram of source separation using end-to-end NAEs during the inference step: estimating the non-negative model activations $\mathbf{H}_{\theta_\mathbf{i}}$ of the sources (left) and estimating the time-domain waveforms of the sources $\mathbf{s}_{\theta_\mathbf{i}}$ (right). The subscript θ indicates that the corresponding variable is a parameter of the inference network and we train to estimate its values for the given mixture.

two distinct inference frameworks. As shown in Section 2.2, the decoders of the pre-trained NAEs act as representative models. Extending the same idea, the decoders of our pre-trained end-to-end NAEs can be used to separate sources in the inference step. We emphasize here that when we refer to the "decoder" in case of end-to-end NAEs, it contains the NAE decoder $\mathcal{D}$ followed by the transposed-1D convolution layer. The block diagram of the inference network using the pre-trained end-to-end decoders is shown in Figure 4.2a. At the outputs of the decoders, we can directly access the waveforms of the individual sources. Consequently, the goal of the inference step is to extract the source waveforms $\mathbf{s_i}$ for $i \in 1, 2, \ldots K$, given the mixture waveform $\mathbf{x_m}$ and the pre-trained decoders. To get these decoder outputs, we need to estimate the non-negative activations $\mathbf{H}_{\theta_\mathbf{i}}$ for all the sources in the mixture. In other words, during the inference step, we optimize towards finding the right inputs to the inference network. The subscript θ serves to denote the trainable parameters of our inference network.

The above inference procedure ignores the encoder in constructing the inference network. Incorporating the pre-trained encoder could potentially

improve separation performance. Thus, we consider an alternative approach where the inference network uses the whole pre-trained end-to-end NAE. As before, we optimize to find the right inputs $\mathbf{s}_{\theta_i}$ to the inference network such that the outputs add up to explain the mixture. In this version of inference, we are optimizing on the space of waveforms as opposed to the space of activations. Figure 4.2b describes this inference approach as a block diagram.

To develop an intuition on the complexity of the inference optimization, we can compare the number of trainable parameters for the two inference approaches for a 1-second test example. For a sampling rate of 16 kHz, a 64 dimensional activation matrix for each source and a stride of 32 samples for the front-end 1D-convolutional layer, the number of trainable parameters can be given by $16,000 \cdot \frac{64}{32} = 32,000$ parameters per source. The second inference approach optimizes to estimate the waveforms as the parameters and trains for $16,000$ parameters per source. Thus, the inference optimization happens on a significantly smaller parameter space compared to training standard neural networks. In addition, applying the inference step for longer test examples can possibly be done in batches, reducing the inference time further.

4.2.1 Cost function

In the case of end-to-end models, using cost functions motivated by SDR have led to several promising results [46, 51]. We can use these waveform based cost functions to train end-to-end NAEs, both during training and inference. For a reference waveform $\mathbf{y}$ and a network output $\mathbf{x}$, we maximize a simplified version of SDR given by $\frac{|\langle \mathbf{x},\ \mathbf{y} \rangle|^2}{\langle \mathbf{x},\ \mathbf{x} \rangle}$. Here, $\langle \mathbf{x},\ \mathbf{y} \rangle$ represents the inner-product operation. Intuitively, we ask to maximize the sample correlation between $\mathbf{x}$ and $\mathbf{y}$ and while minimizing the energy of the solution. The detailed derivations of SDR and other BSS_Eval metrics as waveform based cost functions are described in detail in Section 3.2.1.

4.2.2 Advantages of end-to-end NAEs

We now consider the advantages that end-to-end NAEs have to offer when used for source separation. As stated previously, we can exploit the modeling flexibility afforded by neural networks to construct complex architectures that operate on the waveforms directly. In our experiments, we show that end-to-end NAE models are comparable to discriminative models in terms of separation performance. Although we require an optimization process during inference, we gain a significant advantage. End-to-end NAEs are developed as generalizations of NMF and we continue to retain its modular nature. Consequently, once we learn a model for a source, we can use the model on any mixture that contains the source and extract it, irrespective of the interferences in the mixture. Also, we can directly use the pre-trained models without the need for any data-augmentation, on a variety of test examples with varying characteristics. We also evaluate this capability of end-to-end NAEs in our experiments. In fact, the availability of an inference step, where we try to optimally fit the pre-trained models on an unseen mixture, allows us to use the same models for different test mixtures. In other words, the modularity of end-to-end NAEs is a consequence of having a trainable inference step. Finally, extending the model to operate on new types of sounds becomes extremely easy. All we need to do is to train an end-to-end NAE for the new source. We can then append the pre-trained model to the inference network to separate that source from given mixtures.

4.3 Experiments on End-to-end NAEs

We now present some experiments and their results to evaluate the performance of end-to-end NAEs for supervised single-channel source separation. Primarily, we focus on two experiments: (i) The first experiment is aimed at comparing end-to-end NAEs to end-to-end discriminative source separation models, in terms of their separation performance. (ii) The second experiment is directed towards evaluating their modular nature. We first describe our experimental setup.

4.3.1 Dataset

For our experiments, we use the Device and Produced Speech (DAPS) [55] dataset. We use only the clean speech examples from the dataset for our experiments. Of the 10 male and 10 female speakers, we use 8 male and 8 female speakers to construct the training set. We use the first 3 scripts to construct the training examples out of the 5 scripts available per speaker. This gives about 2 hours of training data for each class. To evaluate separation performance, we generate two test sets as follows: *Test-set-1* is generated from the unused recordings of the speakers that are a part of the training set. The test examples in *Test-set-2* come from the speakers not included in the training set. We down-sample the recordings to 16 kHz and randomly draw 2-sec snippets for training and testing.

4.3.2 Network

For the network configurations, we use a front-end 1D convolutional layer consisting of 256 filters of width 64 samples and a stride of 32 samples. The NAE encoder ($\mathcal{E}$) is formed by a cascade of two 1D-convolutional layers. The two layers have 128 and 64 filters respectively, a filter width of 5 taps and a stride value of 1. Thus, the activation matrix for each source has a dimensionality of 64. The decoder architecture is constructed to invert the activation matrix. Thus, the NAE decoder ($\mathcal{D}$) is constructed using two 1D transposed-convolutional layers of 128 and 256 filters, a filter width of 5 taps and a stride value of 1. Each convolutional layer is followed by a softplus non-linearity and a batch-norm operation. To transform the output of the NAE decoder $\mathcal{D}$ back into the waveform domain, we use a 1D transposed-convolutional layer having the same parameters as the front-end. In our experiments, we compare the performance of NAEs to discriminatively trained source separation models. For the discriminative model, we use the same architecture as an end-to-end NAE.

4.3.3 End-to-end NAEs vs. discriminative models

The first experiment is designed to compare the separation performance of end-to-end NAEs with discriminative source separation models. We train

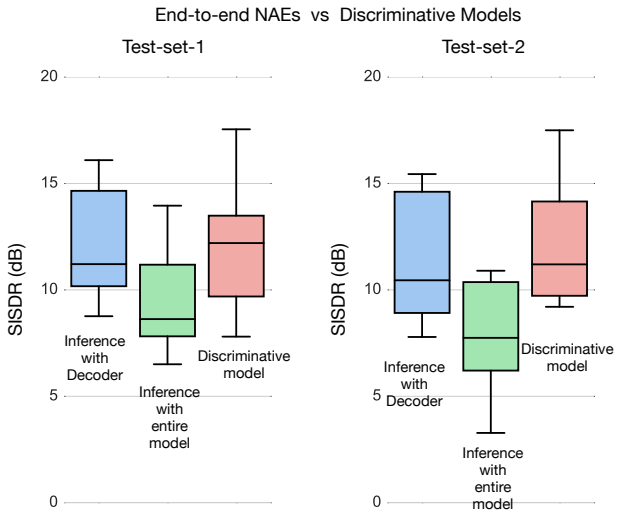

Figure 4.3: SISDR values for 0 dB test mixtures. We see that the separation performance of end-to-end NAEs is comparable with discriminatively trained models.

two end-to-end NAEs, one for the set of male speakers and the other for the set of female speakers. Using these pre-trained models, we apply the inference step on the test mixtures to get the separation results. The discriminative model used for comparison is trained as a denoising autoencoder that separates the female speaker from a 0 dB mixture consisting of a male and a female speaker. We generate these 0 dB training examples by drawing random snippets from the training sets and mixing them. For evaluation, we generate 30 test mixtures mixed at 0 dB for each test set. To reiterate, Test-set-1 is formed from speakers that are included in the training set. Test-set-2 is generated from speakers that are not included in the training set. The results are shown in Figure 4.3 as a box-plot of scale-invariant SDR (SISDR) [56, 57] values. The solid line in the center of the box indicates the median value and the box-boundaries show the inter-quartile range (25^{th} and 75^{th} percentile points).

As shown in Figure 4.3, we compare the separation performance of three models: 1. End-to-end NAE with inference using decoder only (left), 2. End-to-end NAE with inference using entire model (middle), and 3. End-to-end discriminative source separation (right) over both the test-sets. We see that the separation performance of end-to-end NAEs with decoder based

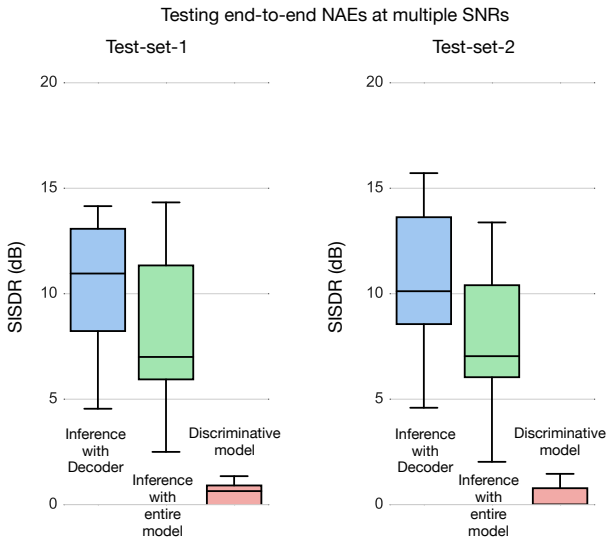

Figure 4.4: SISDR values for test mixtures with SNR varying between −3 dB and 3 dB. We see that the inference step allows us to fit the models and separate sources even for conditions unseen during the training step. The discriminative model cannot deal with this mismatch between the training and test sets.

inference is comparable with end-to-end discriminative models over both the test sets. Using the entire model for inference results in a significant drop in separation performance. This can be attributed to the fact that inference on waveforms does not produce silences as effectively as inference on the activation matrices. Comparing with the performance on Test-set-2, we observe a dip in median SISDR on end-to-end NAE models. Also, the variance in SISDR values increases slightly. However, the separation performance looks about the same overall.

4.3.4 Testing end-to-end NAEs at multiple SNRs

The second experiment aims to evaluate the modular nature of our end-to-end NAEs. We evaluate this by comparing the performance on mixtures with varying signal-to-noise ratio (SNR) levels. In the case of discriminative models, we cannot expect any reasonable separation results when there is a mismatch between the training and test mixtures encountered by the models. But the model-fitting performed during the inference step for end-to-end NAEs allows the use of the pre-trained models even in mismatched cases.

We generate 30 test mixtures with SNR levels varying from -3 dB to 3 dB, taking the male speech as the reference. As before, we construct two test sets of 30 mixtures each: Test-set-1 from speakers included in the training set and Test-set-2 from speakers that are not a part of the training examples. We compare across three models: 1. End-to-end NAE with decoder inference (left) 2. End-to-end NAE with fill model inference (middle) and 3. End-to-end discriminative source separation (right). We use the same models trained for the previous experiment in all the cases. The results are shown in Figure 4.4.

We see that the end-to-end NAEs achieve a good separation even at these varying SNR levels, something not possible in the case of discriminative separation models as shown by their extremely poor performance. However, we also observe an increase in the range and variance of SISDR values for this experiment. As before, we see a drop in the median SISDR values and an increase in the variance, on Test-set-2 compared to Test-set-1. Despite this, the separation performance falls in the same range overall, indicating the efficacy of the trained model.

4.4 Learning from Mixtures

So far, we have focused on learning our end-to-end NAE models on clean sounds. However, in the real world, there can be several use cases and applications where we may not have access to such clean training examples. Such applications are also encountered in trying to train neural networks directly in ambient settings without any human intervention.

In the absence of such clean training examples for the sources, one of the main issues is to define what we mean by a "source" and pass that information on to our separation network. In our case, we can relax the problem by interpreting it as a weakly-supervised source separation problem. Essentially, instead of the clean sounds, we assume that we have access to information about when the sources are active in a given mixture. Locating the start and stop times of the sources in a mixture can be easily done by a human listener "hearing out" the mixture and marking the approximate start and stop times.

Thus, the relaxation we propose is a significant one because we can now completely rely on the mixtures alone to learn our models. The NMF and NAE activations contain information about the start and stop times of the sources. Thus, we can use the start and stop times as priors for the model activations.

The problem of weakly supervised source separation has been addressed for discriminatively trained networks [58]. We address a parallel problem in the context of generative source separation. In order to develop the approach to train our end-to-end NAEs in a weakly supervised setting, we consider the training of weakly supervised NMF models first.

Spectral and temporal priors have been used extensively to learn suitable NMF models in different forms. In [59, 60], the authors propose a method where the user "hums" the sound and the resulting spectrogram is used as a spectro-temporal prior to learn NMF models. In [61] and [62], Ozerov et al. use temporal priors to identify the sources and learn NMF models from mixtures. Spectro-temporal annotations have been used as priors in [6]. In contrast to such prior based methods, Bryan and Mysore propose a posterior regularized NMF model [63]. Posterior regularization generates an approximate time-frequency mask for each source and multiplies the source spectrograms by the masks in each step of the NMF updates. We use a similar strategy to train our weakly supervised end-to-end NAEs.

Figure 4.5 shows the block diagram of our weakly supervised end-to-end NAEs. To train this network, we need to perform additional modifications. The mixture signal is given as an input to both the autoencoders, and the autoencoders are expected to produce the source waveforms at their outputs. However, we do not have access to these waveforms. In our case, we have access to timing information about the sources in different mixtures. We use these to generate a temporal mask for each source. The temporal mask basically remains high for the duration when the source is active. To apply this mask and push our end-to-end NAEs towards the desired solution, we use a strategy similar to the posterior-regularized NMF approach [63]. In the forward pass of the neural network, the output of each layer in the encoder is multiplied by the corresponding temporal mask before passing it onto the next layer. Thus, the information about the sources is conveyed through

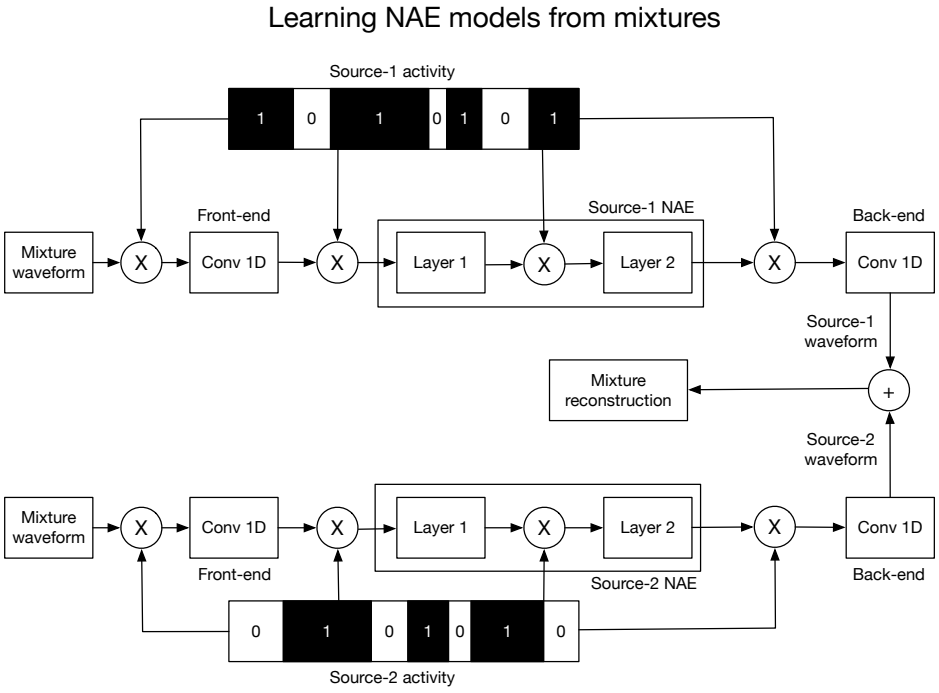

Figure 4.5: Learning end-to-end NAEs from mixtures.

their activity in the mixtures.

In addition, we also need to modify our cost function to suit the task at hand. The first requirement is that the autoencoder outputs $\mathbf{s_1}$ and $\mathbf{s_2}$ should add up to the mixture waveform $\mathbf{x}$. To enforce this condition, we use the SDR cost as the measure of discrepancy and minimize $\text{SDR}(\mathbf{x},\ \mathbf{s_1} + \mathbf{s_2})$. Using the source activities as masks ensures that the succeeding layers receive the right inputs from the previous layer. However, since we do not have access to these masks at inference time, each layer is also expected to produce an output that incorporates the masking information. This is ensured by using additional terms to the cost function. Thus, we also include the terms $||\mathbf{Y}_i - \mathbf{m}_k \odot \mathbf{Y}_i||^2$ into the cost function. Here, $\mathbf{Y}_i$ denotes the output of the i^{th} layer, $\mathbf{m}_k$ denotes the temporal mask of the k^{th} source broadcasted to have the same size as $\mathbf{Y}_i$ and $\odot$ represents the element-wise multiplication operator. Essentially, we ask that the output of each layer be approximately equal to the corresponding masked output of the same layer.

4.4.1 Experiments and evaluation

To evaluate our weakly supervised end-to-end NAEs we perform speech denoising on the noisy ambient mixtures from the DAPS database [55]. In particular, we use noisy mixtures from 3 room recordings: confroom1, office1, and livingroom1. For each of these environments, we have a total of 10 male and 10 female speakers recording 5 scripts each. For our experiments, we try to learn a separate model for male speech, directly from the mixtures. To train these models, we set aside a training set consisting of 8 male speakers. For each of these speakers in the training set, we use the first 3 scripts for training. This amounts to about 2 hours of training mixtures to learn our models. To generate the temporal masks, we use the corresponding synchronized clean versions of the mixtures. We set a small threshold for the noise floor in these clean examples and say that the speaker is active if the energy of the current sample is greater than the selected noise threshold. As described in the initial parts of Section 4.4, we can also obtain this information by listening to the mixtures and manually marking the start and end times of the speech signal. To train our models, we generate 2 second mixtures from the training set. To test these models, we use the test examples to generate 32 mixtures of duration 2 seconds each.

Table 4.1: Denoising performance of end-to-end NAEs (denoted as NAE) trained only from mixtures to spectral subtraction (denoted as SS). We see that end-to-end NAEs consistently outperform spectral subtraction across all the denoising evaluation metrics.

Environment	C_{sig}		C_{bak}		C_{ovl}		PESQ	
	SS	NAE	SS	NAE	SS	NAE	SS	NAE
Confroom1	1.51	2.35	1.42	1.85	1.31	1.90	1.27	1.74
Office1	1.77	2.29	1.44	1.78	1.33	1.69	1.23	1.29
Livingroom1	1.82	2.31	1.52	1.87	1.41	1.74	1.32	1.32

We compare these results to those obtained from spectral subtraction (SS) for a baseline comparison. The two methods are compared in terms of signal distortion (Csig), noise distortion (Cbak), overall quality (Covl) [64] and perceptual evaluation of speech quality (PESQ) [65]. Of these metrics, Csig, Cbak, Covl, and PESQ are closely related to human perception and are intended to imitate a mean opinion score test. Csig indicates a measure of

the distortions introduced into the output, Cbak estimates the interferences remaining due to the noise, Covl summarizes the overall quality of the denoised results, and PESQ finally estimates the speech quality. For all of these metrics, higher is better.

The results of this comparison are shown in Table 4.1. The table gives the mean value of these metrics computed across the 32 mixtures. We see that the learning to denoise using NAEs results in improved performance over spectral subtraction in all the selected environments. The use of end-to-end NAEs allows us to learn models by using large amounts of training data and this helps our models to significantly outperform SS.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Overview

In this dissertation, we developed a new neural network (NN) based framework as an alternative to non-negative audio modeling. To this end, we presented the use of end-to-end non-negative autoencoders (NAEs) as a suitable update to classical non-negative matrix factorization (NMF) models. These end-to-end NAEs combine the advantages of classical methods like NMF while retaining the generalizability and performance of NNs.

To develop end-to-end NAEs, in Chapter 2 we explored the idea of interpreting NMF as a shallow NAE and showed that we can learn equivalent representative models for audio signals through both the methods. The equivalence between these models also extends to single-channel source separation where the models are shown to be comparable in terms of separation performance. The advantages of using NAEs became very clear when using deeper and more complicated multi-layer architectures. These extended versions significantly outperformed shallow NAEs and NMF in source separation experiments.

In the Chapter 3, we showed how we can generalize the idea of short-time Fourier transforms (STFT) to develop front-end and synthesis transform layers. These replacements allowed us to learn optimal and customized time-frequency representations for the audio signal and develop end-to-end neural networks that operate directly on the waveforms. Finally, we showed how these architectures allow us to utilize cost functions that directly map to perceived performance. Our experiments revealed that these models outperform existing architectures significantly in terms of separation perfor-

mance. Through the listening tests, we also compared the separation results of performance-based cost functions for end-to-end separation and drew the following conclusions: (i) A SDR-STOI combination gives the best separation performance in terms of preserving the target source. (ii) Maximizing the SDR also produces the least artifacts among the cost functions tested. (iii) A combination of SIR and SAR gives the best suppression of interfering sources.

In Chapter 4, we combined NAEs and adaptive front-ends to develop end-to-end NAEs and investigated their use for single-channel source separation. We showed that with such updates to their modeling capabilities, generatively trained end-to-end NAEs are comparable to discriminatively trained NNs in terms of separation performance. Although these models have a more computationally intensive inference step, they allow for additional extensions that discriminative models cannot easily facilitate. In addition to having an extensible architecture, the modular nature of end-to-end NAEs allowed us to separate sources from mixtures over a range of signal-to-noise ratio values using the same pre-trained generative models for the sources. We also considered the problem of learning end-to-end NAE models directly from mixtures of sounds. We interpreted the problem as a weakly supervised source separation problem wherein, in addition to the mixtures, we had access to information about when the sources are active in the given mixtures. We used this timing information to generate masks for the sources and incorporate these masks in the forward pass of the network. The results of weakly supervised denoising experiments on real ambient noisy mixtures indicated that with these improvements, we can outperform spectral subtraction.

5.2 Future Directions

In terms of future directions, there appear to be several unexplored territories going forward. In this dissertation, we have particularly focused on single-channel source separation to evaluate the performance of end-to-end NAEs. Given the wide applicability of NMF, there are several allied applications that can potentially benefit significantly by the use of end-to-end NAEs. One of the popular applications of NMF is its use in semi-supervised source

separation where we only have clean training examples for a subset of the sources in the mixture. NMF models have also been used for speech denoising, bandwidth extension and audio imputation (restoring high-quality audio from corrupted or clipped recordings). We can replace NMF by end-to-end NAEs for these applications and potentially achieve better performance. We can also put the ability of end-to-end NAEs to learn from mixtures to good use to perform polyphonic music transcription.

REFERENCES

[1] Y. Xu, J. Du, L.-R. Dai, and C.-H. Lee, "An experimental study on speech enhancement based on deep neural networks," *IEEE Signal Processing Letters*, vol. 21, no. 1, pp. 65–68, 2013.

[2] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen et al., "Deep speech 2: End-to-end speech recognition in English and Mandarin," in *International Conference on Machine Learning*, 2016, pp. 173–182.

[3] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, no. 3, pp. 279–283, 2017.

[4] H. Ze, A. Senior, and M. Schuster, "Statistical parametric speech synthesis using deep neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 7962–7966.

[5] X. Wang and Y. Wang, "Improving content-based and hybrid music recommendation using deep learning," in *Proceedings of the 22nd ACM International Conference on Multimedia*, 2014, pp. 627–636.

[6] N. J. Bryan, G. J. Mysore, and G. Wang, "ISSE: An interactive source separation editor," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2014, pp. 257–266.

[7] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, 2011.

[8] C. Févotte, R. Gribonval, and E. Vincent, "BSS_EVAL toolbox user guide-Revision 2.0," HAL-Inria, France, Tech. Rep. 1706, April 2005.

[9] P. Smaragdis and J. C. Brown, "Non-negative matrix factorization for polyphonic music transcription," in *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No. 03TH8684)*. IEEE, 2003, pp. 177–180.

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

[11] P. Smaragdis, C. Fevotte, G. J. Mysore, N. Mohammadiha, and M. Hoffman, "Static and dynamic source separation using nonnegative factorizations: A unified view," *IEEE Signal Processing Magazine*, vol. 31, no. 3, pp. 66–75, 2014.

[12] T. Virtanen, J. F. Gemmeke, B. Raj, and P. Smaragdis, "Compositional models for audio processing: Uncovering the structure of sound mixtures," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 125–144, 2015.

[13] 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.

[14] A. Ozerov and C. Févotte, "Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 550–563, 2009.

[15] K. W. Wilson, B. Raj, P. Smaragdis, and A. Divakaran, "Speech denoising using nonnegative matrix factorization with priors," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2008, pp. 4029–4032.

[16] N. Mohammadiha and S. Doclo, "Speech dereverberation using nonnegative convolutive transfer function and spectro-temporal modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 2, pp. 276–289, 2015.

[17] Ç. Bilen, A. Ozerov, and P. Pérez, "Audio declipping via nonnegative matrix factorization," in *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2015, pp. 1–5.

[18] A. Ozerov, A. Liutkus, R. Badeau, and G. Richard, "Informed source separation: Source coding meets source separation," in *2011 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2011, pp. 257–260.

[19] M. Turkan and C. Guillemot, "Image prediction based on neighbor-embedding methods," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1885–1898, 2011.

[20] O. Cirakman, B. Gunsel, N. S. Sengor, and O. Gursoy, "Key-frame based video fingerprinting by NMF," in *2010 IEEE International Conference on Image Processing*. IEEE, 2010, pp. 2373–2376.

[21] B. Krausz and C. Bauckhage, "Action recognition in videos using non-negative tensor factorization," in *2010 20th International Conference on Pattern Recognition*. IEEE, 2010, pp. 1763–1766.

[22] M. Cooper and J. Foote, "Summarizing video using non-negative similarity matrix factorization," in *2002 IEEE Workshop on Multimedia Signal Processing*. IEEE, 2002, pp. 25–28.

[23] Z. Chen, A. Cichocki, and T. M. Rutkowski, "Constrained non-negative matrix factorization method for EEG analysis in early detection of Alzheimer disease," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 5. IEEE, 2006.

[24] M. W. Berry and M. Browne, "Email surveillance using non-negative matrix factorization," *Computational & Mathematical Organization Theory*, vol. 11, no. 3, pp. 249–264, 2005.

[25] J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, "Metagenes and molecular pattern discovery using matrix factorization," *Proceedings of the National Academy of Sciences*, vol. 101, no. 12, pp. 4164–4169, 2004.

[26] P. Smaragdis and S. Venkataramani, "A neural network alternative to non-negative audio models," in *Proc. ICASSP*, 2017, pp. 86–90.

[27] C. Févotte, N. Bertin, and J.-L. Durrieu, "Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis," *Neural Computation*, vol. 21, no. 3, pp. 793–830, 2009.

[28] P. Smaragdis, B. Raj, and M. Shashanka, "Supervised and semi-supervised separation of sounds from single-channel mixtures," in *International Conference on Independent Component Analysis and Signal Separation*. Springer, 2007, pp. 414–421.

[29] P. Smaragdis, "Convolutive speech bases and their application to supervised speech separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 15, no. 1, pp. 1–12, 2006.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[31] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[32] S. Squires, A. P. Bennett, and M. Niranjan, "A variational autoencoder for probabilistic non-negative matrix factorisation," *arXiv preprint arXiv:1906.05912*, 2019.

[33] J. Garfolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallet, N. Dahlgren, and V. Zue, "TIMIT acoustic-phonetic continuous speech corpus," University of Pennsylvania Linguistic Data Consortium, LDC93S1, 1993. https://catalog.ldc.upenn.edu/LDC93S1.

[34] T. N. Sainath, R. J. Weiss, A. Senior, K. W. Wilson, and O. Vinyals, "Learning the speech front-end with raw waveform CLDNNs," in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[35] S. Dieleman and B. Schrauwen, "End-to-end learning for music audio," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 6964–6968.

[36] D. Fagot, H. Wendt, and C. Févotte, "Nonnegative matrix factorization with transform learning," in *Proc. ICASSP*, 2018, pp. 2431–2435.

[37] A. Sarroff, "Complex neural networks for audio," Ph.D. dissertation, Dartmouth College, 2018.

[38] Z.-Q. Wang, J. Le Roux, and J. R. Hershey, "Alternative objective functions for deep clustering," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[39] A. A. Nugraha, A. Liutkus, and E. Vincent, "Multichannel audio source separation with deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 9, pp. 1652–1664, Sept 2016.

[40] D. Stoller, S. Ewert, and S. Dixon, "Wave-u-net: A multi-scale neural network for end-to-end audio source separation," *arXiv preprint arXiv:1806.03185*, 2018.

[41] S. Venkataramani, J. Casebeer, and P. Smaragdis, "End-to-end source separation with adaptive front-ends," in *Proc. Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 684–688.

[42] Z.-Q. Wang, J. L. Roux, D. Wang, and J. R. Hershey, "End-to-end speech separation with unfolded iterative phase reconstruction," *arXiv preprint arXiv:1804.10204*, 2018.

[43] D. Rethage, J. Pons, and X. Serra, "A wavenet for speech denoising," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018.

[44] S. Pascual, A. Bonafonte, and J. Serra, "SEGAN: Speech enhancement generative adversarial network," *arXiv preprint arXiv:1703.09452*, 2017.

[45] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "A short-time objective intelligibility measure for time-frequency weighted noisy speech," in *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4214–4217.

[46] S. Venkataramani, R. Higa, and P. Smaragdis, "Performance based cost functions for end-to-end speech separation," in *Proc. APSIPA ASC*, 2018, pp. 350–355.

[47] J. Garfolo, D. Graff, D. Paul, and D. Pallet, "CSR-I (WSJ0) Complete," speech corpus, University of Pennsylvania Linguistic Data Consortium, LDC93S6A, 1993. https://catalog.ldc.upenn.edu/LDC93s6a.

[48] J. B. Allen and L. R. Rabiner, "A unified approach to short-time Fourier analysis and synthesis," *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1558–1564, 1977.

[49] M. Cartwright, B. Pardo, G. J. Mysore, and M. Hoffman, "Fast and easy crowdsourced perceptual audio evaluation," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2016, pp. 619–623.

[50] Y. Luo and N. Mesgarani, "TasNet: Time-domain audio separation network for real-time single-channel speech separation," in *Proc. ICASSP*, 2018.

[51] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, pp. 1256–1266, 2019.

[52] I. Kavalerov, S. Wisdom, H. Erdogan, B. Patton, K. Wilson, J. Le Roux, and J. R. Hershey, "Universal sound separation," in *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE, 2019, pp. 175–179.

[53] E. Tzinis, S. Venkataramani, Z. Wang, C. Subakan, and P. Smaragdis, "Two-step sound source separation: Training on learned latent targets," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020.

[54] E. Tzinis, S. Wisdom, J. R. Hershey, A. Jansen, and D. P. Ellis, "Improving universal sound separation using sound classification," in *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2020, pp. 96–100.

[55] G. J. Mysore, "Can we automatically transform speech recorded on common consumer devices in real-world environments into professional production quality speech?—A dataset, insights, and challenges," *IEEE Signal Processing Letters*, vol. 22, no. 8, pp. 1006–1010, 2014.

[56] J. Le Roux, S. Wisdom, H. Erdogan, and J. R. Hershey, "SDR–half-baked or well done?" in *Proc. ICASSP*, 2019, pp. 626–630.

[57] E. Vincent, R. Gribonval, and C. Févotte, "Performance measurement in blind audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 4, pp. 1462–1469, 2006.

[58] F. Pishdadian, G. Wichern, and J. L. Roux, "Learning to separate sounds from weakly labeled scenes," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 91–95.

[59] P. Smaragdis, "User guided audio selection from complex sound mixtures," in *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, 2009, pp. 89–92.

[60] P. Smaragdis and G. J. Mysore, "Separation by "humming": User-guided sound extraction from monophonic mixtures," in *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE, 2009, pp. 69–72.

[61] A. Ozerov, E. Vincent, and F. Bimbot, "A general flexible framework for the handling of prior information in audio source separation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 4, pp. 1118–1133, 2011.

[62] A. Ozerov, C. Févotte, R. Blouet, and J.-L. Durrieu, "Multichannel nonnegative tensor factorization with structured constraints for user-guided audio source separation," in *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2011, pp. 257–260.

[63] N. Bryan and G. Mysore, "An efficient posterior regularized latent variable model for interactive sound source separation," in *International Conference on Machine Learning*, 2013, pp. 208–216.

[64] Y. Hu and P. C. Loizou, "Evaluation of objective quality measures for speech enhancement," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 1, pp. 229–238, 2007.

[65] A. W. Rix, J. G. Beerends, M. P. Hollier, and A. P. Hekstra, "Perceptual evaluation of speech quality (PESQ)– A new method for speech quality assessment of telephone networks and codecs," in *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 01CH37221)*, vol. 2. IEEE, 2001, pp. 749–752.