Thesis for the Degree of
Doctorof Philosophy

Deep Learning-based Latent Source Analysis for Source-aware Audio Manipulation

by

Woosung Choi

Department of Computer Science and Engineering

Graduate School

Korea University

August 2021

鄭舜榮 教授指導

博 士 學 位 論 文

Deep Learning-based Latent Source Analysis for Source-aware Audio Manipulation

이 論文을 컴퓨터學 博士學位 論文으로
提出함

2021年 06月 21日

高 麗 大 學 校 大 學 院

컴 퓨 터 學 科

崔 宇 成 (印)

崔宇成의 컴퓨터學 博士學位論文 審査를 完了함

2021年 06月 21日

委員長	정 순 영	(印)
委　員	유 헌 창	(印)
委　員	김 현 철	(印)
委　員	서 태 원	(印)
委　員	정 재 화	(印)

Abstract

This dissertation presents deep learning-based latent source analysis for source-aware audio manipulation. It mainly focuses on three audio tasks as follows: (1) dedicated source separation, (2) conditioned source separation, and (3) Audio Manipulation on Specified Sources (AMSS). The first and second tasks are variants of audio source separation, which aims to extract sources of interest from the given mixture. AMSS, which aims to perform audio transformations to user-specified sources of a given audio track according to a given description, is the original task this dissertation proposes. A novel concept called Latent Sources Analysis is introduced for conditioned source separation and AMSS. Models are designed on top of it, showing outstanding performance in both tasks.

This dissertation first introduces a simple but effective frequency transformation method called Time-Distributed Fully connected network (TDF) to improve the source separation performance of dedicated models. The experimental results indicate that injecting TDFs into a traditional U-Net structure can significantly improve the source separation quality.

TDF is extended to Latent Source-attentive Frequency Transformation (LaSAFT) for Conditioned Source Separation by employing latent source analysis. A latent source deals with a more detailed acoustic feature aspect than a symbolic-level source. The experimental results show that a Conditioned-U-Net equipping LaSAFTs outperforms existing conditioned source separation models by analyzing latent sources' frequency patterns.

Finally, this dissertation formulates a novel audio task called AMSS and proposes a textual query language called Audio Manipulation Language (AML) based on Context-Free Grammar. Furthermore, it proposes a neural network called AMSS-Net based on latent source analysis for AMSS. It extracts latent sources and selectively manipulates them while preserving irrelevant sources. Also, an evaluation benchmark for AMSS is proposed to measure the performance of models for AMSS. Experimental results show that AMSS-Net outperforms baselines on several AMSS tasks via objective metrics and empirical verification.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The technological advances of recent decades enable people to create audio or video content with mobile devices such as smartphones. Besides, many social media applications have attracted numerous users. Now they create, edit, and share their multimedia contents with other users. In general, however, it is difficult for non-experts to edit multimedia content such as image, audio, and video. To edit these types of content, users should know how to use editing tools with many buttons, functions, and parameters. For example, removing undesired objects is extremely difficult because such tasks usually require advanced skills to fill eliminated areas (e.g., pixels, frequency bins, or frames) with plausible contents.

Fortunately, many automatic editing tools have been proposed in the computer vision field to fulfill the need for an easy-to-use interface for image editing. Especially, recently proposed data-driven approaches such as image inpainting [20, 64], style transfer [67], and text-guided image manipulation [19, 23] provide next generation image editing interfaces. Users can get rid of unwanted objects in their image, for example, using deep learning-based inpainting [20, 64] methods. Image inpainting was initially proposed to reconstruct missing areas in a damaged image. However, it is also possible to remove objects by intentionally corrupting the input image and inpainting the corrupted with a trained inpainting model. If a user masks objects such as the tower in Figure 1.1 (a), and reconstruct it with model[1] proposed in [20], then she or he can obtain the desired output as shown in Figure 1.1 (c).

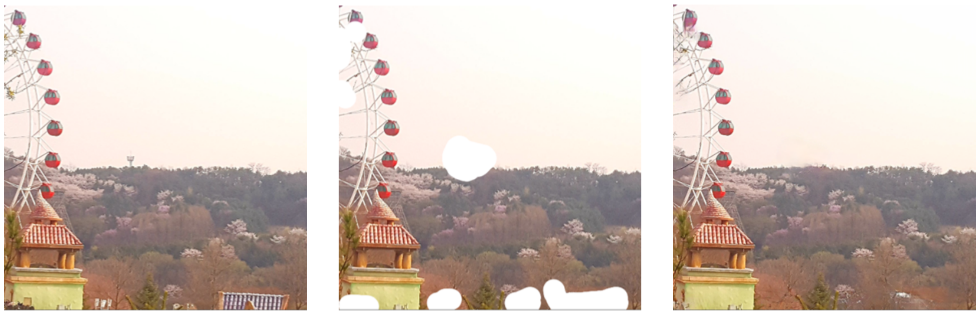

(a) original (b) original + mask (c) inpainted by model

Figure 1.1: Object Removal with an Image Inpainting Model [20]

Recent methods such as ManiGAN and Describe What to Change (DWC) [19, 23] provide more powerful tools. With DWC, users can interactively manipulate their images by simply typing text queries, as shown in Figure 1.2. These methods are helpful for users, especially those who lack prior knowledge of image editing. Users can edit images without a laborious search for appropriate buttons, functions, and appropriate parameter configurations.

Various researchers in the signal processing field also have investigated data-driven approaches to automate audio editing. However, proposed methods are limited in terms of effectiveness compared to those of the computer vision field. Developing an audio editing model is often considered more challenging due to the *transparency*, a unique characteristic of audio signals, as discussed in [62].

An object is said to be *transparent* if multiple sources can have energy in it. Figure 1.3 compares the characteristics of an image and an audio track in terms of transparency. As illustrated in Figure 1.3 (a), a single pixel in the given image usually corresponds to only a single object. A marked pixel belongs to the bird's head in Figure 1.3 (a). The other pixel belongs to a branch of a tree in the figure. Since a pixel usually carries the energy of a specific object, an image pixel is considered *opaque*.

[1]interactive demo site - https://www.nvidia.com/research/inpainting/

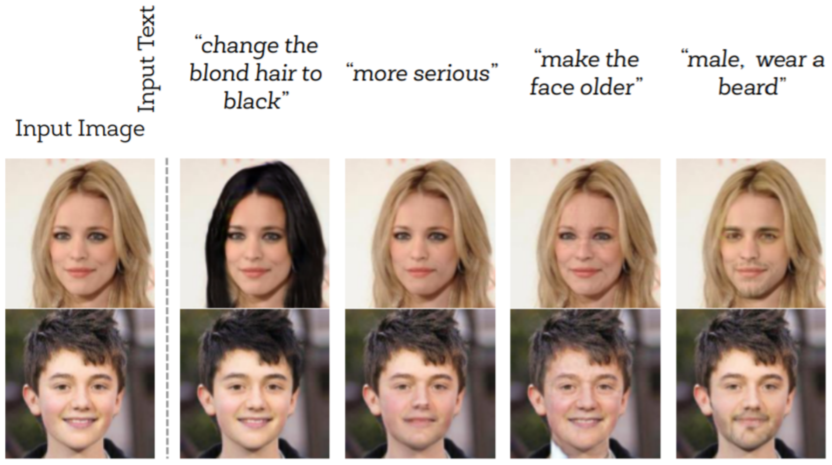
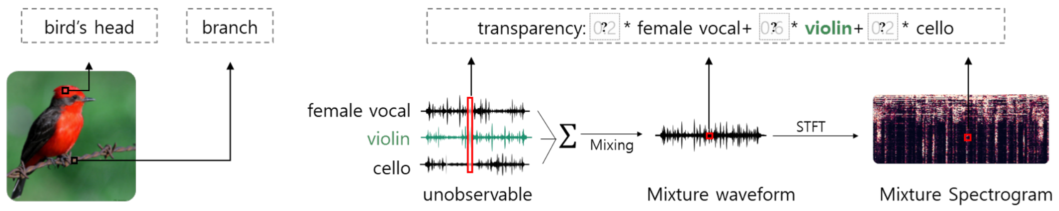


Figure 1.2: An Example of image manipulation with textual queries using Describe What to Change (DWC) model [23]

On the other hand, a sound object does not always belong to a single object in general. A sound object is an accumulated value of objects from different sources. It carries information from multiple sources, no matter how it is represented. It can be represented as a sample in a waveform or a frequency bin in a spectrogram, but both formats deal with multiple sources. For instance, suppose that a user has a waveform which is the mixture of three different sources, namely, female vocal, violin, and cello, as illustrated in Figure 1.3 (b). In contrast to an image pixel, a sound object of a random moment does not correspond to a single object, as shown in the figure. Furthermore, the energy ratio of each source is also blind unless the sources were observable. In general, those source-level audio tracks are not provided for the most non-expert end-users.



(a) A pixel corresponds to a single object (b) A sound object does not only carry information of a single source

Figure 1.3: Sound objects are transparent

This property makes audio editing more challenging. Recall the example of image manipulation introduced in Figure 1.2. As mentioned before, users can manipulate a specific object in an image by writing text queries such as "change the blond hair to yellow" by using trained neural networks. An advanced user of image editing tools also can perform such a task by manipulating target pixels with tools such as a brush. Similarly, suppose that a user wants to make the violin sound softer (i.e., smaller volume) and dryer (i.e., weak reverberation) in the example of Figure 1.3 (b). This task is difficult even for the experienced audio engineers because they must decompose the given track into a set of different sources from scratch; the violin and the others. This task is also known as *Audio Source Separation*. After separating sources successfully, they finally try to manipulate the violin track, which is painful again. It is still laborious to find an ideal Digital Signal Processing (DSP) function and appropriate parameters.

Moreover, audio source separation itself is not trivial at all due to the transparency. It is relatively easy to separate sources that have a restricted frequency range, such as *bass guitars*. For example, we can extract bass guitars from the music mixture by applying a low-pass filter function that filters out signals with higher frequencies than the given threshold. However, this approach is limited when applied to sources with a wide range of frequencies such as speech, singing voices, and drums. Besides, frequency ranges of different sources often overlap, making source separation more challenging. Audio engineers must carefully adjust parameters for each temporal chunk of samples to separate a specific source of which frequency range changes dynamically.

This complex procedure for audio source separation can be automated using the recent development of data-driven approaches. Many deep learning methods for source separation [1, 63, 21, 22, 32, 43, 60] have been proposed during the last decade. They have trained neural networks to predict the separated signal of an individual source for the given input mixture in an end-to-end manner. During the training phase, parameters in networks are optimized for the given source separation tasks. Most methods are trained to separate either a single source [1, 63] or multiple sources [22] simultaneously. The former is called *dedicated models* and the latter is called *multi-head models*. They are preferable due to their simplicity, robustness, and effectiveness.

Several researchers have proposed novel architectures that can perform several tasks with a single model conditioned on an extra signal. Throughout the rest of this thesis, these are called *Conditioned Source Separation* methods. Although the performance of conditioned separation models is usually inferior to that of the corresponding dedicated counterparts, they are preferable for service providers since they can perform multi-tasks with a single instance with limited parameters. A detailed overview of source separation models based on deep learning is given in section 2.2.1.

This thesis aims to develop various methods for audio processing that reduce human labor. This thesis mainly focuses on three audio tasks as follows: (1) dedicated source separation, (2) conditioned source separation, and (3) a novel task called Audio Manipulation on Specified Sources (AMSS). This paper introduces a novel concept called *latent sources*, which deal with more detailed aspects of audio signals than symbol-level sources to improve the performance of dedicated separation models and conditioned models.

On top of source separation methods, neural networks can perform advanced tasks, as computer vision models have shown. Considering that audio editing usually requires expert knowledge of audio engineering or signal processing, this thesis explores a deep learning approach for the third task, AMSS, in conjunction with textual queries to lessen audio editing difficulty. AMSS aims to edit only desired objects that correspond to specific sources, such as vocals and drums, according to a given description while preserving the content of sources that are not mentioned in the description. For AMSS, this thesis defines a query language called Audio Manipulation Language based on Context-Free Grammar [5]. Although many machine learning approaches have been proposed for audio processing [28, 29, 27, 48, 61, 33, 1, 32, 60, 43], to the best of our knowledge, there is no existing method that can directly address AMSS. The proposed method analyzes latent sources relying on the internal features for AMSS. The proposed method can be used for many applications, such as video creation tools making audio editing easy for non-experts. For example, users can decrease the volume of drums by typing simple textual instructions instead of time-consuming interactions with digital audio workstations.

1.1 Structure and Objectives of the Thesis

This thesis focuses on deep neural networks that reduce the difficulty of audio processing. The rest of this thesis is structured as follows: Chapter 2 overviews background technologies and the relevant literature related to this thesis. Chapter 3 proposes a method called *frequency transformation* to capture frequency-to-frequency correlations observed in spectrograms for dedicated source separation. Chapter 4 extends frequency transformation methods to conditioned source separation based on the *Latent Source Analysis* Chapter 5 defines a novel audio processing task called Audio Manipulation on Specified Sources and proposes a deep neural network based on the *Latent Source Analysis* again. Chapter 6 discusses the pros, cons, reusable insights, and potential applications of proposed models. This thesis is concluded in Chapter 7.

1.2 Contributions

The contributions of this thesis are summarized as follows:

- This thesis proposes a simple but effective Frequency Transformation method called Time-Distributed Fully connected network (TDF) that improves the source separation performance of U-Nets.

- This thesis empirically verifies that U-Nets equipping TDFs can outperform existing source separation models, especially for sources with harmonic frequency patterns such as singing voice.

- A novel concept of *Latent Sources Analysis* is proposed, which can be used to improve separation qualities of conditioned source separation models. The proposed concept is reusable to other tasks, including AMSS.

- An extension of frequency transformation block called Latent Source-attentive Frequency Transformation (LaSAFT) is proposed. It is an attention-based novel frequency transformation block that captures instrument-dependent frequency patterns.

- An extension of FiLM call Gated Point-wise Convolutional Modulation (GPoCM) is proposed to modulate internal features for conditioned source separation.

- The experimental results show that a Conditioned-U-Net equipping LaSAFTs outperforms existing conditioned models. The proposed model achieves state-of-the-art performance on several MUSDB18 tasks.

- This work is a pioneer study on selective audio manipulation. This thesis formulates a novel audio task called *Audio Manipulation on Specified Sources* (AMSS) and proposes a textual query language called Audio Manipulation Language based on Context-Free Grammar.

- A supervised training framework for AMSS is proposed based on source observable multi-track datasets and DSP libraries together with evaluation benchmarks for AMSS.

- This thesis proposes AMSS-Net, a novel neural architecture for AMSS. AMSS-Net performs latent source analysis that enables AMSS-Net to manipulate the given audio track delicately according to the text query.

- This thesis validates AMSS-Net architecture by presenting several experimental results, including an ablation study. AMSS-Net shows comparable performance to the state-of-the-art source separation models and performs more complicated source-level manipulation tasks for the given text queries.

1.3 Publications

Some ideas and figures have appeared previously in the following publications.

I Woosung Choi, Minseok Kim, Jaehwa Chung, Daewon Lee, and Soonyoung Jung. 2020. Investigating u-nets with various intermediate blocks for spectrogram-based singing voice separation. In Proceedings of the 21th International Society for Music Information Retrieval Conference.

II Woosung Choi, Minseok Kim, Jaehwa Chung, and Soonyoung Jung. 2021. Lasaft: Latent Source Attentive Frequency Transformation For Conditioned Source Separation. In ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 171–175. https://doi.org/10.1109/ICASSP39728.2021.9413896

III Woosung Choi, Minseok Kim, Marco A. Mart ınez Ram ırez, Jaehwa Chung, and Soonyoung Jung. 2021. AMSS-Net: Audio Manipulation on User-Specified Sources with Textual Queries. arXiv:2104.13553 [eess.AS]

IV Minseok Kim, Woosung Choi, Jaehwa Chung, Daewon Lee, and Soonyong Jung. 2019. Deep Convolutional Music Source Separation with Complex-valued Spectrograms. In the 3rd ICICPE 2019 Conference.

Chapter 2

Background

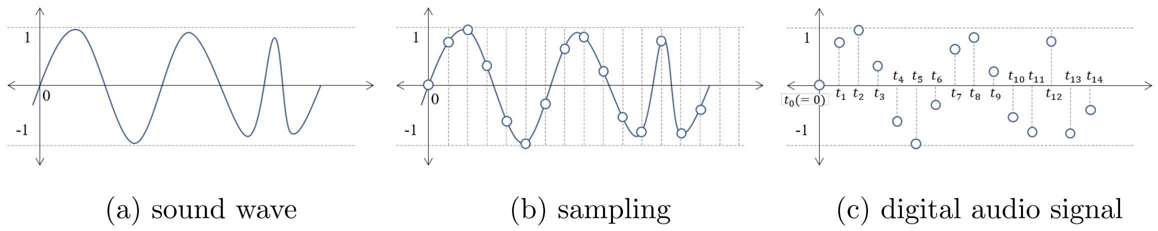
This chapter reviews the literature related to source-aware audio manipulation based on deep learning technologies. Section first 2.1 defines digital audio signals and reviews fundamental signal processing methods. Section 2.2 summarizes deep learning methods for audio processing such as audio source separation (section 2.2.1), modeling audio effects (section 2.2.2), and automatic audio mixing (section 2.2.3).

2.1 Digital Audio Signal Processing

This dissertation focuses on various audio tasks, namely, audio source separation and audio manipulation on specific sources. This section first defines *audio signals* in section 2.1.1, and multi-track in section 2.1.2. Section 2.1.3 overviews fundamental digital signal processing methods for audio and section 2.1.4 reviews the time-frequency analysis which is frequently used in digital signal processing.

2.1.1 Digital Audio Signal

An audio signal is represented as a sequence of numbers in modern computer systems. As described in Figure 2.1, every item of a digital audio signal is sampled from a value of sound wave at a specific moment. A digital audio signal is often denoted by a discrete function $x[n]$, where input n is an integer and $x[n]$ is the sampled value at the timestamp n.



1
0
-1
$t_0(=0)$
t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8 t_9 t_{10} t_{11} t_{12} t_{13} t_{14}

(a) sound wave (b) sampling (c) digital audio signal

Figure 2.1: Digital audio signal

The number of samples per second is called *sampling rate*. One of the most commonly used sampling rates is 44,100 Hz is because it is greater than twice the maximum hearing frequency. Some special purposed systems often use lower sampling rates. For example, a corpus for Automatic Speech Recognition (ASR) called LibriSpeech [36] contains 1000 hours of speech sampled at 16kHz. Since the frequency range of human voice is restricted, using a much larger sampling rate for ASR usually does not significantly improve the performance. For such tasks, lower sampling rates might be preferable. However, it is assumed that the sampling rate of an arbitrary audio signal is 44,100 Hz in this dissertation if not mentioned because MUSDB18 [41], a frequently used dataset throughout this paper, uses 44,100 Hz.

2.1.2 Multi-track Recording and Audio Mixing

Multi-track audio files are commonly used for music production and other fields. They are obtained by multi-track recording, where multiple sources are recorded separately (but not necessarily simultaneously). Multi-track audio, organized in a specific format or a Digital Audio Workstation (DAW) session, provides individual sources instead of a mixed audio track. They are helpful to music producers and sound engineers since they can provide the complete observability of individual sources, which mixed audio files cannot. For example, Native Instruments STEMS Format, or STEMS [1] is an open multi-track audio format. A Stem file contains four different musical sources: drums, bass, melody, and vocals. Every file from the MUSDB18 [41] dataset is encoded in the STEMS format. These files can be used for creative tasks such as remixing to alter a song to suit a different style.

Audio mixing is a task that combines multi-track recordings into a single audio track. Before taking the sum of signals (also known as *mixdown*), mix engineers usually apply various audio effects such as equalization and panning for a better hearing experience to minimize the interference between sources. Audio mixing requires deep background knowledge of sound engineering in general. Recently, some researchers have tried to automate audio mixing with deep learning approaches. Section 2.2.3 introduces such methods.

2.1.3 Audio Effects

As described in [48], audio effects are used to modify perceptual attributes of the given audio signal, such as loudness, spatialization, and timbre. They are also used for better hearing experience to avoid the undesirable interference between different sources as described in section 2.1.2.

[1]https://www.stems-music.com/

For example, audio engineers use *Equalization* effects to cut a specific frequency range of an instrument off so that listeners can focus on the other instrument, which has a more attractive timbre in that frequency range. There are various types of frequency filters in equalization effects. For example, a high-pass filter passes signals with higher frequencies than the given threshold. Similarly, a low-pass filter passes signals with lower frequencies than the given threshold. On the other hand, engineers use *Delay* effects to simulate the reflection of sound waves. Producers and DJs often use Delay effects to implement reverberant environments such as concert halls. There are several types of delay effects, such as reverberation and echo.

Similar to Audio Mixing, several methods have been proposed to automate audio effects. Using these effects also requires prior knowledge of sound engineering because an audio effect is usually controlled with many parameters. For example, an artificial reverberation has many controllable parameters such as room size, delay time, pre-delay time, decay, and high-cut threshold. It is painful even for experts to search the optimal configuration of an effect for the given contents. Deep learning enables to automate this configuration search with the data-driven approach, as described in section 2.2.2. Also, some of the effects above are modeled with deep neural networks in chapter 5.

2.1.4 Time-Frequency Analysis

Many real-world applications, such as, require both time and frequency information. For example, a DAW developer may want to visualize how the spectrum of frequencies varies with time. This visualization help music producers understand how the energy distribution of frequencies changes over time. In this scenario, the developer should implement Time-Frequency analysis methods such as Short-Time Fourier Transformation (STFT).

A Fourier Transform decomposes a time-varying signal into a weighted sum of trigonometric functions. The Short-Time Fourier Transformation performs Fourier Transform to each fixed-length window of samples. Since the computed coefficients are complex-valued, some audio engineers decompose the STFT outputs into Magnitude and Phase spectrogram. The Magnitude spectrogram of a signal can provide information on the energy distribution of frequencies. However, the linear scale of STFT outputs is not preferable for some tasks such as Automatic Speech Recognition (ASR) because human hearing is logarithmic. Therefore, some engineers take log-power spectrogram of the magnitude or even apply Mel-scale transform [7, 46, 10, 9] to the log-power spectrogram. Chapter 3 compares the performance of some types of spectrogram aforementioned in the Music Source Separation task.

2.2 Deep Learning for Audio Processing

Many practical DSP algorithms such as artificial reverberation, noise-canceling, or bandpass filters have been developed for audio engineers. The evolution in deep learning techniques has recently motivated the signal processing community to investigate deep learning approaches for digital audio processing. Deep learning, a part of machine learning technologies, is based on artificial neural networks. The beauty of deep learning is that one can make a machine learn a complicated task end-to-end if provided sufficient training data items and a well-defined loss function. Although it requires abundant background knowledge to design an appropriate architecture and a laborious search for optimal hyper-parameters, a deep learning-based approach is attractive due to its powerful performance.

For example, audio source separation can be automated with deep learning models. As mentioned in chapter 1, audio source separation is not trivial due to the transparency of an audio track. Audio engineers have to listen carefully to adjust DSP parameters such as a threshold frequency to separate sources of interest. However, audio source separation can be automated with a data-driven approach based on deep learning, which is discussed in section 2.2.1. Audio effects such as equalization, distortion, and reverberation can also be modeled with deep learning, which is reviewed in section 2.2.2. Besides, deep learning can model *mixing* procedure, as summarized in 2.2.3.

2.2.1 Audio Source Separation

An audio file usually contains multiple sources. For example, music signals are mixtures of several sub-tracks such as vocals, drums, and bass. Also, meeting recordings have multiple speakers. In some applications such as Speech Enhancement, the unwanted sound, *noise*, is also considered as a source. Noise is generated for several reasons: electronic noise, thermal noise, and physical noise in the real world. Speech Enhancement aims to extract clean speech from the noisy one.

The goal of *Audio Source Separation* is to extract a specific source from a given mixture. Recently, many machine learning-based methods have been proposed for audio source separation. They can be categorized into two groups in terms of the overall estimation method: waveform-to-waveform models and spectrogram-based models. The former method tries to generate the target waveform directly. Wave-U-Net [49], TasNet [25, 24], Conv-TasNet[26], and Demucs [6] are waveform-to-waveform models. TasNet and Conv-TasNet were proposed for speech separation. Speech separation is a special case of audio source separation, of which the goal is to separate individual speech from the overlapping speech signals. The authors of [24] trained TasNet to perform Speech Dereverberation, of which the goal is to extract a clean speech from a reverberant speech. Wave-U-Net and Demucs were proposed for Music Source Separation. They proposed waveform-to-waveform models adopting U-Net [42] structure and trained their models using MUSDB18 [41] dataset, which contains four different sources, namely, vocals, drums, bass, and 'other.' The authors of [6] also trained Conv-TasNet for music source separation using the same dataset. The quality of their separation are well organized and discussed in [6]. Demucs is one of the state-of-the-art models on the MUSDB18 benchmark.

While the waveform-to-waveform models aim to generate the target waveform directly, the spectrogram-based models estimate the spectrogram of the target source. In general, they apply Short-Time Fourier Transform (STFT) on a mixture waveform to obtain the input spectrograms. Then, they estimate spectrograms of the target source and finally restore the vocal waveform with inverse STFT (iSTFT).

A typical example of spectrogram-based models is a U-Net model proposed in [1], which section 3.2 reviews in detail. The proposed model is an encoder-decoder architecture with symmetric skip connections. [37, 65, 54, 53, 55] also proposed similar architectures. Especially, MMDenseNet[54] and its extension called MMDenseLSTM[53] adopted densely connected convolution blocks [16], initially proposed in computer vision, reporting promising results on MUSDB18 tasks. D3Net [55], which uses dilated convolutions with dense connection, currently holds state-of-the-art performance on several MUSDB18 tasks. While the models mentioned above usually use 2-D convolutions as a fundamental building block, DGRU-DGConv [22] uses 1-D convolutions. DGRU-DGConv is another state-of-the-art model on MUSDB18 tasks.

The number of tasks that the model can perform is another criterion to categorize deep learning-based source separation methods. They are categorized into three group, (1) dedicated models, (2) multi-head models, and (3) conditioned models, as shown in Figure 2.2.

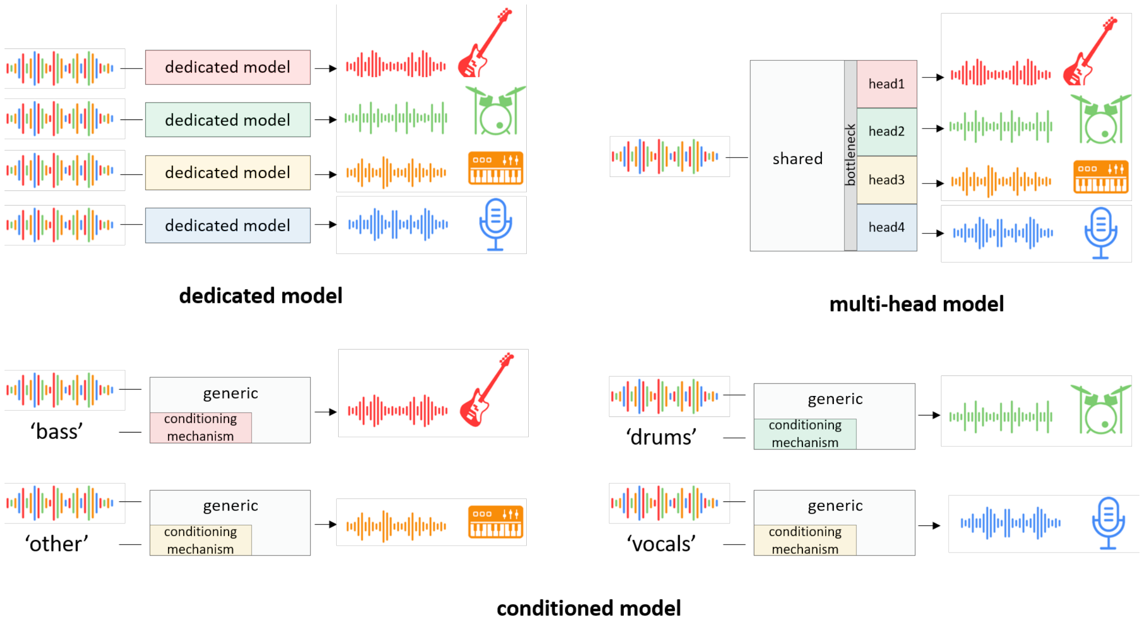


Figure 2.2: Taxonomy of source separation models

Most of the early models for Source Separation are dedicated models. Each of them is dedicated to a single instrument. This approach, however, has a critical drawback: we must train an individual model to separate a source because trained models cannot share parameters.

A multi-head model is a simple extension from dedicated separation to multi-source separation by using multi-head components. They generate several outputs simultaneously at a single inference. For example, DGRU-DGConv [22] generates multiple outputs. However, it also has a drawback in terms of scaling the number of sources: the number of heads increases as the number of sources increases, leading to the performance degradation caused by the shared bottleneck and the inefficient memory usage, as pointed out in Publish II.

While these methods separate either a single source or multiple sources once, conditioned source separation methods [32, 43] isolate the source specified by an input symbol. A conditioned model separates different instruments with the aid of the control mechanism. Since it does not need a multi-head output layer, there is no shared bottleneck. For example, Meta-TasNet [43] and Conditioned U-Net [32] proposed models that separate the source specified by an external symbol such as "vocals" or (1,0,0,0). Since such architectures enable efficient parameter-sharing for multi-source separation, they have attracted considerable interest despite their inferior performance.

Chapter 4 introduces a conditioned source separation model called Latent Source Attentive Frequency Transformation (LaSAFT), which is one of the state-of-the-art conditioned models on several MUSDB18 [41] tasks.

2.2.2 Modeling Audio Effects

As described in section 2.1.3, audio effects are used to modify perceptual attributes of the given audio signal, such as loudness, spatialization, and timbre. Recently, several methods [28, 29, 27, 48, 61] have been proposed for audio effect modeling with deep neural networks. [28] proposed a convolutional network performing equalization (i.e., an audio effect that changes the harmonic and timbrel characteristics of audio signals). [29, 27] proposed convolutional and recurrent networks for nonlinear audio effects with Long and Short-Term Memory [15], such as distortion and Leslie speaker cabinet. [48] presented an efficient neural network for modeling an analog dynamic range compressor enabling real-time operation on CPU.

The authors of [29] have extensively surveyed various audio effects and existing data-driven approaches for modeling them in [29]. They categorized audio effects into several types: nonlinear audio effects with short-term memory, time-dependent nonlinear effects, and time-varying audio effects. There are several effects in each type. For instance, amplifier, distortion and equalization are effects with nonlinear with short-term memory. They summarized methods for modeling each type in detail in [29].

All the methods mentioned above assume an audio input with a single source, while tasks addressed in this dissertation assume that an input is a mixture of several sources. Considering the transparency of sound mentioned in chapter 1, the environment assumed in this dissertation is more challenging than that of the existing methods.

It is also notable that the existing methods are dedicated to a single task (the trained model provides only one audio effect), while models for Conditioned Source Separation (see chapter 4) and AMSS (see chapter 5) can perform several tasks with a single instance. Although their performance is slightly inferior to their counterpart state-of-the-art models, they are still advantageous with respect to the task-scalability and memory space efficiency.

2.2.3 Automatic Mixing

Audio mixing is a task that combines multi-track recordings into a single audio track. Sound engineers apply various audio signal processing techniques such as equalization and panning for a better hearing experience to minimize the interference between sources before they take the sum of signals.

As mentioned in section 2.1.2, some approaches have attempted to automate this procedure with deep neural networks. For example, [47] proposed neural networks based on temporal dilated convolutions to learn neural proxies of specific audio effects and train them jointly to perform audio mixing. The authors of [30] reported that their drum mixing model could produce a virtually indistinguishable mixing result from a professional human-made mix in terms of user preference. Their model is expected to perform all signal processing or transformation involved in producing a musical mix.

These methods mainly focus on developing expert-type mixing models that combine individual sources into a mixture track, regardless of user control or input for the desired type of transformation. Unlike [47, 30], [31] assumed an environment where individual sources are not provided as input. They proposed an algorithmic framework that automatically remixes early jazz recordings, which are often perceived as irritating and disturbing from today's perspective. It first decomposes the given input into individual tracks by means of acoustic source separation algorithms and remixes them using automatic mixing algorithms. [33] proposed a replacement of the source separation and the mixing processes by deep neural networks. [33] mainly focuses on remixing to change the audio mixing style.

Chapter 3

Frequency Transformation for Dedicated Source Separation

This chapter proposes a dedicated source separation model based on a U-Net [42] architecture, which uses simple but effective frequency transformation blocks called Time-Distributed Fully-Connected Layers (TDF). This chapter shows that injecting TDFs into a fully convolutional U-Net can significantly improve the performance of dedicated source separation model and discusses how it improves performance by visualizing the weight matrix of a trained TDF. Specifically, section 3.1 introduces dedicated models for audio source separation. Section 3.2 reviews an existing dedicated separation model based on U-Net [42], which is fully convolutional. Section 3.3 summarizes several spectrogram estimation methods. Section 3.4 introduces a generalized architecture of U-Net, and the concept of *Frequency Transformation*, and section 3.5 proposes TDF. The experimental results are finally summarized in section 3.6 and the results are discussed in section 3.7.

3.1 Dedicated Models for Audio Source Separation

As mentioned in section 2.2.1, machine learning models for source separation can be grouped into three categories: (1) dedicated models, (2) multi-head models, and (3) conditioned models. This chapter focuses on dedicated models to improve their performance with simple but effective blocks called Time-Distributed Fully connected layers (TDF).

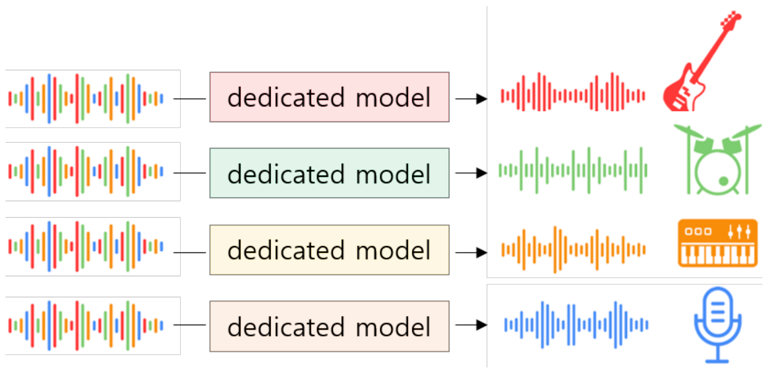


Figure 3.1: Dedicated Models

A dedicated model is trained to separate a single source, as its name implies. In other words, an individual model must be trained per each source of interest as illustrated in Figure 3.1 . Despite this disadvantage, many researchers and engineers prefer this model since it is easy to implement but produces high-quality results. Dedicated models usually outperform the other types with comparable parameters since dedicated models are solely optimized to a single source while the other types have to consider multiple sources simultaneously.

An early method[1] proposed a dedicated model that aims to separate singing voice signals from mixtures using the U-Net architecture [42]. They trained a U-Net to estimate the target spectrogram of singing voice from the spectrogram of the input mixture. This model also can be trained to separate other instruments such as bass or drums if provided a proper training dataset for such instruments. The following section reviews the U-Net architecture for Spectrogram-based Source Separation.

3.2 U-Net for Spectrogram-based Source Separation

While *waveform-to-waveform* methods such as [49, 6] estimate the target waveform from an input mixture waveform, *spectrogram-to-spectrogram* methods [1, 54, 63] predict the spectrogram of the target from a mixture spectrogram. This section introduces an early *dedicated method* [1] that aims to predict spectrograms of singing voice from mixture spectrograms using the U-Net architecture [42]. They formulated the singing voice separation as the translation of a mixed spectrogram into the vocal spectrogram and proposed a translation method adopting U-Net [42], an image-to-image translation model. U-Net, proposed initially for vision tasks, can be easily extended because a spectrogram consists of 2-D (time and frequency) bins, which is similar to a 2-D image consisting of pixels.

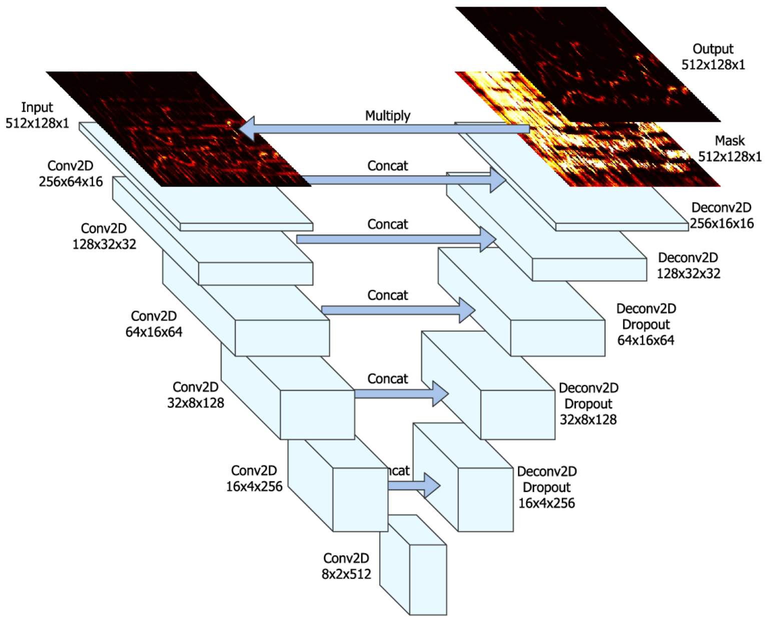


Figure 3.2: U-Net architecture used in [1] for Singing Voice Separation

The U-Net architecture used in [1] is illustrated in Figure 3.2. As shown in the figure, it is a fully convolutional encoder-decoder network with symmetric skip connections. The U-Net f with parameters Θ takes as input the magnitude spectrogram M of a given mixture and outputs $f(M, \Theta)$, a soft mask which is multiplied element-wise with M. The shape of $f(M, \Theta)$ is the same as the input's. It estimates the magnitude spectrogram $\hat{T}$ of singing voice as follows:

$$\hat{T} = f(M, \Theta) \odot M, \text{ where } \odot \text{ denotes the Hadamard product.} \tag{3.1}$$

It can finally generate the signals of singing vocals by applying iSTFT to the complex-valued spectrograms obtained from $\hat{T}$ and the phase information of the mixtures. They trained f to minimize the following loss function L_{unet} for the estimated spectrogram $\hat{T}$ and the ground-truth spectrogram T of singing voice:

$$L_{unet}(M, T; \Theta) = L_1(f(M, \Theta), T) = L_1(\hat{T}, T), \text{ where } L_1(x, y) = ||x - y||_1 \tag{3.2}$$

Treating spectrograms as images, the U-Net architecture proposed in [1] can separate singing voice signals from mixtures. Despite the successful extension to source separation from image segmentation, however, several researchers [54, 63, 62] have pointed out that the power of 2-D fully-CNNs is limited when applied to spectrogram processing. 2-D CNNs are proper building blocks for image processing since they excel at capturing spatially invariant features. However, it does not seem necessary for spectrograms, where each y-coordinate (frequency bin) corresponds to a fixed position in the given frequency range. Furthermore, patterns with very long-ranged dependencies are found along the frequency axis due to the harmonic structure of sounds. Fully 2-D convolutional model might not be ideal to capture such patterns in terms of the *receptive field*.

Figure 3.3 illustrates an example where the magnitude spectrograms of four instruments playing the same note were plotted. If a spectrogram is shifted along the y-axis and the output signal is reconstructed by applying iSTFT, the output would be far different from the original signal since its frequency information has changed. Also, long-range frequency patterns are observable in Figure 3.3. Moreover, it is observable that each instrument produces unique frequency patterns occurred by resonance. Source separation models need to capture those unique frequency-to-frequency dependencies observed in spectrograms since they are critical clues to distinguish sources.

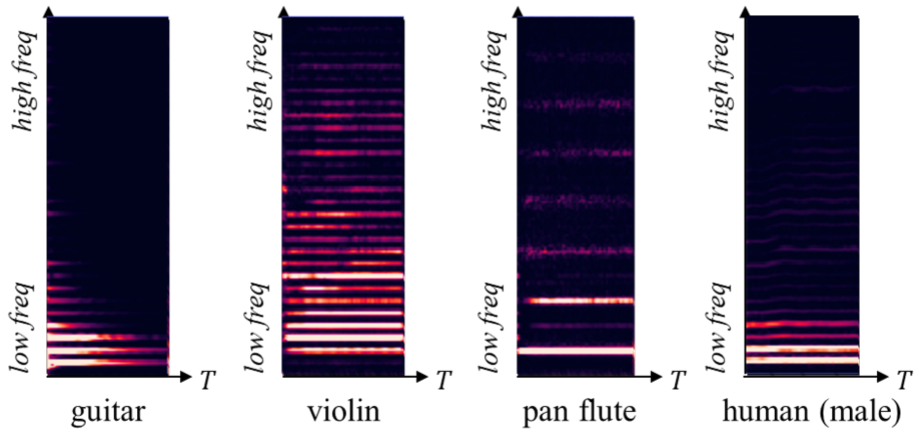


Figure 3.3: Magnitude Spectrograms of different instruments

These patterns can only be recognized by stacking many convolutional layers to enlarge the receptive field. However, a fully connected layers might be more suitable and efficient building blocks since even a single instance of them gives an entire receptive field. For example, a speech enhancement model called Phasen [63], which equips a fully connected layer in building blocks, achieved the state-of-the-art performance for speech enhancement. This type of block is called *Frequency Transformation Block* through the rest of the paper, and section 3.5 reviews a frequency transformation block called TDF and its variant used in detail.

This chapter aims to validate the following assumption: *Injecting frequency transformation blocks into a standard U-Net architecture can significantly improve the separation performance.* Section 3.4 presents a generalized architecture of U-Net. However, there are multiple options when designing a U-Net structure for audio source separation based on spectrograms. Thus, section 3.3 first summarizes possible spectrogram estimation methods. Section 3.5 proposes TDF blocks which improves the performance of a standard U-Net. Section 3.6 compares the performance of two models (namely, the fully convolutional U-Net and the enhanced U-Net presented in section 3.5), which share the generalized architecture.

3.3 Spectrogram Estimation Methods

This section reviews different methods that have been used for source separation based on spectrogram. They are categorized into four groups as follows: (1) Direct Magnitude Estimation (DME), (2) Magnitude Mask Estimation (MME), (3) Direct Complex-valued Spectrogram Estimation (DCSE), and (4) Complex-valued spectrogram Mask Estimation (CME). Figure 3.4 summarizes their data flows.

As illustrated in Figure 3.4, every method takes a raw STFT output $M_{complex}$, which is complex-valued. While DME and MME only feed the magnitude spectrogram M_{mag} into their neural networks, DCSE and CME feed the original $M_{complex}$ to their networks. Also, DME and MME aim to estimate the magnitude spectrogram of the target source $\hat{T}_{mag}$, the others aim to estimate the raw complex-valued spectrogram $\hat{T}_{mag}$ of the target.

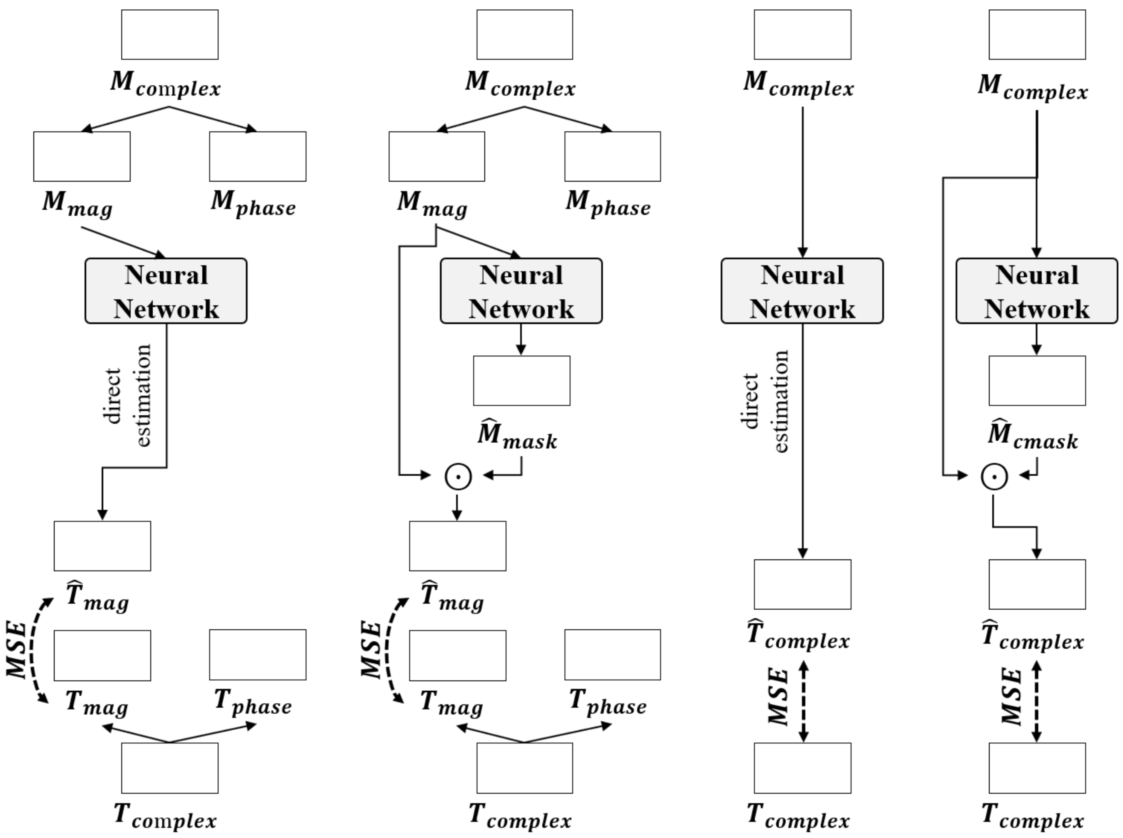
3.3.1 Direct Magnitude Estimation

The Direct Magnitude Estimation (DME) method takes the magnitude spectrogram M_{mag} of a mixture signal as input (or the log-scale magnitude $log(1 + M_{mag})$). As illustrated in Figure 3.4. (a), it directly estimates the magnitude spectrogram $\hat{T}_{mag}$ of the separated target signal. To obtain the target signal, it combines $\hat{T}_{mag}$ with M_{pahse} to construct the complex-valued spectrogram $\hat{T}_{complex}$ and apply iSTFT to $\hat{T}_{complex}$. DME-based models are usually trained to reduce the MSE or MAE between $\hat{T}_{mag}$ and the ground-truth target magnitude spectrogram T_{mag}.

Although it may seem impractical to estimate the target magnitude in an unbounded manner, some studies [22, 53] have empirically shown that DME-based models can show outstanding results. For example, DGRU-DGConv [22], one of the state-of-the-art models, is trained to directly estimate $\hat{T}_{mag}$ for a given input $log(1 + M_{mag})$.

3.3.2 Magnitude Mask Estimation

The Magnitude Mask Estimation (MME) method takes the magnitude spectrogram M_{mag} of a mixture signal as input (or the log-scale magnitude $log(1 + M_{mag})$). MME-



(a) DME (b) MME (c) DCSE (d) CME

Figure 3.4: Spectrogram Estimation Methods

based models are trained to learn a binary or soft mask $\hat{M}_{mask}$ that is multiplied with M_{mag} to obtain the estimated magnitude spectrogram $\hat{T}_{mag}$ of the target source (that is, $\hat{T}_{mag} = \hat{M}_{mask} \odot M_{mag}$) as illustrated in Figure 3.4. (b). To obtain the separated target signal, it combines $\hat{T}_{mag}$ with M_{phase} to construct the complex-valued spectrogram $\hat{T}_{complex}$ and apply iSTFT to $\hat{T}_{complex}$ for the target signal reconstruction. The Mean Square Error (MSE) (, or Mean Absolute Error (MAE)) between $\hat{T}_{mag}$ and the ground-truth target magnitude spectrogram T_{mag} is usually used for its loss function. The U-Net introduced in section 3.2 uses this method for target spectrogram estimation.

3.3.3 Direct Complex-valued Spectrogram Estimation

In DME and MME, the phase M_{phase} of the mixture is reused to generate the target signal. However, M_{phase} is usually not the same as the phase T_{phase} of the target signal. Thus, magnitude estimation methods usually perform poorer than phase-aware estimation methods [63, 52, 6] due to the loss of phase information. For phase-aware estimation, some methods [63, 52] estimate the phase as well as the magnitude of the target while other [11, 35] methods directly estimate the target complex-valued spectrogram. The latter method is called Direct Complex-valued Spectrogram Estimation (DCSE). The DCSE method takes the complex-valued spectrogram $M_{complex}$ of a mixture signal and directly estimates the complex-valued spectrogram $\hat{T}_{complex}$ of the target, as shown in Figure 3.4. Models are trained to reduce the MSE between $\hat{T}_{complex}$ and the ground-truth target spectrogram $T_{complex}$.

When developing a DCSE model, it is natural to adopt Deep Complex Networks (DCNs) [56], which have building blocks with complex-valued parameters such as complex convolutions and complex-valued activation functions, because the input and corresponding output are complex-valued. Alternatively, conventional real-valued neural networks can be used by viewing the real and the imaginary parts of a spectrogram as separate channels. This approach is called *Complex-as-Channels (CaC)* throughout the rest of the paper.

3.3.4 Complex Mask Estimation

The Complex Mask Estimation (CME) method is a soft masking version of the DCSE method. It predicts a mask $\hat{M}_{mask}$ instead of $T_{complex}$. CME-based models generate the target complex spectrogram $T_{complex}$ by multiplying $\hat{M}_{mask}$ by $M_{complex}$ (that is, $\hat{T}_{complex} = \hat{M}_{mask} \odot M_{complex}$) as described in Figure 3.4. (d)). For the last activation to obtain $\hat{M}_{mask}$, it can use tanh, sigmoid or more advanced activation functions for complex-valued features such as MODReLU [2], $\mathbb{C}$ReLU[56], and zReLU [13].

3.4 Baseline U-Net Architecture

This section presents a generalized U-Net architecture for spectrogram-based source separation, which is shared by models in section 3.6. The baseline U-Net estimates the spectrogram $\hat{T}$ from a given spectrogram M of the mixture. For the sake of simplicity, it is assumed that M and $\hat{T}$ are complex-valued spectrograms unless it is explicitly mentioned that they are magnitude spectrograms. Also, it is assumed that the baseline model directly estimates target spectrograms instead of masks.

As shown in Figure 3.5, the baseline U-Net consists of an encoder and decoder: the encoder embeds M into a down-sized representation, and the decoder takes it and estimates target spectrogram $\hat{T}$. It has down-sampling layers with stride two and up-sampling layers, which double the scale of an input tensor. The number of down-sampling layers and up-sampling layers are the same, as in the conventional U-Net architecture [42]. It has skip connections, which are the most important components in U-Net-like structures [42, 1, 65, 54, 53]. Each of them connects an internal feature map in the encoder to the corresponding feature map in the decoder with the same scale. It enables the U-Net to effectively recover fine-grained details of the spectrogram during up-sampling.

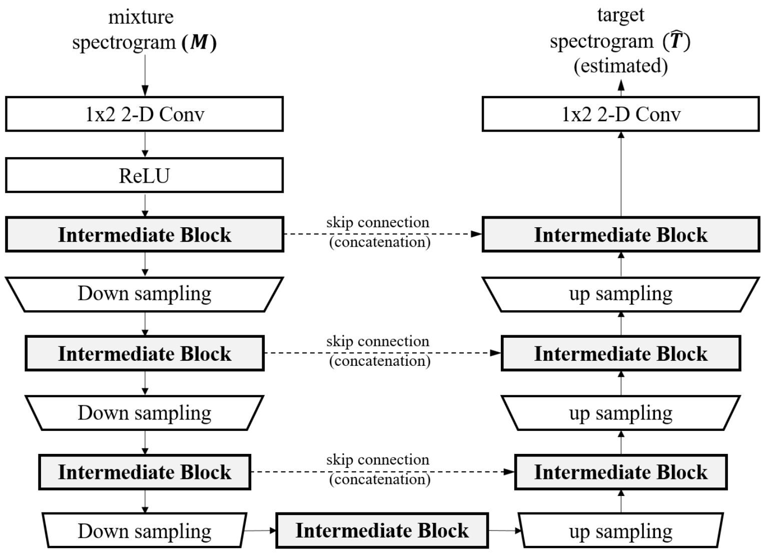


Figure 3.5: Baseline U-Net Architecture

As shown in Figure 3.5, the baseline network has several *intermediate blocks*. For the sake of simplicity, it is assumed that each intermediate block maps an input tensor to an equally-sized tensor, possibly with a different number of channels. In other words, the size of an internal tensor is changeable only with down-sampling layers and up-sampling layers. Exceptionally, it uses two additional convolution layers that slightly change the size of tensors. It uses them to control the number of channels and make the number of frequency bins (i.e., the height of spectrogram in Figure 3.3) even-numbered.

Before describing these two convolutions, some additional notations are introduced as follows. The input of the l-th intermediate block is denoted by $X^{(l-1)}$, and the output by $X^{(l)}$. The size of $X^{(l-1)}$ is denoted by $c_{in}^{(l)} \times T^{(l)} \times F^{(l)}$, where $c_{in}^{(l)}$ represents the number of channels and and $T^{(l)} \times F^{(l)}$ represents the size of the spectrogram-like tensor. Also, the size of $X^{(l)}$ is denoted by $c_{out}^{(l)} \times T^{(l)} \times F^{(l)}$, where $c_{out}^{(l)}$ is the number of channels.

Using these notations, the input of the first block is $X^{(0)}$, and its size is $c_{in}^{(1)} \times T^{(1)} \times F^{(1)}$. As illustrated in Figure 3.5, the first convolution layer, which takes M as input, generates $X^{(0)}$. It is a 1×2 2-D convolution with $c_{in}^{(1)}$ output channels, followed by ReLU [12]. $c_{in}^{(1)}$ is set to be 24 to increase the number of internal channels. The final convolution layer is a 1×2 convolution with c output channels, where c is the number of channels of the input spectrogram. The parameter $c_{in}^{(1)}$ is empirically set to be 24 in the experiments. Models with smaller $c_{in}^{(1)}$ (e.g., 12) are trained faster, but usually perform inferior than models with larger size of $c_{in}^{(1)}$.

Since there are multiple options for intermediate blocks, one can implement various variants of U-Nets for dedicated source separation based on this architecture. For example, one can design a fully convolutional U-Net by employing a block of densely connected 2-D Convolutions used in [54, 53], called Time-Frequency Convolutions (TFC) block. A TFC block extracts features by considering both the time and the frequency dimensions. It treats spectrogram-like features as image features. The baseline intermediate, TFC, is introduced as follows:

A baseline intermediate block: Time-Frequency Convolutions

The Time-Frequency Convolutions (TFC) is a block of densely connected 2-D CNNs, as shown in Figure 3.6. This dense block is comprised of densely connected composite layers, where each layer is defined as three consecutive operations: 2-D convolution, BN [17], and ReLU [12]. It is applied to the spectrogram-like input representation in the time-frequency domain. Every convolution layer in a dense block has kernels of size (k_F, k_T). Its 2-D filters are trained to capture features along both frequency and temporal axes jointly.

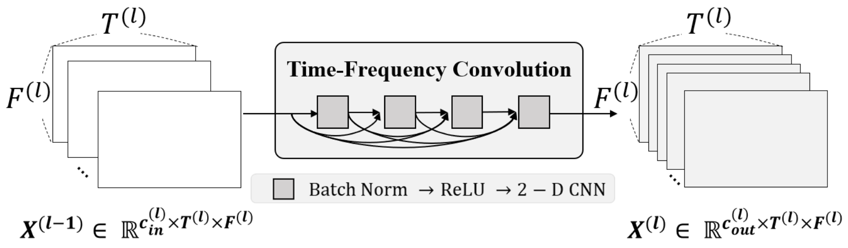

$T^{(l)}$
$F^{(l)}$
Time-Frequency Convolution
Batch Norm → ReLU → 2 − D CNN
$T^{(l)}$
$F^{(l)}$
$X^{(l-1)} \in \mathbb{R}^{c_{in}^{(l)} \times T^{(l)} \times F^{(l)}}$
$X^{(l)} \in \mathbb{R}^{c_{out}^{(l)} \times T^{(l)} \times F^{(l)}}$

Figure 3.6: Time-Frequency Convolutions

3.5 Frequency Transformation Block

Some existing models based on the U-Net architecture use CNNs (e.g., [3]) for intermediate blocks to extract timbre features of the target source. However, the authors of [63] reported that conventional CNN kernels are limited for this task. They observed that long-range correlations exist along the frequency axis in the spectrogram of voice signals, which Fully-connected Neural Networks (FCNs) can efficiently capture. They proposed a model called Phasen for speech enhancement, which uses the Frequency Transformation Block (FTB) that has a single-layered FCN without bias. This FCN is applied to each frame of the internal representation in a *time-distributed* manner. Capturing harmonic correlation with TFBs, Phasen achieved the state-of-the-art speech enhancement performance.

Inspired by TFB, this section proposes the Time-Distributed Fully connected network (TDF) block, applied to a single frame of a spectrogram-like feature map. It aims to extract time-independent features that help source separation without using inter-frame operations. This section also proposes a block that combines TFC and TDF, called TFC-TDF block.

Time-Distributed Fully-connected networks

As illustrated in Figure 3.7, a TDF block is applied to each channel of each frame separately and identically.

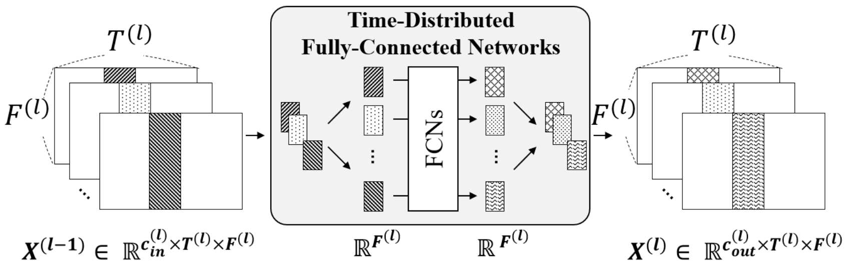

$T^{(l)}$
$F^{(l)}$
Time-Distributed Fully-Connected Networks
FCNs
$T^{(l)}$
$F^{(l)}$
$X^{(l-1)} \in \mathbb{R}^{c_{in}^{(l)} \times T^{(l)} \times F^{(l)}}$
$\mathbb{R}^{F^{(l)}}$
$\mathbb{R}^{F^{(l)}}$
$X^{(l)} \in \mathbb{R}^{c_{out}^{(l)} \times T^{(l)} \times F^{(l)}}$

Figure 3.7: Time-Distributed Fully-connected networks

Suppose that the l-th intermediate block in the U-Net structure takes input $X^{(l-1)}$ into an output $X^{(l)}$. As shown in Figure 3.7, a fully-connected network is applied separately and identically to each frame (i.e., $X^{(l-1)}[i,j,:]$) in order to transform an input tensor in a time-distributed fashion. While an FTB of Phasen [63] is single-layered, a TDF block can be either single- or multi-layered. Each layer is defined as consecutive operations: a fully-connected layer, Batch Norm (BN), and ReLU. If it is multi-layered, then each internal layer maps an input to the hidden feature space, and its final layer maps the internal vector to $\mathbb{R}^{F^{(l)}}$. The number of hidden units is $\lfloor F^{(l)}/bn \rfloor$, where the bottleneck factor is denoted by bf. The number of parameters can be reduced if a TDF has a bottleneck factor of $bf > 2$.

Time-Frequency Convolutions with TDF

Another proposed block is the Time-Frequency Convolutions with Time-Distributed Fully-connected networks (TFC-TDF) block. It utilizes two different blocks inside: a TFC block and a TDF block. Figure 3.8 describes the structure of a TFC-TDF block. It first maps the input $X^{(l-1)}$ to a same-sized representation with $c_{out}^{(l)}$ channels by applying the TFC block. Then the TDF block is applied to the dense block output. A residual connection is also added for efficient gradient flow.

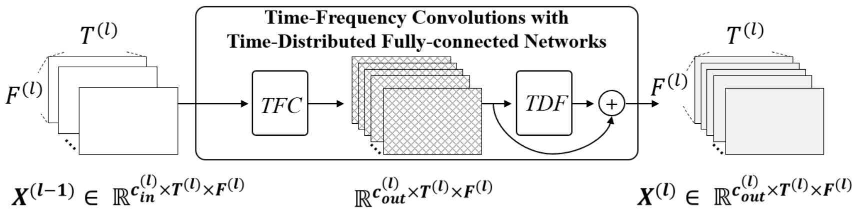

Time-Frequency Convolutions with
Time-Distributed Fully-connected Networks
$T^{(l)}$
$F^{(l)}$
TFC
TDF
$T^{(l)}$
$F^{(l)}$
$X^{(l-1)} \in \mathbb{R}^{c_{in}^{(l)} \times T^{(l)} \times F^{(l)}}$
$\mathbb{R}^{c_{out}^{(l)} \times T^{(l)} \times F^{(l)}}$
$X^{(l)} \in \mathbb{R}^{c_{out}^{(l)} \times T^{(l)} \times F^{(l)}}$

Figure 3.8: Time-Frequency Convolutions with TDF

Phasen [63] has shown that inserting time-distributed operations into intermediate blocks can improve speech enhancement performance. Section 3.6 validates whether it also works for music source separation or not, using MUSDB18 [41] benchmark.

3.6 Experiment

This section evaluates U-Nets with different types of blocks based on different estimation methods mentioned above. Section 3.6.1 presents the experimental setup including dastaset and model configurations. Section 3.6.2 compares four different spectrogram estimation methods introduced in section 3.3. Section 3.6.3 empirically validates the assumption: "Injecting frequency transformation blocks into a standard U-Net architecture can significantly improve the separation performance." The proposed architecture is compared with other state-of-the-art models in section 3.6.4. Ablation studies are presented in section 3.6.5. Section 3.6.6 presents a novel task to visualize artifacts created by DSCE methods.

3.6.1 Setup

Dataset

In this section, models are evaluated for the multi-channeled Musical Source Separation task on the MUSDB18 [41] dataset. The train and test datasets of MUSDB18 have 100 and 50 musical tracks respectively, all stereo and sampled at 44100 Hz. Each track file consists of the mixture and four source audios: 'vocals,' 'drums,' 'bass' and 'other.'

The default validation set (14 tracks) for validation is used as defined in the *musdb* package. Thus 86 tracks is used as training items in the training phase. Data augmentation [57] was applied to obtain mixture audio clips comprised of source audio clips from different tracks.

Model Configurations

U-Nets with different blocks based on different estimation methods are implemented for experiments. Each model is based on the U-Net architecture (section 3.4). $c_{in}^{(1)}$, the number of internal channels is set to 24, as mentioned in section 3.4. Each model uses a single type of block for its intermediate blocks. For STFT parameters, an FFT window size of 2048 and a hop size of 1024 for STFT are used. However, a larger window size is used in some models for a fair comparison with state-of-the-art methods. In other words, an FFT window size of 4096 and a hop size of 1024 are used for large models.

Each model is trained and evaluated for separating a single instrument. They are individual models and each does not share any parameters with others. Models in section 3.6.2 contains 7 intermediate blocks, and models in section 3.6.4 contains 9 intermediate blocks. Each TFC block has 5 convolution layers with 3×3 kernels and growth rate [16] of 24. Each TDF has a bottleneck factor of 16.

Weights of each model were optimized with RMSprop [14] with learning rate $lr \in [0.0005, 0.001]$ depending on model depth. Each model is trained to minimize the mean square error between $\hat{T}$ and $\mathbf{T}$. Mean Squared Error (MSE) between target and estimated signal (waveform) as the validation metric for validation is measured in each epoch for validation.

The official evaluation tool[1] is used, which is provided by the organizers of the SiSEC 2018 [50] to measure Source-to-Distortion Ratio (SDR) [59]. The evaluation metric is the median SDR value over all the test set tracks to obtain the overall SDR performance for each run, as done in the SiSEC2018. The average of 'median SDR values' over three runs for each model is reported for each model.

3.6.2 Comparison of Spectrogram Estimation Methods

Table 3.1 summarizes the SDR performance of models based on four different spectrogram estimation methods introduced in section 3.3. It is observable that the SDR performance of DCSE models is higher than that of the other models by large margins. Thus, the DCSE method is selected for the comparison with state-of-the-art models in section 3.6.4. Both CME methods were slightly inferior to the DCSE method, and there were no significant SDR differences between the CME with sigmoid function (denoted by CME-σ) and the CME with hyperbolic tangent (denoted by CME-tanh). Three complex-valued methods usually performed better than magnitude estimation models for every instrument. It means that considering phase information is essential for separation qual-

[1]https://github.com/sigsep/sigsep-mus-eval

ity.

esimation	vocals	drums	bass	other	mean
DME	6.01	4.17	3.98	4.16	4.58
MME	5.28	3.35	3.96	4.45	4.26
CME-σ	7.04	4.88	4.99	4.51	5.35
CME-tanh	6.89	4.90	5.05	4.55	5.35
DCSE	**7.05**	**5.38**	**5.62**	**4.61**	**5.66**

Table 3.1: Comparison of TFC-TDF-U-Nets on spectrogram estimation methods: Source-to-Distortion Ratio (SDR)

The SIR performance and ISR performance show a similar trend, as shown in Table 3.3 and Table 3.2. The DCSE method usually performs better than the other model except for few cases, but only a small margin.

framework	vocals	drums	bass	other	ALL
DME	12.68	7.92	8.68	7.79	9.27
MME	**13.61**	8.06	9.23	8.02	9.73
CME-σ	13.08	7.78	7.56	7.53	8.99
CME-tanh	12.75	7.68	9.01	7.52	9.24
DCSE	13.17	**9.27**	**9.95**	**8.28**	**10.17**

Table 3.2: Comparison of TFC-TDF-U-Nets on spectrogram estimation methods: Image to Spatial Distortion Ratio (ISR)

Compared to SDR and SIR, the SAR performance shows a different trend: the DCSE method does not always have higher SAR. It might be because direct estimation methods suffer from many artifacts due to their unbounded nature. Such artifacts is visulalized in section 3.6.6 for an evidence.

framework	vocals	drums	bass	other	mean
DME	11.51	7.97	4.87	6.46	7.70
MME	9.12	6.90	5.57	6.58	7.04
CME-σ	13.60	10.55	8.68	9.50	10.58
CME-tanh	13.53	**12.20**	8.89	8.58	10.80
DCSE	**14.38**	11.97	**11.36**	**9.97**	**11.92**

Table 3.3: Comparison of TFC-TDF-U-Nets on spectrogram estimation methods: Source-to-Interferences Ratio (SIR)

framework	vocals	drums	bass	other	mean
DME	6.16	4.30	3.67	4.48	4.65
MME	6.02	4.74	3.63	**5.04**	4.86
CME-σ	**6.96**	4.37	**5.63**	3.95	**5.23**
CME-tanh	6.70	4.37	4.25	4.01	4.83
DCSE	**6.96**	**5.13**	4.82	3.92	5.20

Table 3.4: Comparison of TFC-TDF-U-Nets on spectrogram estimation methods: Sources-to-Artifacts Ratio (SAR)

3.6.3 Injecting frequency transformation blocks into a standard U-Net architecture

This section validates the assumption: "Injecting frequency transformation blocks into a standard U-Net architecture can significantly improve the separation performance." To verify it, DCSE-based U-Nets with different time-frequency blocks were implemented for singing voice separation tasks. All models are trained on 3 seconds (128 STFT frames) of music. Since the number of frequency bins is much larger than the number of frames, models with more than 7 neural transforms use both 2×2 or 2×1 sized down/up-sampling layers to scale the frequency axis more than 3 times while maintaining the number of scales in the temporal axis to 3. Exceptionally, different down/up-sampling strategies were used to investigate the effect of down/up-sampling in the temporal axis.

model	sampling	# blocks	# params	SDR
TFC	O	17	1.56M	6.89
TFC	X	17	1.56M	6.75
TFC-TDF	O	7	0.99M	7.07
TFC-TDF	O	17	1.93M	**7.12**

Table 3.5: Evaluation results of Time-Frequency Blocks.

Every TFC block is set to have 5 convolution layers with kernel size 3×3. The growth rate is set to be 24. By using this TFC block configuration, a TFC-based U-Net is trained and the result is in the first row of Table 3.5.

The model in the second row is set to use different down/up-sampling layers to investigate the effect of down/up-sampling in the temporal axis. Every kernel size used in each down/up-sampling layer of this model is 2×1 to preserve the temporal resolution while scaling frequency resolution.

The first two rows of Table 3.5 summarize the experiment results of two TFC-based models. The model that preserves the temporal resolution (i.e., the second row) was slightly inferior to the other model. It is also notable that the U-Nets with TFC blocks achieve comparable results with state-of-the-art methods in 3.6.4, even using lower frequency resolution.

The third and fourth rows of Table 3.5 shows promising results regarding the U-Nets with TFC-TDF blocks. The same TFC setting above is used, and bf is set to 16 for each TDF. The 7-blocked U-Net with TFC-TDFs outperforms the other 17-blocked models with nearly twice as less number of layers. These results show that inserting TDFs into intermediate blocks can be helpful for music source separation as well as for Speech Enhancement [63]. Also, results show that it is achievable with fewer parameters by using FCNs with a bottleneck layer.

3.6.4 Comparison with State-of-the-art Models

Table 3.6 shows the SDR performance of state-of-the-art models along with ours on the MUSDB dataset: (MMDenseLSTM [53], D3Net [55], Meta-TasNet [43], UMX [51], DGRU-DGConv [22], Demucs [6], and Conv-Tasnet [26, 6]). For a fair comparison, the size of networks is enlarged (see section 3.6.1). The proposed model is a U-Net with nine TFC-TDF blocks based on DCSE method, which achieves the best SDR performance on 'vocals' and 'other' even when compared to models that use extra training data and outperforms models without extra data for bass separation. In the case of drums and others, it shows comparable performance to other methods. Also, it is worth noting that previous spectrogram-based models adopt Multi-channel Wiener Filtering as a post-processing method to enhance SDR further, while ours directly use the signal reconstruction output without such post-processing.

3.6.5 Ablation Study

This section shows how a U-Net model with TFC-TDF blocks works. Table 3.7 shows the performance of three different singing voice separation models on the MUSDB benchmark. One of them (the first row in the table) is the reference model that achieves the state-of-the-art SDR performance (7.98 dB). The other models have different configurations.

model	w2w	extra	vocals	drums	bass	other
Demucs [6]	✓	✗	6.84	**6.86**	**7.01**	4.42
Conv-Tasnet [6, 26]	✓	✗	6.43	6.02	6.20	4.27
Meta-TasNet [43]	✓	✗	6.40	5.91	5.58	4.19
UMX [51]	✗	✗	6.32	5.73	5.23	4.02
DGRU-DGConv [22]	✗	✗	6.85	5.85	4.86	4.65
MMDenseLSTM [53]	✗	✗	6.60	6.43	5.16	4.15
D3Net [55]	✗	✗	7.24	7.01	5.25	4.53
Proposed [I]	✗	✗	**7.98** $\pm$ .07	6.11 $\pm$ .13	5.94 $\pm$ .08	**5.02** $\pm$ .07
Demucs [6]	✓	1.5k	7.29	**7.58**	**7.60**	4.69
Conv-Tasnet [26, 6]	✓	150	6.74	7.11	7.00	4.44
MMDenseLSTM [53]	✗	804	7.16	6.81	5.4	4.8
D3Net [55]	✗	1.5k	7.80	7.36	6.20	5.37

Table 3.6: Comparison with state-of-the-art models on MUSDB dataset

TFC-TDF block VS TFC block

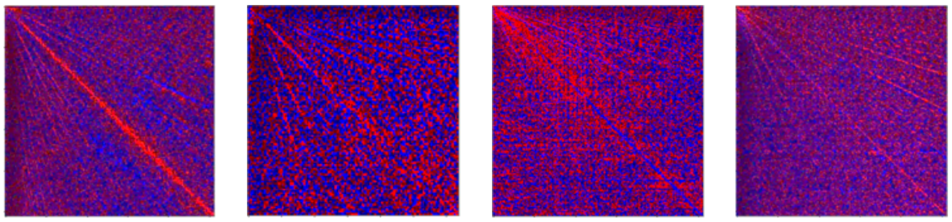
The TDF block captures long-ranged patterns along the frequency axis. Replacing each TFC-TDF block with a simple TFC block makes the resulting model cannot exploit such frequency correlation feature enhancement. The last row in Table 3.7 shows the performance of the resulting model. Its low SDR indicates that using TDF with TFC block in multi-scales significantly improves the quality of singing voice separation. However, Its SDR is comparable to that of existing spectrogram-based models unless it is not trained with an extra dataset.

Bottleneck Layers in TDFs

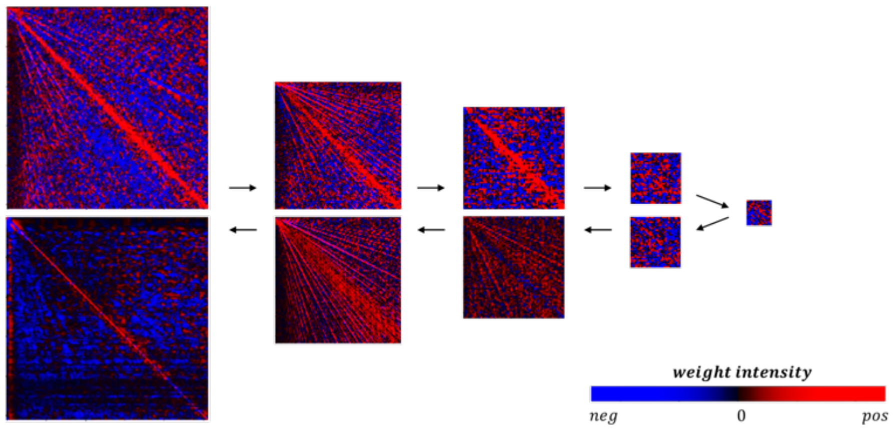
The second row in Table 3.7 shows the SDR performance of TFC-TDF-U-Net architecture without bottleneck layers in TDFs. Its performance is slightly lower than that of the reference model. However, it is worthy to say a large amount of the number of parameters

block	bottleneck	# params	SDR
TFC-TDF	✓	2.24M	**7.98**
TFC-TDF	✗	12.00M	7.72
TFC-TDF (DME)	✓	2.24M	7.24
TFC	N/A	0.83M	6.60

Table 3.7: Ablation Study of TFC-TDF-U-Nets (Vocals)



(a) vocals (b) bass (c) drums (d) other



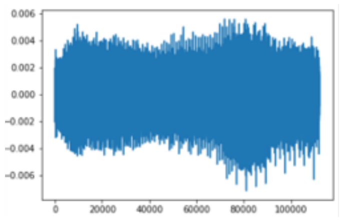
(e) Singing voice in multi-scales

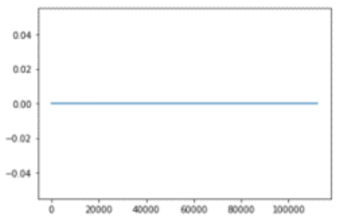
Figure 3.9: Weight Visualization in single-layered TDFs

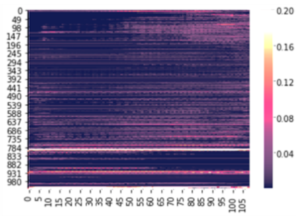
can be reduced in each TDFs: about $(bf)^2/2$ times smaller than a single-layered TDF, where bf is a bottleneck factor.

Despite its low SDR, single-layered TDFs provide us insight into how it enhances frequency correlation features. Inspired by [63], we also visualized the weight matrix after training, as shown in Figure 3.9. Figure 3.9 (a),(b),(c) and (d) visualize the weight matrix of the first TDFs for each model. Each matrix is optimized to enhance timbre features for its instrument by capturing different frequency dependencies observed along the frequency axis. Also, we can observe that each TDF still performs well in multi-scales.

3.6.6 Silent Mixture Waveform Separation


(a) output signal (DCSE)


(b) output signal (CME)


(c) magnitude (DCSE)

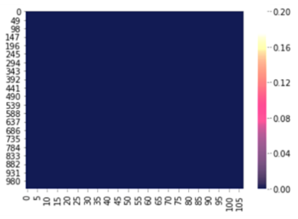

(d) magnitude (CME)

Figure 3.10: Result for Silent Mixture Signal Separation Task

This experiment feeds a silent mixture signal (i.e., a zero-valued signal) to networks and visualizes their outputs. The desirable output is also a silent signal. The output of the CME-based network is silent, as shown in Figure 3.10 (b) and (d). However, it turns out that DCSE-based TFC-TDF-U-Net generates a noise signal as shown in Figure 3.10 (a). Models without bias parameters (for TDFs and the final convolution) also generated such artifacts. Despite its high SDR performance, we could such very small artifacts constantly arising throughout playing a separated track.

Its magnitude spectrogram has none-zero TF bins as illustrated in Figure 3.10 (c). Since $M_{complex}$ is zero-valued $\hat{T}_{complex} = \hat{M}_{mask} \odot M_{complex}$ is also zero-valued in the CME method. Similar observation was reported in [40], where they fed a wight noise signal to neural networks with different up-sampling methods and visualized the spectrogram of the output signal.

3.7 Discussion

This section validates the following assumption: *Injecting frequency transformation blocks into a standard U-Net architecture can significantly improve the separation performance.* It also provides resuable insights as follows:

- Using down/up-sampling is important for CNN-based blocks, especially in the frequency dimension.
- Injecting a time-distributed block to a time-frequency block can improve SDR.
- U-NET with TFC-TDFs shows promising results for the separation of other sources as well as vocals.
- It is possible to reduce parameters without performance degradation by using bottlenecked TDFs.
- Although its SDR performance is high, DCSE with TFC-TDF-U-Net suffers from many artifacts.
- Visualization of artifacts is provided with a simple task called silent mixture waveform separation.

This work is not limited to the U-Net-architecture nor music source separation. Blocks can be used as core components in more complex architectures as well. We can use different types of blocks for a single model, meaning that much space remains for improvement.

The observations give reusable insights when one designs a source separation model. For example, one can use the artifact spectrogram of silent mixture waveform separation (Figure 3.10) to cancel out constant artifacts generated by DCSE-based models. Also, they are reusable when one designs spectrogram-based models for other Music Information Retrieval (MIR) tasks. For instance, one can adopt the TFC-TDF block for music generation since it can model complex frequency correlations such as overtones with the aid of TDFs.

3.8 Conclusion

This section verifies that injecting frequency transformation blocks into a standard U-Net architecture can significantly improve the separation performance. The experiments provide abundant material for future works by comparing several U-Nets with different types of blocks. Also, the proposed models outperform state-of-the-art methods on several MUSDB18 tasks. Empirical justification is given to explain how it works with an ablation study. Future work can extend this model to utilize attention networks to model long-term dependencies observed in both the frequency and the temporal axis. It is also revealed that direct estimation methods might suffer from many artifacts by examining the silent mixture waveform separation task. For future work, one can develop the DCSE method with smaller artifacts.

Chapter 4

Frequency Transformation for Conditioned Source Separation

This chapter aims to extend the TDF block proposed in chapter 3 for dedicated source separation to conditioned source separation. This chapter adopts the Conditioned-U-Net (CUNet) [32] as a baseline for conditioned source separation. Although the prior experiment (see section 4.6) indicates that it improves the performance to insert TDF blocks to a CUNet simply, it does not inherit the spirit of the TDF block, with which a TDF block is trained to capture frequency-to-frequency dependencies of the target source.

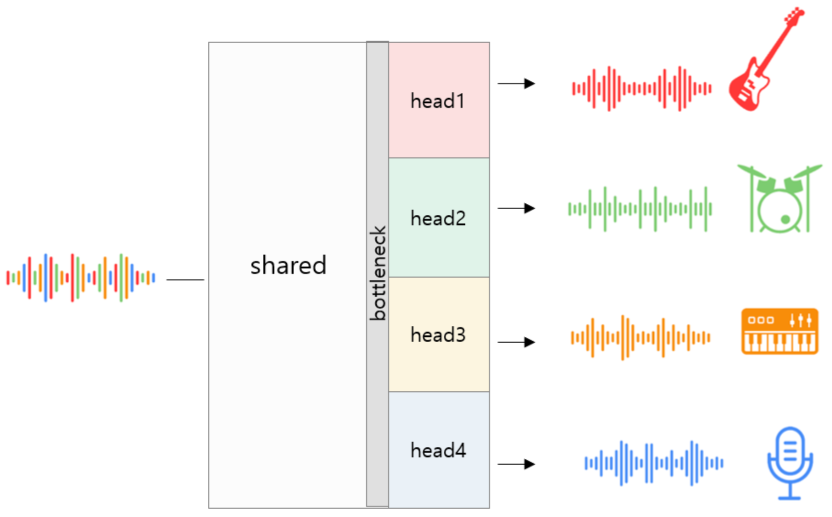
By inheriting the property mentioned above, this chapter proposes the Latent Source-Attentive Frequency Transformation (LaSAFT), a novel frequency transformation block that can capture instrument-dependent frequency patterns by exploiting the scaled dot-product attention [58]. This chapter also proposes the Gated Point-wise Convolutional Modulation (GPoCM), a new modulation that extends the Feature-wise Linear Modulation (FiLM) [39]. The proposed CUNet with LaSAFT and GPoCMs outperforms the existing methods on several MUSDB18 [41] tasks. The ablation study shows that adding LaSAFT or replacing FiLMs with GPoCMs improves separation performance.

The rest of this chapter is organized as follows. Section 4.1 defines the conditioned source separation task and section 4.2 defines a *latent source* which is the key concept used in this thesis. Section 4.3 summarizes the baseline CUNet architecture. Section 4.4 proposes the LaSAFT block and section 4.5 proposes the PoCM mechanism to modulate internal features. The experimental results are summarized in section 4.6.

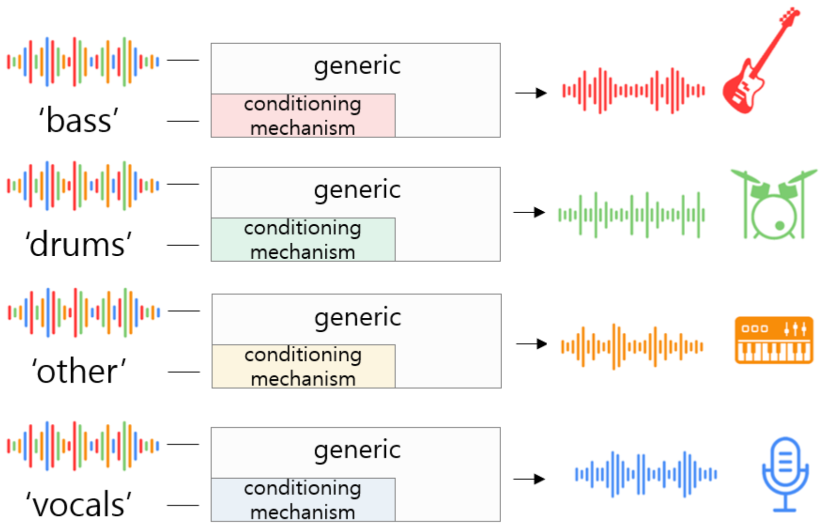
4.1 Conditioned Models for Audio Source Separation

Many state-of-the-art models based on neural networks for audio source separation are dedicated models. However, this approach forces us to train an individual model for each instrument. Moreover, trained models cannot use the commonalities between different instruments. A simple extension to multi-source separation is to generate several outputs at once, as shown in Figure 4.1 (a). For example, the model proposed in [22] generates multiple outputs at once. Although it shows promising results, this approach still has a scaling issue: the number of heads increases as the number of instruments increases, leading to (1) performance degradation caused by the shared bottleneck, (2) inefficient memory usage.

Alternatively, it is possible to adopting conditioning learning [32, 43] for muti-source separation. A conditioned model separates different instruments with the aid of the control mechanism, as illustrated in Figure 4.1 (b). Since it does not need a multi-head output layer, there is no shared bottleneck. For example, the Conditioned-U-Net (CUNet) [32] extends the U-Net [42, 1] by exploiting Feature-wise Linear Modulation (FiLM) [39]. It takes as input the spectrogram of a mixture and a control vector that indicates which instrument a user wants to separate and outputs the estimated spectrogram of the target instrument.



(a) multi-head models



(b) conditioned models

Figure 4.1: Multi Source Separation Task

4.2 Latent Source Analysis for Conditioned Separation

Recent spectrogram-based methods [63] employed Frequency Transformation (FT) blocks to capture frequency patterns as discussed in chapter 3. Capturing frequency-to-frequency dependencies of the target source with FT layers, neural networks with these FT blocks can show outstanding performance on several source separation tasks. For example, chapter 3 shows that it is possible to improve source separation performance by inserting TDFs to a conventional fully 2-d convolutional U-Net, achieving state-of-the-art SDRs on two MUSDB18 [41] tasks, namely, 'vocals' and 'other' separation tasks.

However, it was above their expectation for the authors of Publication I that the U-Net with TFC-TDF blocks performs well in the 'other' separation task. A TDF block is expected to learn frequency patterns of a specific source, but there are many instruments in 'other' source in MUSDB18 [41] including piano, guitar, organ, brass, and synthesizer. This result indicates that a TDF block can also improve the performance of a U-Net by capturing the frequency patterns observed across multiple instruments. Similarly, inserting TDFs into a CUNet also enhances the source separation quality, as discussed in section 4.6.

On top of this observation, the motivation of this work is to use multiple TDFs for multiple *latent sources.* The concept of latent source has been introduced in recent source separation methods [60]. [60] trained their model to separate the given input into a variable number of latent sources, which can be remixed to approximate the original mixture. By carefully taking the weighted sum of separated latent sources, extracting the desired source, such as clean speech, is possible. The model proposed in this chapter also uses the concept of *latent source* for conditioned source separation.

In this chapter, a *latent source* is defined as a concept that deals with a more detailed aspect of acoustic features than a symbolic-level source (e.g., 'vocals'). Humans are used to categorizing symbolic-level classes of sources, but there are many sub-classes in a single class, as mentioned in the example of 'other.' Latent sources are helpful, especially for such complex classes, to which there are various sub-classes belong. On top of this concept, this chapter proposes a novel frequency transformation block for conditioned source separation, assuming that a weighted sum of latent sources can represent a source.

4.3 Baseline Architecture: Conditioned U-Net

The baseline is a generalized CUNet [32] architecture. It consists of (1) the Generic U-Net and (2) the Condition Generator.

1. **The Generic U-Net** is a U-Net [42] which takes a mixture spectrogram as input and outputs the estimated target spectrogram. It applies FiLM layers to modulate intermediate features with condition parameters generated by the condition generator. The
2. **Condition Generator** takes as input a condition vector and generates condition parameters. A condition vector is a one-hot encoding vector that specifies which instrument we want to separate.

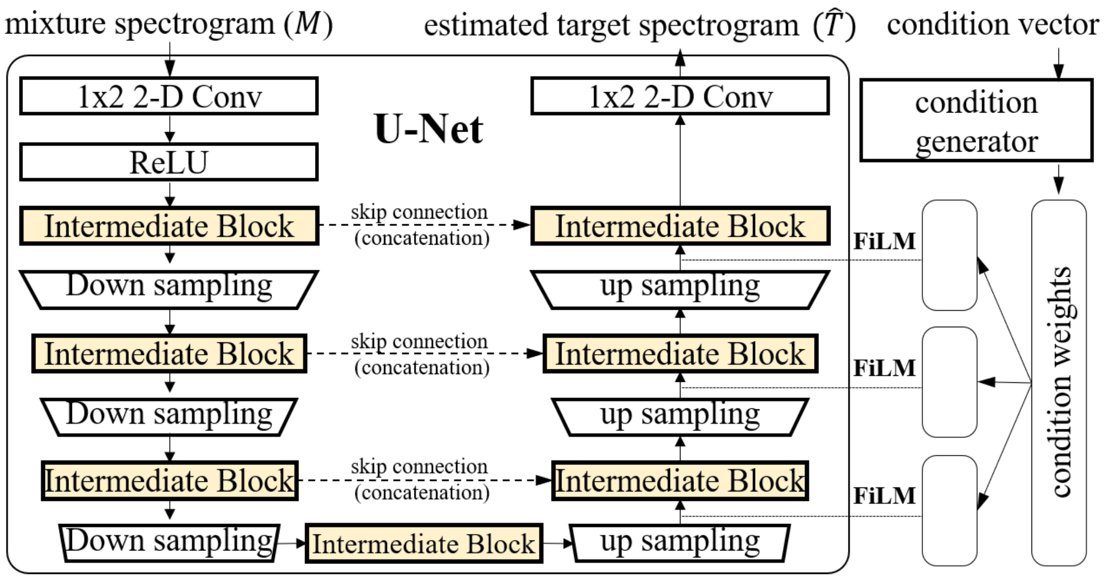


Figure 4.2: The baseline architecture

Subsection 4.3.1 describes the generic U-Net, which is conditioned on external information. Subsection 4.3.2 describes the condition generator in detail.

4.3.1 The Generic U-Net

The baseline architecture adopts the U-Net used in section 3.4 as a backbone structure. As shown in the left part of Figure 4.2, it consists of an encoder and a decoder: the encoder transforms the input mixture spectrogram M into a downsized spectrogram-like representation. The decoder takes it and returns the estimated target spectrogram $\hat{T}$. It should be noted that spectrogram M and $\hat{T}$ are complex-valued adopting the Complex-as-Channel (CaC) separation method described in section 3.3.3. Recall that real and imaginary parts are viewed as separate channels in CaC. Thus, if the original mixture waveform is c-channeled (i.e., $c = 2$ for stereo), then the number of channels of M and $\hat{T}$ is $(2c)$.

There are four components in the structure: 1 X 2 Convolution Layers, Intermediate blocks, Down/Up-sampling layers, and skip connections. *1 X 2 Convolution Layers* are used for adjusting the number of channels. Since the target spectrogram is complex-valued, there is no activation function. It is inherited from the U-Net of 3.4. *An Intermediate Block* transforms an input spectrogram-like tensor into an equally-sized tensor. For each block in the baseline, the baseline uses a Time-Frequency Convolution (TFC) described in 3.4, a block of densely connected 2-D convolution layers [16]. L denotes the number of intermediate blocks in the encoder. The decoder also has L blocks. There is an additional block between them. *A Down/Up-sampling Layer* halves/doubles the scale of an input tensor, which is implemented with a strided/transposed-convolution. *Skip Connections* concatenate output feature maps of the same scale between the encoder and the decoder. They help the U-Net recover fine-grained details of the target.

The only different part between this U-Net and that of section 3.4 is that this U-Net has FiLM layers in the decoder for condition-aware modulation. Unlike in the U-Net baseline of the previous section, it modulates internal features in the decoder by applying FiLM layers, as shown in the right part in Figure 4.2. Applying a FiLM is an effective way to condition a network, which applies the following operation to intermediate feature maps.

A Film layer is defined as follows:

$$FiLM(X_c^i|\gamma_c^i, \beta_c^i) = \gamma_c^i \cdot X_c^i + \beta_c^i \tag{4.1}$$

where γ_c^i and β_c^i are parameters generated by the condition generator, and X^i is the output of the i^{th} decoder's intermediate block, whose subscript refers to the c^{th} channel of X as shown in Figure 4.3.

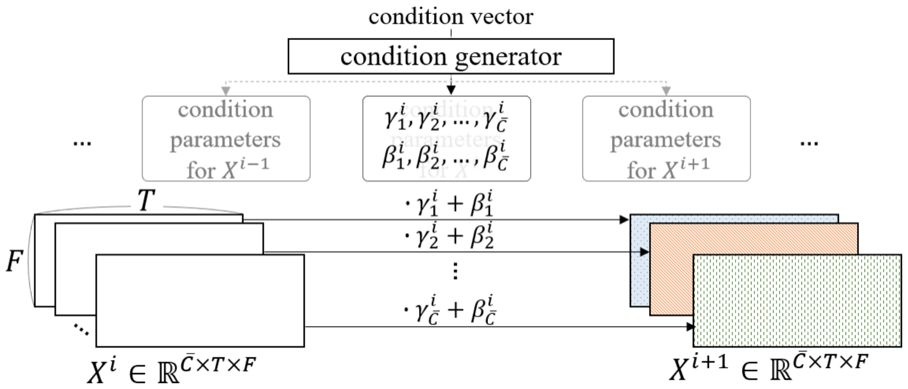
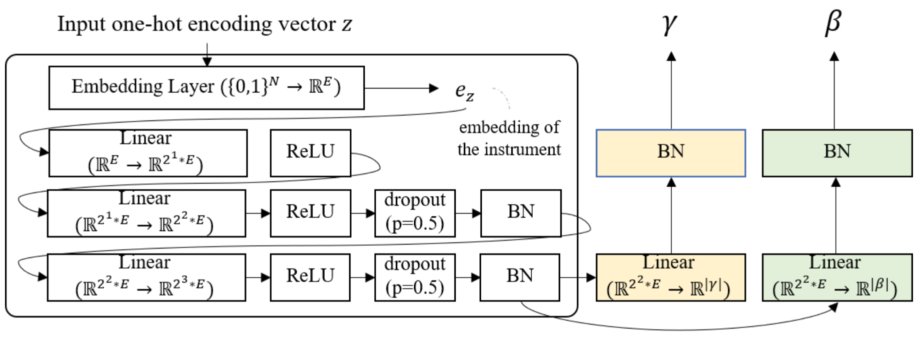

condition vector
condition generator
condition parameters for X^{i-1}
$\gamma_1^i, \gamma_2^i, ..., \gamma_{\bar{C}}^i$
$\beta_1^i, \beta_2^i, ..., \beta_{\bar{C}}^i$
condition parameters for X^{i+1}
T
F
$\cdot \gamma_1^i + \beta_1^i$
$\cdot \gamma_2^i + \beta_2^i$
$\cdot \gamma_{\bar{C}}^i + \beta_{\bar{C}}^i$
$X^i \in \mathbb{R}^{\bar{C} \times T \times F}$
$X^{i+1} \in \mathbb{R}^{\bar{C} \times T \times F}$

Figure 4.3: FiLM layers

4.3.2 The Condition Generator

The condition generator is a network that predicts condition parameters $\gamma = (\gamma_1^1, ..., \gamma_{\bar{C}}^L)$ and $\beta = (\beta_1^1, ..., \beta_{\bar{C}}^L)$. The proposed condition generator is similar to 'Fully-Connected Embedding' of the CUNet [32] except for the usage of the embedding layer. It takes as input the one-hot encoding vector $z \in \{0, 1\}^{|\mathcal{I}|}$ that specifies which one we want to separate among $|\mathcal{I}|$ instruments. The condition generator projects z into $e_z \in \mathbb{R}^E$, the embedding of the target instrument, where E is the dimension of the embedding space.

It then applies a series of fully connected (i.e., linear or dense) layers, which doubles the dimension. We use ReLU [12] as the activation function for each layer and apply a dropout with $p = 0.5$ followed by a Batch Normalization (BN) [17] to the output of each last two layers. The last hidden units are fed to two fully-connected layers to predict $\gamma = (\gamma_1^1, ..., \gamma_{\bar{C}}^L) \in \mathbb{R}^{|\gamma|}$ and $\beta = (\beta_1^1, ..., \beta_{\bar{C}}^L) \in \mathbb{R}^{|\beta|}$, where $|\gamma| = |\beta| = \bar{C}L$.



Input one-hot encoding vector z
Embedding Layer ($\{0,1\}^N \to \mathbb{R}^E$)
e_z
embedding of the instrument
Linear ($\mathbb{R}^E \to \mathbb{R}^{2^1*E}$)
ReLU
Linear ($\mathbb{R}^{2^1*E} \to \mathbb{R}^{2^2*E}$)
ReLU
dropout (p=0.5)
BN
Linear ($\mathbb{R}^{2^2*E} \to \mathbb{R}^{2^3*E}$)
ReLU
dropout (p=0.5)
BN
Linear ($\mathbb{R}^{2^2*E} \to \mathbb{R}^{|\gamma|}$)
Linear ($\mathbb{R}^{2^2*E} \to \mathbb{R}^{|\beta|}$)
BN
BN
γ
β

Figure 4.4: The condition generator

4.4 Frequency Transformation for Conditioned Separation

In this section, naive approaches are introduced to extend the frequency transformation block to conditioned source separation. Then, the Latent Source Attentive Frequency Transformation (LaSAFT) method is proposed.

4.4.1 Naive Approaches

Injecting TDFs to the baseline is a very straightforward approach to extend the TDF block to conditioned source separation. However, it does improve the SDR performance by capturing the common frequency patterns observed across all instruments as mentioned before (see section 4.6).

An alternative approach is to use multiple instances with if-then-else branches to control data flows. For example, suppose that there are four sources users want to separate. Then, the CUNet with this naive approach creates four instances of TDFs in each intermediate block. Assuming each TDF corresponds to a specific source and the data flow is controllable with the if-then-else branch, it is trivial to implement a naive extension of frequency transformation to conditioned source separation.

However, these approaches do not inherit the spirit of TDF. The following subsection proposes a delicate frequency transformation method for conditioned source separation.

4.4.2 LaSAFT: Latent Source-attentive Frequency Transformation Blocks

This subsection proposes the Latent Source Attentive Frequency Transformation (LaSAFT) by adopting the scaled dot-product attention mechanism [58]. Compared to the structure of the existing TDF shown in Figure 4.5, a LaSAFT block has a complex structure, as illustrated in Figure 4.6. Figure 4.6 describes the key idea of LaSAFT, which is to use multiple TDF blocks to analyze various latent sources and attentively aggregate their results concerning the given query.

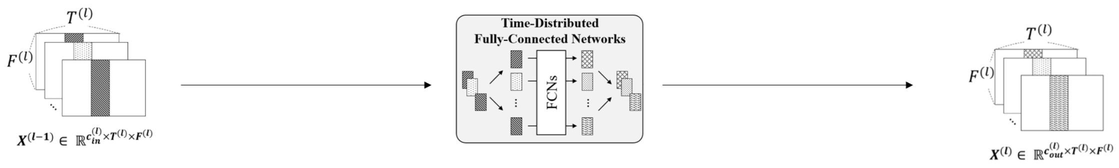

$T^{(l)}$
$F^{(l)}$
$X^{(l-1)} \in \mathbb{R}^{c_{in}^{(l)} \times T^{(l)} \times F^{(l)}}$
Time-Distributed Fully-Connected Networks
FCNs
$T^{(l)}$
$F^{(l)}$
$X^{(l)} \in \mathbb{R}^{c_{out}^{(l)} \times T^{(l)} \times F^{(l)}}$

Figure 4.5: TDF: Time Distributed Fully Connected Layers

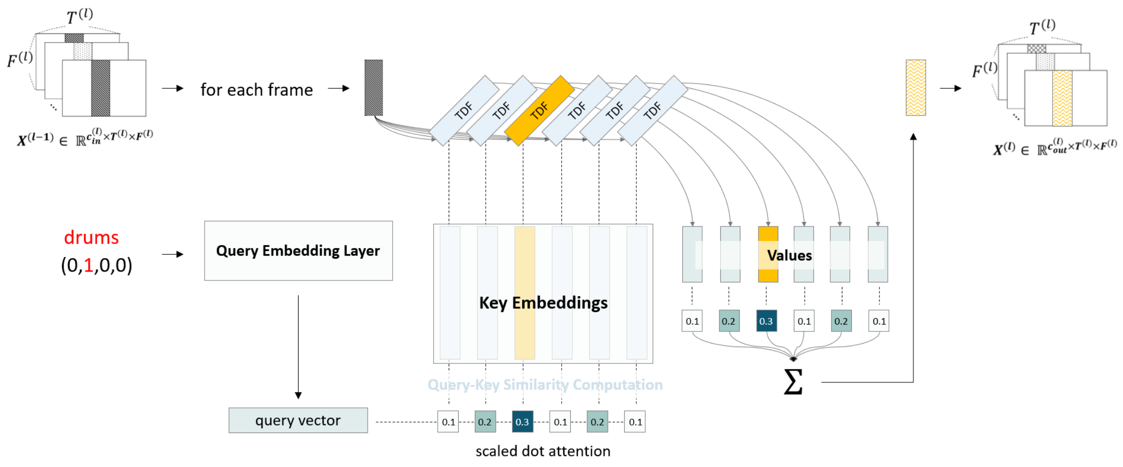


Figure 4.6: Conceptual view of a LaSAFT block

Technically, LaSAFT first duplicates $|\mathcal{I}_L|$ copies of the second layer of the TDF, as shown in the right side of Figure 4.7, where $|\mathcal{I}_L|$ refers to the number of *latent instruments*. $|\mathcal{I}_L|$ is not necessarily the same as $\mathcal{I}$ for the sake of flexibility. For the given frame $V \in \mathbb{R}^F$, it obtains the $|\mathcal{I}_L|$ latent instrument-dependent frequency-to-frequency correlations, denoted by $V' \in \mathbb{R}^{F \times |\mathcal{I}_L|}$. It uses components on the left side of Figure 4.7

to determine how much each *latent source* should be attended. LaSAFT takes as input the instrument embedding $z_e \in \mathbb{R}^{1\times E}$. It has a learnable weight matrix $K \in \mathbb{R}^{|\mathcal{I}_L|\times d_k}$, where the dimension of each instrument's hidden representation is denoted by d_k. By applying a linear layer of size d_k to z_e, it obtain $Q \in \mathbb{R}^{d_k}$. It finally computes the output of the LaSAFT as follows:

$$Attention(Q, K, V') = softmax(\frac{QK^T}{\sqrt{d_k}})V' \tag{4.2}$$

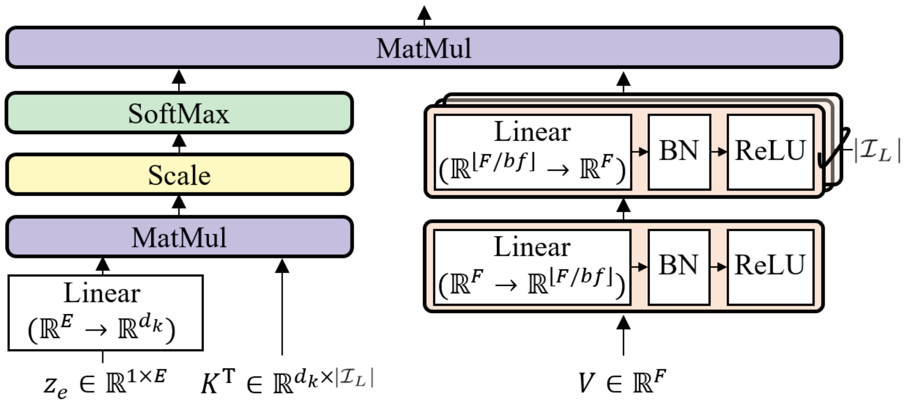

MatMul
SoftMax
Scale
MatMul
Linear
$(\mathbb{R}^E \to \mathbb{R}^{d_k})$
$z_e \in \mathbb{R}^{1\times E}$
$K^\mathsf{T} \in \mathbb{R}^{d_k \times |\mathcal{I}_L|}$
Linear
$(\mathbb{R}^{\lfloor F/bf \rfloor} \to \mathbb{R}^F)$
BN
ReLU
$|\mathcal{I}_L|$
Linear
$(\mathbb{R}^F \to \mathbb{R}^{\lfloor F/bf \rfloor})$
BN
ReLU
$V \in \mathbb{R}^F$

Figure 4.7: Latent Source Attentive Frequency Transformation

A LaSAFT is applied after each TFC in the encoder and after each Film/GPoCM layer in the decoder. There is another skip connection for the final output of each block (i.e., it outputs $X' + lasaft(X')$, where X' is an input of the $lasaft$).

4.5 Feature Modulation

The conventional CUNet [32] uses FiLM layers to modulate internal features for the given condition. FiLM operations used in [32] are efficient and straightforward to implement but might be limited in terms of expressive power because they do not have inter-channel operations. This section proposes novel feature modulation methods called PoCM and GPoCM, which have inter-channel computation.

Subsection 4.5.1 proposes PoCM and subsection 4.5.2 introduced the gated version of PoCM, called GPoCM.

4.5.1 PoCM: Point-wise Convolutional Modulation

The PoCM is an extension of FiLM. While FiLM does not have inter-channel operations, PoCM has them as follows:

$$PoCM(X_c^i|\omega_c^i, \beta_c^i) = \beta_c^i + \sum_j \omega_{cj}^i \cdot X_j^i \tag{4.3}$$

where $\omega_c^i = (\omega_{c1}^i ..., \omega_{c\bar{C}}^i)$ and β_c^i are condition parameters, and X^i is the output of the i^{th} decoder's intermediate block, as shown in Figure 4.8. Since this channel-wise linear combination can also be viewed as a point-wise convolution, it is named *PoCM*. With inter-channel operations, PoCM can modulate features more flexibly and expressively than FiLM.

4.5.2 GPoCM: Gated Point-wise Convolutional Modulation

In the decoder, the following 'Gated PoCM (GPoCM)' is used instead of PoCM. Although substituting each FiLM with a raw PoCM improves the performance, GPoCMs also worked and were more robust than PoCM because the output generated by GPoCM is bounded. Also, GPoCM can easily make the output near zero-valued, which is natural and intuitive in source separation tasks where near-zero internal features are expected to generate near-silent output signals.

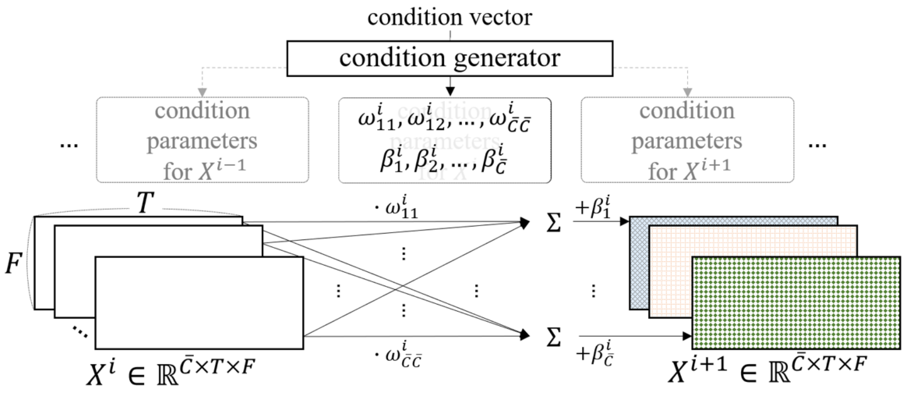

condition vector
condition generator
condition parameters for X^{i-1}
$\omega_{11}^i, \omega_{12}^i, \dots, \omega_{\bar{C}\bar{C}}^i$
$\beta_1^i, \beta_2^i, \dots, \beta_{\bar{C}}^i$
condition parameters for X^{i+1}
T
F
$\cdot\,\omega_{11}^i$
$\cdot\,\omega_{\bar{C}\bar{C}}^i$
Σ
$+\beta_1^i$
$+\beta_{\bar{C}}^i$
$X^i \in \mathbb{R}^{\bar{C}\times T\times F}$
$X^{i+1} \in \mathbb{R}^{\bar{C}\times T\times F}$

Figure 4.8: PoCM layers

GPoCm is defined as follows:

$$GPoCM(X_c^i|\omega_c^i, \beta_c^i) = \sigma(PoCM(X_c^i|\omega_c^i, \beta_c^i)) \odot X_c^i \tag{4.4}$$

where σ is a sigmoid and $\odot$ means the Hadamard product.

4.6 Experiments

This section validates the architectural choices with an ablation study. It also compares the performance of the proposed model with other state-of-the-art conditioned source separation models.

4.6.1 Experiment Setup

The same dataset was used for the experiment as section 3.6, MUSDB18 dataset [41]. It contains 86 tracks for training, 14 tracks for validation, and 50 tracks for the test. Each track is stereo, sampled at 44100 Hz, consisting of the mixture and four sources (i.e., $|\mathcal{I}|=4$): 'vocals,' 'drums,' 'bass,' and 'other.'

Models were trained using Adam [18] with learning rate $lr \in \{0.0005, 0.001\}$ depending on model depth. Each model is trained to minimize the mean squared error between the ground-truth STFT output and the estimated. For validation, the mean absolute error of the target signal and the estimated was measured again. Data augmentation [57] was applied on the fly to obtain mixture clips comprised of sources from different tracks.

For the evaluation metric, Source-to-Distortion Ratio (SDR) [59] was used by using the official tool[1] for MUSDB18. The median SDR value over all the test set tracks for each run is obtained, and the mean SDR over three runs was reported in tables. More details are available online[2].

4.6.2 Results

An ablation study is provided in this subsection to validate the effectiveness of the proposed methods compared with the baseline. In every model in Table 4.1, the configuration of the STFT parameters is as follows: FFT window size of 2048 and hop size of 1024 for spectrogram estimation. The configuration of the baseline (FiLM CUNET) is as follows: we use 7-blocked CUNet (i.e., $L = 3$), and we use a TFC for each block with the same configuration used in Publication I, where 5 convolution layers with kernel size

[1]https://github.com/sigsep/sigsep-mus-eval
[2]https://github.com/ws-choi/Conditioned-Source-Separation-LaSAFT

3×3 are densely connected, and the growth rate [16] is set to be 24. $\bar{C}$ is set to be 24 as in Publication I. The dimensionality of the embedding space of conditions (i.e., $\mathbb{R}^E$) is set to be 32. $|\mathcal{I}_L|$, the number of latent instruments is set to 6.

In Table 4.1, it is observable that a considerable performance degradation when employing the existing method (FiLM CUNet), compared to the *dedicated* U-Net, which is trained separately for each instrument with the same configuration. Injecting TDF blocks to the baseline (FiLM CUNet + TDF) improves SDR by capturing the common frequency patterns. Replacing TDFs with LaSAFTs (FiLM CUNet + LaSAFT) significantly improves the average SDR score by 0.51dB, indicating that the proposed LaSAFTs are more appropriate for multi-instrument tasks than TDFs. The proposed model (GPoCM CUNet + LaSAFT) outperforms the others, achieving comparable but slightly inferior results to dedicated models.

model	vocals	drums	bass	other	AVG
dedicated (chapter 3)	7.07	5.38	5.62	4.61	5.66
FiLM CUNet	5.14	5.25	4.20	3.40	4.49
+ TDF	5.88	5.70	4.55	3.67	4.95
+ LaSAFT	6.74	5.64	5.13	4.32	5.46
GPoCM CUNet	5.43	5.30	4.43	3.51	4.67
+ TDF	5.94	5.46	4.47	3.81	4.92
+ LaSAFT	**6.96**	**5.84**	**5.24**	**4.54**	**5.64**

Table 4.1: An ablation study: *dedicated* means U-Nets for the single source separation, trained separately. FiLM CUNet refers the baseline in section 4.3. The last row is our proposed model.

Also, a comparison against existing state-of-the-art models on the MUSDB18 benchmark is provided in Table 4.2 The first seven rows of Table 4.2 show SDRs of state-of-the-art models, which are not conditioned on an external condition. The SDR performance of each non-proposed method is taken from the respective papers.

The eighth row describes the results of the dedicated U-Nets with nine TDF-TDF blocks based on DCSE, introduced in chapter 3. For fair comparison, the proposed 'CUNet with LaSAFT + GPoCM' with the same frequency resolution as the other state-of-the-art models [22, 55] (FFT window size = 4096) was used. Similar to the results in Table 4.1, the proposed model was slightly inferior to its dedicated counterpart. As shown in the last row of Table 4.2, the proposed model yields comparable results against the existing methods and even outperforms the others (except for the dedicated counterparts) on 'vocals' and 'other.' Notably, the proposed usually performs better than the other conditioned source separation model called Meta-TasNet [43].

The original CUNet with a window size of 4096 were omitted in this experiment because they were expected to show inferior performance to the proposed. The proposed model with a window size of 2048 significantly outperformed the generalized CUNet as shown in Table 4.1 and the generalized model usually performed better than the original.

LaSAFT's ability, which allows the proposed model to extract latent instrument-attentive frequency patterns, significantly improves the SDR of 'other' since it contains various instruments such as piano and guitars. Also, existing works [63] and Publication I have shown that FT-based methods are beneficial for voice separation, which explains our model's excellent SDR performance on vocals.

model	conditioned?	vocals	drums	bass	other	AVG
Demucs [6]*	✗	6.84	6.86	**7.01**	4.42	**6.28**
Conv-Tasnet [6, 26]*	✗	6.43	6.02	6.20	4.27	5.73
UMX [51]	✗	6.32	5.73	5.23	4.02	5.32
DGRU-DGConv [22]	✗	6.85	5.85	4.86	4.65	5.55
MMDenseLSTM [53]	✗	6.60	6.43	5.16	4.15	5.59
D3Net [55]	✗	7.24	**7.01**	5.25	4.53	6.01
Nachmani[34]*	✗	6.92	6.15	5.88	4.32	5.82
dedicated (chapter 3)	✗	**7.98**	6.11	5.94	**5.02**	6.26
Meta-TasNet[43]*	✓	6.40	**5.91**	5.58	4.19	5.52
LaSAFT+GPoCM (*proposed*)	✓	**7.33**	5.68	**5.63**	**4.87**	**5.88**

Table 4.2: A comparison SDR performance of our models with other systems. '$*$' denotes model operating in time domain.

4.7 Discussion

This section first summarizes the differences between the original CUNet [32], our baseline, and the proposed model. Then, it derives reusable insights which can be adopted in other domain as well.

The original CUNet[32] is the most relevant work. It was extended in this section by employing LaSAFT and GPoCM. Besides, the baseline used in the section is also different from it. The differences between them are summarized as follows: (1) the proposed model's U-Net is based on a generalized U-Net for source separation used in section 3.4, but the U-Net of [32] is more similar to the original U-Net [42], and (2) the proposed architecture applies FiLM/GPoCM to internal features in the decoder, but [32] applies it in the encoder. The authors of [32] tried to manipulate latent space in the encoder, assuming the decoder can perform as a general spectrogram generator, which is 'shared' by different sources. However, it was founded that this approach is not practical since it makes the latent space (i.e., the decoder's input feature space) more discontinuous. It was observed that applying FiLMs in the decoder was consistently better than applying FiLMs in the encoder.

For multi-source separation, [43] employed meta-learning, which is similar to conditioning learning. It also has external networks that generate parameters for the target instrument. Whiling the proposed focus on modulating internal representations with GPoCM, [43] focuses on generating parameters of the masking subnetwork. Also, models proposed in [43, 34] operate in the time domain, while ours in the time-frequency domain.

While other multi-source separation methods [22, 55] estimate multiple sources simultaneously, the proposed method tries to condition a shared U-Net. It is expected that the proposed can be easily extended to more complicated tasks such as audio manipulation.

4.8 Conclusion

This chapter proposes LaSAFT that captures source-dependent frequency patterns by extending TDF to fit the multi-source task. This section also proposes GPoCM that modulates features more flexibly and expressively than FiLM. The experimental results indicate that employing our LaSAFT and GPoCM in CUNet can significantly improve SDR performance. Possible future works include the following topics; reducing the number of parameters and the memory usage of LaSAFT to consider more latent instruments and extending the proposed architecture to audio manipulation tasks, where we can condition the model by providing various instructions.

Chapter 5

Latent Source Analysis for Audio Manipulation on Selective Sources

This chapter addresses a novel problem called Audio Manipulation on Specific Sources (AMSS). Section 5.1 formally defines AMSS and Section 5.2 introduces a structured query language for AMSS. Designing a neural network for AMSS is challenging because the sources of a given mixture track are usually not observable. This chapter proposes a neural network called AMSS-Net that extracts a feature map containing *latent sources* from the given mixture audio and selectively manipulates them while preserving irrelevant latent sources. Section 5.4 describes the AMSS-Net architecture. Another challenge is that existing datasets cannot be directly used for supervised training AMSS-Net. To this end, section 5.3 proposes a training framework for AMSS that uses a 'source observable multi-track dataset' such as MUSDB18 [41]. Audio transformations are applied onto specific sources of a given multi-track using general methods from Digital Signal Processing (DSP) libraries to generate an AMSS triple on the fly. Section 5.5 summarizes and discusses the various experimental results. Section 5.6 discusses various aspects including pros and cons of the AMSS-Net, and section 5.7 concludes this chapter.

5.1 Audio Manipulation on Specified Sources

In recent days, social media applications have attracted many users to create, edit, and share their audio, audio-visual, or other types of multimedia content. However, it is usually hard for non-experts to manipulate them, especially when they want to edit only the desired objects. Fortunately, for image manipulation, recently proposed methods enable non-expert users to edit the desired objects while leaving other contents intact. These machine learned-based methods can decrease human labor for image editing and let non-experts manipulate their image without abundant knowledge of tools that are usually complicated to use, as mentioned in chapter 1.

On the other hand, little attention has been given to machine learning methods for automatic audio editing. It is challenging to edit specific sound objects (e.g., decrease the volume of *cicada buzzing noise*) in the given audio.

Although many machine learning approaches have been proposed for audio processing [28, 29, 27, 48, 61, 33, 1, 32, 60, 43], to the best of knowledge, there is no existing method that can directly address AMSS. This chapter proposes a novel end-to-end neural network that performs AMSS according to the given textual query. Designing a neural network for AMSS is straightforward if the sources of a given mixture track are observable. However, this section assumes that they are not observable because most audio data does not provide them in general. In the assumed environment, modeling AMSS is very challenging because a sound object (e.g., a sample in a wave, a frequency bin in a spectrogram) is 'transparent'([62]); a pixel in an image usually corresponds to only a single visual object, whereas a sound object carries information of multiple sources, as shown in Figure 1.3. Thus, different approaches are required for AMSS from the existing image manipulation techniques.

Considering that audio editing usually requires expert knowledge of audio engineering or signal processing, this chapter explores a deep learning approach in conjunction with textual queries to lessen audio editing difficulty. Specifically, this chapter addresses a novel task named Audio Manipulation on Specific Sources (AMSS), which aims to edit only desired objects that correspond to specific sources, such as vocals and drums, according to a given description while preserving the content of sources that are not mentioned in the description. AMSS can be used for many applications such as video creation tools making audio editing easy for non-experts. For example, users can decrease the volume of drums by typing simple textual instructions instead of time-consuming interactions with digital audio workstations. Before modeling AMSS with deep neural networks, this section gives a formal definition of AMSS as follows:

Definition 1. *Audio Manipulation on Specified Sources: for a given audio track A and a given description S, AMSS aims to generate a manipulated audio track A' that semantically matches S while preserving contents in A that are not described in S. Assume that A contains multiple sources, and S describes the desired audio transformation and the targets, which we want to manipulate. Assume that S can be represented as a one-hot encoding or a textual query. In this thesis, we assume that S is a textual query written in the Audio Manipulation Language described in section 5.2.*

Although several machine learning methods have been proposed for audio processing [28, 29, 27, 48, 61, 33, 1, 32, 60, 43], to the best of knowledge, there is no existing method that can directly address AMSS. This chapter proposes a novel end-to-end neural network that performs AMSS according to the given textual query.

Designing a neural network for AMSS is straightforward if the sources of a given mixture track are observable. However, this dissertation assumes that they are not observable because most audio data does not provide them in general. In the assumed environment, modeling AMSS is very challenging because a sound object is transparent, as discussed in chapter 1 and shown in Figure 1.3. Thus, unconventional approaches are required for AMSS from the existing image manipulation techniques. To address this challenge, section 5.4 proposes a neural network called AMSS-Net that extracts a feature map containing *latent sources*, proposed as key concepts in the previous chapter, from the given mixture audio and selectively manipulates them while preserving irrelevant latent sources.

It is also challenging that existing datasets cannot be directly used for supervised training AMSS-Net. If a training dataset of triples $\{(A^{(i)}, A'^{(i)}, S^{(i)})\}_{i=1}^{N}$, where $S^{(i)}$ is an AMSS description, $A^{(i)}$ is a mixture, and $A'^{(i)}$ is the manipulated audio according to $S^{(i)}$ is provided, a neural network *net* can be trained in a supervised manner by minimizing $\sum_{i=1}^{N} loss(net(A^{(i)}, S^{(i)}), A'^{(i)})$, where *loss* is a distance metric such as L_2. Unfortunately, there were no datasets currently available that directly target AMSS. To solve this issue, section 5.3 proposes a novel training framework for AMSS that uses a 'source observable multi-track dataset' such as MUSDB18 [41]. Audio transformations are applied onto specific sources of a given multi-track using general methods from Digital Signal Processing (DSP) libraries to generate an AMSS triple on the fly.

This chapter finally verifies that it is possible to train a neural network for AMSS. As a proof-of-concept, this chapter focuses on modifying specified sources' sonic characteristics (e.g., loudness, panning, frequency content, and dereverberation). More complex manipulations such as distortion are not addressed in this thesis. As discussed in section 5.6, more complex manipulations need more sophisticated architectures than the proposed.

Throughout the rest of the paper, we define an *AMSS task* to be a set of instructions dealing with the same manipulation method. Table 5.1 lists nine AMSS tasks modeled in this chapter. Note that an *AMSS task class* is defined as a set of similar AMSS tasks.

class	task	DSP operations
volume control	separate	masking the others
	mute	masking targets
	increase vol	re-scaling (increase)
	decrease vol	re-scaling (decrease)
volume control (multi channel)	pan left	re-scaling (left $>$ mean $>$ right)
	pan right	re-scaling (left $<$ mean $<$ right)
filter	lowpasss	Low-pass Filter
	highpass	High-pass Filter
delay	dereverb	reverb*

Table 5.1: List of AMSS tasks modeled in this paper: $(*)$ denotes reversed generation process (the line 5 in Algorithm 1)

5.2 Audio Manipulation Language

As mentioned earlier, this chapter assumes that S is given as a textual query, such as "apply light lowpass to drums". This assumption is plausible because textual querying enables us to naturally represent any pair of a transformation function and its target sources with particular options. For example, one can control the degree of audio effects (which corresponds to the parameter settings of DSP functions) by simply inserting adjectives such as *light*, *medium*, or *heavy* into the query. It also can provide easy extensibility to natural language interfaces, which will be addressed in future works.

Accordingly, this section proposes an Audio Manipulation Language based on a probabilistic Context-Free Grammar (CFG) [5] for AMSS. For the sake of simplicity, a subset of production rules (i.e., Rules (5.1a)-(5.1f)) that define the query language's syntax for the *filter class* were presented. The Full CFG is given in Appendix 7.1. It is also available online[1].

$$< desc > \rightarrow < cls_f > \tag{5.1a}$$

$$< cls_f > \rightarrow \textbf{\textit{apply}} \;\; < opt - filter > \;\; \textbf{\textit{to}} \;\; < srcs > \tag{5.1b}$$

$$< opt - filter > \rightarrow \; < opt > < filter > \; | < filter > \tag{5.1c}$$

$$< opt > \rightarrow \textbf{\textit{light}} \; |\textbf{\textit{medium}} \; |\textbf{\textit{heavy}} \tag{5.1d}$$

$$< filter > \rightarrow \textbf{\textit{lowpass}} \mid \textbf{\textit{highpass}} \tag{5.1e}$$

$$\begin{aligned} < srcs > \rightarrow\, & \textbf{\textit{vocals}} \mid \textbf{\textit{drums}} \mid \textbf{\textit{bass}} \\ & \mid \textbf{\textit{vocals, drums}} \mid \textbf{\textit{vocals, bass}} \mid \textbf{\textit{drums, vocals}} \\ & \mid \textbf{\textit{drums, bass}} \mid \textbf{\textit{bass, vocals}} \mid \textbf{\textit{bass, drums}} \\ & \mid \textbf{\textit{vocals, drums, bass}} \mid \textbf{\textit{vocals, bass, drums}} \\ & \mid \textbf{\textit{drums, vocals, bass}} \mid \textbf{\textit{drums, bass, vocals}} \\ & \mid \textbf{\textit{bass, vocals, drums}} \mid \textbf{\textit{bass, drums, vocals}} \end{aligned} \tag{5.1f}$$

[1]https://kuielab.github.io/AMSS-Net/aml.html

In the above rules, bold strings are terminal symbols, and strings enclosed in angle brackets are non-terminal symbols. Each rule is of the form $A \rightarrow \alpha|\beta|...$, which means that A can be replaced with α or β. In a CFG, a rule is applied to replace a single non-terminal symbol with one of the expressions. Starting from the first symbol $< desc >$, it is able to generate a *valid* query string by recursively applying rules until there is no non-terminal symbol.

For example, "***apply medium lowpass to vocals, drums***" can derived from $<$ $desc >$ as follows:

$$
\begin{aligned}
< desc > &\rightarrow < cls_f > && \text{(5.2a)}\\
&\rightarrow \boldsymbol{apply}\ \ < opt - filter >\ \ \boldsymbol{to}\ \ < srcs > && \text{(5.2b)}\\
&\rightarrow \boldsymbol{apply}\ \ < opt >< filter >\ \ \boldsymbol{to}\ \ < srcs > && \text{(5.2c)}\\
&\rightarrow \boldsymbol{apply\ medium}\ \ < filter >\ \ \boldsymbol{to}\ \ < srcs > && \text{(5.2d)}\\
&\rightarrow \boldsymbol{apply\ medium\ lowpass\ to}\ \ < srcs > && \text{(5.2e)}\\
&\rightarrow \boldsymbol{apply\ medium\ lowpass\ to\ vocals,\ drums} && \text{(5.2f)}
\end{aligned}
$$

It can be also producible; "***apply lowpass to vocals, drums***" if we choose $< filter >$ instead of $< opt >< filter >$. Since default option for *lowpass* level is set to be *medium*, those two queries have the same meaning.

Notably, rules (5.1b)-(5.1e) are dependant on a AMSS task class, and Rule (5.1f) is dependant on a given multi-track audio. In this thesis, four AMSS task classes are used as shown in Table 5.1. Since the experiment uses MUSDB18[41] dataset of which track contains three named instruments (i.e., vocals, drums, bass), the right-hand side of Rule (5.1f) is set to have all the possible permutations (15 expressions in total).

5.3 Training Framework for AMSS

This section proposes a novel training framework that uses a multi-track dataset. It generates an AMSS triple (A, A', S) on the fly by applying DSP library transformations or audio effects onto target sources of a given multi-track audio file. For instance, suppose that a randomly generated query string S is given as 'apply lowpass to drums' using the CFG and a multi-track that consists of three sources, namely, a vocal track a_0, a bass track a_1, and a drum track a_2. The proposed training framework takes the linear sum (i.e., $A = \sum a_j$) to generate corresponding input A. For the target audio A', it computes $\sum_{j \notin \tau} a_j + \sum_{j \in \tau} f(a_j)$, where τ and f is the set of target sources and DSP function described in S, respectively. The framework applies a Low-pass Filter (LPF) to a_2, and takes the sum for A' as follows: $A' = a_0 + a_1 + LPF(a_2)$. By doing so, it can generate an AMSS triple on the fly for a given description S.

Audio restoration tasks such as dereverberation requires swapping A and A' because the goal is to remove effects. For example, the framework applies reverb to a_2, takes the sum for $A' = a_0 + a_1 + reverb(a_2)$, and returns (A', A, S) instead of (A, A', S) for the description "remove reverb from drums."

Algorithm 1 AMSS Training Triple Generation

```
Input: multi-track $\{a_j\}_{j=1}^n$, a set $\mathcal{G}$ of triple generators
Output: a triple $(A, A', S)$
1: randomly sample a generator $g$ from $\mathcal{G}$
2: $S, f, \tau \leftarrow g.gen\_random\_generate()$
3: $A \leftarrow \sum_{j=1}^n a_j$
4: $A' \leftarrow \sum_{j \notin \tau} a_j + \sum_{j \in \tau} f(a_j)$
5: if g is for removing effect then
6:     $(A, A', S) \leftarrow (A', A, S)$
7: end if
8: return $(A', A, S)$
```

The training framework has a set $\mathcal{G}$ of triple generators. A generator $g \in \mathcal{G}$ has a subset of CFG for text query generation and the corresponding DSP function f, which is used for computing A'. g also has an indicator that describes whether g is for applying or removing the effect. For a multi-track $\{A_j\}_{j=1}^n$ and a set $\mathcal{G}$ of triple generators, the framework generates an AMSS triple by using Algorithm 1, where n denotes the number of sources. For example, if a random generated query is “apply low-pass to drums,” then the corresponding triple is generated as shown in Figure 5.1.

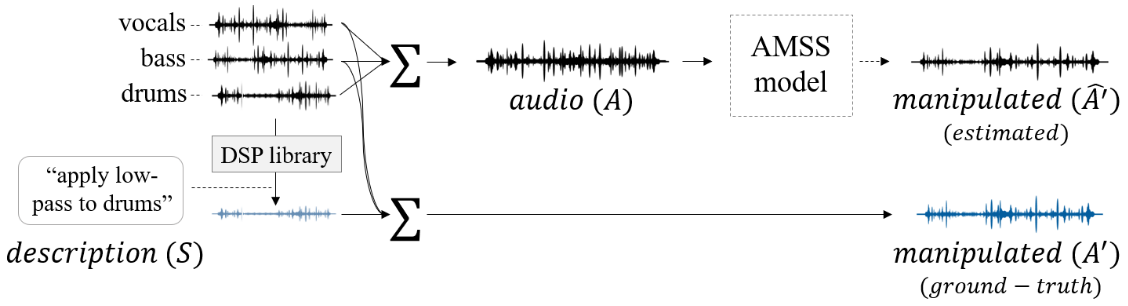


Figure 5.1: An example of AMSS Training Triple Generation Processes

5.4 AMSS-Net Architecture

The AMSS-Net takes an audio track A with a text description S as input and outputs the manipulated audio track $\hat{A}'$ as shown in Figure 5.2. It consists of two sub-networks, i.e., a Description Encoder E_{desc} and a Spectrogram Encoder-Decoder Network $SEDN$. E_{desc} extracts word features $w \in \mathbb{R}^{L \times E}$ from S, where E denotes the dimension of the word features and L denotes the number of words, to analyze the meaning of S. $SEDN$ takes as input word features w and the complex-valued spectrogram A_{spec} of the input audio A. Conditioned on the word feature w, it estimates the complex-valued spectrogram $\hat{A}'_{spec}$, from which $\hat{A}'$ can be reconstructed using iSTFT. An AMSS-Net is trained by minimizing the L_2 loss between the ground-truth spectrogram A'_{spec} of A' and the estimated $\hat{A}'_{spec}$.

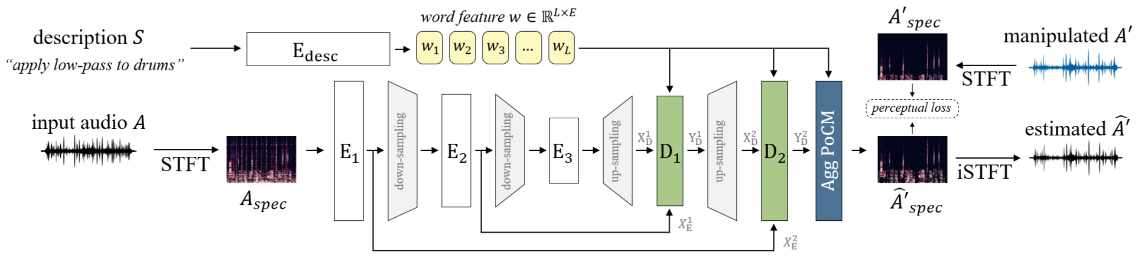


Figure 5.2: AMSS-Net Architecture

5.4.1 Description Encoder

The description encoder E_{desc} encodes the given text description S written in the Audio Manipulation Language (section 5.2) to word features $w \in \mathbb{R}^{L \times E}$, where the dimension of each word feature is denoted by E and the number of words in S by L. It embeds each word to a distributed representation using a word embedding layer and then encodes the embedded representation using a bidirectional Recurrent Neural Network (Bi-RNN) [44]. Pre-trained word embeddings such as GloVe[38] are used for the initialization of the weight of the embedding layer since they were trained to capture the syntactic and semantic meaning of words.

5.4.2 Spectrogram Encoder-Decoder Network

The Spectrogram Encoder-Decoder Network $SEDN$ estimates the complex-valued spectrogram $\hat{A}'_{spec}$, conditioned on the extracted word features $w \in \mathbb{R}^{L \times E}$. It is an encoder-decoder network that has the same number of down-sampling layers and up-sampling layers as depicted in Figure 5.2. It extracts down-sampled representations from A_{spec} in the encoding phase and generates up-sampled representations in the decoding phase. The output of the last decoding block is fed to the *Aggregate PoCM* (see section 5.4.3) that generates the output $\hat{A}'_{spec}$.

As illustrated in Figure 5.2, it has direct connections between the encoding blocks and their counterpart decoding blocks, which help decoding blocks recover fine-grained details of the target. Instead of concatenation or summation commonly used in several U-Net-based architectures [1, 32, 6], this section proposes a Channel-wise Skip Attention (CSA) mechanism (section 5.4.2) that attentively aggregates latent source channels to reconstruct the original channels.

As shown in Figure 5.2, $SEDN$ consists of several components: *encoding blocks*, *down/up-sampling layers*, *decoding blocks*, and an *Aggregate PoCM*. Strided convolutions and transposed convolutions are used for down-sampling and up-sampling, respectively. Other components are introduced in section 5.4.2, section 5.4.2, and section 5.4.3.

Spectrogram Encoding Block

$SEDN$ uses multiple encoding blocks in the encoding phase to capture common sonic properties residing in the input spectrogram. The k^{th} encoding block E_k transforms an input spectrogram-like tensors into an equally-sized tensor. TFC-TDF described in 3.5 is adopted for each encoding block, which applies densely-connected 2-d convolutions to the given spectrogram-like representations followed by a fully connected layer that enhances features of frequency patterns observed in the frequency axis. As observed in chapter 4, TDF blocks inserted in a Conditioned U-Net can improve the overall performance by capturing frequency-to-frequency dependencies observed across all sources. This is why TFC-TDF blocks are selected as fundamental blocks in the encoder of AMSS-Net.

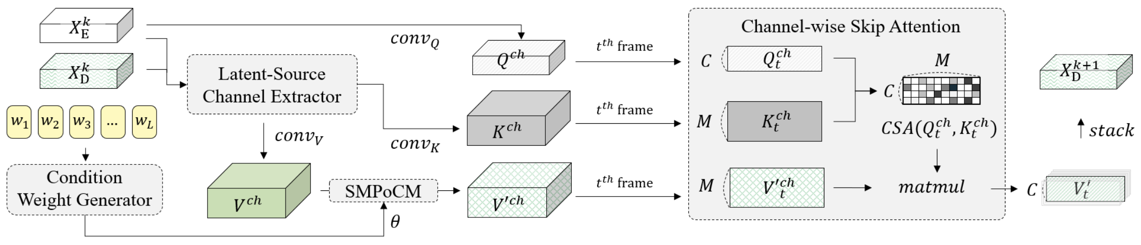


Figure 5.3: k^{th} decoding block (D_k)

Spectrogram Decoding Block

In the decoding phase, $SEDN$ uses multiple decoding blocks. Each decoding block first extracts a feature map in which each channel corresponds to a specific latent source. It selectively manipulates them conditioned on the AMSS description and aggregates channels using a channel-wise attention mechanism to minimize information loss during channel reconstruction.

As shown in Figure 5.3, the k^{th} decoding block D_k takes three inputs: (1) X_D^k: features from the previous decoding block, (2) X_E^k: features from the skip connection and (3) $w \in \mathbb{R}^{L \times E}$: word features. The first block takes the up-sampled features from the encoder instead because it has no previous decoding block.

Each decoding block consists of four components: Latent Source Channel (LSC) Extractor, Condition weight generator, Selective Manipulation via PoCMs (SMPoCM), and Channel-wise Skip Attention (CSA). illustrate the overall workflow in Figure 5.3.

Latent Source Channel Extractor

Assuming it is possible to learn representations of latent sources that deal with more detailed acoustic features than symbolic-level sources, the Latent Source Channel (LSC) extractor aims to generate a feature map V^{ch}, in which each channel deals with a latent source. Figure 5.4 visualizes the conceptual view of latent source channels. Each channel is a spectrogram-like representation of size $T^{(k)} \times F^{(k)}$ dealing with a specific latent source. For example, the blue channel in Figure 5.4 deals with the acoustic features observed in the bass drum. It can also generate an audio track from a single latent source channel.

In section 5.5.4, visualization and discussion of the audio generation are provided.

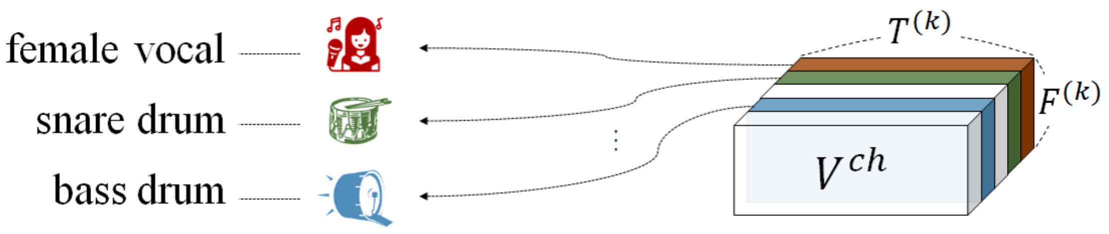


Figure 5.4: Conceptual View of Latent Source Channels

The LSC extractor takes $X_E^k, X_D^k \in \mathbb{R}^{C \times T^{(k)} \times F^{(k)}}$ as input and extracts feature maps with M latent source channels, where C refers to the number of channels, $T^{(k)} \times F^{(k)}$ refers to the shape of the spectrogram-like features, and M denotes the number of latent sources. It first concatenates X_E^k and X_D^k to obtain $[X_E^k; X_D^k] \in \mathbb{R}^{2C \times T^{(k)} \times F^{(k)}}$, and applies a TFC-TDF block to $[X_E^k; X_D^k]$ to obtain acoustic features X^k for latent source separation.

To extract a feature map $V^{ch} \in \mathbb{R}^{M \times T^{(k)} \times F^{(k)}}$ with M latent source channels, it applies a 1×1 convolution to X^k. This convolution is denoted by $conv_V$ since its role is a *value generator* in the context of the channel-wise skip attention mechanism section 5.4.2 as shown in Figure 5.3. Similar to $conv_V$, it applies another 1×1 convolution called $conv_K$ to X^k, of which role is a *key generator*, to obtain K^{ch}. By the guide of the channel-wise skip attention mechanism, the LSC extractor is expected to extract *latent source channels* from the mixture features so that each channel deals with a latent source.

The optimal M depends on datasets. For example, there are 4 types of sources (i.e., 'vocals,' 'drums,' 'bass,' and 'other') in the MUSDB18 [41] dataset. Humans are used to categorizing such symbolic-level classes of sources, but there are more latent sources we have to consider for AMSS, as described before. After carefully listening to the training dataset, one can derived 8 different latent sources as follows: (female, male vocals), (kick, snare, hat), (bass), and (piano, guitar). Any pair in these latent sources must not be treated as a single source in AMSS-Net because each has its unique timbre. That is why the proposed method uses M of 8 in the experiment section.

It is also recommended the empirical optimization for the optimal M. Adjusting M using a small portion of the training dataset (also known as a development set) usually provides a better configuration for the model after fining M based on dataset analysis. If GPUs' memory sizes are limited, it is also possible to use a smaller M. Since we use the multi-head attention mechanism in Channel-wise Skip Attention (CSA), which is introduced in section 5.4.2, using a slightly smaller M does not significantly impact the performance. Multi-heads can relax this by letting each head focus on a different subset of them.

Selective Manipulation via PoCMs

Since a decoding block must manipulate specific features while preserving other features, selective manipulation requires a more sophisticated modulation than existing methods such as Feature-wise Linear Modulation (FiLM) [39], or Point-wise Convolutional Modulation (PoCM).

SMPoCM, an extension of PoCM, is proposed to manipulate features for the given AMSS task selectively. As shown in Figure 5.3, it takes as input V^{ch} and condition parameters $\theta = (\theta_s, \theta_m, \theta_i)$, generated by the LSC extractor and the condition weight generator, respectively. It outputs $V'^{ch} \in \mathbb{R}^{M \times T^{(k)} \times F^{(k)}}$, a selectively manipulated feature.

Inspired by Long Short-term Memory (LSTM) [15], the SMPoCM uses three different PoCMs: (1) a selective gate PoCM with θ_s to determine how much we should manipulate each latent source channel, (2) a manipulation PoCM with θ_m to manipulate specific features, and (3) an input gate PoCM with θ_i to determine how much we should emit the manipulated features.

Before defining SMPoCM formally, the behavior of PoCM is summarized as follows: a PoCM is a point-wise convolution (i.e., 1×1 convolution) of which the condition weight generator provides parameters. A definition of SMPoCM is provided as follows: $\text{SMPoCM}(X|\theta) = i \odot \mathit{tanh}\ (\text{PoCM}(s \odot X, \theta_m)) + (1 - s) \odot X$, where s is defined as $\sigma(\text{PoCM}(X, \theta_s))$, i is defined as $\sigma(\text{PoCM}(X, \theta_i))$, $\odot$ is Hadamard product, and σ is a sigmoid function. The total number of parameters in θ is about $3(M^2 + M)$ (i.e., three point-wise convolutions).

SMPoCM naturally models the selective modulation required for modeling AMSS tasks. For example, if i^{th} latent source should be preserved for the given input, then the i^{th} channel of s would be trained to have near-zero values.

Condition Weight Generator

Given $w \in \mathbb{R}^{L \times E}$, the condition weight generator generates parameters $\theta = (\theta_s, \theta_m, \theta_i)$. AMSS-Net exploits the attention mechanism to determine which word should be attended to θ_s, θ_m, and θ_i respectively.

The weight generator has a learnable matrix $\Theta \in \mathbb{R}^{3 \times d_k}$, where the cardinality of each PoCM task embedding is denoted by d_k. It also has two linear layers $linear_k$ and $linear_v$ that embed w to $w_{key} \in \mathbb{R}^{L \times d_k}$ and $w_{value} \in \mathbb{R}^{L \times d_k}$ respectively. To determine which word we should attend for each task, it computes the scaled attention [58] as follows: $\alpha^{wg} = softmax(\frac{\Theta w_{key}}{\sqrt{d_k}})w_{value}^T$. Finally, it generates θ_s, θ_m, and θ_i as follows: $\theta_s = linear_s(\alpha^{wg}[0, :])$, $\theta_s = linear_m(\alpha^{wg}[1, :])$, and $\theta_s = linear_i(\alpha^{wg}[2, :])$, where $linear_s$, $linear_m$, and $linear_i$ are fully-connected layers.

Channel-wise Skip Attention

Inspired by skip attention [66] and [4], AMSS uses a Channel-wise Skip Attention (CSA) mechanism. It attentively aggregates M latent source channels to restore the number of channels to the same as the input (i.e., C). The goal of CSA is to minimize information loss during channel reconstruction to preserve other features that are irrelevant to the description.

Figure 5.3 overviews the workflow required to prepare the query feature Q^{ch}, key feature K^{ch}, and the value feature V'^{ch}. To obtain $Q^{ch} \in \mathbb{R}^{C \times T^{(k)} \times F^{(k)}}$, it applies $conv_Q$, a 1×1 convolution to X_E^k. It is notable that Q^{ch} represents the original audio feature which is not conditioned by any decoding units. For each frame, CSA aims to capture channel-to-channel dependencies between X_E^k that encodes the original acoustic features of A and the feature map K^{ch} of isolated latent sources obtained by the LSC extractor. It is worth noting that it uses K^{ch} for computing attention matrix instead of V'^{ch} since V'^{ch}, which SMPoCM modulated, no longer has the same information as X_E^k.

For $Q_t^{ch} = Q^{ch}[:, t, :]$ and $K_t^{ch} = K^{ch}[:, t, :]$, it computes the scaled dot product attention matrix [58] as follows:

$$CSA(Q_t^{ch}, K_t^{ch}) = softmax(\frac{Q_t^{ch}(K_t^{ch})^T}{\sqrt{F^{(k)}}}) \tag{5.3}$$

The attention weight $CSA(Q_t^{ch}, K_t^{ch})_{i,j}$ represents the correlation between the i^{th} channel of the original audio features and the j^{th} latent source channel of the decoded audio features. Finally, it is able to obtain the decoding block's output y_D^k, where $y_D^k[:, t, :]$ is defined as $CSA(Q_t^{ch}, K_t^{ch})V'^{ch}[:, t, :]$.

One can argue that Q_t^{ch} and K_t^{ch} must also be conditioned on the given word features, because it has been thought natural to use an individual transformation for each query, key, and value in attention, where the goal is to generate an equivalent output in a different form for the given input (e.g., English-to-French or Text-to-Image). The input and output must have the same information in such tasks.

However, the input and output usually carry different information in AMSS. For example, input is a mixture of (vocals+drums+bass+other) while output is a mixture of (LPF(vocals)+drums+bass+other) for a task with the query "apply lowpass to vocals." On top of this observation, the attention for AMSS must be rethought.

For AMSS, the goal is not only to manipulate target sources but also to preserve the others. To provide the entire uncorrupted spectrum of latent sources in the mixture, a nearly-raw output of the encoder block was used to obtain Q_t^{ch}. It is independent of word features. For manipulation, SMPoCM was used to manipulate V_t^{ch} to obtain $V_t'^{ch}$. To generate the final output V_t' of a decoding block, we have to attentively aggregate channels with the guide of the attention.

Suppose that $V_t'^{ch}$ was reused to compute the attention matrix with Q_t^{ch}. At first glance, it might seem this approach is more appropriate than the proposed method. However, the fundamental goal is the ideal reconstruction weight matrix that preserves the other sources. The attention between Q_t^{ch} and $V_t'^{ch}$ cannot provide the ideal attention since this decoding block already corrupted $V_t'^{ch}$ with word features by this decoding block. In short, this attention is a corrupted similarity matrix in the perspective of preservation. Meanwhile, K_t^{ch} might have been manipulated by previous blocks but not by this decoding block. We have considered that the attention matrix computed with the features of the pure mixture (i.e., Q_t^{ch}) and the K_t^{ch} , which is independent of the word feature in this block, is ideal.

The above is why the proposed model only manipulates a value representation, but it is still plausible that applying a FiLM, a PoCM, a GPoCM or even a simple linear transformation to Query and Key features might improve the performance. The proposed method adopts none of them for several reasons: they are dropped because they do not seem mandatory or critical for the given task; to clarify that the given task is different from the existing 'translation' tasks; to simplify models.

5.4.3 Aggregate PoCM

The Aggregate Pocm is similar to the SMPoCM other than two key differences. First, the Aggregate PoCM only has one PoCM that is not followed by any activation. Second, the Aggregate PoCM reduces the number of channels from C to 4 (i.e., the number of channels of the two-channeled complex-valued spectrogram) while the SMPoCM's input and output have the same number of channels.

5.5 Experiments

5.5.1 Experiment Setup

This section evaluates the proposed model both qualitatively and quantitatively on various AMSS tasks described in Table 5.1 using the MUSDB18 [41] dataset. This section compares its performance with baselines to verify the architecture.

Training Framework

MUSDB18 dataset contains 86 tracks for training, 14 tracks for validation, and 50 tracks for the test. Each track is stereo, sampled at 44100 Hz, and each data tuple consists of the mixture and four sources: vocals, drums, bass, and other. Training framework is implemented based on MUSDB18 and pysndfx[2], a Python DSP library. All models are trained based on this framework with 9 AMSS tasks listed in Table 5.1. For each task, triple generators are developed based on the Audio Manipulation Language (section 5.2) and MUSDB18. The 'other' source is excluded since it is not a single instrument. With the set $\mathcal{G}$ of triple generators and randomly generated multi-tracks obtained by data augmentation [57], AMSS triples are generated by using Algorithm 1 for training.

Training Environment

Models are trained using Adam [18] with learning rate $lr \in [0.0001, 0.001]$. Each model is trained to minimize the L_2 loss between the ground-truth and estimated spectrograms. For validation, the L_1 loss of target and estimated signals is used. It takes about two weeks to converge when models are trained with a single 2080Ti GPU.

5.5.2 Model Configurations

To validate the effectiveness of AMSS-Net, experimental comparisons are provided with the two baselines. One does not use CSA (AMSS-Net w/o CSA), and the other does

[2]https://pypi.org/project/pysndfx/

not use SMPoCM in decoding blocks (AMSS-Net w/o SM). The baseline model without CSA uses an LSC extractor with LaSAFT block instead of TFC-TDF to compensate the absence of CSA. The model without SMPoCM uses a single PoCM with tanh activation in its decoding block. An AMSS-Net has about 4.3M, a baseline without CSA has about 4.9M, and a baseline without SMPoCM has about 2.4M.

For hyper-parameter setting, a similar configuration of models with an FFT window size of 2048 is used as in section 4.6. Every model has three encoding blocks, two decoding blocks, an additional Aggregate PoCM block. The number of latent sources is assumed to be eight (i.e., $M = 8$). Adopting the multi-head attention mechanism [58] for CSA, the number of heads is set to 6. The STFT parameter of each model is as follows: an FFT window size of 2048 and a hop size of 1024.

model	vocals	drums	bass	other	AVG
Meta-TasNet	6.40	5.91	5.58	4.19	5.52
LaSAFT-GPoCM-Net$_{12}$	**7.33**	5.68	**5.63**	**4.87**	**5.88**
LaSAFT-GPoCM-Net$_{11}$	6.96	5.84	5.24	4.54	5.64
AMSS-Net$_{separate}$	6.78 ± .12	**5.92** ± .03	5.10 ± .06	4.51 ± .10	5.58 ± .90
AMSS-Net	**6.34** ± .016	5.53 ± .07	**4.33** ± .13	**3.99** ± .07	**5.05** ± .99
w/o CSA	6.03 ± .24	5.53 ± .12	4.22 ± .09	3.73 ± .23	4.88 ± .99
w/o SMPoCM	6.08 ± .25	**5.63** ± .08	4.28 ± .19	3.76 ± .04	4.94 ± .99

Table 5.2: A comparison SDR performance. LaSAFT-GPoCM-Net$_x$ uses FFT window size of 2^x

	pan left			pan right			decrease volume			increase volume		
	voc	drum	bass	voc	drum	bass	voc	drum	bass	voc	drum	bass
reference loss	4.62	8.20	2.18	4.66	8.30	2.13	4.06	6.99	2.00	6.68	9.70	3.48
AMSS-Net	**3.32** ±0.19	**4.81** ±0.16	**2.11** ±0.13	**3.34** ±0.19	**4.85** ±0.19	**2.11** ±0.17	**2.72** ±0.23	**3.78** ±0.18	**1.93** ±0.13	**3.49** ±0.2	**4.36** ±0.18	**2.9** ±0.19
w/o CSA	3.65 ±0.17	5.46 ±0.2	2.61 ±0.33	3.63 ±0.11	5.53 ±0.23	2.58 ±0.3	3 ±0.13	4.26 ±0.14	2.32 ±0.33	4.47 ±0.12	5.11 ±0.18	3.66 ±0.4
w/o SMPoCM	4.34 ±0.2	5.51 ±0.16	3.23 ±0.19	4.23 ±0.18	5.5 ±0.11	3.19 ±0.17	3.55 ±0.16	4.41 ±0.16	2.92 ±0.17	4.16 ±0.14	5.2 ±0.15	3.4 ±0.17

	lowpass filter			highpass filter			dereverberation			mean		
	voc	drum	bass	voc	drum	bass	voc	drum	bass	voc	drum	bass
reference loss	9.02	16.27	1.12	6.72	8.22	4.74	9.43	11.07	5.61	6.46	9.82	3.04
AMSS-Net	**7.32** ±0.19	**10.92** ±0.15	**2.65** ±0.09	5.65 ±0.28	**6.9** ±0.3	**3.8** ±0.11	**6.12** ±0.29	**6.72** ±0.23	**4.18** ±0.24	**4.57** ±1.70	**6.05** ±2.33	**2.81** ±0.85
w/o CSA	7.44 ±0.07	12.37 ±0.5	3.15 ±0.21	**5.49** ±0.25	7.41 ±0.67	4.23 ±0.4	6.18 ±0.12	7.08 ±0.07	4.28 ±0.1	4.84 ±1.53	6.75 ±2.60	3.26 ±0.83
w/o SMPoCM	8.11 ±0.1	12.06 ±0.31	3.67 ±0.23	6.81 ±0.21	7.84 ±0.24	4.85 ±0.42	6.68 ±0.12	7.17 ±0.24	4.68 ±0.05	5.41 ±1.67	6.81 ±2.47	3.71 ±0.75

Table 5.3: A RMSE-MFCC Comparison of the proposed models with baselines, over 7 AMSS tasks applied to vocals, drums, and bass

5.5.3 Quantitative Analysis

Evaluation of separate and mute tasks (Table 5.2)

To evaluate *separate* and *mute* tasks, Source-to-Distortion (SDR) [59] metric was computed by using the official tool[3]. The MUSDB18 tasks are separating vocals, drums, bass, and other, which corresponds to the following AMSS: 'separate vocals,' 'separate drums,' 'separate bass,' and 'mute vocals, drums, bass.' Using the SDR metric for these two tasks allows to compare the proposed model's source separation performance with other state-of-the-art models for conditioned source separation. Following the official guideline, this section reports the median SDR value over all the test set tracks for each run and report the mean SDR over three runs (with a different random seed).

Table 5.2 summarizes the result, where AMSS-Net shows comparable or and slightly inferior performance compared to state-of-the-art conditioned separation models, namely, Meta-TasNet [43] and LaSAFT-GPoCM-Nets. AMSS-Net outperforms the baselines for all sources except for drums, where the gap is not significant. It is worth noting that ours can perform other AMSS tasks and show promising results on source separation tasks. If AMSS-Net is trained solely for MUSDB18 tasks (AMSS-Net$_{separate}$), then it shows comparable performance to LaSAFT-GPoCM-Net$_{11}$ whose FFT window size is the same as that of the proposed model.

Evaluation of other AMSS tasks (Table 5.3)

Since there is no reference evaluation metric for AMSS, this section proposes the new evaluation benchmark. The benchmark script is available online[4]. For evaluation metric, Mel-frequency Cepstral coefficients (MFCC) for A' and $\hat{A}'$ are extracted, and the Root Mean Square Error (RMSE) of them are computed, since MFCC approximates the human perception of the given track. This metric is referred as RMSE-MFCC. This section reports the mean RMSE-MFCC value over all the test set tracks for each run and report the mean RMSE-MFCC over three runs.

[3]https://github.com/sigsep/sigsep-mus-eval
[4]https://github.com/kuielab/AMSS-Net/blob/main/task2_eval.py

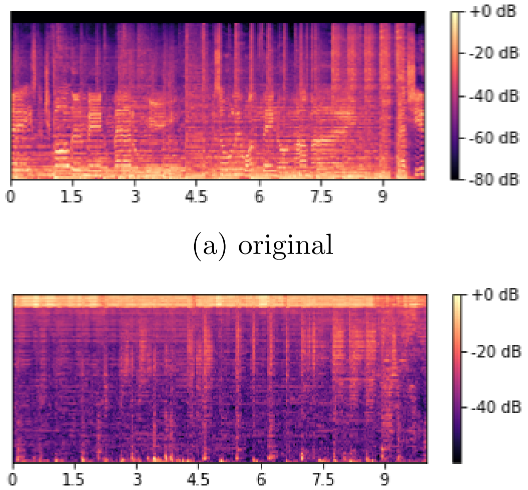
Table 5.3 summarizes the results, where *reference loss* is the RMSE-MFCC of A and A'. Reference loss provides information about the amount of manipulation needed to model each AMSS task. As described in Table 5.3, AMSS-Net outperforms all the AMSS task but a task of “highpass filter to vocals.” Significantly, the model without SMPoCM is inferior to AMSS-Net for every task. It indicates that SMPoCM significantly contributes to improving the quality of AMSS results. CSA also improves the performance of AMSS-Net for most of the AMSS tasks. CSA might degrade the performance for more difficult AMSS tasks by forcing the model to over-correlate the latent source channels with the mixture channels. However, it reduces artifacts created during progressive manipulation as described in section 5.5.5

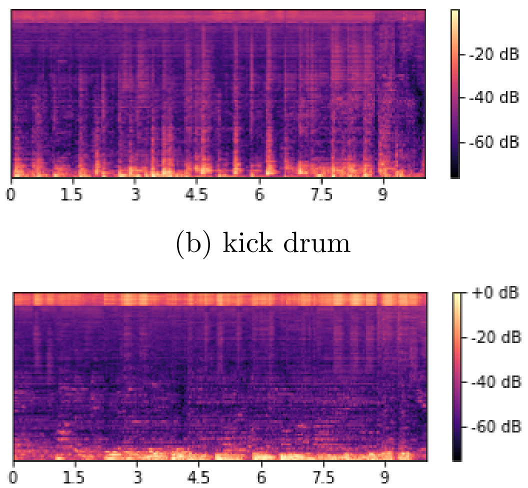
5.5.4 Latent Source Channels

As mentioned in the LSC extractor (section 5.4.2), AMSS-Net is designed to perform latent source-level analysis. Such analysis enables AMSS-Net to perform delicate manipulation for the given AMSS task. To verify that AMSS-Net decoding blocks can extract a feature map in which each channel corresponds to a specific latent source, an audio track from a single latent source channel is generated. All channels is masked in the manipulated feature map V'^{ch} except for a single latent source channel. Then it is fed to the remaining sub-networks to generate the audio track during the last decoding block.

Figure 5.5 shows interesting results of generated audios, which remind us the conceptual view of the latent source in Figure 5.4. For the given input track of Figure 5.5 (a), an audio track is generated after masking all channels except for the fifth channel in the second head group, then the result sounds similar to the low-frequency band of drums (i.e., kick drum) as illustrated in Figure 5.5 (b). AMSS-Net can keep this channel and drop other drum-related channels to process “apply lowpass to drums.” However, a latent source channel does not always contain a single class of instruments. For example, the latent channel of the fourth row in the table deals with several instruments. Some latent sources were not interpretable to the authors. Generated samples are available online.[5]

[5] https://kuielab.github.io/AMSS-Net/latent_source.html





(a) original (b) kick drum (c) high hat (d) non-percussion

Figure 5.5: Mel-Spectrogram of single latent-source channel

5.5.5 Progressive Manipulation

This subsection shows that it is eable to repeatedly apply the proposed method to manipulated audio tracks, which is also known as Progressive Manipulation used in conversational systems described in [23]. Figure 5.6 shows an example of progressive manipulation.

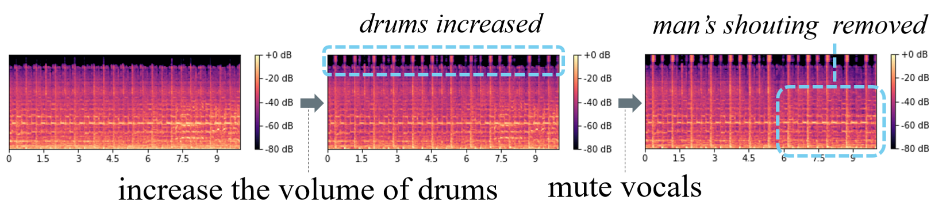
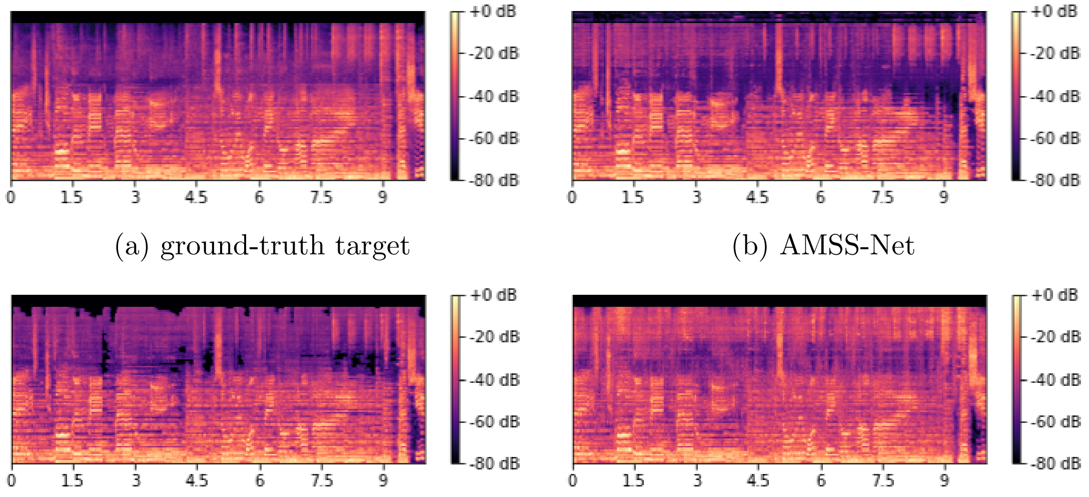


Figure 5.6: An Example of Progressive Manipulation

However, methods based on neural networks sometimes suffer from artifacts [59], which are not present in the original source. Although they sound negligible after a single manipulation task, they can be large enough to be perceived after progressively applying several. To investigate artifacts created by progressive manipulation, the same AMSS task "apply highpass to drums" is issued to a track in a progressive manner. Figure (a) shows the Mel-spectrogram of the ground-truth target. Blurred areas are observable in the high-frequency range in Mel-spectrograms of Figure 5.7 (c) and (d). Compared to them, Figure 5.7 (b) is more similar to the ground-truth target. Via hearing test, it was observed perceivable artifacts 5.7 in the results of baselines. The AMSS-Net contains minor artifacts compare to them because each decoding block of AMSS-Net has a CSA mechanism, a unique structure that prevents unwanted noise generated by intermediate manipulated features. Generated samples are available online[6].



(a) ground-truth target

(b) AMSS-Net

(c) w/o CSA

(d) w/o SMPoCM

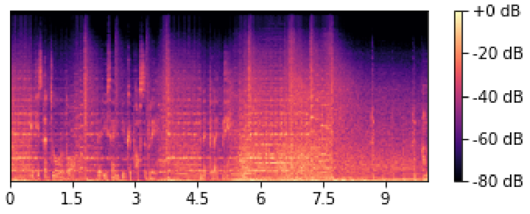
Figure 5.7: Mel-Spectrogram Comparison after applying 20 times of 'apply highpass to durms' in a progressive manner

5.5.6 Controlling the level of audio effects

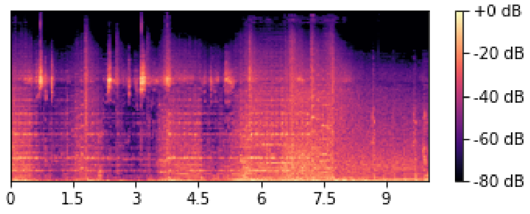
As mentioned in section 5.2, it is possible to train AMSS-Net to perform more detailed AMSS such as "apply heavy lowpass to vocals". As shown in Figure 5.8, users can control the level of audio effects by simply injecting adverbs instead of a laborious search for an

[6]https://kuielab.github.io/AMSS-Net/progresive.html

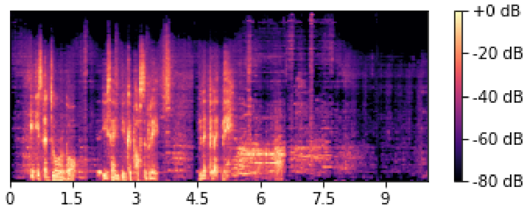
appropriate parameter configuration. Generated samples are available online[7].



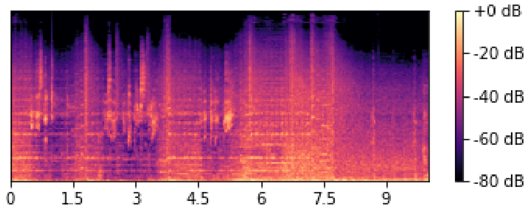
(a) original



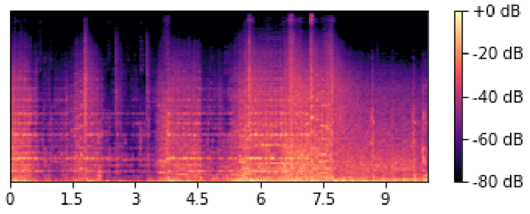
(b) heavy highpass to vocals



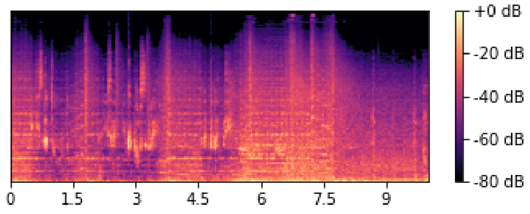
(c) separate vocals



(d) medium highpass to vocals



(e) mute vocals



(f) light highpass to vocals

Figure 5.8: Controlling the level of highpass with adjectives

[7]https://kuielab.github.io/AMSS-Net/control_level.html

5.6 Discussion

AMSS-Net shows promising results on several AMSS tasks. AMSS-Net can also be trained with a more complicated AMSS training dataset based on a realistic audio mixing dataset such as *IDMT-SMT-Audio-Effects* dataset [45]. However, this study is limited to model relatively simpler AMSS tasks. One can extend this work to provide more complex AMSS tasks such as distortion and reverberation. Each AMSS task in this paper only deals with a single type of manipulations, but one can also extend this work to provide multiple types of tasks such as "apply reverb to vocals and apply lowpass to drums" at once. Also, this work is easily extendable to support a more user-friendly interface. For example, adopting unsupervised training frameworks such as Mixture of Mixture (MoM) [60] to train AMSS on annotated audio datasets such as clotho[8] might enable a natural language query interface.

5.7 Conclusion

In this chapter, a novel task called AMSS is formulated. Also, this chapter proposes AMSS-Net, which generates feature maps in which each channel deals with a latent source and selectively manipulates them while preserving irrelevant features. AMSS-Net can perform several AMSS tasks, unlike previous models such as LaSAFT-GPoCM-Net. The experimental results show that AMSS-Net outperforms baselines on several tasks. Future work will extend it to provide more complex AMSS tasks such as distortion and reverberation by adopting state-of-the-art methods such as Generative Adversarial Networks (GAN).

Chapter 6

Conclusion

This dissertation presents deep learning-based latent source analysis for source-aware audio manipulation. It proposes two neural networks for source separation and one neural network for more advanced audio task called Audio Manipulation on Specified Sources (AMSS). In each domain, the proposed networks show promising results achieving the state-of-the-art performance on the existing benchmark or the proposed benchmark for AMSS. Especially, a novel concept called Latent Sources Analysis is introduced for conditioned source separation and AMSS. On top of the novel method called latent source analysis, the proposed models outperform existing models in several tasks.

Specifically, this thesis verifies that injecting frequency transformation blocks, called TDFs, into a standard U-Net architecture can significantly improve the separation performance. Empirical justification is given to explain how it works with an ablation study. Furthermore, this thesis extends TDFs to conditioned source separation by exploiting the novel concept of latent sources. This thesis shows that LaSAFT, which aims to capture source-dependent frequency patterns by extending TDF to fit the multi-source task, can considerably improves the SDR performance. Also GPoCM is proposed, which modulates features more flexibly and expressively than FiLM. The experimental results indicate that employing our LaSAFT and GPoCM in CUNet can significantly improve SDR performance.

The final goal of this work is to develop a neural network that can perform source-specific audio manipulation according to the given text query, which is also called AMSS, in this thesis. The proposed AMSS-Net can perform several AMSS tasks, unlike previous models. The experimental results show that AMSS-Net outperforms baselines on several tasks. I believe AMSS-Net can be widely used in various audio signal processing software such as Smart-Phone-Applications, DAWs, or DAW-plugins, as many users have loved the ML-based applications. Decreasing the difficulty of audio editing will make more users create, edit, manipulate, and share their audio files. Future work will extend it to provide more complex AMSS tasks such as distortion and reverberation by adopting state-of-the-art methods such as Generative Adversarial Networks (GAN).

Chapter 7

Appendix

7.1 Appendix A - Context-Free Grammar for Audio Manipulation Language

1. $< desc > \rightarrow < cls_v > \| < cls_p > \| < cls_f > \| < cls_d >$

2. $< cls_v > \rightarrow < vc_{hard} > \| < vc_{soft} >$

3. $< vc_{hard} > \rightarrow < mask - vol > < srcs >$

4. $< mask - vol > \rightarrow$ mute $\|$ remove $\|$ get rid of $\|$ eliminate separate $\|$ isolate $\|$ extract

5. $< vc_{soft} > \rightarrow < rescale - vol >$ the volume of $< srcs >$

6. $< rescale - vol > \rightarrow$ increase $\|$ decrease

7. $< cls_p > \rightarrow$ pan $< opt - pan >$ to the $< direction >$ side

8. $< opt - pan > \rightarrow < src > \| < src >$ completely

9. $< direction >$ left $\|$ right

10. $< cls_f > \rightarrow$ apply $< opt - filter >$ to $< srcs >$

11. $< opt - filter > \rightarrow < opt >< filter > \parallel < filter >$

12. $< opt > \rightarrow$ light $\parallel$ medium $\parallel$ heavy

13. $< filter > \rightarrow$ lowpass $\parallel$ highpass

14. $< cls_d > \rightarrow$ remove reverb from $< srcs >$

15. $< srcs > \rightarrow$ vocals $\parallel$ drums $\parallel$ bass $\parallel$ vocals, bass $\parallel$ vocals, drums $\parallel$ drums, vocals $\parallel$ drums, bass $\parallel$ bass, vocals $\parallel$ bass, drums $\parallel$ vocals, bass, drums $\parallel$ vocals, drums, bass $\parallel$ drums, vocals, bass $\parallel$ drums, bass, vocals $\parallel$ bass, vocals, drums $\parallel$ bass, drums, vocals $\parallel$

Bibliography

[1] Jansson Andreas, Humphrey Eric, Montecchio Nicola, Bittner Rachel, Kumar Aparna, and Weyde Tillman. 2017. Singing voice separation with deep u-net convolutional networks. In *18th International Society for Music Information Retrieval Conference*. 23–27.

[2] Martin Arjovsky, Amar Shah, and Yoshua Bengio. 2016. Unitary evolution recurrent neural networks. In *International Conference on Machine Learning*. PMLR, 1120–1128.

[3] Pritish Chandna, Marius Miron, Jordi Janer, and Emilia Gómez. 2017. Monoaural audio source separation using deep convolutional neural networks. In *International conference on latent variable analysis and signal separation*. Springer, 258–266.

[4] Long Chen, Hanwang Zhang, Jun Xiao, Liqiang Nie, Jian Shao, Wei Liu, and Tat-Seng Chua. 2017. Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5659–5667.

[5] N. Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2, 3 (1956), 113–124. https://doi.org/10.1109/TIT.1956.1056813

[6] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. 2019. Mu-

sic Source Separation in the Waveform Domain. *arXiv preprint arXiv:1911.13254* (2019).

[7] W Dixon Ward. 1970. Musical Perception. *Foundations of Modern Auditory Theory* 1 (1970), 1270–77.

[8] Konstantinos Drossos, Samuel Lipping, and Tuomas Virtanen. 2020. Clotho: An audio captioning dataset. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 736–740.

[9] Harvey Fletcher. 1938. Loudness, masking and their relation to the hearing process and the problem of noise measurement. *The Journal of the Acoustical Society of America* 9, 4 (1938), 275–293.

[10] Harvey Fletcher and Wilden A Munson. 1937. Relation between loudness and masking. *The Journal of the Acoustical Society of America* 9, 1 (1937), 1–10.

[11] Szu-Wei Fu, Ting-yao Hu, Yu Tsao, and Xugang Lu. 2017. Complex spectrogram enhancement by convolutional neural network with multi-metrics learning. In *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 1–6.

[12] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 315–323.

[13] Nitzan Guberman. 2016. On complex valued convolutional neural networks. *arXiv preprint arXiv:1602.09046* (2016).

[14] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. 2012. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on* 14 (2012), 8.

[15] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780.

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[17] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*. 448–456.

[18] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1412.6980

[19] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip HS Torr. 2020. Manigan: Text-guided image manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7880–7889.

[20] Guilin Liu, Fitsum A Reda, Kevin J Shih, Ting-Chun Wang, Andrew Tao, and Bryan Catanzaro. 2018. Image inpainting for irregular holes using partial convolutions. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 85–100.

[21] Jen-Yu Liu and Yi-Hsuan Yang. 2018. Denoising Auto-encoder with Recurrent Skip Connections and Residual Regression for Music Source Separation. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 773–778.

[22] Jen-Yu Liu and Yi-Hsuan Yang. 2019. Dilated Convolution with Dilated GRU for Music Source Separation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 4718–4724. https://doi.org/10.24963/ijcai.2019/655

[23] Yahui Liu, Marco De Nadai, Deng Cai, Huayang Li, Xavier Alameda-Pineda, Nicu Sebe, and Bruno Lepri. 2020. Describe What to Change: A Text-guided Unsupervised

Image-to-Image Translation Approach. In *Proceedings of the 28th ACM International Conference on Multimedia*. 1357–1365.

[24] Yi Luo and Nima Mesgarani. 2018. Real-time Single-channel Dereverberation and Separation with Time-domain Audio Separation Network. In *Proc. Interspeech 2018*. 342–346. https://doi.org/10.21437/Interspeech.2018-2290

[25] Yi Luo and Nima Mesgarani. 2018. TaSNet: Time-Domain Audio Separation Network for Real-Time, Single-Channel Speech Separation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 696–700. https://doi.org/10.1109/ICASSP.2018.8462116

[26] Yi Luo and Nima Mesgarani. 2019. Conv-tasnet: Surpassing ideal time–frequency magnitude masking for speech separation. *IEEE/ACM transactions on audio, speech, and language processing* 27, 8 (2019), 1256–1266.

[27] Marco A Martínez Ramírez, Emmanouil Benetos, and Joshua D Reiss. 2020. Deep learning for black-box modeling of audio effects. *APPLIED SCIENCES-BASEL* 10, 2 (2020).

[28] Marco A Martínez Ramírez and Joshua D Reiss. 2018. End-to-end equalization with convolutional neural networks. In *21st International Conference on Digital Audio Effects (DAFx-18)*.

[29] Marco A Martínez Ramírez and Joshua D Reiss. 2019. Modeling nonlinear audio effects with end-to-end deep neural networks. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 171–175.

[30] Marco A Martínez Ramírez, Daniel Stoller, and David Moffat. 2021. A Deep Learning Approach to Intelligent Drum Mixing with the Wave-U-Net. *Journal of the Audio Engineering Society* 69, 3 (2021), 142–151.

[31] D. Matz, Estefanía Cano, and J. Abeßer. 2015. New Sonorities for Early Jazz Recordings Using Sound Source Separation and Automatic Mixing Tools. In *ISMIR*.

[32] Gabriel Meseguer-Brocal and Geoffroy Peeters. 2019. CONDITIONED-U-NET: Introducing a Control Mechanism in the U-net For Multiple Source Separations.. In *20th International Society for Music Information Retrieval Conference*, ISMIR (Ed.).

[33] Stylianos Ioannis Mimilakis, Estefanıa Cano, Jakob Abeßer, and Gerald Schuller. 2016. New sonorities for jazz recordings: Separation and mixing using deep neural networks. In *2nd AES Workshop on Intelligent Music Production*, Vol. 13.

[34] Eliya Nachmani, Yossi Adi, and Lior Wolf. 2020. Voice Separation with an Unknown Number of Multiple Speakers. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 7164–7175. http://proceedings.mlr.press/v119/nachmani20a.html

[35] Zhiheng Ouyang, Hongjiang Yu, Wei-Ping Zhu, and Benoit Champagne. 2019. A Fully Convolutional Neural Network for Complex Spectrogram Processing in Speech Enhancement. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 5756–5760.

[36] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 5206–5210. https://doi.org/10.1109/ICASSP.2015.7178964

[37] Sungheon Park, Taehoon Kim, Kyogu Lee, and Nojun Kwak. 2018. Music Source Separation Using Stacked Hourglass Networks. In *The 19th International Society for Music Information Retrieval Conference*.

[38] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation.. In *EMNLP*, Vol. 14. 1532–1543.

[39] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C Courville. 2018. FiLM: Visual Reasoning with a General Conditioning Layer. In *AAAI*.

[40] Jordi Pons, Santiago Pascual, Giulio Cengarle, and Joan Serrà. 2021. Upsampling artifacts in neural audio synthesis. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 3005–3009.

[41] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. 2017. MUSDB18 - a corpus for music separation. https://doi.org/10.5281/zenodo.1117371 MUSDB18: a corpus for music source separation.

[42] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[43] David Samuel, Aditya Ganeshan, and Jason Naradowsky. 2020. Meta-learning Extractors for Music Source Separation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 816–820.

[44] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing* 45, 11 (1997), 2673–2681.

[45] Michael Stein, Jakob Abeßer, Christian Dittmar, and Gerald Schuller. 2010. Automatic detection of audio effects in guitar and bass recordings. In *Audio Engineering Society Convention 128*. Audio Engineering Society.

[46] John C Steinberg. 1937. Positions of stimulation in the cochlea by pure tones. *The Journal of the Acoustical Society of America* 8, 3 (1937), 176–180.

[47] Christian J. Steinmetz, Jordi Pons, Santiago Pascual, and Joan Serrà. 2021. Automatic Multitrack Mixing With A Differentiable Mixing Console Of Neural Audio Effects. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech*

and Signal Processing (ICASSP). 71–75. https://doi.org/10.1109/ICASSP39728.2021.9414364

[48] Christian J Steinmetz and Joshua D Reiss. 2021. Efficient Neural Networks for Real-time Analog Audio Effect Modeling. *arXiv preprint arXiv:2102.06200* (2021).

[49] Daniel Stoller, Sebastian Ewert, and Simon Dixon. 2018. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In *The 19th International Society for Music Information Retrieval Conference*.

[50] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. 2018. The 2018 signal separation evaluation campaign. In *International Conference on Latent Variable Analysis and Signal Separation*. Springer, 293–305.

[51] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. 2019. Open-Unmix - A Reference Implementation for Music Source Separation. *Journal of Open Source Software* 4, 41 (Sept. 2019), 1667. https://doi.org/10.21105/joss.01667

[52] Naoya Takahashi, Purvi Agrawal, Nabarun Goswami, and Yuki Mitsufuji. 2018. PhaseNet: Discretized Phase Modeling with Deep Neural Networks for Audio Source Separation.. In *Interspeech*. 2713–2717.

[53] Naoya Takahashi, Nabarun Goswami, and Yuki Mitsufuji. 2018. MMDenseLSTM: An efficient combination of convolutional and recurrent neural networks for audio source separation. In *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE, 106–110.

[54] N. Takahashi and Y. Mitsufuji. 2017. Multi-Scale multi-band densenets for audio source separation. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. 21–25. https://doi.org/10.1109/WASPAA.2017.8169987

[55] Naoya Takahashi and Yuki Mitsufuji. 2020. D3Net: Densely connected multidilated DenseNet for music source separation. *arXiv preprint arXiv:2010.01733* (2020).

[56] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. 2018. Deep Complex Networks. In *International Conference on Learning Representations*.

[57] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. 2017. Improving music source separation based on deep neural networks through data augmentation and network blending. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 261–265.

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[59] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. 2006. Performance measurement in blind audio source separation. *IEEE transactions on audio, speech, and language processing* 14, 4 (2006), 1462–1469.

[60] Scott Wisdom, Efthymios Tzinis, Hakan Erdogan, Ron J. Weiss, Kevin Wilson, and John R. Hershey. 2020. Unsupervised Sound Separation Using Mixture Invariant Training. In *NeurIPS*. https://arxiv.org/pdf/2006.12701.pdf

[61] Alec Wright, Eero-Pekka Damskägg, Lauri Juvela, and Vesa Välimäki. 2020. Real-Time Guitar Amplifier Emulation with Deep Learning. *Applied Sciences* 10, 3 (2020). https://doi.org/10.3390/app10030766

[62] Lonce Wyse. 2017. Audio Spectrogram Representations for Processing with Convolutional Neural Networks. In *Proceedings of the First International Conference on Deep Learning and Music*. 37–41.

[63] Dacheng Yin, Chong Luo, Zhiwei Xiong, and Wenjun Zeng. 2020. Phasen: A phase-and-harmonics-aware speech enhancement network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 9458–9465.

[64] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. 2018. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 5505–5514.

[65] W. Yuan, S. Wang, X. Li, M. Unoki, and W. Wang. 2019. A Skip Attention Mechanism for Monaural Singing Voice Separation. *IEEE Signal Processing Letters* 26, 10 (Oct 2019), 1481–1485. https://doi.org/10.1109/LSP.2019.2935867

[66] Weitao Yuan, Shengbei Wang, Xiangrui Li, Masashi Unoki, and Wenwu Wang. 2019. A Skip Attention Mechanism for Monaural Singing Voice Separation. *IEEE Signal Processing Letters* 26, 10 (2019), 1481–1485.

[67] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*. 2223–2232.

Acknowledgement

I would like to appreciate all the supports and helps I received from the committee chair, Professor Soonyoung Jung, my advisor. I could not have completed my Ph.D. program without his guidance and advice. I appreciate the other dissertation committee members, Professor Heonchang Yu, Hyeoncheol Kim, Taeweon Suh, and Jaehwa Chung. Thanks to their thoughtful comments and advice, I could have revised the dissertation to significantly improve its quality. I express my special gratitude to Professor Jaehwa Chung, who constantly inspires my research. He has been my lecturer, senior alumnus, and project leader member for more than ten years. Thank you, Professor Hong-Jun Jang, who has taught and guided me to become a better researcher. Also, my sincere thanks go to Professor Jeongmin Chae, Kinam Park, Daewon Lee, and Byoungwook Kim. I am so lucky to know you, Dr. Marco A. Martínez Ramírez, who has contributed a lot to the writhing of this dissertation and other works.

Thank you to all the alumni of Intelligence Engineering Lab. Thank you for being my best friends Kyoungseok Hyun and Jonghyun Min. I miss the sleepless nights that we spent studying and discussing various issues. I thank Taemin Lee, the kindest and most brilliant person that I have met. Many thanks go to Minseok Kim, who was a perfect coworker for me. I believe you will be an outstanding researcher. I am wishing you lots of success and happiness in your new job, Hyundo Jung. It was a great honor and a privilege to work with new members of the Intelligence Engineering Lab: Min Shin, Jinsung Kim, Yeongseok Jeong, and Jun-Hyung Lee. I hope you have good luck in all you are involved in. Thank you to many other labmates; Taehyung Lim and Gromyko Diana. Besides, I would like to thank fellows in the department of computer science and engineering; Han-Yee Kim, Rohyoung Myung, Eunhun Lee and Sang Woo Park.

I am externally grateful to my parents for their love and support throughout my life. They have encouraged me to be myself and pursue my passions. I also express my thanks to my beloved sisters and my brother in law. Thank all of you for helping me to accomplish my dreams.