國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Master Thesis

使用機器學習進行無伴奏合唱的歌聲分離

Machine Learning for Source Separation of A Cappella Music

Quinn Myles McGarry

指導教授：張智星 博士

Advisor: Jyh-Shing Roger Jang, Ph. D.

中華民國 112 年 8 月

August, 2023

doi:10.6342/NTU202303444

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

使用機器學習進行無伴奏合唱的歌聲分離

Machine Learning for Source Separation of A Cappella Music

本論文係 QUINN MYLES MCGARRY 君（學號 R10922158）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 112 年 7 月 30 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department of Computer Science and Information Engineering on 30 July 2023 have examined a Master's thesis entitled above presented by QUINN MYLES MCGARRY (student ID: R10922158) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

張智星	曹昱	蘇黎
（指導教授 Advisor）		

系主任/所長 Director: 洪士灝

doi:10.6342/NTU202303444

Acknowledgements

I would first like to say thank you to my advisor Roger Jang, who made it possible for this project to be completed. I would also give my sincerest thanks to 王鈞右 and 陈欣惠 for giving so much of their time to help with my progress. I'd like to express my gratitude for my parents who are the reason I am here, in a literal way, and my friends, on this side of the world and the other who are the reason I am here, in a less literal way.

This one's for the boys.

doi:10.6342/NTU202303444

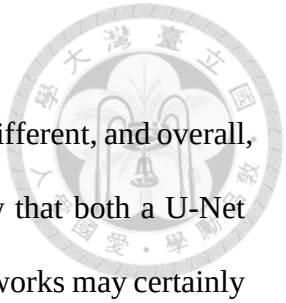
doi:10.6342/NTU202303444

Abstract

In recent years there have been many studies done on the problem of speech separation, which attempts to separate audio of multiple people speaking simultaneously into the audio of each speaker individually. However, audio source separation of multiple simultaneous singers is still not well explored and remains a challenge. This is mainly due to the fact that when people are singing their voices tend to "blend" together much more than when speaking, and multiple vocal lines are often singing the same words, and potentially frequencies, in unison. In order to deal with these issues, we propose a new U-Net based model specifically for the purpose of a cappella singing separation of two singers and compare it to three state-of-the-art speech separation models.

There is a large variety in the results of our experiments. The U-Net based network excels at separating music taken from choir datasets, with a max mean SDR of 9.76 dB, but achieves poor results at separating random combinations of singers. The best speech separation network is capable of separating random combinations of singers quite well, with a max mean SDR of 7.64 dB after finetuning but is uncapable of separating samples where the singers are singing the same lyrics simultaneously. This singing separation score is also much lower than the same model's mean SDR for speech separation of 9.04 dB.

doi:10.6342/NTU202303444



These results are quite nuanced and show that singing separation is a different, and overall, more difficult task than speech separation. However, they also show that both a U-Net based network, and one based on contemporary speech separation networks may certainly be capable of performing well on it.

Keywords: Machine learning, Audio source separation, Music source separation, Speech separation, Music information retrieval, Singing separation, A cappella separation, U-Net, TasNet

doi:10.6342/NTU202303444

Table of Contents

doi:10.6342/NTU202303444

doi:10.6342/NTU202303444

List of Figures

doi:10.6342/NTU202303444

doi:10.6342/NTU202303444

List of Tables

doi:10.6342/NTU202303444

Denotation

CNN: Convolutional neural network

dB: decibels

kHz: kilohertz

LSTM: Long short-term memory

ReLU: Rectified linear unit

RNN: Recurrent neural network

SATB: Soprano, Alto, Tenor, Bass. The four standard voice parts of a choir

SAR: Signal-to-artifact ratio

SDR: Signal-to-distortion ratio

SDRi: Signal-to-distortion ratio improvement

SIR: Signal-to-interference ratio

SI-SDRi: Scale invariant signal-to-distortion ratio improvement

Stem: Individual audio track; part of an entire song, or mixture of audio track

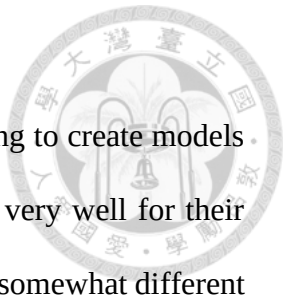
doi:10.6342/NTU202303444

Chapter 1 Introduction

1.1 Audio Separation

Attempting to separate a recording of multiple speakers simultaneously speaking with the purpose of returning each of their voices individually, which is sometimes known as the "cocktail party problem," has been an area of research in computer science for many years. A closely related problem, which has also been an area of research for many years, is attempting to take a song as input in order to return the audio of each instrument, as well as the singer, individually. These are usually known as "speech separation" and "music source separation" respectively. In both of these cases the goal is to retrieve from the original audio mixture, whether it be speech or music, source signals which are as clear as possible. These both have many useful applications, for example, speech separation can be used to take unintelligible audio, such as field-audio captured by journalists which has multiple people talking over each other, and easily recover what each individual is saying. If doable in real time it can also be used to make audio processing devices such as hearing aids. Music source separation can be used by music producers to recover individual parts of songs to be used as samples in new songs. Removal of certain instruments can also be used to make "backing tracks" which are helpful for musicians who want practice playing along with any song they wish.

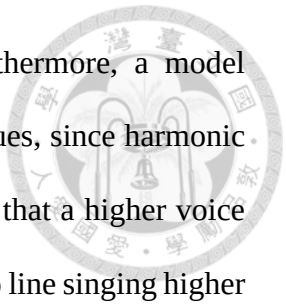
doi:10.6342/NTU202303444



Recently, a lot of work has been done in the realm of machine learning to create models which can perform these tasks very well. While these models work very well for their intended purposes, it is difficult to say how they will perform given a somewhat different task, that being separating simultaneous singing, also known as a cappella separation. At first this may sound very similar to the other two problems mentioned, and while it is closely related, it does differ in some important ways, which might make it more challenging for these previously successful networks.

Firstly, music source separation models function by being able to recognize the differences between instruments, and split audio tracks based on these differences. However, even human voices which are quite different in timbre, such as a male and female voice, are closer to each other than a human voice is to most, if not all, other instruments. Because of this, it would be reasonable to predict that music source separation models may not perform the best at this task. What about speech separation models then? Intuitively, speech and singing are quite closely related, but in the case of singing separation a few new problems are introduced which are not present in speech separation. One issue is that in the case of singing, people's voices may be more similar to each other than they would be when speaking, as they naturally tend to "blend" together. This is also pushed to the extreme as it is not uncommon for singers to record themselves singing over their own voice multiple times to create layers of harmony. This is a situation which would never be encountered in speech separation. In addition to potentially having the same voice, singers will also often be singing the same words. This means that if the model is trained to either recognize a difference in the speaker's voices or recognize differences in the spoken phonetics, which are two attributes that are often used to

doi:10.6342/NTU202303444



separate speech, it could run into issues in these scenarios. Furthermore, a model separating voices based only on harmonic content can also cause issues, since harmonic lines can often overlap, sing the same notes, and even cross over so that a higher voice part is singing lower notes than a lower voice part, for example the alto line singing higher notes than the soprano line. All of these are situations which would likely cause a standard speech separation model to fail. These may seem like extreme examples, but they do occur in reality, and it is important to consider them.

It is also important to clarify that we will be simplifying things here to only involve two singers, and to only include what would be referred to as "classical" a cappella, as opposed to "modern" a cappella. The main distinction here being that modern a cappella has seen the introduction of "beatboxing", a type of vocal percussion meant to imitate the sounds of drums. In our experiments we will only be attempting to separate the type of vocal singing that is typical of a choir, or other classical singing group.

1.2 Possible Cases

Since we will be dealing with only two singers in our experiments, there are four general cases that the singing mixtures can fall into based on a.) if there is one voice or two voices and b.) if they are singing the same lyrics or different lyrics. Firstly, there can be two different singers singing different words (Figure 1.1 - Top left). This is the simplest scenario since there is the most information to differentiate them, including differences in the timbre of the voices, as well as phonetics, and potentially differences in frequency. This is also the most similar case to standard speech separation. The next step up in difficulty is still having two different singers but having them sing the same words at the

doi:10.6342/NTU202303444

same time (Figure 1.1 - Top right). In this arrangement the difference in words is removed as a point to use for separation, but the difference in voices, and potentially the difference in frequencies is still usable. The third case is the same singer singing different words (Figure 1.1 - Bottom left). Here there is no difference in voice to be used for separation, but the difference in words is reintroduced. It is likely these middle two cases are somewhat similar in difficulty, but without testing it is hard to say for sure. The most difficult case would be the same voice singing the same words (Figure 1.1 - Bottom right). The only potential variable here would be the difference in frequency and, as we discussed, this is not necessarily a reliable way to separate the two voice lines. In our experiments we will mainly be dealing with the first case, as well as some audio from the second case.

doi:10.6342/NTU202303444

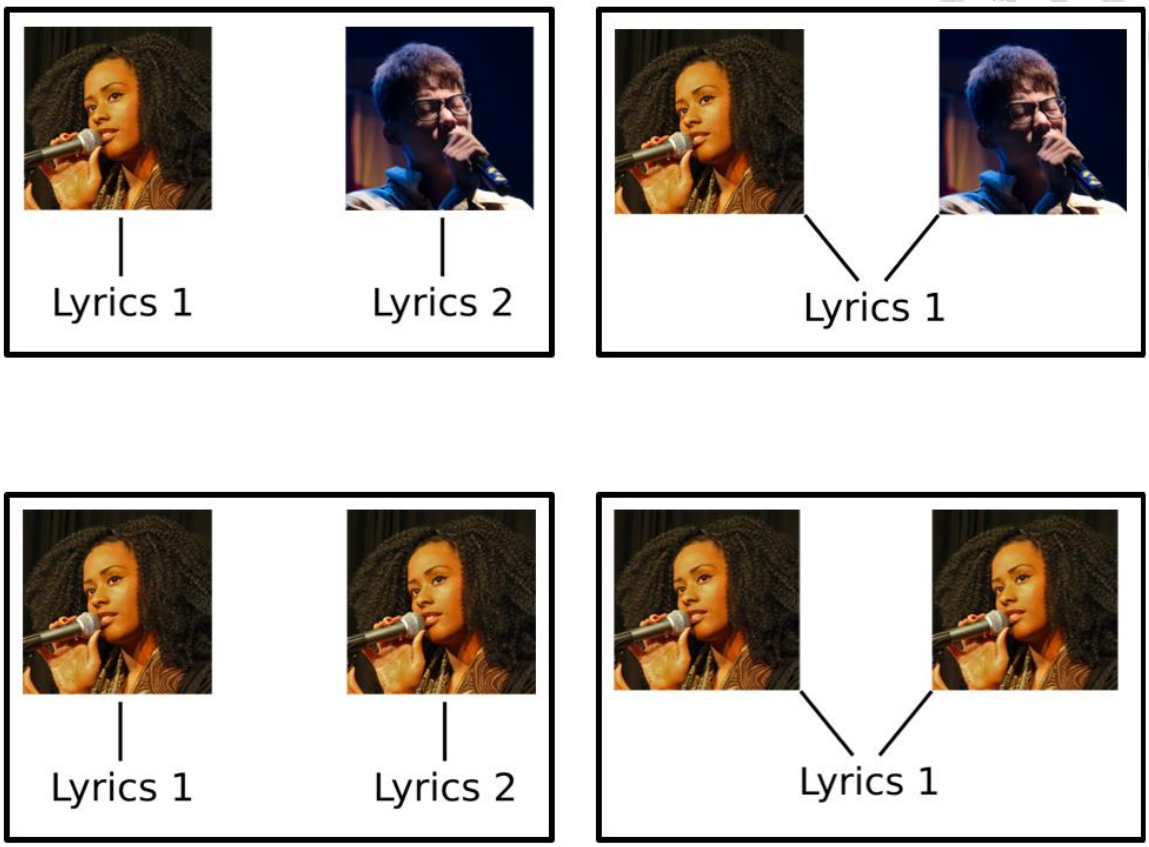


Figure 1.1. Possible configurations of two simultaneous singers shown visually. (Top left) Two different singers singing different lyrics. (Top right) Two different singers singing the same lyrics. (Bottom left) Two simultaneous recordings of the same singer singing different lyrics. (Bottom right) Two simultaneous recordings of the same singer singing the same lyrics.

1.3 Applications

There are some practical applications for a high-performance singing separation model. The first would be the ability to make customized audio tracks for singers to practice singing in harmony. Currently, there does not exist a real solution for singers to easily practice singing in harmony with other real voices. Their best options are to either sing along with a full recording of all the singers, or take the time to create their own recording,

doi:10.6342/NTU202303444

neither of which are ideal. The ability to generate this kind of audio automatically would be a large advantage for singers trying to practice. Second, similar to what is already being done with music separation software, isolated vocals could be taken from existing songs and used by music producers as samples in the creation of new songs. Finally, since there exist many folk songs which are purely vocal, with complex harmony, and which are not yet transcribed, the ability for a musicologist to separate this type of song into its individual lines and transcribe each of them separately would make this process much simpler and faster.

doi:10.6342/NTU202303444

Chapter 2 Literature Review

2.1 Speech Separation

Since speech separation papers typically report a metric known as SDRi or SI-SDRi, which is separate from the metric of SDR which we will be utilizing, and will be discussed further in section 3.5, the specific performance score of these speech separation models will not be included.

2.1.1 TasNet

The original TasNet was quite a groundbreaking speech separation model and has served as the basis for many more, such as Conv-TasNet [2], Dual-Path RNN [3] and, SepFormer [4], since its release in 2018. The name stands for “Time-domain Audio Separation Network”, and as the name suggests, it separates audio in the time domain, as opposed to the frequency domain like the majority of music separation networks. Separating in the time domain allows for an increase in the maximum possible quality, when compared to only using the frequency domain, as no phase information is lost during the process. It

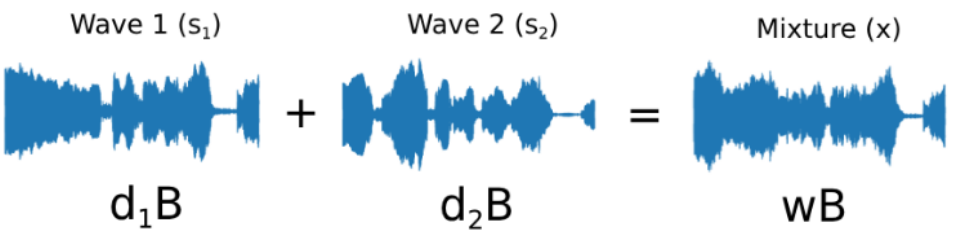
doi:10.6342/NTU202303444

also allows for lower latency applications, such as use in hearing aids, since there is no costly conversion to and from spectrograms [1].

The main principle of TasNet is fairly simple: A mixture of sources (speaking audio), we will call it *x*, will be represented by a basis signal *B* multiplied by a weight vector *w*. Each individual source we can call s_i, which is equal to *B* multiplied by its corresponding weight vector, d_i (Figure 2.1 - 1). Since the mixed signal *x* is the sum of all the source signals s_i, that also means the mixed weight vector *w* is the sum of all the source weight vectors d_i (Figure 2.1 - 3). This then means we can find a mask vector for each s_i, we will call it m_i, which we can multiply by *w* to return d_i (Figure 2.1 - 4), and then multiply d_i by *B* to return s_i. Once this is done for every s_i we are finished the separation process.

doi:10.6342/NTU202303444

$$\begin{cases} \mathbf{x} = \mathbf{w}\mathbf{B} \\ \mathbf{s}_i = \mathbf{d}_i\mathbf{B} \end{cases} \quad (1)$$



(2)

$$\mathbf{w} = \sum_{i=1}^{C} \mathbf{d}_i \quad (3)$$

$$\mathbf{d}_i = \mathbf{m}_i \odot \mathbf{w} \quad (4)$$

Figure 2.1. Principle of TasNet. (1) The mixture signal can be represented by a weight vector *w* multiplied by a basis signal *B*. Each source can be represented by their own weight vector d_i multiplied by the same signal *B*. (2) the mixture waveform is the sum of all source waveforms which means (3) *w* is the sum of all d_i for *C* sources. This means that (4) for each d_i there must exist a mask vector m_i which can be multiplied by *w* to return d_i.

TasNet, and most of its subsequent evolutions are divided into three modules (Figure 2.2). The "Encoder" module first estimates the weight vector *w* which will be used for the separation. Next, the "Separation" module estimates each of the mask vectors m_i for each source *i*. This module does the bulk of the work and is the section which undergoes the majority of the changes between each evolution of the network. Finally, is the "Decoder"

doi:10.6342/NTU202303444

module which renders the audio from each of the mask vectors, m_i, and the basis signal B [1]. The block diagram for this model is shown in Figure 2.2.

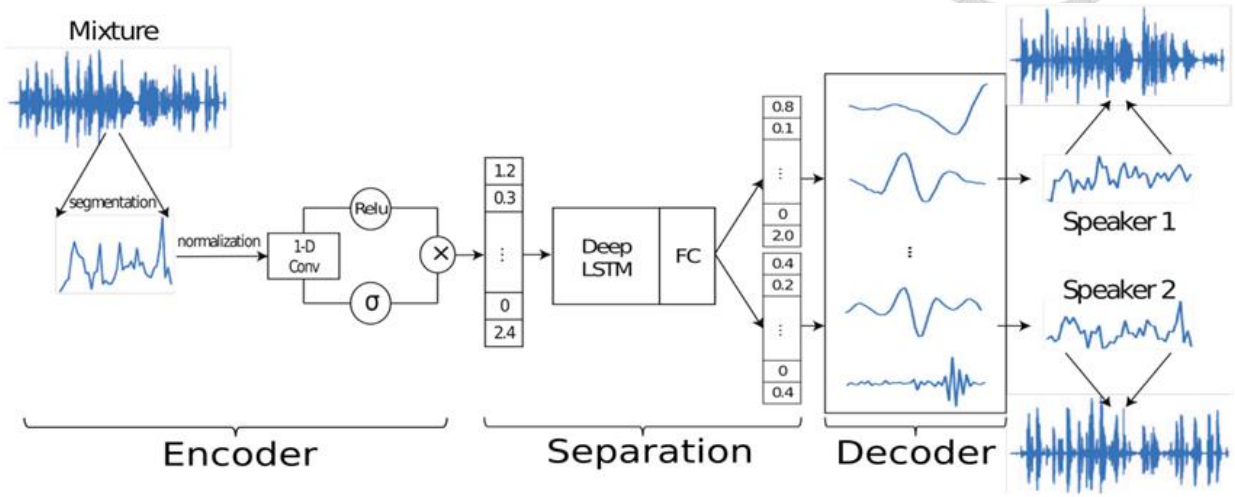


Figure 2.2. Block diagram of TasNet network. Split into "Encoder", "Separation" and "Decoder" modules. [1].

TasNet is trained on the WSJ0-2mix dataset, which contains 40 hours of data, split into 30 hours of training data and 10 hours of validation data [1].

2.1.2 Conv-TasNet

Conv-TasNet was released in 2019 and is the first evolution of TasNet, which shares many similarities with the original. The main difference between the two is that Conv-TasNet replaces the deep LSTM based separation module of the original with a separation module composed of several 1-D convolutional layers. The structure of this can be seen in Figure 2.3. This approach allows for parallel processing, with the use of skip connections, which is able to increase the performance of the model, as well as the

doi:10.6342/NTU202303444

efficiency, with a much smaller model size and faster processing times than the original TasNet [2].

Conv-TasNet was trained using both the WSJ0-2mix dataset as well as the WSJ0-3mix dataset, and thus is able to separate both two and three simultaneous speakers [2].

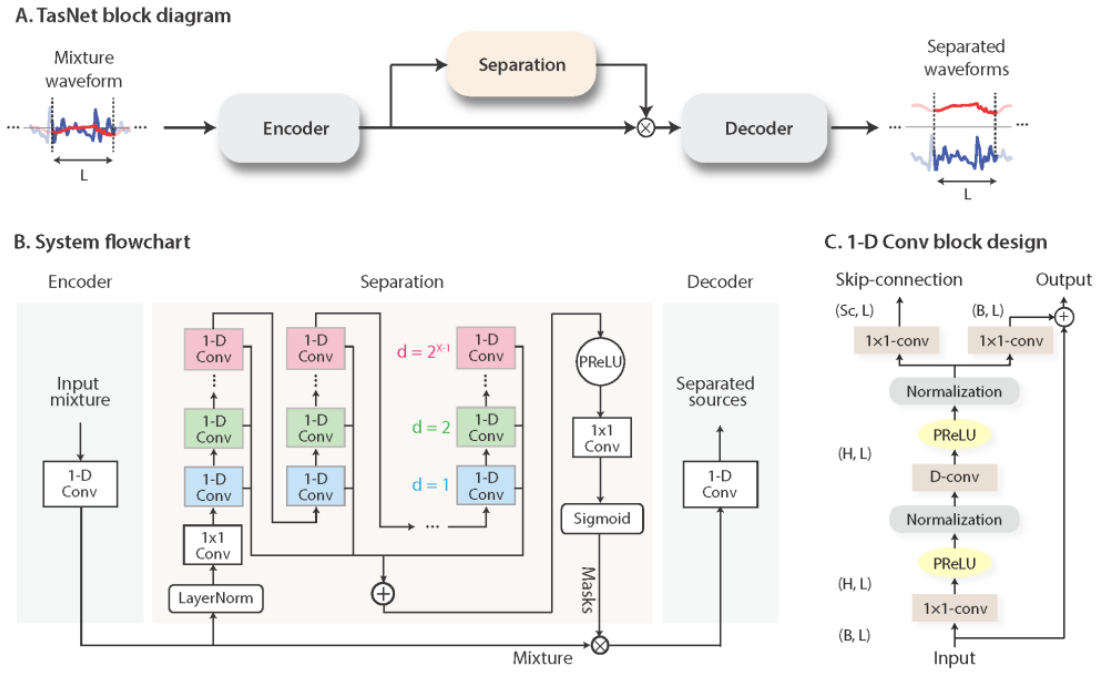


Figure 2.3. Block diagram for Conv-TasNet. (A) High-level diagram of Conv-TasNet, showing the Encoder, Separation, and Decoder modules as well as the skip connection between the Encoder and Decoder. Input and output waveform(s) are also shown. (B) Flow chart of Conv-TasNet architecture. (C) All layers of a single 1-D block in Conv-TasNet. [2].

doi:10.6342/NTU202303444

2.1.3 Dual-path-RNN

This network from 2020 and proposed in [3] is essentially a continuation of the TasNet and Conv-TasNet framework and operates on the exact same high-level encoder-separation-decoder network architecture. The first difference is that this network implements a technique called “segmentation”. Here, the input sequence is cut into segments then, with a default of 50% overlap between segments, they are concatenated into “chunks”, these chunks are then stacked to form a 3D tensor (Figure 2.4 - A). The 3D structure is then fed into two RNNs. The first RNN is applied to each segment individually in sequence, in a “left-to-right” method, and is meant to gather low-level information. The second RNN is applied across all the segments at once, in a “front-to-back” method and is meant to gather high-level information (Figure 2.4 - B). The 3D structure is then outputted and reformatted to its original dimensions using the “overlap-add” method (Figure 2.4 - C) [3].

There are a couple large improvements in this architecture over its predecessors. One is that the ability to analyze both the high-level and low-level structure gives it a very large performance boost in terms of SDRi scores compared to previous models. The other is that cutting the input into chunks and processing it as a 3D structure means increasing the time efficiency from O(N) to O($\sqrt{N}$) [3].

This model was trained on the WSJ0-2mix dataset and is only able to separate two simultaneous speakers [3].

doi:10.6342/NTU202303444

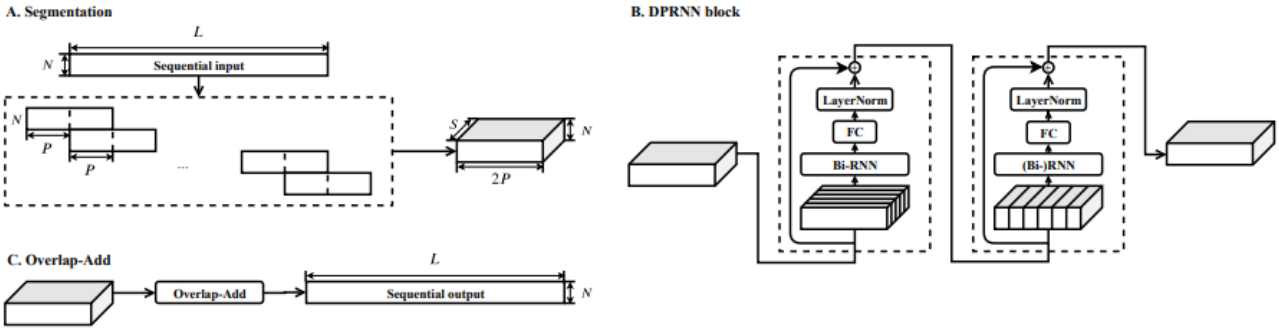


Figure 2.4. Block diagram of Dual-Path RNN. (A) Example of "Chunking" method: here *L* is the length of the sequence, *N* is the number of feature dimensions and *P* is the percentage which the sequences will overlap. (B) Block diagram of the dual RNN architecture. The first RNN reads "left-to-right" collecting local dependencies, the second reads "front-to-back" collecting global dependencies. (C) Example of the overlap-add method, used to return the 3D tensor back to its original dimensions. [3].

2.1.4 SepFormer

One of the next iterations of the TasNet setup is SepFormer from 2021 [4]. This network is similar in principle to Dual-Path RNN, although it replaces the RNNs with transformers. From Dual-Path RNN, SepFormer inherits the ideas of "chunking", as well as using two transformers in a row, one to collect local information, labeled the "IntraTransformer" and the other to collect global information, labeled the "InterTransformer" (Figure 2.5 - Middle) [4]. The largest advantage of SepFormer over the previous models is that transformer networks are able to take advantage of true parallelization, which is not possible in any of the previous networks. This parallelization allows for training and inference to be very fast compared to the previous architectures, leading to an increase in performance and efficiency over all the previous models [4]. The structure of SepFormer is shown in Figure 2.5.

doi:10.6342/NTU202303444

SepFormer is trained on the WSJ0-2mix dataset and the WSJ0-3mix dataset and is able to separate two or three simultaneous speakers [4].

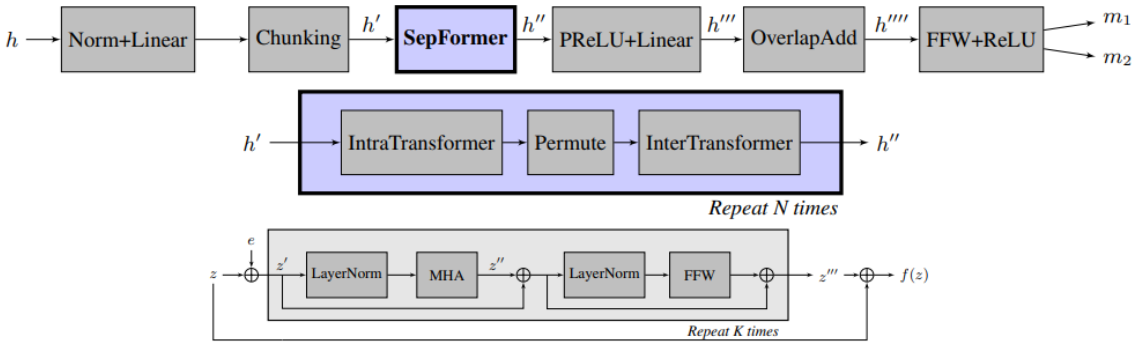


Figure 2.5. Block diagram of SepFormer. Here *h* is a "STFT-like" representation of the audio signal. (Top) The overall network architecture. (Middle) A lower-level representation of the "SepFormer" separation module (light blue block of the top level), with its two transformers, separated by a permutation function. (Bottom) The individual architecture of each transformer block. [4].

2.1.5 Separate and Diffuse

As of writing this, the highest score achieved in a speech separation paper is *Separate And Diffuse* [5] which was published in 2023. Unlike the previous models, this network takes advantage of diffusion to raise the upper limit of what is possible for strictly deterministic models. More specifically, this paper uses a combination of a previously trained deterministic speech separation model, and diffusion-based vocoder. Applying the vocoder to the output of the separation model is able to raise the quality passed what the separation model alone is capable of. The model also trains an alignment module which stops a potential phase shift from occurring in the vocoder. The whole process

doi:10.6342/NTU202303444

architecture is shown in Figure 2.6. When used in tandem with the SepFormer model a SI-SDRi of 23.9 was achieved, compared to the classical upper bound of 23.1. This model is also able to achieve state-of-the-art results for 3, 5, 10 and 20 speakers, but is not able to surpass the classical upper bound in these cases. The vocoder model in this paper was trained on the Wall Street Journal WSJ0 dataset and the LibriSpeech dataset [5]. This model is currently not publicly available.

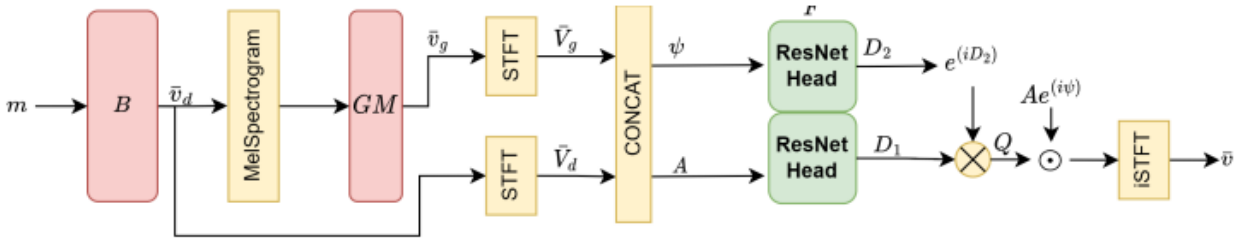


Figure 2.6. Separate and Diffuse suggested architecture. Here, *m* is a mixture of speakers, *B* is any source separation model and *GM* is a diffusion-based vocoder. The "ResNet Head" blocks are the learned alignment networks which prevent a phase shift from occurring. [5].

2.2 Music Source Separation

2.2.1 Deezer Spleeter

One of the main inspirations for the custom singing separation network we will be creating here is Spleeter by Deezer from 2020 [6]. This network uses a fairly simple U-Net based architecture and is able to split a song into up to five parts: vocals, drums, bass, piano and "other". This model is trained on an internal dataset from the Deezer company. It separates each part by having the network predict a soft mask of the spectrogram of the entire audio input, for each of the individual parts it is trying to separate. Spleeter is able

doi:10.6342/NTU202303444

to rival the state-of-the-art models of the time with mean SDR scores of 6.55 dB for vocal, 5.10 dB for bass, 5.93 dB for drums and 4.24 dB for “other”, in its 4-stem separation mode on the standard musdb18 dataset [6].

2.2.2 BandSplit RNN

One of, if not the current highest performing music separation network is BandSplit RNN from 2022 [7]. Like many other similar networks, it also uses spectrograms but chooses to use complex-valued spectrograms to maintain the phase information. BandSplit RNN follows a fairly complex process of splitting the input spectrogram into frequency-bands of predetermined sizes, before normalizing and remerging them into a full-band tensor. This full-band tensor then undergoes a process very similar to Dual-Path RNN [3] wherein it is put through two RNNs in a row, the first analyzing the temporal dimension, and the second analyzing the frequency dimension. Finally, the tensor is processed through a mask estimation module which calculates the tensor mask for each of the target sources. This process is shown in Figure 2.7. BandSplit RNN was able to achieve very high results scoring mean SDR values of 10.01 dB for vocals, 6.80 dB for bass, 8.92 dB for drums, 6.01 dB for “other”, and 8.24 dB overall [7].

BandSplit RNN was trained on the commonly used MUSDB18-HQ dataset [7].

doi:10.6342/NTU202303444

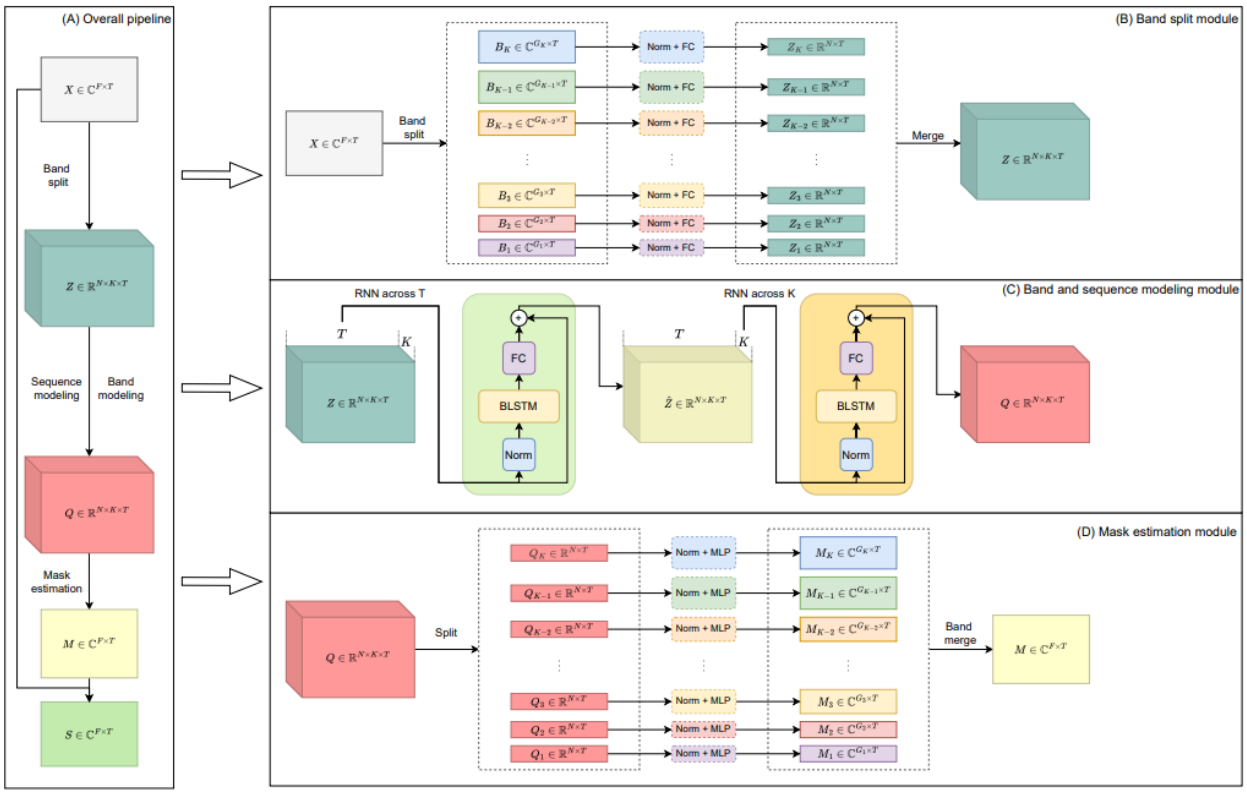


Figure 2.7. Block diagram of BandSplit RNN. (A) Overall network diagram. (B) Band split module, which divides the input into specific frequency bands, normalizes, then remerges them. (C) Band and sequence modeling module, which contains two sequential RNNs to collect audio dependencies. (D) Mask estimation module, which creates the final masks used to estimate each audio part. [7].

2.3 U-Net

The U-Net architecture was originally proposed in [8] for use in biomedical imaging but has seen many other uses since its conception in 2015. It is a type of convolutional neural network containing two symmetrical sides. First, the layers of the encoding side downsample the vertical and horizontal resolution of the input tensor using max pooling layers, while subsequently increasing the number of feature channels. The decoding

doi:10.6342/NTU202303444

layers then do the opposite by upsampling the tensor resolution, while simultaneously decreasing the number of feature channels. Each of the symmetrical layers in the network are also connected via a “skip” connection. These skip connections concatenate each layer of the encoding side of the network with the corresponding layer of the decoding side of the network. These connections allow for the extraction of information at both the local and global level and are very powerful for information extraction [8].

The standard setup for the U-Net architecture is for each level of the encoder section to have two sets of convolutional 2D layers. The first of these layers takes the input at the current number of feature channels, and outputs it at the newly desired number of feature channels, typically double the input, with the exception of the first layer, which increases the input from 1 feature channel. These are then followed by a rectified linear unit activation function. The second of these layers typically does not raise or lower the number of feature channels or the resolution. These are then followed by a 2x2 max pooling layer which downsamples the resolution of the image to one fourth of its original size, halving both the x and y dimensions. This entire process is repeated for the desired number of encoding level before reaching the “center” section, which does the same thing except without the 2x2 max pooling layer. Instead, a 2x2 upsampling convolutional layer begins to be used at the end of every section to double the x and y dimensions of the tensor, as well as halve the number of feature channels. The output of the center section is then sent into the decoder section, which has an equal number of levels as the encoder. Each time the tensor is upsampled it can be concatenated, via the skip connection, with its corresponding tensor on encoding side of the “U” (the tensor with the same dimensions as it). After the tensor is concatenated it needs to be cropped to the correct dimensions, and then goes through the same convolutional layer and rectified linear unit activation

doi:10.6342/NTU202303444

functions as all the previous levels. The final section of the network functions the same way but returns the image to one feature channel via a 1x1 convolutional layer [8]. All of this is shown visually in Figure 2.8.

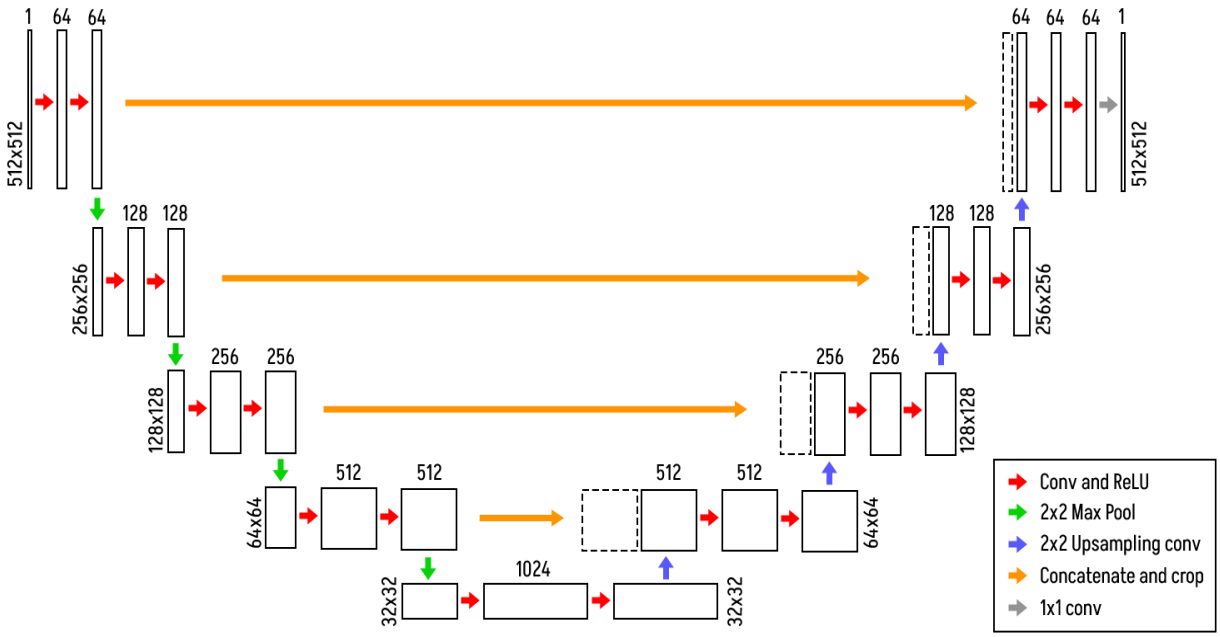


Figure 2.8. Diagram of the U-Net convolutional network. The legend on the bottom right shows the type of procedure each colour of arrow represents. Each rectangle represents a tensor with the input to the network being on the top left and the output of the network being on the top right. The number on the top of each tensor shows the current number of feature channels. The numbers on the left or right of each tensor show its current 2-dimensional resolution. The dotted rectangles represent the tensor being concatenated via the “Concatenate and crop” connection before it has been cropped.

doi:10.6342/NTU202303444

Chapter 3 Methodology

3.1 Datasets

This project will mainly utilize three different datasets, with one supplementary dataset. The *Dagstuhl ChoirSet* [9] is an audio dataset recorded from 13 different singers over four choir vocal parts, soprano, alto, tenor and bass, otherwise known as SATB, for the two choir songs, Locus Iste by Anton Bruckner, and Tebe Poem by Dobri Hristov. Each singer is recorded on three different types of microphones, a dynamic microphone, a headset microphone and a larynx microphone. For our purposes we will be exclusively using the dynamic microphone recordings, as they provide the clearest audio. The dataset also contains CSV representations for each part of both songs; however, we will not be using this in our experiments.

The *Choral Singing Dataset* [10] is a choir dataset recorded from 16 different singers, also over each of the four SATB voice parts. This dataset contains recordings of the songs Locus Iste by Anton Bruckner, Niño Dios by Francisco Gurrero and a Catalan folk song titled El Rossinyol. Unfortunately, there is a one song overlap between these two datasets, so we will only be using the recordings of Locus Iste by Anton Bruckner from this dataset. The Choral Singing Dataset also contains MIDI representations of each song, but we will

doi:10.6342/NTU202303444

not be using those for our purposes. The total audio of both choir datasets together, not including the overlapping song, amounts to approximately 40 minutes.

Finally, the MedlyDB dataset is a general music dataset with 196 songs between two versions [11], [12]. The songs in this dataset contain the full "mixed" version of each song as well as the individual audio tracks, or "stems," they are composed of. Unfortunately, since it is such a general dataset, not all the songs have vocals, and many are purely instrumental. In addition to this, even though every stem is meant to be an individual part of the song, some songs still have vocal harmony, or even instrumental parts recorded along with the lead singer's vocal track, which makes them unusable for our experiments. Because of this, each song in the dataset needs to be curated to first check if it has vocals at all, then find the correct vocal track, and finally remove any non-vocal part, if possible. This is a time-consuming process and is unrealistic to do for all 196 songs, but it also gives the opportunity to edit out any long periods of silence from the audio. This is important since many lead vocal tracks in these songs tend to be predominantly silence. Keeping this silence would likely not yield the best results as it would make many of the samples being used have only one singer at a time, or just have no audio whatsoever. On this relatively small scale it was possible to do this removal manually, however on a larger scale it could, and should be made automatic via voice-activity detection (VAD) or end-point detection (EPD). A total of 24 vocal tracks were taken from this dataset, all from different songs. The total audio of these vocal tracks amounts to approximately 61 minutes.

Initial testing was also done on trying to use music source separation software to retrieve the vocal tracks from songs, which could then be used in the same way we will use these

doi:10.6342/NTU202303444

vocal tracks. We found when attempting this that the overall quality of the separated vocals would be too low to use for our purposes, as we wanted to have as clean audio as possible.

Although the audio from all of these datasets is generally quite clean, it does have the problem of audio “leaking” in some sections. This problem comes from multiple audio tracks being recorded simultaneously in the same environment. Even with relatively high-quality microphones, if audio sources are too close together, audio other than the main source can be picked up in the background of the main recording. When tracks are used in a full song this is usually not an issue, but since in our case we would like the highest possible quality isolated vocals, this is a somewhat limiting factor. This effect occurs more in the choir recordings, since they are necessarily all recorded together, but can also be heard on some of the vocal tracks in the non-choir songs.

Since a cappella data is limited, it would be possible during future research to use these datasets to create different two-singer scenarios for training and testing. The most basic scenario, which we will be using, is simply to combine two different singers singing different words. This is done by randomly selecting vocal tracks and combining them together. The second configuration is the same singer singing different words. This can be created by dividing a song and recombining it with other samples of itself. The final possible configuration which can be created, is two different voices singing the same words. Firstly, it should also be noted that due to the nature of choir music, some of the choir combinations we are using, naturally fall into this category. Artificially, this can be achieved by taking a sample and recombining it with a pitch-shifted version of the same sample. Intuitively it may seem like using the same voice would result in the scenario of

doi:10.6342/NTU202303444

the *same* voice singing the same words, however, the timbre of a voice is determined mostly by the frequencies which do not change when a person is speaking or singing. Because of this, pitch shifting the voice essentially has the effect of changing the voice entirely. This also means that it is essentially impossible to "fake" the scenario of having the same singer singing the same words. Using these different combinations, it should be possible to test exactly how separation models react to each of these three different scenarios, as well as potentially train them how to perform better on each.

Finally, for testing the speech separation performance of the speech separation models we will be using the LibriSpeech Dataset. This is a very large-scale dataset at a total of 1000 hours of speech data, although we will only need a very small portion of this data. This dataset is available in 16 kHz, however, due to the limitations of the speech separation models, we will only be using 8 kHz audio while working with them, this applies to both the speech and singing data.

To create our speaking samples, we simply took 200 5-second (see section 3.2) speech segments from the LibriSpeech dataset and randomly combined them such that no segment was used more than once, leaving us with 100 mixed speaking segments. Since the LibriSpeech dataset is organized by the source of the audio, if we take all our segments from different sources, it is very unlikely that there will be any overlapping speakers. As the speech separation models take audio samples as input, no post-processing is needed.

While we did not end up using any data augmentation techniques in our experiments, there are a few which would potentially be applicable here. The first would be pitch transposition, where we would simply shift the pitch of entire audio tracks up or down

doi:10.6342/NTU202303444

and treat them as though they were essentially new data, recombining them into new mixtures. Next is time shifting, wherein we shift the timing of one audio track relative to the one it is mixed with to create a new mixture using the same songs. Finally, we could simply recombine the solo audio tracks we use here to create new mixtures, since we are only using a small number of the possible unique combinations of songs. Even though this procedure can be done a very large number of times, there may be some disadvantages to using the same audio tracks too many times, such as the model simply memorizing the entirety of song, leading to quick overfitting.

3.2 Data Preprocessing

Since choirs, and thus the audio data from them, are normally divided into four voice parts, it seemed natural to use the combinations of those parts as the way to create our singing mixtures. As such, we kept each of the choir songs together in their SATB groups, then created mixtures out of each of the six possible unique pairings: alto with bass, alto with soprano, etc. This way the parts are combined as to maintain their original orientation, and each combination of parts is represented. This seemed similar to the way the model may be used in practice. Although the songs from MedlyDB do not have the same musical context as the choir songs, we mirrored the method of forming combinations, as it seemed to work well. In this case, we simply arranged all the vocal tracks by duration and gathered each individual group of four, so that there were no overlapping songs. Then for each group took all six possible pairings in the same way we did for the choir songs. This is shown in Figure 3.1 - 1 “Merge”.

doi:10.6342/NTU202303444

Once we have created all the combinations of songs we will use, both the mixtures, and the solo audio tracks must be divided into shorter audio files. This is to prevent running out of memory while training, and also to help make the data more manageable, as it makes it easier to add and remove data in smaller quantities. For training we decided to use audio segments of five seconds. This was a short enough length for the network to process without any problems, but also was long enough to retain a relatively large amount of information per sample. This is shown in Figure 3.1 - 2 “Cut”.

After the audio segments have been cut to the proper length, they need to be converted into spectrograms (Figure 3.1 - 3 “Convert”) using short-time Fourier transforms (STFT), which are then processed directly by the network (Figure 3.2). Spectrograms work very well for our purposes here, however, since we are not using complex-valued spectrograms, this means they only retain the magnitude of the audio and contain no phase information. This lack of information can lead to ambiguity when converting them back to audio, creating audio artifacts, and this leads to an upper bound in the possible audio quality. All of the audio pre-processing steps are shown visually in Figure 3.1.

doi:10.6342/NTU202303444

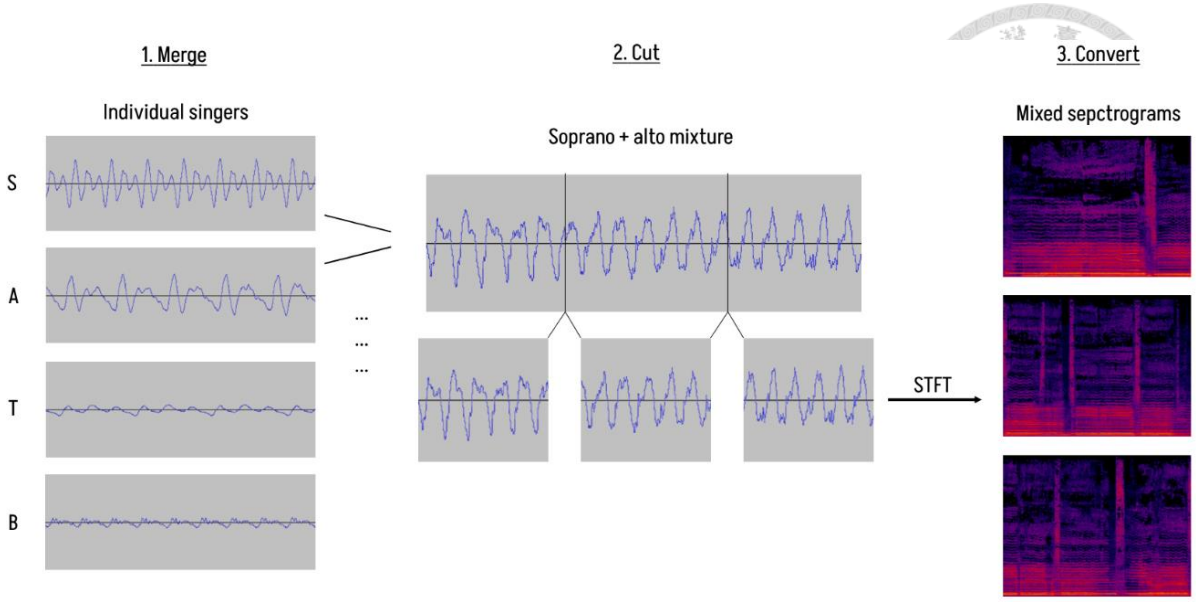


Figure 3.1. Full audio processing diagram. (1) Group of four audio tracks merged into all six possible unique pairings. (2) All audio tracks cut into shorter segments, in our case five seconds in length. (3) All audio segments converted into spectrograms via STFT.

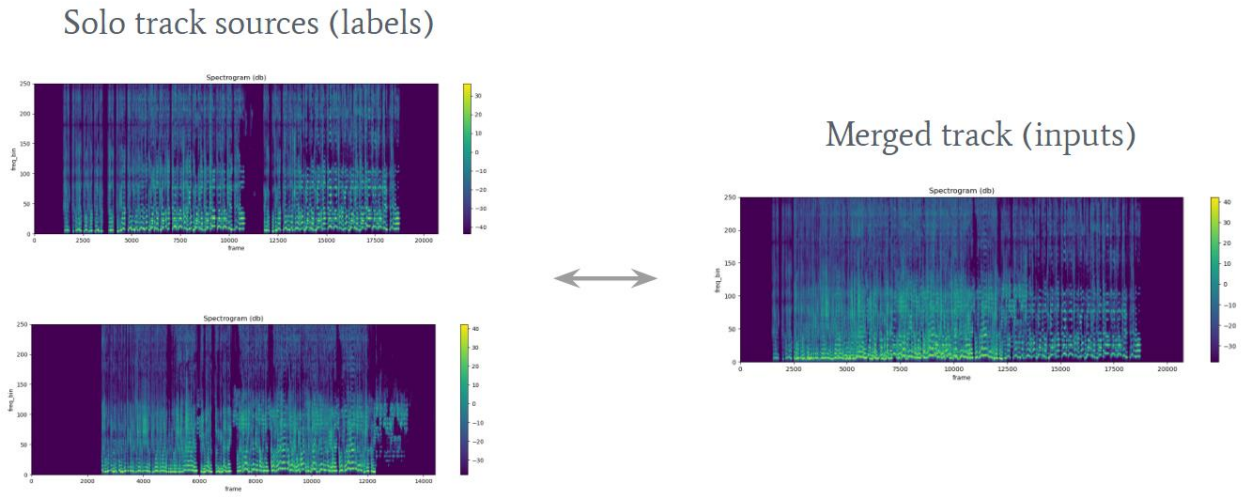


Figure 3.2. Spectrogram labels and inputs. This figure shows an example of input and ground truth label spectrograms. On the left are the spectrograms of two separate singing clips which act as the ground truth labels. On the right is the spectrogram of both singing tracks combined, which acts as our input.

doi:10.6342/NTU202303444

All singing data used with our custom U-Net models will be using mono audio with a consistent 44.1 kHz sample rate, converted into spectrograms using an n_fft value of 2046, a hop length of 512 and a window size of 0. This is done using python's "librosa" package.

3.3 Data Postprocessing

After the spectrograms have been estimated by the model, they have to be converted back into audio waves so they can be both heard and tested for quality. There were, unfortunately, some unforeseen issues which arose at this step. The best way to reconstruct the audio from the predicted spectrograms would be with the Griffin-Lim function. The Griffin-Lim function aims to reconstruct the phase of a spectrogram by initially using pure noise as the phase information, and iteratively applying inverse short-time Fourier transforms (iSTFT), followed by short-time Fourier transforms to the spectrogram, which refines the phase information over time [13]. This method is not perfect, but considering no phase information is available at the beginning, it works fairly well, and is better than a simple iSTFT for aural purposes. The problem, however, came when trying to use the testing program (the evaluation metrics will be discussed in the section 3.6) since, as it turns out, the evaluation function assumes that the audio you are testing will be perfectly in phase with the audio you are testing it against. If this is not the case, then the results will be skewed extremely negatively. As such, it was not possible to use the Griffin-Lim function while testing our model performance. Instead, the phase information had to be taken directly from the original audio mixture that was being separated and applied to each of the solo singing predictions, that way the phases would align, and the test results would be accurate. This was also achieved using the librosa python package. The simplified python code looks something like this:

doi:10.6342/NTU202303444

```
#Retrieve the phase of the original mixture
stft = librosa.stft(audio_mix, n_fft=2046, hop_length=512)
stft_phase = np.angle(stft)

#Apply the phase to the predicted spectrograms
stft_prediction1 = prediction1 * np.exp(1j * stft_phase)
stft_prediction2 = prediction2 * np.exp(1j * stft_phase)

#Use iSTFT to return the new spectrograms to time domain
predicted_audio1 = librosa.istft(stft_prediction1, hop_length=512, n_fft=2046)
predicted_audio2 = librosa.istft(stft_prediction2, hop_length=512, n_fft=2046)
```

It should be noted that the Griffin-Lim function was used to generate results for listening purposes, as it is less prone to audio artifacts.

3.4 New Network

All of the existing networks we will be considering are based off of the original TasNet architecture [1] and thus share similar properties. One of these is that they all function in the time domain, instead of the frequency domain. This is cited in the original TasNet paper [1] as a way to reduce latency, since to function well, the resolution of the generated spectrogram needs to be quite high, and this conversion takes time, and also to potentially improve quality, since there are limitations to converting non-complex-valued spectrograms back to audio. Since in our case latency is not a concern, and the upper bound of what is possible without taking into account the loss of phase information is not being pushed, we believe it is worth trying something very different than TasNet for the sake of comparison.

The type of architecture chosen here is the U-Net [8] (described is section 2.3), which has been shown to be quite powerful in other music source separation networks, including

doi:10.6342/NTU202303444

being used in current state-of-the-art models such as Hybrid Transformer Demucs [14]. Since this type of network is designed for segmenting images, it is able to make good use of limited data sets, can extract high-level and low level-detail, and is very efficient to run. As spectrograms can simply be treated as greyscale images, it seems to be an excellent choice for our purposes.

There are two different general formats this model can take. Both of these formats take a spectrogram of the mixture of two singers as the input, but the difference comes in the spectrogram(s) the model will output. The first type will be known as the “parallel” type of model. In this type of model one spectrogram of the singing mixture is given as the input, and two separate spectrograms are returned simultaneously as the output, one for each singer. A diagram of this type of model is shown in Figure 3.3. The other approach is the “subtractive” type of model. In this type of model, the mixture is still given as the input, but only the spectrogram of the first singer is returned as the output. Then, since the original mixture is the addition of both singers in the time-domain, we can use spectrogram subtraction to subtract our estimation of the first singer from the original mixture to return the spectrogram of the second singer. A diagram of the subtractive approach is shown in Figure 3.4.

doi:10.6342/NTU202303444

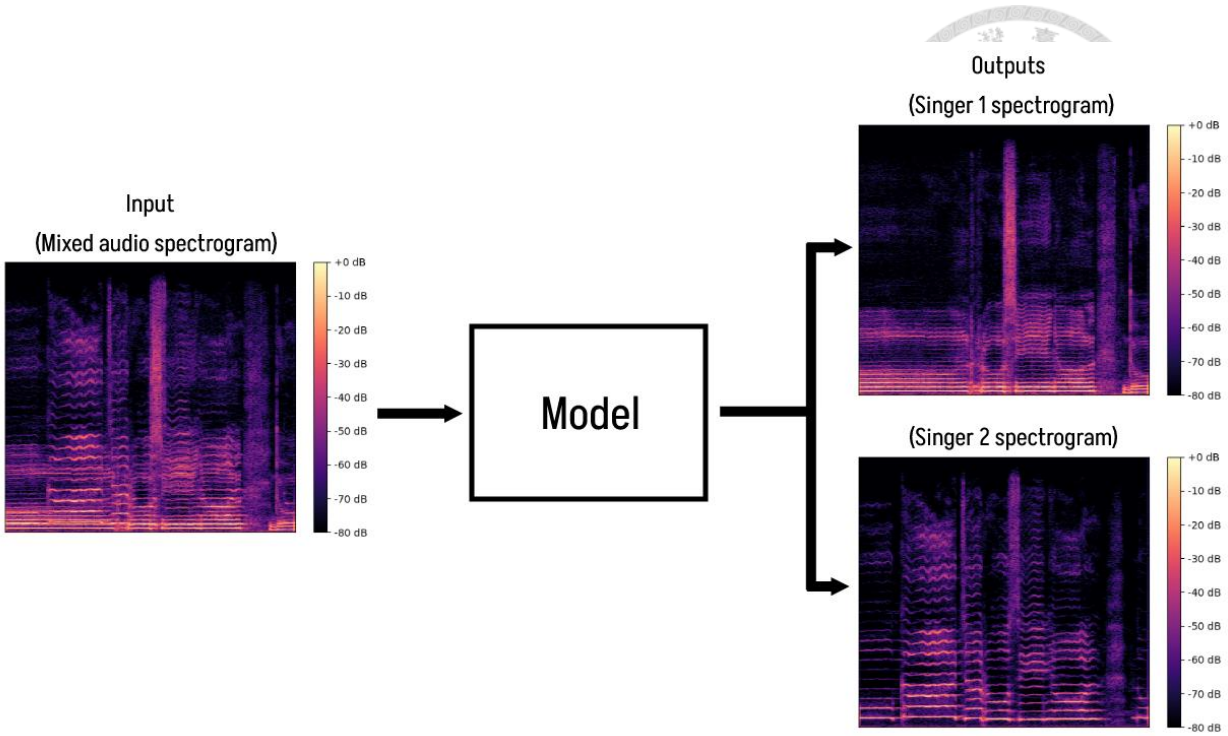


Figure 3.3. Diagram of the parallel-type model. The mixed singer spectrogram is given to the model and both predictions are returned simultaneously.

doi:10.6342/NTU202303444

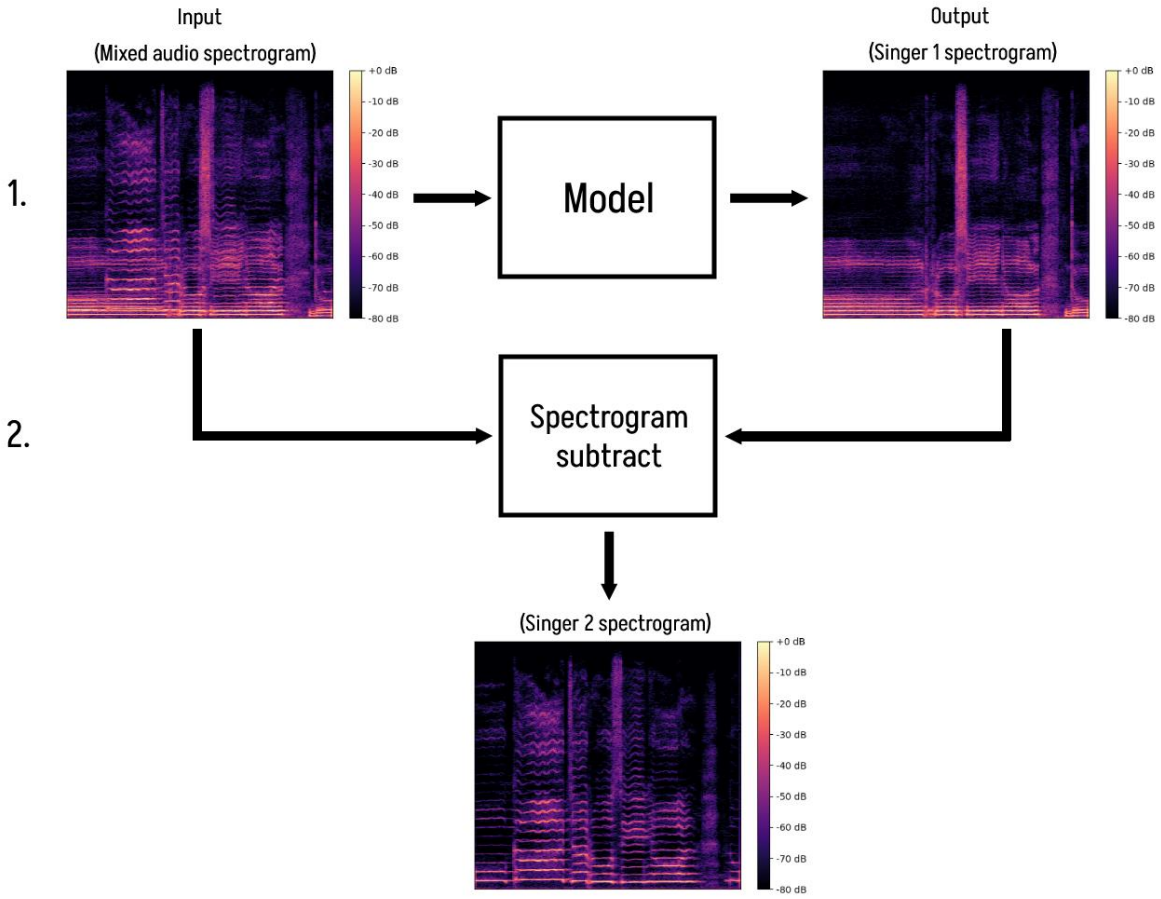


Figure 3.4. Diagram of subtractive-type model. This takes place in two steps. Step 1.) the model is given a mixture and predicts one out of the two singers. Step 2.) The prediction is subtracted from the original spectrogram to return the spectrogram of the second singer.

When it comes to practically setting up the network, using the subtractive approach is somewhat simpler. For the input we just use the spectrogram of the mixture, and for the label we use the spectrogram of the first of the two vocals. Then we only need to use one input and one output in the network and to use a single loss function to compare the prediction to the ground truth spectrogram. In a simplified version of the python code, calculating the loss looks something like this:

doi:10.6342/NTU202303444

```
for input, output in dataloader:
    prediction = model(input)
    loss = loss_function(prediction, output)
```

This method does add an additional step to the post-processing, where the spectrogram subtraction must occur, but that is also quite straightforward and relatively low-cost. However, this version has a fundamental issue. Since there is no phase information in either spectrogram, their subtraction will lead to further audio artifacts due to the ambiguity in the possible phase alignments. Because of this, this method should likely be avoided, at least when not using complex-valued spectrograms.

Setting up the parallel method takes a bit more effort. Firstly, since both labels are being estimated at the same time, our network needs to return two output channels instead of one. This is accomplished by adding a second "output" layer at the end of the network, essentially adding a second gray arrow layer in Figure 2.8, one to output the spectrogram of each singer. Once both predictions are returned, both of their losses have to be taken individually, and the sum of them is used as the total loss. The simplified python code looks something like this:

```
for input, output_1, output_2 in dataloader:
    prediction_1, prediction_2  = model(input)
    loss_1 = loss_function(prediction_1, output_1)
    loss_2 = loss_function(prediction_2, output_2)
    loss = loss_1 + loss_2
```

The entire process from a single mixed audio track to two separated audio tracks is then:

1. Convert audio mixture from a waveform to a spectrogram.

doi:10.6342/NTU202303444

2. a.) Process the two-singer spectrogram through the U-Net model to return the spectrogram of first singer, which can then be subtracted from the mixture spectrogram to return the spectrogram of the second singer.

   Or

   b.) Process the two-singer spectrogram through the U-Net model to return the spectrograms of both individual singers.
3. Convert each spectrogram back to a waveform using the Griffin-Lim function or an iSTFT.

The final network used for our models is quite similar to that discussed and shown in section 2.3. The network we have created also contains four downsampling and upsampling levels and has a maximum of 1024 feature channels. The main differences are that the kernel size of the "horizontal" 2D convolutional layers (red arrows in Figure 2.8) are increased to a very large 8x8, from the basic 3x3 used in the original paper [8] and each are followed by a batch normalization layer, before the ReLU activation function. We are also utilizing the "parallel" type format mentioned above.

This large kernel size means the model requires a relatively high amount of video RAM to process, at approximately 10GB when training with spectrograms corresponding to five seconds of 44.1kHZ audio.

The training objective used in the final network is the sum of mean-absolute-error (MAE) loss for both the spectrograms being predicted. MAE loss was found to perform better than other loss functions tested, mainly mean-squared-error (MSE) loss. This may be because MSE loss is more sensitive to changes in data, and the data being used here

doi:10.6342/NTU202303444

encompasses a very large scope, meaning MAE loss is able to make more, smaller changes, leading to a more generalized model. Using MAE loss is also quite common in other music source separation models, such as the recent Band-split RNN [7]. Using the SDR metric directly would be another possible way to calculate the loss, but since it is much more computationally expensive than other loss functions, it would be unrealistic to use in this case. It was also found that the Adam optimization algorithm with a learning rate of 1e-4 produced the best results.

3.5 Experiments

The experiments done here will center around testing different combinations of models and datasets and will be split into two main parts. 1.) Testing the performance of our newly created singing separation network, and 2.) Testing the performance of previously created speech separation models on separating singing.

First, we start with our newly created U-Net based singing separation network from the previous section. Since the full dataset we are using is divided into two smaller subcategories, choir music and non-choir music, it is natural for us to also train, and test models based on this. We will therefore train three different models, one only using the choir music, one only using the non-choir music, and one using a combination of both choir and non-choir music. Conveniently, both choir datasets have recordings of multiple singers for each voice part (SATB), which allows us to also test the effect of changing only the singers while keeping the same test sample times and voice parts. For both the choir and non-choir models we can withhold an entire “group” of tracks, which is to say one entire song for the choir, and four separate songs for the non-choir, from training, and

doi:10.6342/NTU202303444

test the performance on samples from songs which have not been seen in training as well. Finally for the model trained on the entire dataset (both choir and non-choir) we can test how it performs on the choir and non-choir test samples individually, as well as on them together.

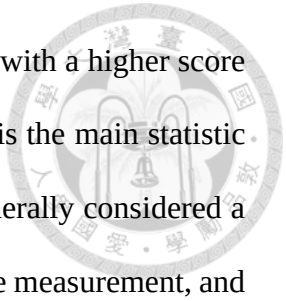
The mixed data model is trained on 976 5-second samples, split into 384 choir samples and 592 non-choir samples. It is validated on 123 5-second samples, split into 48 choir samples and 75 non-choir samples. The choir and non-choir only models are trained and validated on their respective portions of the mixed model training and validation data. The test samples for each model are discussed in detail in section 4.1. Note that all data distribution was strictly done using pseudo-random number generation via the python "random.py" library.

The second part of these experiments is to test the performance of previously made speech separation models. The point of this section is to determine if speech separation is a completely different task from singing separation, and if possible, compare their differences. Each speech separation model used will be tested on 100 5-second samples of two simultaneous speakers, as well as 100 5-second samples of two simultaneous singers, so the difference in results can be compared. We will be testing Conv-TasNet [2], Dual-Path RNN [3] and SepFormer [4].

3.6 Evaluation Metrics

The main evaluation metric used in these experiments is the signal-to-distortion ratio, or SDR, which is essentially the ratio between the energy of the ground truth signal and the

doi:10.6342/NTU202303444



energy of noise caused by the separation within the separated signal, with a higher score being better. We will be measuring SDR in decibels (dB). The SDR is the main statistic that is reported by most music source separation papers, and it is generally considered a good indication of the quality of a separation. However, this is just one measurement, and it can also be worth taking into account other metrics such as the signal-to-interference ratio (SIR), signal-to-artifact ratio (SAR), and also qualitative observations of the separated audio. As noted in section 2.1, speech separation papers usually report signal-to-distortion ratio *improvement* (SDRi) and/or scale invariant signal-to-distortion ratio improvement (SI-SDRi), which we will not be using here.

To perform these evaluations the python library "mir_eval.py" will be used, which contains many useful functions for music information retrieval, including one which automatically evaluates and returns SDR, SAR and SIR.

3.7 Training Specifications

Our custom models were trained on a machine running Ubuntu 20.04.5 LTS using PyTorch 1.13.1 and CUDA 11.6.1. It had an Intel i7-7800X 3.50GHz CPU, an NVIDIA GeForce GTX 1080 Ti GPU with 11 GB of VRAM, and 126 GB of RAM. The models were all trained for a total of 100 epochs, with the model trained only on choir data taking approximately 50 minutes per epoch, the model trained only on non-choir data taking approximately 80 minutes per epoch, and the model trained on both choir and non-choir data taking approximately 115 minutes per epoch.

doi:10.6342/NTU202303444

Chapter 4 Results

4.1 Custom Models

In this section we will report the results of each of the three custom models we have trained. For clarification, we will herein be referring to the test samples which come from songs also used in the training of that particular model as “Regular test data”. Test samples coming from songs never seen in the training data are referred to as coming from “withheld songs”. The bracketed number in each row of the “Test” column of each table indicates the number of test samples in that subset. As mentioned, all measurements are reported in decibels.

First, the results of the model trained on both the choir and non-choir data is shown in Table 4.1 and Figure 4.1. In total there are 123 test samples, which are divided into 71 non-choir test samples and 52 choir test samples. As can be seen, even though the performance of this model on the non-choir data is very low, the performance on the choir data is quite high. The overall average, however, is lowered by the non-choir data, which makes up a larger portion of the entire dataset, and this leaves the total average being subpar.

doi:10.6342/NTU202303444

Table 4.1. Test results of the singing separation model trained on mixed data.

Test	Mean SDR	Median SDR
Total testing set (123)	3.27	3.80
Choir data only (52)	6.80	6.93
Non-choir data only (71)	0.67	1.58

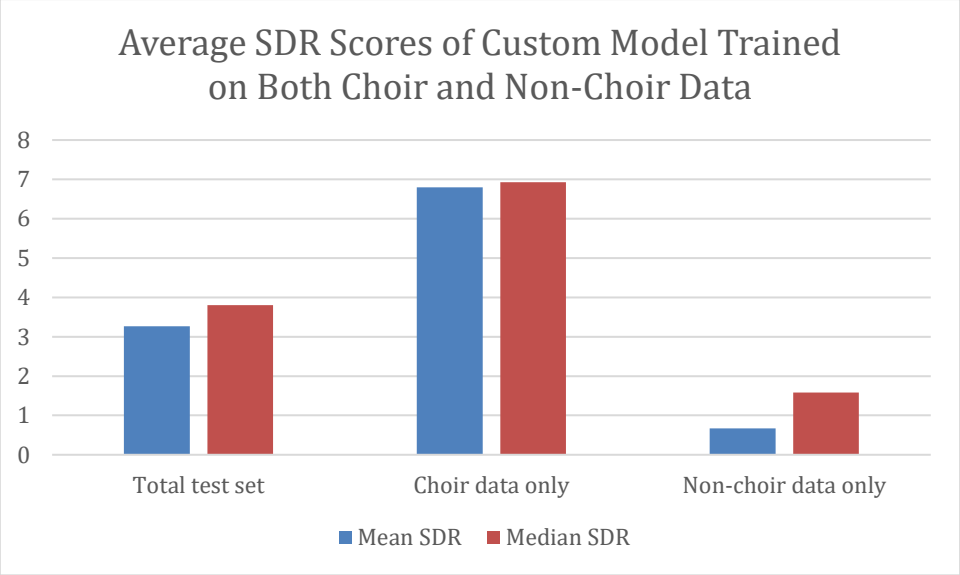


Figure 4.1. Bar graph of results of singing separation model trained on mixed data.

We can then look at the model trained solely on choir data, where one of the choir songs has been entirely withheld to use for testing. The results of these tests are shown in Table 4.2 and Figure 4.2. The test samples for this model are grouped into 38 “regular” test samples, an additional 38 samples using the same points in time and same voice parts, however with all the singers exchanged, and a final 50 test samples from the one withheld song. Clearly, the results here are generally much higher than the previous model, with very high SDR scores in the first two test sets, only showing slight drop in SDR when different singers are used. Testing on the withheld song did lead to a substantial drop, although it still did not result in poor average SDR scores.

doi:10.6342/NTU202303444

Table 4.2. Test results of singing separation model trained only on choir data.

Test	Mean SDR	Median SDR
Regular test data (38)	9.76	9.81
Regular test data from different singers (38)	9.22	8.82
Data from withheld songs (50)	5.65	5.80

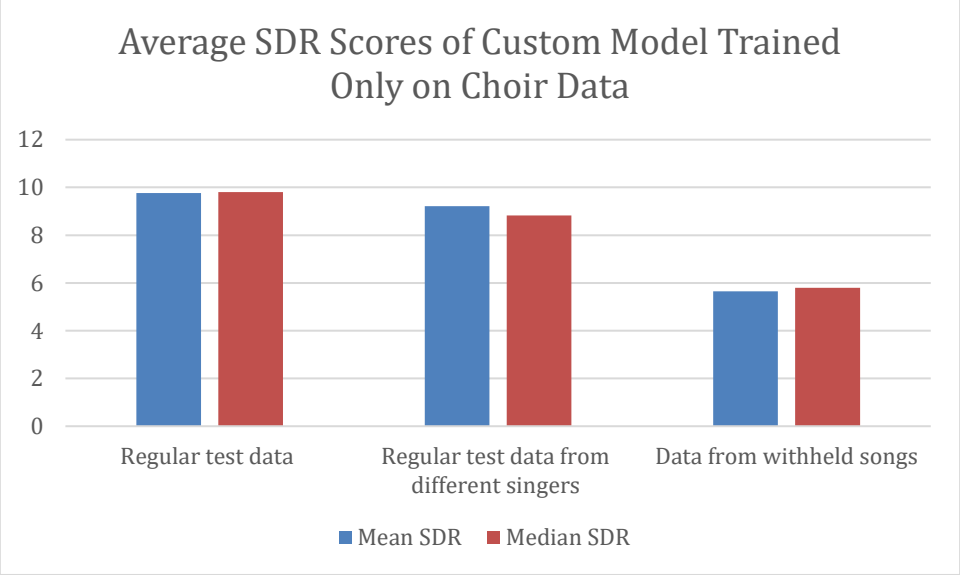


Figure 4.2. Bar graph of results of singing separation model trained only on choir data.

Lastly, we have the dataset containing only the non-choir data. The results of this model are shown in Table 4.3 and Figure 4.3. Test samples here are divided into 59 “regular” test samples, and 50 test samples taken from the withheld group of songs. Withholding these songs is equivalent to withholding one full song from the choir data, and also has the effect of introducing all new singers to the testing pool, since each song has a different singer. Testing on the regular test samples does not result in terrible average scores, but changing to the withheld song test samples does drop the average SDRs to fairly poor values.

doi:10.6342/NTU202303444

Table 4.3. Test results of singing separation model trained only on non-choir data.

Test	Mean SDR	Median SDR
Regular test data (59)	4.09	4.54
Data from withheld songs (50)	1.99	2.47

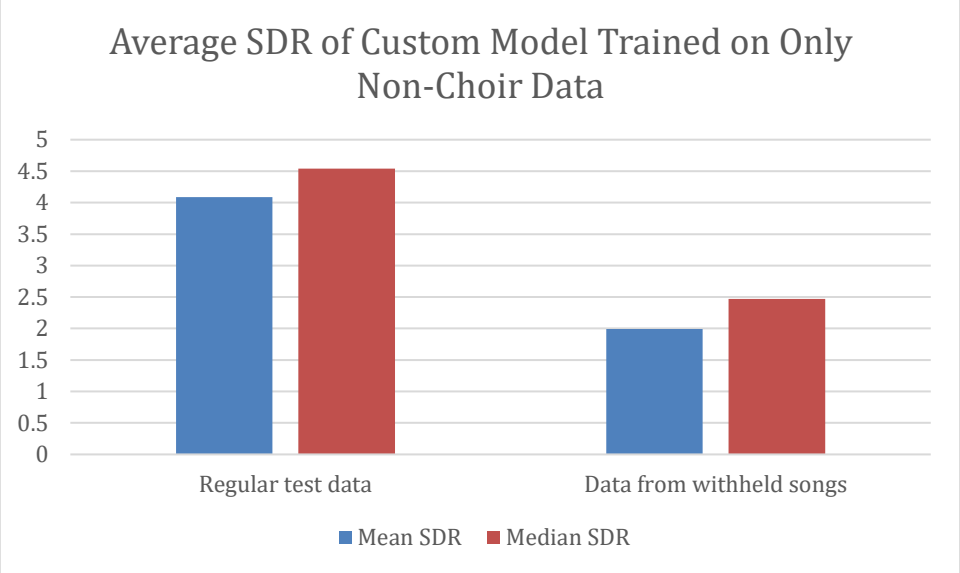


Figure 4.3. Bar graph of results of singing separation model trained only on non-choir data.

4.2 Speech Separation Models

Next, we look at the results of the speech separation models. For each of the speech separation models we report the mean and median averages over 100 samples for both mixtures of two singers and mixtures of two speakers. All of these results are shown in Table 4.4 and Figures 4.4, 4.5 and 4.6. Conv-TasNet and Dual-Path RNN score similarly in both speech and singing separation, with Dual-Path RNN scoring somewhat higher for speech separation, and Conv-TasNet scoring slightly higher for singing separation.

doi:10.6342/NTU202303444

SepFormer outperformed both of the other models, achieving a very high score for speech separation, and a surprisingly good score for singing separation as well.

Table 4.4. Test results of all three speech separation models on both singing and speech separation tests.

Model	Test	Mean SDR	Median SDR
Conv-TasNet	100 singing samples	1.43	1.68
	100 speaking samples	6.12	6.86
Dual-Path RNN	100 singing samples	1.30	1.38
	100 speaking samples	6.60	7.80
SepFormer	100 singing samples	4.69	5.71
	100 speaking samples	9.04	9.51

doi:10.6342/NTU202303444

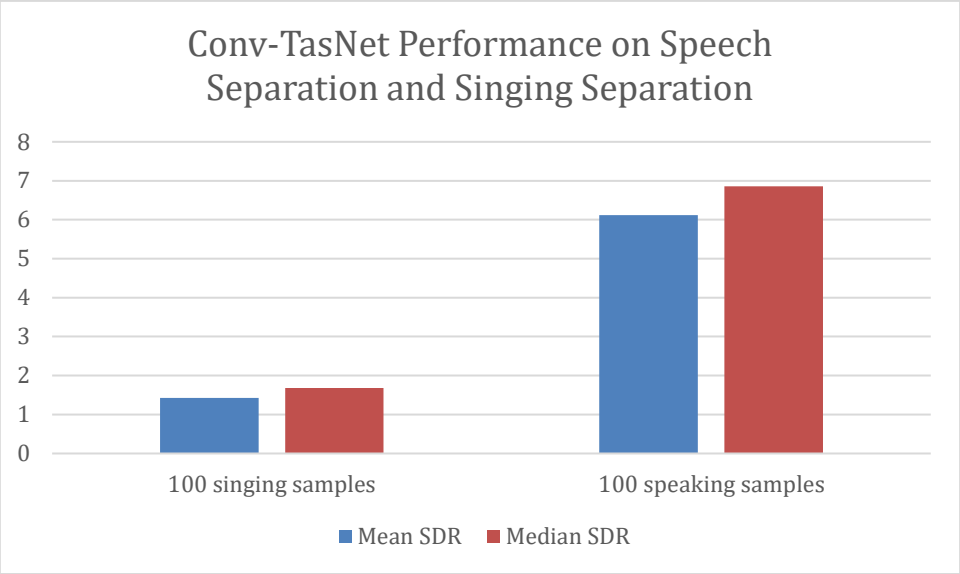


Figure 4.4. Bar graph of comparison of speech separation and singing separation using Conv-TasNet.

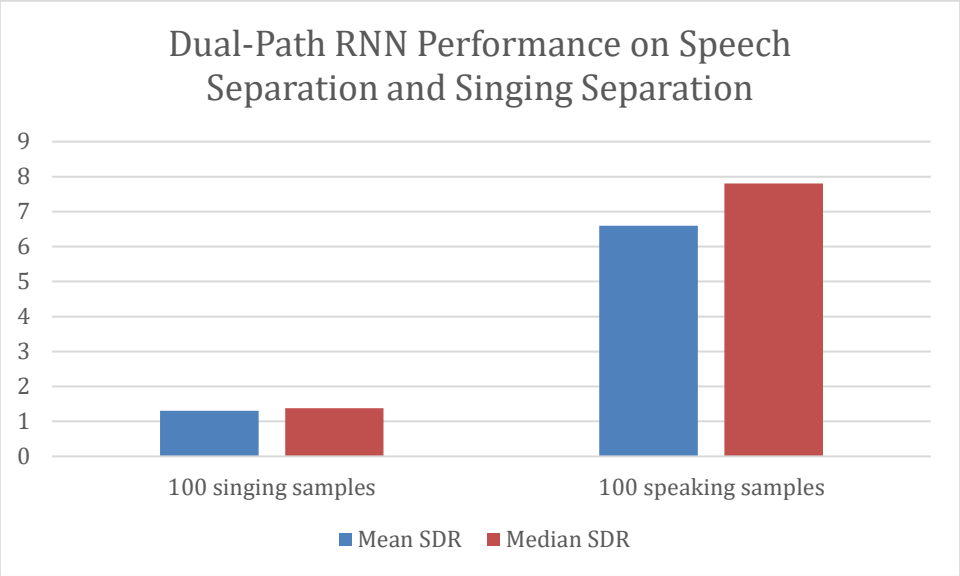


Figure 4.5. Bar graph of comparison of speech separation and singing separation using Dual-Path RNN.

doi:10.6342/NTU202303444

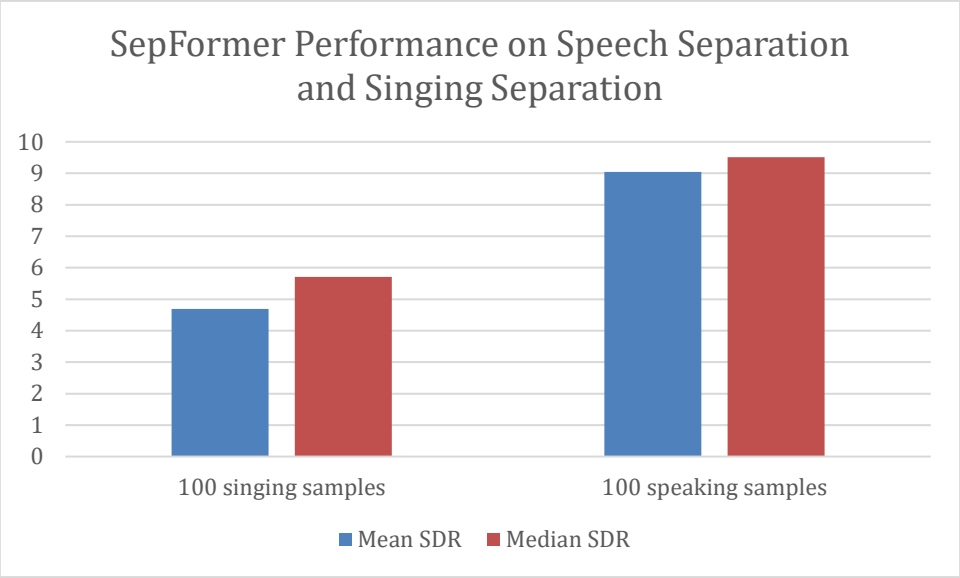


Figure 4.6. Bar graph of comparison of speech separation and singing separation using SepFormer.

4.3 Finetuning SepFormer

Since the performance of the pretrained SepFormer model on singing separation was quite good, we decided it would be worth attempting to finetune the model to perform better on singing separation specifically. As the performance of the other two speech separation models were drastically lower, we saw no reason to also try finetuning these.

This was fairly simple to accomplish using a modified version of the SepFormer training program, which loaded the encoder, separation, and decoder modules from the pretrained model, instead of creating new ones. The dataset used here was the same one used to train our own singing separation models, a mixture of both the choir and non-choir songs, with one choir song and 4 non-choir songs (comprising one full group) withheld from the set

doi:10.6342/NTU202303444

to use for testing. The learning rate was also lowered from the original 1.5e-4 to 0.5e-4. This finetuning program was trained for 20 epochs.

The results of the finetuned version of the model compared to the regular pretrained version of the model, including the difference in performance between the regular test samples and withheld test samples, are shown in Table 4.5, Figure 4.7 and Figure 4.8. In addition, a breakdown of the performance of the pretrained SepFormer model on the choir and non-choir portions of the withheld songs specifically is shown in Table 4.6 and Figure 4.9. This will be discussed in more detail in section 5.1.

Table 4.5. Test results on singing samples from the regular test group and withheld song test group using both the pretrained SepFormer model, and the model finetuned on singing data.

Model	Test	Mean SDR	Median SDR
Pretrained	100 regular singing samples	4.69	5.71
	100 withheld singing samples	4.17	8.34
Finetuned	100 regular singing samples	7.64	9.35
	100 withheld singing samples	6.22	11.00

doi:10.6342/NTU202303444

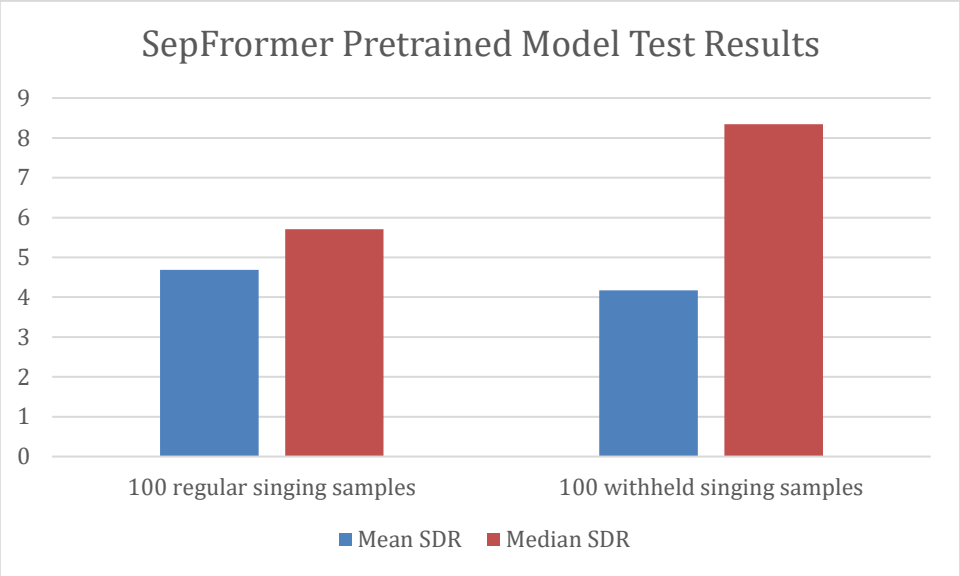


Figure 4.7. Bar graph of results of regular test samples and withheld song test samples using pretrained SepFormer model.

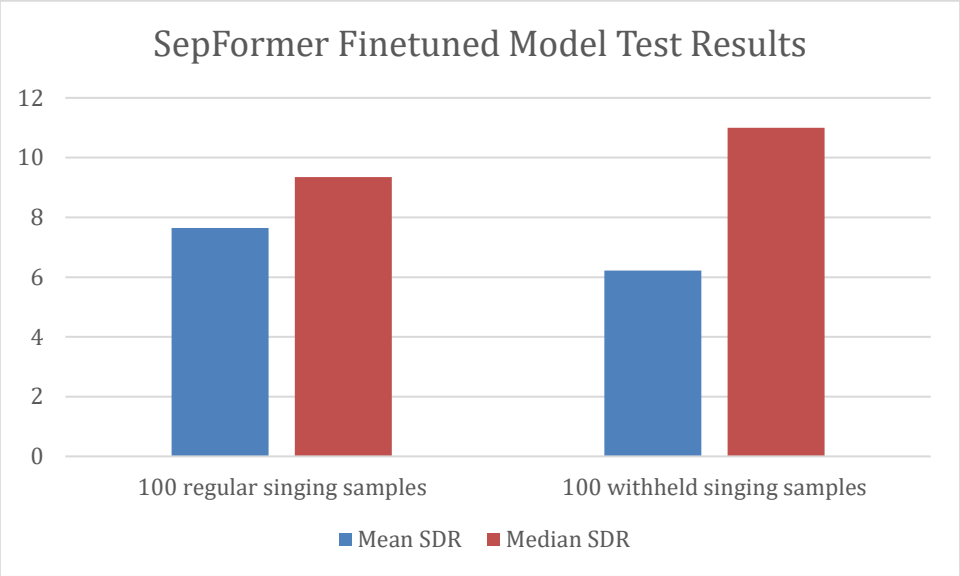


Figure 4.8. Bar graph of results of regular test samples and withheld song test samples using finetuned SepFormer model.

doi:10.6342/NTU202303444

Table 4.6. Test results from the pretrained SepFormer model tested individually on the choir and non-choir portions of the withheld song test samples.

Model	Test	Mean SDR	Median SDR
Pretrained	100 withheld singing samples	4.17	8.34
	Only choir samples	-6.71	-7.16
	Only non-choir samples	8.66	11.07

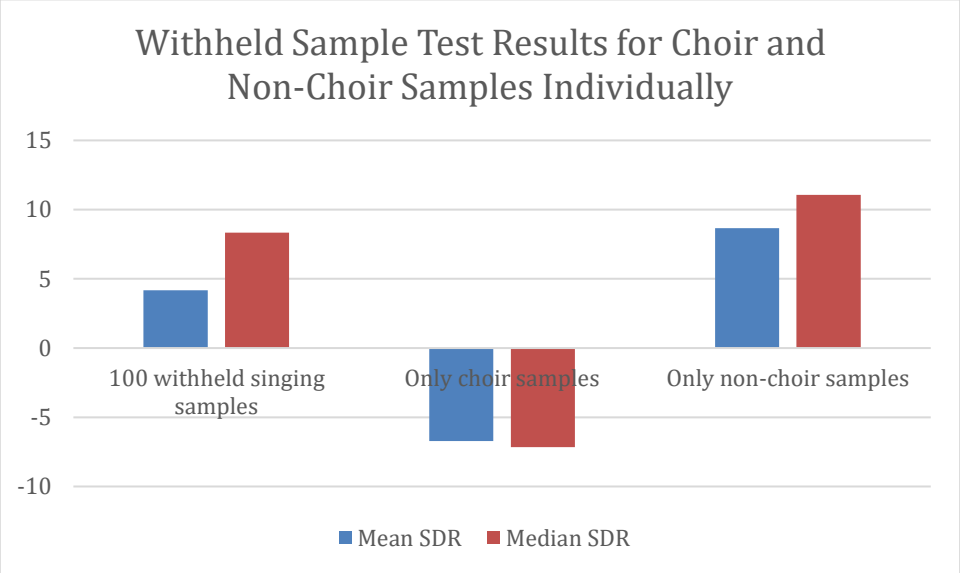
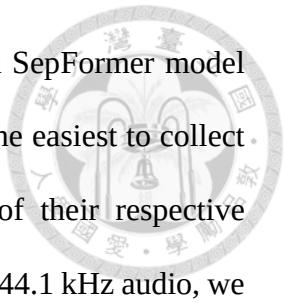


Figure 4.9. Bar graph of results of SepFormer pretrained model tested individually on the choir and non-choir portions of the withheld song test samples.

4.4 Subjective Listening Test

Due to a lack of time, a large-scale subjective test was not possible. However, for the sake of comprehensiveness we decided to still examine some of the audio test results qualitatively. The main purpose for this is that the SDR value is only one metric and does not necessarily reflect the listening quality of the final audio, which is arguably more

doi:10.6342/NTU202303444



important. We will be testing listening examples from the finetuned SepFormer model and from our choir singing separation model, since these should be the easiest to collect high quality examples from and should be fairly representative of their respective networks. Because our custom model has the advantage of outputting 44.1 kHz audio, we will be equalizing all the audio examples to 8 kHz, the output quality of SepFormer.

We wanted to find samples which had fairly good SDR scores (at least 10 dB for each singer), as we wanted to test the models when they were performing well. However, since the models perform so differently, it was difficult to find examples of the same samples achieving very similar SDR scores. So, we had to settle for comparing different samples which had achieved similar results, and this might have some impact on the results.

After listening to several samples from both models, it seems the actual difference in quality created by the models themselves is rather negligible. What was made clear from this test is that the “leaking” issue discussed in section 3.1 is certainly more apparent in the choir than in the non-choir audio, and this makes the non-choir separations sound more “isolated”. However, it appears that that is just a result of the data itself, and not of the models. It is also possible that due to the low audio quality used for these tests, it is more difficult to discern some of the details which would have made larger differences between the performance of the two models clearer.

doi:10.6342/NTU202303444

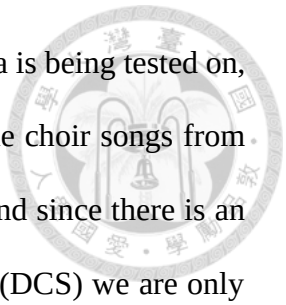
Chapter 5 Discussion

5.1 Custom U-Net Models

It is clear from the results of the previous section that the performance of our new U-Net based network relies heavily upon the category of data it is being trained with and tested on. The most obvious conclusion to be drawn is that the network reacts much better to the choir data than the non-choir data. This is demonstrated both by the mixed data model scoring much higher on the choir test data than the non-choir test data (Table 4.1 and Figure 4.1), and also by the choir-only model generally performing much better than either of the other two (Table 4.2 and Figure 4.2). It is also important to remember that in the case of the mixed data model the choir data represents a smaller portion of the total training data, at approximately 40%, meaning the network must be learning to separate it particularly well.

There are a couple reasons why the network might be taking much more strongly to the choir data than to the non-choir data. Firstly, is simply that there is more audio data with the same singers. Naturally, having more data to learn specific voices will increase the performance of a model when it encounters more data that has the same voices. It is likely that effect is happening here, but it also seems the full answer is somewhat more

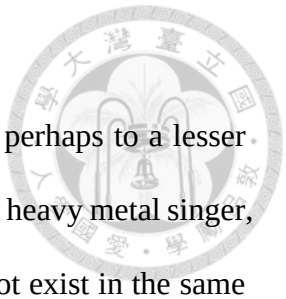
doi:10.6342/NTU202303444



complicated than that. On both of the two models which the choir data is being tested on, (shown in Table 4.1, Figure 4.1, Table 4.2 and Figure 4.2) one of the choir songs from the Choral Singing Dataset (CSD) is being withheld from training, and since there is an overlap of one song between this dataset and the Dagstuhl ChoirSet (DCS) we are only using one song from DCS and two songs from CSD, which means the same voices are only used for two songs; the ones shared by the CSD. In addition to this, Table 4.2 and Figure 4.2 show that if we switch out only the singers of the test samples, the mean performance of the choir model only drops by 0.54 dB which, while still a decrease, is rather marginal. It may be possible that the addition of only one song is enough to substantially increase the model's response to certain voices, however, the lack of the performance dropping significantly when used on other singers does show this is not the only effect at play.

One potential possibility is that, instead of just learning specific voices, the model becomes more accustomed to the particular vocal style. Choir singers tend to intentionally sing in a specific *classical* style. A large reason for this is because all the singers of the choir want to combine their voices as much as possible to create the sound of one unified voice. Because of this, choir singers can end up sounding very similar to one another, and this may explain why the models perform much better on the choir singing, the network has much more data to learn the specific choir *style.* This can also be somewhat equated with the main inspiration for this network, the U-Net based music source separation models, which learn the sound of different instruments to separate them from one another. The difference in our case is that the models are learning to recognize a particular type of singer rather than different instruments, but in both cases, they excel at learning some specific auditory patterns.

doi:10.6342/NTU202303444



This singing style phenomenon also exists in other genres, although perhaps to a lesser extent. Most people can easily envision a stereotypical jazz, country or heavy metal singer, although the emphasis on combining different singers' voices does not exist in the same way. However, it is possible that given the same volume of data from one of these genres a model could be trained to perform just as well as the choir model does here, and perhaps given a large enough volume of data a more generalized model could be trained as well.

The issue with the data taken from the MedlyDB dataset seems to come from its size and generality. Since MedlyDB is a very general database, the songs cover a wide range of genres and singing styles which are very distinct from one another and only give the model one song to learn from. This is essentially the opposite of the choir data, which has longer durations of similar sounding data. This is made especially difficult since we are using a relatively small dataset of only 24 non-choir songs total. In the case of the non-choir songs the network is trying to learn too many different things at once, without enough data to learn any of them very well.

Another possible explanation for the observed differences in performance comes from the fact that, while the choir songs are all written with the inherent logic of music theory, the non-choir songs are all random combinations. Since each part of the choir (SATB) has its own role, and the vocal parts as well as their combinations are obeying the rules of music theory, such as using scales, chords, and standard rhythms, this may have given the models a much stronger structure for predicting future samples from these songs. In contrast to this, the non-choir songs, while individually belonging to a musical scale (or scales) and using rhythm, when combined have no relation to each other in terms of music

doi:10.6342/NTU202303444

theory, likely being in different keys and at different tempos, and this gives a less reliable structure for the model when predicting future samples.

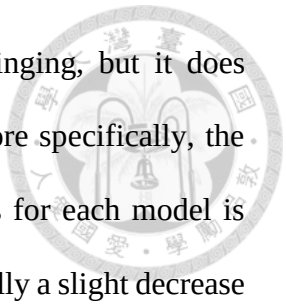
This leads into another approach for how a singing separation network could be built, which is to separate the singers based on the syntactic qualities of the songs themselves. This could function similarly to natural language processing networks, which are able to derive meaning based on the rules of a language. In this case the network would be trained to learn the harmonic role each voice part plays and would attempt to separate the singers based on these qualities. Theoretically, this type of approach should work on songs that follow the standard rules of music theory, which should be the vast majority of music, especially within the realm of classical choir, as well as pop, music and is definitely a method which is worth exploring.

One disadvantage of this is that it would mean all the audio data used would have to be from music with standard harmony and rhythm, like the choir data we have used here, and could not come from random combinations. This would make the data collection much more difficult and time-consuming and reduce the possible scope of currently available data significantly.

5.2 Speech Separation Networks

Shifting attention onto the speech separation models we can observe a much different, and very interesting set of results. The most obvious and surface level result is that, in all cases (Table 4.4), the speech separation models perform significantly worse on singing samples than they do on speaking samples. This is not surprising since the models were

doi:10.6342/NTU202303444



designed and trained specifically for separating speech and not singing, but it does quantitatively show that the two types of data are quite distinct. More specifically, the difference in mean SDR between the speaking and singing samples for each model is around 4 to 5 dB, depending on the model. Interestingly, there is actually a slight decrease in singing separation performance when moving from Conv-TasNet to Dual-Path RNN, despite the increase in the speaking separation performance (Table 4.4, Figure 4.4 and Figure 4.5). Since the mean SDR increments are quite small, at a decrease of only 0.13 dB and an increase of 0.48 dB, it is difficult to say without further experimentation whether this decrease is some kind of anomaly, or if the singing samples really respond better to the purely convolutional network than they do to the recurrent network. Regardless though, this isn't of too much importance since the transformer-based SepFormer model by far surpasses both of the others for speaking, as well as, singing separation.

The pretrained SepFormer model ended up scoring much better than expected on singing separation with a mean SDR score of 4.69 dB (Table 4.4), an improvement of 3.26 dB over the next highest, but also with speech separation, scoring a mean SDR of 9.04 dB, an improvement of 2.44 dB points next highest. Both of these are dramatic improvements over the other two speech separation models which, when compared to each other, only show small differences in performance. It is somewhat difficult, however, to compare this single result to the models we have created, since their performance changes quite a bit in different situations. In the simplest terms we can say: When separating choir songs, our singing separation models that have been trained on choir data seem to perform better, but when separating non-choir songs, the speech separation model (SepFormer) seems to perform better. There is a small amount of uncertainty even in this statement though, since

doi:10.6342/NTU202303444

there are still some variables, such as the speech separation models taking 8 kHz samples while our singing separation models take 44.1 kHz samples. Additionally, while testing our models on the withheld choir songs, and with different singers should give a solid idea of the performance in a general case, having choir test samples which are completely separate from either choir dataset would give further reassurance on the true general performance of the model. Unfortunately, these resources are not available at the moment.

As can be clearly seen in Table 4.5, Figure 4.7 and Figure 4.8, finetuning the SepFormer model using our singing dataset makes quite a noticeable improvement to the model's performance. The improvement margin is somewhat larger when testing on the "regular" test data, at 2.95 dB, which is not surprising since these samples are taken from songs used in the finetuning. However, the performance improvement on the withheld samples, which represent a more general case, is still significant at 2.05 dB. Since this testing set is fairly limited in terms of variety, it is somewhat difficult to say what the true general performance is, but we can say that the finetuning has definitely improved it. With more singing data, and possibly testing other similar speech separation models, it is probable that performance could be improved even further using this method.

During the finetuning process another interesting result became apparent. Looking at Table 4.5 we can see that, for the pretrained model, the mean SDR value of the withheld singing samples is actually somewhat lower than that of the regular testing samples. Given the small dataset scale this might be possible due to coincidence, however, looking at the median SDR, we can see it has also increased significantly from the regular testing samples. Together these results warranted further investigation.

doi:10.6342/NTU202303444

To test this, the samples from the withheld song test data were split into their choir and non-choir subsets and tested separately using the pretrained SepFormer model. The results of these tests are shown in Table 4.6 and Figure 4.9. This immediately made it clear what was occurring, the model was performing extremely poorly on these choir samples, with average SDR scores in the negatives. There is something about this song in particular which makes it much more difficult for this model to separate than others. This song is Niño Dios by Francisco Guerrero, and when listening to the song one can hear that much of the song has singers singing simultaneous lyrical lines, in fact, during most, if not all of the song at least two of the vocal parts are singing the same lyrics. The speech separation model is not trained to deal with this at all and that is likely what is causing such a large drop in average SDR scores for this song in particular. While the other choir songs also contain some overlapping lyrics, it is for much smaller percentages of the overall songs, leading to better average performance.

Also of note is that this is the same withheld song used for testing our custom choir model. In this case the model was able to achieve a mean SDR of 5.65 dB (Table 4.2). This shows that one advantage of the custom model is that it is much better able to deal with this more complex type of polyphonic singing. How both models react to each of the different types of polyphonic singing is an area worthy of further testing, although without the adequate data in each category it is not possible to rigorously test all cases.

Although the performance of SepFormer, and presumably other similar models, is quite impressive and does suggest that using a TasNet based model for singing separation is indeed a potential option, there are a few things to consider. The first is the datasets being used. Since the first TasNet, all the subsequent models have been trained using, at the

doi:10.6342/NTU202303444

very least, the WSJ0 2-mix data set, which is 40 hours of data in total. SepFormer specifically also uses the “dynamic mixing” data augmentation method [4], and many other models are trained using additional datasets such as LibriSpeech and WHAM. These networks are made to be trained with large dataset of audio. Our dataset, on the other hand, has only around 101 minutes of audio data in total, and that is when the entire dataset is in use. We can then see that, even with this huge discrepancy in data volume, the results are not extremely different. This leads to the belief that, given a larger volume of data, it is possible that our custom model would potentially be able to perform better than a TasNet based model. Without legitimate testing it is very difficult to say the difference increasing the volume of data will have on the performance of each model, and at what point, if any, the general performance of our custom model will surpass that of SepFormer.

doi:10.6342/NTU202303444

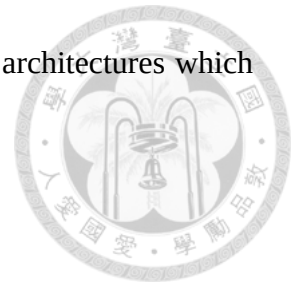
Chapter 6 Conclusions

6.1 Final Conclusions

Here, we have created a custom network for the task of a cappella music separation, as well as tested three premade speech separation models to observe how they would perform at the task. While the performance of the custom network was mixed, dealing very well with choir music but quite poorly with non-choir music, it does show good potential considering the dataset used for training was very limited. The older speech separation models performed poorly, but the newer SepFormer model ended up dealing quite well with singing, although still not as well as it dealt with speech, which indicates singing separation is likely a more difficult task. Finetuning the SepFormer model using our singing dataset was able to raise its performance even more to quite an impressive score. However, we also saw that SepFormer's performance dropped significantly when trying to separate two singers singing the same lyrics, while our custom model was able to deal with the same situation relatively well. This is an important point to remember moving forward, since in real applications it will become much more common to have to deal with the other two-singer scenarios, making a model which can adapt to these situations much more useful. These different situations will likely have to be individually

doi:10.6342/NTU202303444

tested, however, it does seem that there are many possible network architectures which could be capable of performing this task, given the proper training.



6.2 Future Work

There is plenty of room for improvement in this area and there are clear ways to start. Firstly, the U-Net architecture could be expanded, and there are other successful music source separation models currently working on this, such as [14] which uses a bi-U-Net based model. Next, a large improvement could be made by implementing phase compensation, this can be done using a separate loss function for the phase, or by using waveforms instead of spectrograms. There are music source separation models currently which do attempt to account for this phase loss such as Wave-U-Net which operates in the time domain [15]. It also seems, from the results we have found, that it is worth investigating both the performance of the U-Net and TasNet based models, to determine if there is one which is clearly superior at this task, and perhaps even attempting some architectures we have not seen yet.

Once the most rudimentary cases are performing sufficiently, we can then start moving on to the more difficult scenarios, including trying to separate more than just two singers, and trying to deal with the more complex cases which can arise such as separating the voices of the same singer, and singers singing the same lyrics simultaneously.

doi:10.6342/NTU202303444

References

[1] Yi Luo and Nima Mesgarani, “Tasnet: time-domain audio separation network for real-time, single-channel speech separation,” in *Proc. ICASSP. IEEE*, pp. 696–700, 2018.

[2] Y. Luo and N. Mesgarani, “Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation,” *IEEE/ACM Trans. Audio. Speech, Lang. Process*., vol. 27, no. 8, pp. 1256–1266, 2019.

[3] Y. Luo, Z. Chen, and T. Yoshioka, “Dual-path RNN: efficient long sequence modeling for time-domain single-channel speech separation,” in *Acoustics, Speech and Signal Processing (ICASSP), 2020 IEEE International Conference on*. IEEE, pp. 46–50, 2020.

[4] Subakan, C., Ravanelli, M., Cornell, S., Bronzi, M., and Zhong, J. “Attention is all you need in speech separation.” in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 21–25. IEEE, 2021.

doi:10.6342/NTU202303444

[5] Shahar Lutati, Eliya Nachmani, Lior Wolf, “Separate And Diffuse: Using a Pretrained Diffusion Model for Improving Source Separation”, *arXiv:2301.10752v1*, 2023.

[6] Romain Hennequin, Anis Khlif, Felix Voituret, Manuel Moussallam, “Spleeter: a fast and efficient music source separation tool with pre-trained models.” *Journal of Open Source Software*., vol. 5, no. 50, pp. 2154, 2020.

[7] Yi Luo, Jianwei Yu, “Music Source Separation with Band-split RNN”, *arXiv:2209.15174*, 2022.

[8] Olaf Ronneberger, Philipp Fischer, Thomas Brox, “U-Net: Convolutional Networks for Biomedical Image Segmentation”, *arXiv:1505.04597*, 2015.

[9] Sebastian Rosenzweig, Helena Cuesta, Christof Weiß, Frank Scherbaum, Emilia Gómez, MeInard Müller, “Dagstuhl ChoirSet: A Multitrack Dataset for MIR Research on Choral Singing”, *Transactions of the International Society for Music Information Retrieval.*, vol. 3, no. 1, pp. 98–110, 2020.

[10] Helena Cuesta, Emilia Gómez, Agustín Martorell, & Felipe Loáiciga, “Choral Singing Dataset”. *15th International Conference on Music Perception and Cognition (ICMPC)*. 2018.

doi:10.6342/NTU202303444

[11] R. Bittner, J. Salamon, M. Tierney, M. Mauch, C. Cannam and J. P. Bello, "MedleyDB: A Dataset of Multitrack Audio for Music Research", in *15th International Society for Music Information Retrieval Conference*, Oct. 2014.

[12] Bittner, R., Wilkins, J., Yip, H., & Bello, J, MedleyDB 2.0: New Data and a System for Sustainable Data Collection. *New York, NY, USA: International Conference on Music Information Retrieval (ISMIR-16).* 2016.

[13] D. Griffin and Jae Lim, "Signal estimation from modified short-time Fourier transform," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236-243, April 1984.

[14] Simon Rouard, Francisco Massa, Alexandre Défossez, “Hybrid Transformers for Music Source Separation”, *arXiv:2211.08553*, 2022.

[15] Daniel Stoller, Sebastian Ewert, and Simon Dixon, “Wave-u-net: A multi-scale neural network for end-to-end audio source separation,” *arXiv preprint arXiv:1806.03185*, 2018.

doi:10.6342/NTU202303444