Score-Informed Source Separation of Choral Music

Matan Gover

Department of Music Research
Schulich School of Music
McGill University
Montreal, Canada

October 2019

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Arts.

Abstract

Audio source separation is the act of extracting one or more sources of interest from a recording involving multiple sound sources. In recent years, remarkable progress has been made in the development of source separation techniques, enabling applications such as background noise reduction, separation of multiple speakers or multiple instruments, and creation of 'karaoke' tracks by separating vocals and accompaniment in songs.

To our knowledge, this thesis is the first to study the application of source separation to choral music. Choral music recordings are a particularly challenging target for separation due to their inherent acoustical complexity. Every choir singer has a distinctive voice timbre, and the combination of multiple voices singing in harmony, with slight pitch mistunings and imperfect synchronization, creates a highly-variable 'choral timbre'. While choir singers aim to blend their voices, source separation aims to undo that blend. Source separation of choral music enables applications such as fine-grained editing, analysis, and automatic creation of practice tracks (recordings of individual choir parts used by singers as an aid for learning new music) from professional choir recordings.

In this thesis, we address choral music separation using a deep learning separation method called Wave-U-Net. To separate choral music, Wave-U-Net must be trained using a large dataset of choral recordings in which each choir part is recorded separately. Due to the scarcity of such recordings, we create a dataset of synthesized Bach chorale harmonizations. In a series of experiments on this dataset, we show that Wave-U-Net performs significantly better than a baseline technique that is based on non-negative matrix factorization (NMF). We propose a simple change in the way Wave-U-Net is trained that leads to a substantial improvement in separation of more than two sources.

To further improve separation results, we introduce *score-informed Wave-U-Net*, a variant of Wave-U-Net that incorporates the musical score of the piece being separated. The musical score has potential to aid separation because it contains detailed pitch and timing information for every note in the piece. We experiment with different methods of representing the musical score and feeding it into Wave-U-Net. Experiment results show that score-informed Wave-U-Net attains significantly improved separation performance compared to the original Wave-U-Net. Moreover, for increased control over the separation process, we devise a 'score-guided' technique in which the user indicates which notes should be extracted from a recording by simply indicating the desired notes' pitches and times.

Résumé

La séparation de sources sonores consiste à extraire une ou plusieurs sources présentant un attrait significatif d'un enregistrement contenant plusieurs sources sonores. Ces dernières années, de nombreux progrès ont été réalisés concernant le développement de techniques pour la séparation de sources sonores, permettant des applications telles que la réduction de bruit de fond, la séparation de plusieurs chanteurs ou instruments ainsi que la création de pistes « karaoké » en séparant les voix des instruments.

À notre connaissance, cette thèse est la première à présenter une étude concernant l'application des techniques de séparation de sources à la musique chorale. La séparation d'enregistrements de musique chorale constitue une tâche particulièrement difficile du fait de leur complexité acoustique intrinsèque. Chaque chanteur a un timbre de voix distinctif et la combinaison de nombreuses voix chantant en harmonie, avec de légers désaccords et une synchronisation imparfaite, crée un « effet de chorus » extrêmement variable. Alors que les choristes cherchent à fusionner leurs voix, la séparation de sources vise à annuler cette fusion. La séparation de sources permet l'édition, l'analyse et la création automatique de pistes audio pour les séances de répétition (enregistrements de parties individuelles du chœur utilisés par les chanteurs pour faciliter l'apprentissage de nouvelles pièces) à partir d'enregistrements professionnels.

Dans cette thèse, nous abordons la séparation de musique chorale en utilisant une méthode d'apprentissage profond pour la séparation de sources appelée Wave-U-Net. Pour sa phase d'apprentissage afin de séparer la musique chorale, Wave-U-Net nécessite une grande base de données contenant des enregistrements choraux avec chaque partie du chœur enregistrée séparément. En raison de la rareté de tels enregistrements, nous avons créé un ensemble de données à partir d'harmonisations de chœurs de Bach synthétisées. Dans une série d'expériences basées sur cet ensemble de données, nous montrons que Wave-U-Net est nettement plus performant qu'une technique basée sur une factorisation matricielle non négative (NMF). De plus, nous proposons un changement mineur dans la façon dont Wave-U-Net est formé, ce qui conduit à une amélioration substantielle de la séparation de deux ou plusieurs sources.

Afin d'améliorer les résultats des techniques de séparation, nous introduisons *score-informed Wave-U-Net*, une variante de Wave-U-Net qui intègre la partition musicale de la pièce à séparer. La partition peut potentiellement aider à la séparation des sources du fait qu'elle contient des informations précises concernant la hauteur et le temps de chaque note. Nous expérimentons différentes méthodes de représentation de la partition musicale et de son intégration dans Wave-U-Net. Les résultats de ces expériences montrent que la technique Wave-U-Net intégrant les partitions musicales est significativement plus performante dans la séparation de sources que la méthode Wave-U-Net d'origine. De plus, afin d'avoir un meilleur contrôle du processus de séparation, nous avons développé une technique « guidée par la partition » dans laquelle l'utilisateur peut indiquer les notes à extraire d'un enregistrement en sélectionnant les hauteurs et temps des notes souhaitées.

Acknowledgments

I thank my supervisor Professor Philippe Depalle for guiding me in my research. Philippe's knowledge, professionalism, and continued enthusiasm for his work greatly inspired me to challenge myself and learn new topics. I am extremely grateful for the generous financial assistance I have received from the Jenny Panitch Beckow Memorial Scholarship given by the Jewish Community Foundation of Montreal, a stipend granted to me by Philippe Depalle through NSERC, and a Graduate Excellence Fellowship granted by the Schulich School of Music. I thank Professors Ichiro Fujinaga and Marcelo Wanderley who have taught me enlightening graduate seminars, and the entire Music Technology faculty for fostering a vibrant learning environment. I thank my colleagues from the Music Technology area and especially my friend Behrad Madahi for insightful conversations and enjoyable pastimes. I thank Compute Canada for providing the high performance computing infrastructure that facilitated this research. My family, whom I love the most, were a constant source of support and encouragement despite being far away, and for them I am eternally grateful. Finally, I thank my loving and beloved partner, Jhonatan, for his constant love and compassion and for his support in carrying out this research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we set out to investigate the application of source separation to choral music. Informally speaking, our goal is to "separate the voices of the choir". More specifically, we aim to take a recording of choral music and extract from it individual recordings for each of the four choir sections: soprano, alto, tenor, and bass. This act of 'de-mixing' a recording is known as audio source separation.

In formal terms, audio source separation refers to extracting one or more sound sources of interest from a recording that involves multiple sound sources (Vincent et al., 2018). Source separation has many applications in speech, music, and environmental sound processing. For example, when a user speaks to a voice-activated virtual assistant such as Apple's Siri or the Google Assistant, the assistant's software performs source separation behind the scenes in order to separate the user's speech from background noise and interferences, such as music playing in the same room or the speech of other people. Separating the user's speech from other sounds improves the assistant's ability to understand what the user said. Humans do not perform source separation per se but they have an impressive ability to segregate sound sources and understand speech in noisy environments (Bregman, 1990).

Source separation has many musical applications (Cano et al., 2019). It can be employed to separate instruments in a recording or even to separate individual notes played by the same instrument. This can be used, for example, to fix the tuning or timing of certain notes. Source separation can also be used to generate 'karaoke' tracks and a cappella renditions by separating the accompaniment and the lead vocals in songs.

Despite the popularity of musical source separation techniques, it appears that this thesis is the first to attempt the application of source separation to choral music. In the next section, we explain our motivation for this research.

1.1 Motivation

Choral singing, and more generally, multi-part singing, is a widespread cultural phenomenon that exists in many societies around the world, and its roots are believed to lie in very early stages of human evolution (Jordania, 2015). In medieval Europe, monks in the Christian Church developed choral singing practices that later evolved into the musical style that we know today as "classical" choir music (Erickson, 2001). Today, choral singing flourishes in various forms: from amateur choirs to professional concert choirs, church choirs, school choirs, and many others. In Canada, for example, an estimated 10% of the population sing in choirs (Hill, 2017).

Many choral singers cannot read music, which makes it difficult for them to practice their parts between rehearsals. To aid singers, conductors often create *practice tracks*, which are individual recordings of each choir part. Recording such tracks for every piece sung by the choir is time-consuming, so choirs sometimes purchase professionally produced tracks instead.[1] From this need arises an idea: source separation could be used to automatically create practice tracks by extracting individual parts from professional choir recordings. Thus, given a good choral source separation method, high-quality practice tracks could be created at no cost.

Given that source separation has not been applied to choral music before, we do not expect this thesis to yield a full solution. Nonetheless, we are motivated by the potential practical applications and by the hope that investigating this new task may lead to ideas that are applicable to other source separation scenarios.

1.2 Challenges

At the outset, separation of choral music would seem a challenging task. The singing voice is produced by a complex biological mechanism that yields sounds with intricate timbral

[1]Example websites offering choral practice tracks: ChoralPractice (`https://choralpractice.com`), Choral Rehearsal Tracks (`https://choralrehearsaltracks.com`), PraiseCharts (`https://www.praisecharts.com/products/choir-practice-tracks/`), Choral Tracks (`https://choraltracks.com`), Choralia (`http://www.choralia.net`).

characteristics that vary considerably between individuals (Sundberg, 1987). Therefore, processing the singing voice is difficult compared to many musical instruments (Rodet, 2002). Choirs are composed of multiple singers singing simultaneously with slight variations in pitch and in tempo, where every singer has a unique voice timbre. It follows that choral music has extremely varied acoustical characteristics, and could pose a particular challenge for source separation techniques compared to other types of music.

Furthermore, an important goal in choral performance is achieving *blend* between singers, so that the choir is perceived by listeners as one coherent sound source (Smith & Sataloff, 2013). This blend can naturally hinder the operation of an algorithm wishing to separate the choir. Choral music is often recorded in highly reverberant spaces such as churches, and the reverberations constitute yet another hurdle for separation. Finally, choirs are seldom recorded in a 'one voice per track' setting (Ihalainen, 2008), and this lack of multi-track recordings makes it harder to design and validate source separation systems.

1.3 Overview of Thesis

Despite the challenges outlined above, we believe that recently developed source separation techniques are powerful enough to tackle the case of choral music. In this thesis, we present our research towards this goal. We start in Chapter 2 by reviewing relevant background literature concerning source separation, choral music, and deep learning. In Chapter 3, we present a dataset of synthesized Bach chorale harmonizations that we create in order to design and test our separation techniques. In Chapter 4, we establish baseline separation performance for choral music using a method based on non-negative matrix factorization. In Chapter 5, we apply a deep learning-based separation technique called Wave-U-Net (Stoller et al., 2018b) to choral music and test its performance in a series of experiments. We extend Wave-U-Net in Chapter 6 to incorporate musical scores into the separation process, and conduct several experiments to determine the effectiveness of this extension. Finally, in Chapter 7 we summarize the main conclusions of this thesis and suggest possible future research directions.

Chapter 2

Background

2.1 Audio Source Separation

Audio source separation is the act of extracting one or more source signals from an audio recording involving several sound sources. The human auditory system is able to perceive separate sound sources in a scene even though the ears only receive the mixture of all sources. For example, when listening to a piano and violin duet, a human listener can easily identify that two different instruments are playing, and can track the notes of every instrument independently to a certain degree. As another example, when two people are conducting a conversation in a noisy room with many people speaking in the background, the listener is able to focus on the speaker's voice and to understand their words; this was termed the "cocktail party effect" by Cherry (1953). Considering that for humans this is a skill that is acquired and exercised without any effort, source separation turns out to be a surprisingly difficult task for a computer to perform.

Source separation has been an active field of research for many decades and is considered a core problem in audio signal processing (Vincent et al., 2018, p. 3). Applications of source separation abound in music, speech, and environmental audio processing. For example, it is used in hearing aids and conference calling systems to amplify human speech while reducing ambient noise. In automatic speech recognition systems, source separation (and more specifically, speech enhancement) is used to separate and enhance the voice of a main speaker in preparation for recognizing the speaker's utterances. In music processing, source separation is used to automatically produce karaoke (instrumental) tracks from song recordings, or to extract a specific instrument from a multi-instrument recording. Another application is 'upmixing' of mono recordings to multiple channels for playback with multiple speakers or in 3D audio settings. The large number of real-world applications and their potential for commercial value constitute a motivation for many researchers to investigate and constantly improve the state of the art in source separation.

It is important to note that source separation can be applied to many kinds of signals other than audio. For example, in image processing source separation can be used to separate images that are overlaid on top of each other with partial transparency. In medicine, source separation is used on electrode measurements from an electroencephalogram (EEG) device to extract the brain activity signal while removing interferences caused by muscle activity. In this work, however, we focus strictly on audio source separation.

2.1.1 Definition

In order to define the audio source separation task in a more precise manner, we first give definitions for the terms *sound source* and *mixture*, after Vincent et al. (2018, pp. 4–5). In general terms, a sound source is simply an object that emits sound, but in the context of source separation we use the term to refer specifically to the sound that is emitted by that object. A sound source can be a point source (located at one point in space) or a diffuse source (spread over a whole region in space), but here we refer only to point sources. Let us assume a scene containing J sound sources, where each source is represented by a signal s_j where $j \in [1, J]$.[1]

Let us now assume that the scene is recorded using I microphones, producing a multi-channel recording. We denote each channel in the recording by x_i where $i \in [1, I]$. As sound propagates from the sound sources to the microphones, it is transformed by some unknown acoustic transfer function. Thus, for every pair of sound source and microphone we define the spatial image c_{ij} representing the sound emitted by source j as captured by microphone i. Every channel x_i is defined as the combination of all sound sources recorded by a single microphone:

$$x_i(t) = \sum_{j=1}^{J} c_{ij}(t)$$

The multi-channel mixture $\boldsymbol{x}$ is then defined as a vector-valued function that is a com-

[1]Throughout this thesis, we use the following notation conventions: a denotes a scalar, $\boldsymbol{a}$ denotes a vector, and $\boldsymbol{A}$ denotes a matrix.

bination of the signals captured by all microphones:

$$\boldsymbol{x}(t) = [x_1(t), \dots, x_I(t)]^T$$

The purpose of source separation is to obtain the original source signals s_j given only the mixture $\boldsymbol{x}$.

2.1.2 Scenarios

The large number of scenarios in which source separation is applied has led the research community to devise several typologies to categorize these scenarios (Vincent et al., 2018, pp. 6–7). The first categorization relates to the number of sound sources and the number of microphones: if there are more microphones than sound sources, the scenario is called *overdetermined*; if there is an equal number of sources and microphones, the mixture is *determined*; and when the number of sources is smaller than the number of microphones, it is *underdetermined*. This distinction stems from the fact that overdetermined and determined mixtures can be separated without additional information other than the recording itself, by directly reversing the linear process that created the mixture (assuming that process is known and invertible). Such methods, however, cannot often be used effectively in practical applications since most recordings are made in *underdetermined* scenarios or contain interferences such as background noise.

A second categorization characterizes scenarios based on the amount of prior information that is used to guide the separation. When no prior information is used other than the mixture itself, the scenario is dubbed *blind* source separation. In *semi-blind* separation, a limited amount of information about the recording or the nature of the sources can be used to guide the separation. For example, if the sources are known to be certain musical instruments, the separation technique could rely on some timbral characteristics of those instruments. The prior information could also be in the form of more general assumptions on source characteristics such as spectral or temporal smoothness. Lastly, in an *informed* source separation scenario there is some form of highly-detailed information that is used to guide separation. In speech separation, for example, that information could be a phonetic transcript of a conversation that is synchronized with the mixture recording. In informed music separation, the musical score could be used to indicate the instruments, fundamental frequencies, and timings that are to be found in the mixture. Score-informed separation will be discussed in depth in Section 2.5.

Blind source separation methods typically cannot work in underdetermined scenarios, hence their real-world applications are limited. In practice, almost any separation method uses some amount of prior information, ranging between very little or very general information (semi-blind separation) to highly specific and detailed information (informed scenarios).

2.1.3 Methods

Several historical paths were taken to approach the task of source separation: microphone array processing techniques emerged from the field of telecommunications for speech localization and enhancement; statistical methods such as independent component analysis and matrix factorization were developed to tackle blind separation with no prior information; and perceptually-informed methods were developed to mimic the function of the human auditory system. These techniques are described in detail below. According to Vincent et al. (2018, p. 10), it seems that these diverging historical paths are now tending to converge: new techniques have emerged that combine insights from all three types of techniques, and the distinguishing lines between them have become less clear.

Spatial filtering techniques (also known as *beamforming*) work on multi-channel recordings acquired by an array of microphones. The separation relies on the distribution of the microphones in space. In order to perform separation, the mixture signal is passed through a linear time-varying system, where parameters for the system are estimated in various ways from the mixture taking into account knowledge about the microphone array and the environment (Van Trees, 2002). Spatial filtering techniques are widely used in applications where the mixture is captured by carefully-placed microphone arrays, such as speech enhancement and noise cancelling for conference calling devices. In other applications, however, these techniques are seldom relevant because the microphone configuration is not well-known.

A second class of techniques is based on Computational Auditory Scene Analysis (CASA) (Brown & Cooke, 1994; Wang & Brown, 2006). CASA techniques are inspired by perceptual studies that have investigated the ways in which the human brain segregates sounds into auditory streams (Bregman, 1990). The systems aim to mimic the processing that occurs in the human auditory system to varying degrees of accuracy. Typically, the first stage in CASA-based methods is converting the mixture signal into a time-frequency representation that approximates the frequency-based selectivity of the basilar membrane in the human ear.

One such representation is the commonly-used gammatone filterbank (Patterson et al., 1987). Subsequently, algorithms informed by psychoacoustical cues are used to group components in the time-frequency representation into auditory streams. Grouping cues can be spatial (cross-channel correlations), spectral (e.g., harmonicity or frequency co-modulation), or temporal (onset and offset synchronicity or amplitude co-modulation).

Independent Component Analysis (ICA) is a source separation technique that aims to produce sources that are as statistically independent as possible (Comon, 1994; Hyvärinen & Oja, 2000). In other words, rather than relying on specific properties of each source, it models sources as stochastic random processes and strives to maximize their independence from each other. Since it does not assume any prior information about the signals, ICA is considered a blind source separation technique (Cardoso, 1998). ICA can work well in applications where components are not related in any way to each other, such as separating speech from background noise. In music, however, sources are often highly correlated to each other leading to poor ICA performance. Furthermore, basic ICA cannot be used in underdetermined scenarios; the number of sources cannot exceed the number of observed signals. To overcome this issue, Independent Subspace Analysis (Casey & Westner, 2000) transforms the observed mixture into a time-frequency representation, and treats every frequency band as a separate channel on which ICA is performed. However, separating in the frequency domain introduces another problem: since every frequency band is separated individually, components output by ICA are not ordered consistently across time frames, and a method must be devised to group separated components into sources along the time axis (Mitianoudis & Davies, 2003).

Data-based methods

Recently, an increasing amount of source separation techniques are based on training data rather than utilizing statistical properties or hand-crafted features (Vincent et al., 2018, p. 11). The nature of the training data and the way it is used vary among methods. In *source-based training*, a separate statistical model for each source is trained using isolated signals of that source. The model embodies statistical characteristics of the sounds generated from that source. For example, for singing voice separation, the model of the singing voice would be trained using a corpus of solo singing voice recordings, and the model of the instrumental accompaniment would be trained using a corpus of instrumental tracks. On the other hand, *separation-based training* methods treat the separation task holistically;

they optimize machine learning models to predict source signals given the mixture in an end-to-end manner. In this case, the model embodies the characteristics of the mapping from mixture to sources.

A further distinction is made between *supervised* and *unsupervised* training. In supervised training, every training example is associated with ground truth data that helps guide the optimization process. In supervised separation-based training, every item in the training dataset normally consists of a mixture and the set of all individual sources that generated it. In supervised source-based training, a training example could consist of a recording of a single note played by an instrument along with an annotation specifying the fundamental frequency of the played note. In unsupervised training, on the other hand, models are optimized using recordings only, without any associated labels and without pairings of mixtures and individual sources.

As an example for training-based methods, Hidden Markov models (HMM) have often been used for source separation with unsupervised source-based training. An HMM is a statistical technique that models sequential data using a set of states, state transition probabilities, and probabilities of states generating certain observed outputs. The set of states and probabilities can be estimated using a training dataset. Roweis (2000), for example, used an HMM trained only on recordings of single speakers to separate mixtures of multiple speakers. The HMM is used to generate a time-frequency mask that filters the desired source from a mixture.

In the last decade, techniques based on artificial neural networks ("deep learning") have gained widespread popularity in many fields, and in audio processing in particular. In recent years, deep learning techniques for source separation increasingly achieve state-of-the-art results in various separation scenarios. Deep learning models for source separation are normally trained using separation-based training, either supervised or unsupervised. We give an in-depth review of deep learning in Section 2.2. We cover audio applications of deep learning, including source separation specifically, in Section 2.3. In Section 2.3.3, we describe a specific deep learning source separation technique called Wave-U-Net on which we base our original work.

Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) is a source separation technique that was originally proposed for image decomposition (Lee & Seung, 1999), but it can be applied to any type of signal. Before the advent of deep learning techniques, NMF was one of the most prominent techniques for source separation due to its simplicity and extensibility.

The basic assumption of NMF is that the components from which a mixture is composed are all non-negative. Hence, in the domain of audio, NMF cannot be applied directly on time-domain signals because they do assume negative values. Rather, NMF is applied on magnitude spectrograms (or magnitudes of other time-frequency representations) in which values are always non-negative (Cichocki et al., 2006).

Let us denote the mixture magnitude spectrogram with $\boldsymbol{X}$. NMF produces an approximate factorization $\hat{\boldsymbol{X}}$ to represent the spectrogram $\boldsymbol{X}$ as a product of two separate matrices: a matrix of basis signals ($\boldsymbol{W}$) and a matrix of activations ($\boldsymbol{H}$):

$$\hat{\boldsymbol{X}} = \boldsymbol{W}\boldsymbol{H}.$$

In this way, the mixture is estimated as a superposition of basis signals that assume time-varying magnitudes. This interpretation can be seen clearly when the matrix multiplication is viewed as a sum of column-by-row multiplications: every basis signal spectrum (column of $\boldsymbol{W}$) is scaled by a time-varying activation value (row of $\boldsymbol{H}$) and the resulting matrices are summed to produce the final mixture estimate $\hat{\boldsymbol{X}}$. See Figure 2.1 for an illustration. The mixture spectrogram $\boldsymbol{X}$ inherently assumes only non-negative elements, and the separation process is constrained to produce $\boldsymbol{W}$ and $\boldsymbol{H}$ that contain only non-negative elements as well.

NMF factorizes a signal into spectral components, but it does not supply a way to group these components into sources. For example, if in a recording of a flute and cello duet every instrument plays two pitches, a successful run of NMF would produce four basis signals corresponding to the four pitches contained in the recording. However, NMF would not indicate which notes belong to which instrument. Normally, a post-processing technique such as spectral clustering is used in order to group components produced by NMF into sources.

NMF algorithms work by minimizing an objective function that measures the divergence between the mixture $\boldsymbol{X}$ and its estimation $\hat{\boldsymbol{X}}$. Several measures have been proposed for this divergence, including squared Euclidean distance (Lee & Seung, 1999), the Kullback-

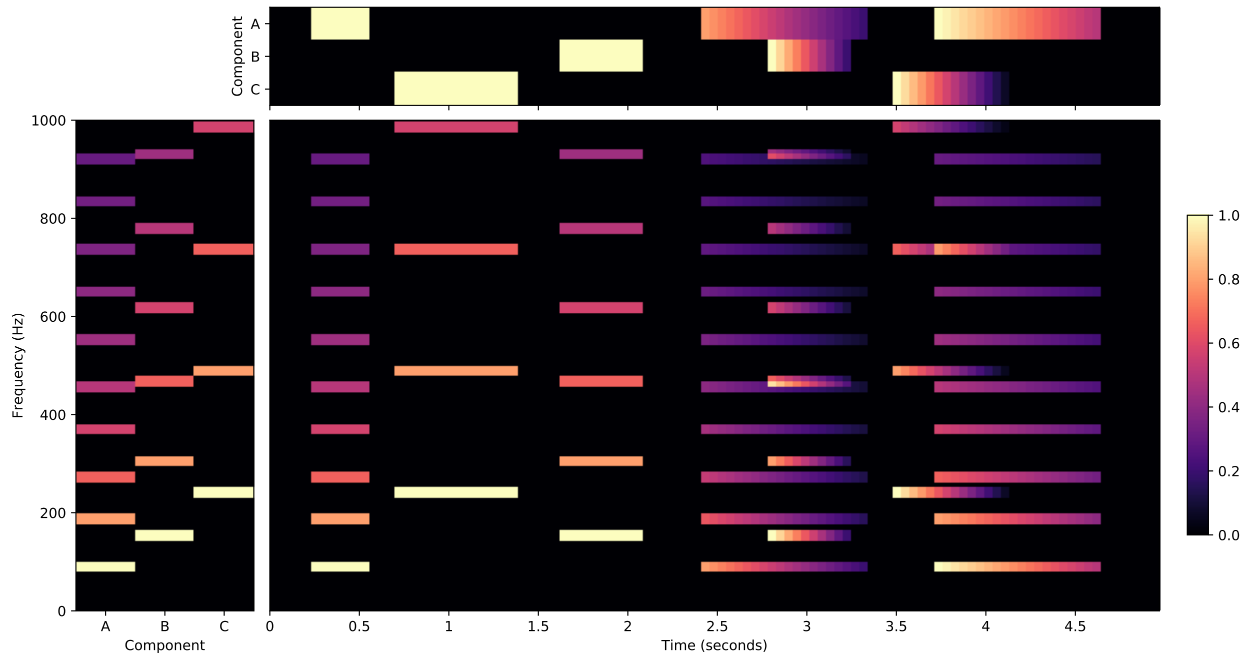


Figure 2.1 Example of non-negative matrix factorization. Left: components ($\boldsymbol{W}$), top: activations ($\boldsymbol{H}$), center: spectrogram approximation ($\hat{\boldsymbol{X}} = \boldsymbol{W}\boldsymbol{H}$).

Leibler divergence (Lee & Seung, 2000), and the Itakuro-Saito divergence (Févotte et al., 2009). Later, the beta-divergence objective function was proposed as a generalization of all of the above (Févotte & Idier, 2011). The choice of objective function can yield to radically different separation results. Additionally, NMF is sensitive to the initial values of $\boldsymbol{W}$ and $\boldsymbol{H}$.

Several algorithms have been proposed to solve NMF, including multiplicative update rules (Lee & Seung, 2000), alternating least squares (Finesso & Spreij, 2006), and hierarchical alternating least squares (Cichocki & Phan, 2009).

NMF in its basic form is often not sufficient to produce well-separated basis signals, but it is amenable to adding additional spectral and temporal constraints on sources. A large part of the literature has focused on formulating such constraints, which naturally vary according to the task at hand. Constraints include group sparsity (Lefèvre et al., 2011), temporal continuity (Smaragdis et al., 2014), and harmonicity (Hennequin et al., 2010). Musical scores can also be used to enforce a semantically meaningful factorization, as we discuss in detail in sections 2.5 and 4.

2.1.4 Evaluation

Given the fact that a large number of source separation methods are designed by the research community, the challenge of evaluating and comparing the various methods arises. It would be desirable to have the ability to compare two methods or to obtain an individual quantitative measure of a system's performance. The existence of such a measure would make it easier to choose techniques based on their performance. However, since separation quality is somewhat subjective in nature, designing an evaluation scheme turns out to be a challenge in its own. It would be possible, of course, to conduct subjective evaluation studies by asking human subjects to rate or compare various techniques. This task was undertaken on several occasions (Barker et al., 2015; Emiya et al., 2011; Ward et al., 2018a).

Conducting subjective studies, however, is costly and time consuming. For this reason, the research community has made several attempts to devise objective evaluation metrics for separation performance. Evaluation using these metrics normally requires a test dataset that contains mixtures along with the set of true separated sources corresponding to each mixture. A separation algorithm's performance is measured by comparing the true separated sources to the source estimations produced by the algorithm. Standard criteria have been established to perform this comparison: source-to-distortion ratio, source-to-interferences ratio, source-to-noise ratio, and source-to-artifacts ratio (Vincent et al., 2006). The existence of several criteria stems from the observation that several factors can affect the perceived separation quality. The BSS Eval toolbox (Févotte et al., 2005)[2] is a MATLAB software library that computes these criteria given mixtures, true sources, and source estimations. BSS Eval stands for Blind Source Separation Evaluation, but it can also be used for non-blind scenarios (as defined in Section 2.1.2). Recently, version 4.0 of BSS Eval was released[3] (Stöter et al., 2018). This latest version is implemented using the Python programming language and introduces significant speed-ups in computation.

Emiya et al. (2011) propose another set of objective measures that are released as a freely-available software toolkit under the name PEASS (Perceptual Evaluation methods for Audio Source Separation). PEASS metrics are similar to BSS Eval in that they compare true sources to algorithm-produced estimations, but unlike BSS Eval, PEASS was designed based on subjective tests and it uses a perceptual salience measure to determine the weighting of

[2] `http://bass-db.gforge.inria.fr/bss_eval`
[3] `https://sigsep.github.io/sigsep-mus-eval/`

various measures so that they correlate well with subjective evaluations. Ward et al. (2018b) compared the BSS Eval and PEASS evaluation toolkits and checked how their evaluations correlate with human evaluations. The authors conclude that PEASS and BSS Eval have comparable performance, with each excelling at a different metric. These results show the elusiveness of devising a truly objective evaluation metric that works equally well for all kinds of mixtures.

2.2 Deep Learning

Deep learning is a set of machine learning techniques that have been used successfully to solve a broad range of AI tasks (LeCun et al., 2015). Machine learning algorithms learn a data distribution from a given training dataset, in the hope that the learned distribution generalizes well to examples outside of that dataset. In classic machine learning algorithms (e.g., k-nearest neighbors), generalization relies on smoothness of the estimated data distribution. A local smoothness prior, however, is not sufficient to estimate complex data distributions. Deep learning, on the other hand, is powered by the assumption that data was generated by a combination of multiple, possibly hierarchical, features. Under this assumption, the number of distinguishable regions in the data distribution grows exponentially in relation to the number of training examples. This ability to make predictions based on learned non-local features enables deep learning to tackle a wider variety of AI tasks without using task-specific engineered features.

Deep learning provides a powerful framework for supervised learning. In a supervised learning setting, model parameters are optimized using a training dataset consisting of matching input and output vectors. In general, it appears that given a large enough dataset and a large enough model, deep learning is able to tackle most tasks that are defined as a mapping from an input vector to an output vector and are easy for a person to perform rapidly.

2.2.1 Artificial Neural Networks

The most prevalent type of deep learning model is a feed-forward neural network, also known as a multilayer perceptron. A feed-forward neural network approximates some function $f*$ from input x to output y by defining a mapping $y = f(x; \boldsymbol{\theta})$ and optimizing the parameters $\boldsymbol{\theta}$ to obtain the values that result in the best approximation of $f*$ according to a training dataset and a predefined loss function. The model is called feed-forward because information flows through it from x through f to y without any feedback connections back into f.

Models are customarily built in a chain structure, as a set of composed functions $f_k,\ k \in [1, n]$, so that $f = f_n \circ f_{n-1} \circ \ldots \circ f_1$. Functions f_k are called the model's *layers*. f_n is called the *output layer* and f_1 to f_{n-1} are the model's *hidden layers*.

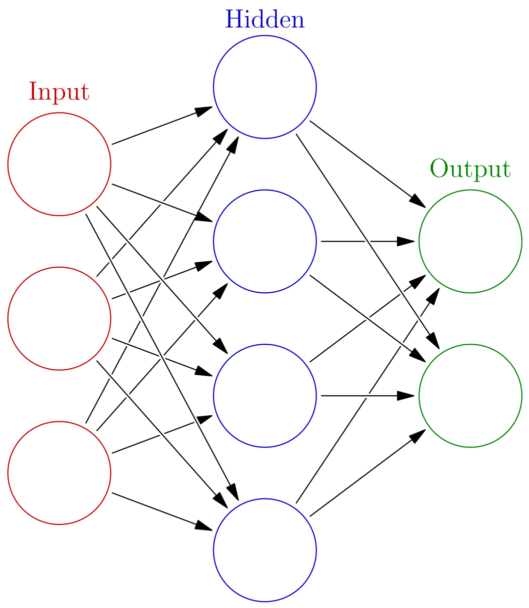


Figure 2.2 Illustration of the structure of a feed-forward neural network. Each circle represents an artificial neuron and each arrow represents a unidirectional connection between one neuron's output and another neuron's input. The depicted network is fully-connected: every neuron is connected to all the neurons in the following layer. The shown network has only one hidden layer for simplicity, but in practice networks often have several hidden layers. Image by Wikipedia user Grosser.ca, licensed under CC-BY-SA 3.0.[5]

These models are called artificial neural networks because they are loosely inspired by neuroscience. Brains contain a large collection of interconnected neurons, where each neuron receives the outputs of several other neurons and produces one output. Artificial neural networks can be seen in a similar light: instead of viewing f as a mapping between vectors, we can see each layer as a set of units ('neurons') that act in parallel, where each unit in layer k receives several outputs from layer $k - 1$ and produces one output which is sent to layer $k + 1$. See Figure 2.2 for an illustration.

[5] `https://creativecommons.org/licenses/by-sa/3.0/`

An artificial neuron is defined by:

$$h(\boldsymbol{x}) = g(\boldsymbol{w}^T \boldsymbol{x} + b), \tag{2.1}$$

where $\boldsymbol{x}$ is the neuron's input vector, $\boldsymbol{w}$ is a vector of weights, b is a scalar bias and g is a nonlinear *activation function* that is inspired by the firing rate of a biological neuron.

Consequently, a layer in a feed-forward neural network is defined recursively as an affine transformation on the output of the previous layer followed by an activation function:

$$\begin{aligned} h_k(\boldsymbol{x}) &= g(\boldsymbol{W}_k h_{k-1}(\boldsymbol{x}) + \boldsymbol{b}_k), \\ h_0(\boldsymbol{x}) &= \boldsymbol{x}, \end{aligned} \tag{2.2}$$

where $\boldsymbol{W}_k \in \mathbb{R}^{D_k} \times \mathbb{R}^{D_{k-1}}$ is a matrix of layer parameters, $\boldsymbol{b} \in \mathbb{R}^{D_k}$ is a bias vector, and g is the activation function.

Considering feed-forward neural networks are built from such simple mathematical constructs, their approximation capabilities are surprisingly powerful. According to the universal approximation theorem (Cybenko, 1989; Hornik et al., 1989), multi-layer feedforward networks are capable of approximating any measurable function from one multidimensional space to another with an arbitrarily small non-zero amount of error, given a sufficient number of hidden units and under mild assumptions on the activation function.

This universal approximation theorem implies that feed-forward neural networks have the power to represent almost any function, but it does not make any guarantee on the number of parameters that are required in order to do that. Furthermore, it does not prove that there exists an optimization algorithm that could find the parameters for a neural network given a set of training examples. That is, the theorem proves there must be a network that can model a target function, but we may never be able to find that network's parameters. For these reasons, the universal approximation theorem is not sufficient to guarantee the success of neural networks for real-world tasks. In fact, for many years machine learning researchers had believed that neural networks have limited applications due to the difficulties in training them. In the past decade, however, advances in training algorithms and network architecture design have made neural networks a method that is well-suited for a wide variety of tasks.

Activation Function

As mentioned earlier, a nonlinear activation function is applied to the output of every hidden unit in a neural network (see Equation 2.1). Early neuroscience-inspired techniques (McCulloch & Pitts, 1943) lacked nonlinear activation functions and were thus limited in power: linear models cannot learn even a simple function such as XOR, and any multi-layer network without nonlinear activation functions can be reduced to a single-layer model (Goodfellow et al., 2016, p. 14). The activation functions are thus one of the main sources of the representational power of neural networks. The characteristics of the activation function greatly affect the efficiency in which neural networks can be trained, but the effects of functions on training are not completely understood. In many cases, activation functions are chosen based on empirical results rather than on a theoretical understanding.

Figure 2.3 shows several common activation functions. Currently, the most commonly used activation function is a rectified linear unit, known as ReLU. A ReLU lets positive input pass unchanged but discards negative input. It is defined as $f(x) = \max(x, 0)$. In the early days of neural networks, a commonly used function was the logistic sigmoid: $f(x) = \frac{1}{1+e^{-x}}$, which generates values between 0 and 1. The hyperbolic tangent is also used as an activation function. It is in fact simply a rescaled version of the logistic sigmoid, with values between -1 and 1: $f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, .

One motivation for using the logistic sigmoid was that it was considered to be a plausible emulation of the firing of a biological neuron. Nonetheless, experimental results have shown that using the hyperbolic tangent leads to more efficient training compared to the logistic sigmoid, due to the fact that its outputs are more likely to have a mean that is closer to zero (LeCun et al., 1998). More recently, ReLUs were found to be superior to sigmoids for many use cases. Experimental results have shown that using ReLUs leads to faster converge during training and also finds minima of equal or greater quality (Glorot et al., 2011). ReLUs have thus become the most prevalent activation function. They are also more biologically plausible than sigmoids, because biological neurons do not have any output until their inputs cross a certain threshold.

2.2.2 Training

Training a machine learning model is the process of finding appropriate parameters for the model so that it approximates a desired function. In the case of feed-forward neural networks,

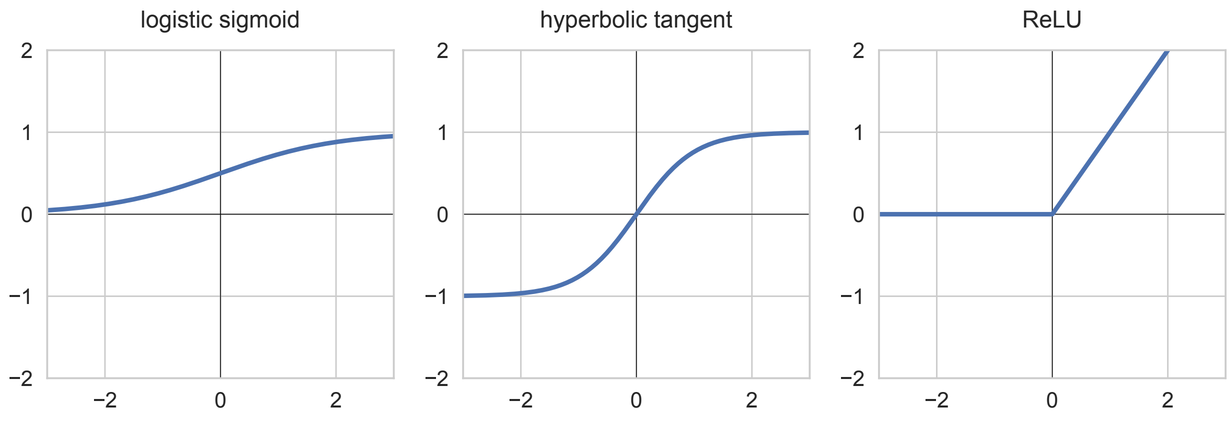


Figure 2.3 Neural network activation functions

the parameters to train are the layer weights and biases: $\boldsymbol{W}_k$ and $\boldsymbol{b}_k$ from Equation 2.2.

The most common training method for neural networks is supervised training (LeCun et al., 2015). In supervised training, the function to be approximated is defined using a set of training examples (the *training dataset*), where each training example is a pair consisting of an input and its true output. The training dataset is normally assembled by collecting a large set of examples from the network's input domain and attaching a human-annotated label to each example representing the so-called *ground truth* that the model is supposed to learn.

A desirable property in a good machine learning model is being able to generalize well to examples outside of the training set. Hence, the purpose of training is to have the model learn a distribution that is built out of the concepts that underlie the training data, rather than simply memorizing the correct answer for each training example. A model that memorizes training examples is called *overfitted*. Overfitting occurs when the model works well on the training dataset but not as well on real-world test data (Srivastava et al., 2014). There are several techniques to prevent overfitting during training (Goodfellow et al., 2016, p. 221); in general, when the dataset is large enough (and the model size remains fixed) the model is forced to generalize rather than memorize. In order to prevent an overfitted model from being selected, a separate set of examples is prepared: the *test dataset*. The test dataset must not be used during training; it is used to measure the performance of a model on examples that it has not encountered during training.

Loss Functions

Training a neural network requires defining a *loss function*, also known as an objective function (Goodfellow et al., 2016, p. 79). The loss function measures how well a specific function models the training dataset. In supervised training, the loss function measures the difference between the model's predictions and the true data from the training dataset. The training process aims to minimize that difference in order to maximize the quality of a model's predictions. For instance, if a model is trained to predict a person's age from their picture, the loss function could be chosen to be the difference between the predicted age and the actual age, averaged over all training examples.

In deep learning, the choice of loss function used to train neural networks is usually motivated by the principle of maximum likelihood (Goodfellow et al., 2016, pp. 172–174). Maximum likelihood estimation is a method used to find model parameters under which the likelihood of observing the given training data is maximized. Under the framework of maximum likelihood, the loss function is simply the cross-entropy between the training data and the model distribution.

The *mean squared error* loss function is of special interest in this regard, because minimizing the mean squared error is equivalent to maximum likelihood estimation assuming a Gaussian model distribution (Goodfellow et al., 2016, pp. 133–134). Mean squared error is thus the most widely used loss function in many domains (Zhao et al., 2017). To define the mean squared error, let $\boldsymbol{y}_i$ be the empirical output vector in a certain training example, $\hat{\boldsymbol{y}}_i$ be the model's prediction for that training example, and m be the number of examples in the training dataset. The mean squared error is then defined as the squared L^2 norm (Euclidean norm) of the difference between the empirical output and model output, averaged over all training examples:

$$\mathrm{MSE} = \frac{1}{m}\sum_{i=1}^{m} \|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\|_2^2$$

The mean squared error loss function is the de facto standard in image processing, but Zhao et al. (2017) have shown that in some cases using the *mean absolute error* produces better results. The mean absolute error is defined as the L^1 norm of the difference between the empirical output and model output, averaged over all training examples:

$$\mathrm{MAE} = \frac{1}{m}\sum_{i=1}^{m} \|\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i\|_1$$

2.2.3 Gradient-Based Optimization

In the previous section we explained that training a neural network is an optimization process that finds appropriate parameters to minimize a chosen loss function. This section explains how that optimization is performed. Classic convex optimization methods cannot be used to train neural networks, because the nonlinearity introduced by neural network activation functions causes the loss function to become non-convex. For this reason, neural networks are trained using gradient-based optimization, also known as gradient descent.

Gradient descent is a general optimization method that is not specific to deep learning. It is an iterative technique that uses a function's gradient to make small steps towards finding a local minimum in the function (Boyd & Vandenberghe, 2004, p. 466). In the case of neural networks, the optimized function is the loss function. The gradient of the loss with respect to the model parameters indicates the direction in which the model parameters should be adjusted so that the loss increases the fastest. Thus, in order to *decrease* the loss the parameters must be adjusted in the opposite direction: the direction of the negative gradient. Hence, at every training iteration, the model parameters $\boldsymbol{\theta}$ are updated by the negative gradient of the loss L multiplied by a small value ϵ called the *learning rate*:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \epsilon \nabla_{\boldsymbol{\theta}} L$$

Unlike convex optimization methods, gradient descent does not provide any guarantees on the number of iterations required to bring the error to a minimum, nor does it guarantee ever converging to such a minimum. Moreover, gradient descent is sensitive to the initial values of parameters: different starting points may lead to different final results. Despite these limitations, in practice gradient descent was found to converge to very small local minima after a sufficiently large number of training iterations (Goodfellow et al., 2016, pp. 171–172).

Gradient descent requires computing the value of the loss function at each training iteration. Most loss functions, such as the mean squared error described above, have a computational cost that depends on the number of examples in the training dataset. In the standard case where the training dataset contains millions of examples, computing the loss function over all training examples at each training iteration is unfeasible; it would lead to very long training times. For this reason, most modern neural networks are trained using *stochastic gradient descent*.

Stochastic gradient descent is a variant of the basic gradient descent algorithm, in which instead of computing the loss function over all training examples, it is computed on a small batch of examples drawn uniformly from the training dataset at each iteration. Since the loss function is computed on a small batch at every iteration, it can be seen as an estimation for the true value of the loss function had it been computed on all training examples (Robbins & Monro, 1951). This estimated loss value is then used to compute the gradient and adjust model parameters iteratively. Stochastic gradient descent is what enables neural network training to scale easily to millions of training examples (Bottou, 2010).

Back-Propagation

Back-propagation (Rumelhart et al., 1986) is an algorithm used to speed up the computation of the gradient of functions that have a large number of inputs. It is used in training of neural networks to efficiently compute the gradient of the loss function at each training iteration. The motivation for back-propagation arises from the fact that the gradient of a neural network loss function is simple to derive analytically using the chain rule of calculus, but actually computing its value using the analytical expression would be inefficient because many subexpressions would be evaluated multiple times. In back-propagation, the values of these repeated subexpressions are stored in variables so that they are calculated only once. For typical neural network architectures built out of a large number of operations with many parameters, back-propagation speeds up training considerably. For a full mathematical formulation of back-propagation, see Goodfellow et al. (2016, pp. 203–221).

2.2.4 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a special kind of neural networks in which at least one of the layers uses the convolution operation to define its weights in place of general matrix multiplication (Lecun, 1989). Convolution is a mathematical operation that expresses the amount of overlap between two functions as one function is shifted over the other. From a signal processing point of view, convolution is equivalent to the operation of a linear time-invariant system (Orfanidis, 1995, p. 104). Convolution is denoted using the star operator ($*$) and is defined on two discrete functions x and h as follows:

$$(x * h)[n] = \sum_{m=-\infty}^{\infty} x[m]h[n-m] \tag{2.3}$$

In the context of neural networks, x is a layer's input and h is a kernel with learned parameters (Goodfellow et al., 2016, pp. 321–324). Since the values of x and h are stored in arrays, the functions are defined over a finite range only and are assumed to be zero outside of this range, and the infinite summation in Equation 2.3 reduces to a finite summation. In a given convolutional layer, the kernel is normally much smaller than the layer's input. Thus, the operation of a convolutional layer can be seen as pattern recognition: a fixed kernel (defining the pattern) is shifted over the layer's input, and at each point the layer computes the amount of 'matching' between the kernel and the input. Compared to a fully-connected layer, where a full matrix of weights is learned to determine the layer's output, a convolutional layer has only to learn a small kernel. This reduction in the amount of parameters and the 'pattern recognition' property of the convolution operation make convolutional layers a successful regularization strategy for neural networks and enable CNNs to achieve good generalization capacity in practice.

Training a CNN is done in the same manner as any other feed-forward neural network. Most neural network libraries do not implement the convolution operation directly using the definition in Equation 2.3. Rather, they take advantage of the fact that convolution is equivalent to multiplication by a *Toeplitz matrix*. A Toeplitz matrix is a matrix in which every row is equal to the row above it shifted by one element, and multiplying a vector by such a matrix is equivalent to convolution. Thus, a convolutional layer is defined like any other layer (see Equation 2.2) but the weight matrix $\boldsymbol{W}$ is constrained to be a Toeplitz matrix.

The convolution operation used in practice in neural networks has several important differences from the standard convolution operation defined in Equation 2.3. In the basic convolution operation, one of the functions is time-reversed before computing the inner product. In CNNs, however, there is no practical reason for performing the time-reversal, and it is simply omitted. This leads to an amusing realization: convolutional neural networks do not actually use the convolution operation; they use cross-correlation instead, but continue to call the operation convolution by convention. Cross-correlation is defined as:

$$(x \star h)[n] = \sum_{m=-\infty}^{\infty} x[m]h[n+m] \tag{2.4}$$

Until now we have only discussed convolutions (or cross-correlations) that are one-dimensional, that is, they that are applied across one axis only, which in the example case of audio signals is the time axis. However, if inputs are multidimensional then convolutions can also be applied over two or more axes simultaneously. For example, two-dimensional convolutions are the most common type of convolution used in image processing CNNs because images have two dimensions: width and height. In two-dimensional convolution the kernel is also two-dimensional and the summation is done across both dimensions. Two-dimensional cross-correlation is then defined as:

$$(x \star h)[m,n] = \sum_{i} \sum_{j} x[i,j]h[m+i,n+j] \tag{2.5}$$

In CNNs, it is often desired to treat some input dimensions differently than others. For example, image processing models process images that are two-dimensional, but each image also has three *channels*: red, green, and blue. Similarly, in audio processing it is common to process stereo audio, thus the input has two channels: left and right. In addition, it is desired that a convolutional layer could have multi-channel outputs, because in standard CNN architectures convolutional layers are stacked on top of each other, each layer producing an output tensor (*feature map*) with different dimensions that is passed on to the next layer.

Hence, it is desired that a convolutional layer would process multi-channel inputs and produce multi-channel outputs, where the number of input channels is not necessarily equal to the number of output channels. For this reason, a convolutional layer is defined using *multiple* cross-correlation operations: for each output channel, each input channel is cross-correlated with a different kernel, and the results are summed (Goodfellow et al., 2016, pp. 337–338). To define a convolutional layer mapping an input with I channels (each channel being a vector $\boldsymbol{x}_i$) to an output with J channels (each channel being a vector $\boldsymbol{c}_j$), we define $i \cdot j$ different kernels $\boldsymbol{h}_{ij}$, one kernel per pair of input and output channels. Every output channel is defined as a sum of cross-correlations with all input channels:

$$\boldsymbol{c}_j = \sum_{i=1}^{I} \boldsymbol{h}_{ij} \star \boldsymbol{x}_i \tag{2.6}$$

Note the important difference between multidimensional convolution (Equation 2.5) and multi-channel convolution (Equation 2.6). In multi-channel convolution, we ‘slide’ a convolution kernel across the axes of every channel separately, whereas multidimensional convolution is applied across several dimensions simultaneously. In practice, neural networks often use convolutions that are both multidimensional and multi-channel.

The activation function and bias (see Equation 2.2) are applied after computing $\boldsymbol{c}_j$, so the final layer output $\boldsymbol{y}_j$ for channel j is:

$$\boldsymbol{y}_j = g(\boldsymbol{c}_j + \boldsymbol{b}_j) \tag{2.7}$$

where g is the activation function and $\boldsymbol{b}_j$ is the bias vector for output channel j.

CNNs often employ several variants of the convolutional layer defined in Equation 2.7. Namely, strided convolutions are used to skip some elements in the output feature map; input padding is used for increased control over the size of the output feature map; and dilated convolutions are used to increase the *receptive field* of a layer while keeping the number of kernel weights fixed. A layer’s receptive field is the number of input units that are connected to each unit in that layer. See Dumoulin & Visin (2016) for a complete review of convolution arithmetic as used in deep learning.

CNNs are normally built as a stack of convolutional layers with other operations interspersed in-between the layers. In particular, *pooling* operations are used as a form of downsampling to reduce the size of feature maps by aggregating values locally. The earliest used form of pooling was average pooling (LeCun et al., 1990), in which the feature map is split into small patches and the average value of each patch is carried forward to the next layer. In max pooling (Ranzato et al., 2007), only the largest value from each patch is carried forward. Scherer et al. (2010) have shown empirically that max pooling yields better results than average pooling, leading to widespread adoption of max pooling in the research community.

The modern breakthrough of CNNs occurred in 2012, when Krizhevsky et al. (2012) famously used CNNs to obtain state-of-the-art results on the ImageNet Large-Scale Visual Recognition Challenge for image classification (Russakovsky et al., 2015), beating other teams by a large margin in classification accuracy. The model was dubbed AlexNet. The stacked convolutional layers processed an input image in multiple scales, allowing AlexNet to create increasingly abstract representations in each layer by combining the feature maps produced by the preceding layer. See Figure 2.4 for an illustration of the most abstract image features as learned by the first layer of AlexNet.

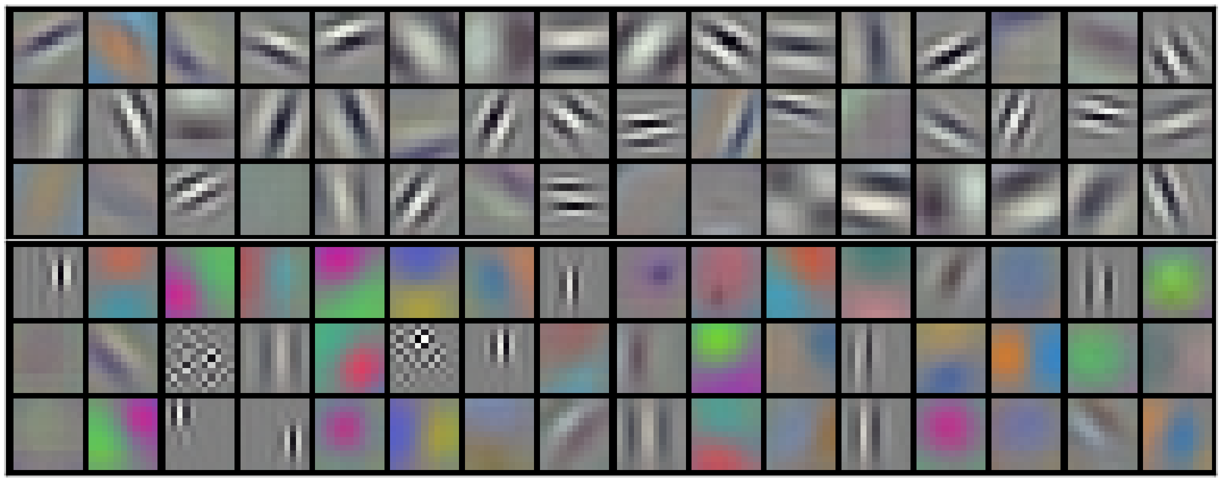


Figure 2.4 Learned convolution kernels from the first convolutional layer in AlexNet, a CNN for image classification. Reproduced from Krizhevsky et al. (2012) with permission.

2.3 Deep Learning for Audio Processing

Recently, the use of deep learning techniques has achieved widespread success in many applications, both scientific and commercial (LeCun et al., 2015). Applications include computer vision, natural language processing, bioinformatics, finance, robotics control, and recommendation systems. As could be expected, audio processing is no exception. In the last half decade, deep learning is increasingly being applied to tasks that were originally solved using classical signal processing techniques, such as synthesis, speech recognition, speech enhancement, sound event classification, music information retrieval, and source separation (Purwins et al., 2019).

In many cases, deep learning models for audio are based on counterparts from the field of image processing. For example, some models for classification of audio events are based on models that were originally developed to classify images. This transfer of techniques is motivated by the early developments in image processing due in part to the existence of larger datasets. However, researchers working on deep learning models for audio sometimes incorporate knowledge and techniques derived from classical signal processing methods; this can lead to innovative techniques that achieve state-of-the-art results (Pons et al., 2016; Ravanelli & Bengio, 2018).

Purwins et al. (2019) give a thorough and recent review of deep learning techniques for audio processing. The authors give a general formulation of an audio task: receiving an input audio sequence and outputting a certain label or labels. Tasks are categorized according to the number of labels that are predicted (one label for the whole input sequence, one label per time step in the input sequence, or a sequence of labels) and the type of these labels (each label can be a single class, a set of classes, or a numeric value). For example, a single-instrument recognition task predicts one label for the whole input sequence and the label consists of a single class which is the name of the instrument. In the speech enhancement task, on the other hand, the output sequence is the same length as the input sequence: the model predicts one numeric label per time step in the input.

2.3.1 Audio Feature Selection

Models differ in the type of input features they use: all models receive audio as input, but not all work on raw audio in the time domain. Many techniques apply a transformation on the audio before it is fed into the deep learning model. The most common transformation is converting the signal to a time-frequency representation, such as the Short-Time Fourier Transform (STFT). Most natural sounds appear sparser when converted into a time-frequency representation (many frequency bins in a spectrogram assume very small amplitude values at most time points), which makes them suitable to be used as a basis for extracting meaningful information (Lewicki, 2002). A time-frequency representation is two-dimensional, as opposed to raw audio that is one-dimensional; this makes it easier to apply models that are adapted from image processing tasks. However, applying image processing models directly on spectrograms is bound to yield suboptimal performance due to the inherent differences between images and spectrograms. Images can be rotated but their content will be preserved; in spectrograms, each dimension has a fundamentally different meaning: the x-axis denote time and the y-axis denotes frequency. Furthermore, natural audio characteristics such as harmonicity yield to correlations between sets of frequency bins in a spectrogram, and these correlations have no counterpart in natural images.

Time-frequency features other than standard spectrograms are also common. In many classical signal processing tasks, the most prominent set of audio features was the mel-frequency cepstral coefficients (MFCCs) (Furui, 1986). However, MFCCs prove to be less efficient for deep learning models, perhaps due to the fact that they remove important information contained in the signal. For this reason, many deep learning models use log-mel spectrograms instead. Log-mel spectrograms are similar to MFCCs in that they both use the outputs of a filter bank that is inspired by the human auditory system, but MFCCs include an additional transformation which log-mel spectrograms do not. Log-mel spectrograms are especially common for speech-related tasks. For musical tasks, constant-Q spectrograms are also common. Constant-Q spectrograms are especially suited for musical tasks because they maintain transposition invariance: transposing an audio excerpt up or down by some amount yields only to a vertical shift in the spectrogram, whereas in other representations it would also lead to scaling.

Finally, some models eschew all manually-designed audio features and choose to use raw audio as input. Raw audio was considered difficult to use for many traditional signal processing applications because of its denseness. However, some neural networks architectures are able to efficiently process raw audio. In particular, CNNs can model a sequence of samples in a hierarchical manner by creating a stack of convolutional layers, where each layer processes the signal at a different time-scale. Since the operation of a filter is equivalent to the convolution operation, and the STFT that is used to produce a spectrogram can be seen as a set of filters, it could be said that a convolutional neural network effectively creates audio features that are similar in nature to a spectrogram but have parameters that are learned using task-specific training data.

In the above description, we were concerned only with how the model's inputs were represented. For tasks in which the output is audio, such as synthesis and source separation, there is a similar challenge with the model's output. If the model outputs raw audio, no further work is necessary. The WaveNet model (Oord et al., 2016), for example, generates audio sample-by-sample where the prediction probability for every sample is conditioned on all previous samples. However, if the model outputs spectrograms (STFT, log-mel, or other), some post-processing stage must convert the spectrograms to raw audio. This is traditionally done using the Griffin-Lim algorithm that recovers the phase from a spectrogram that contains magnitudes only (Griffin & Lim, 1984). However, this algorithm usually does not recover the phase well enough to achieve satisfactory audio quality. Deep learning models can yield better results when trained to convert spectrograms to audio. For example, Shen et al. (2018) conditioned WaveNet on log-mel spectra to generate raw audio.

2.3.2 Source Separation using Deep Learning

Deep learning has emerged as a major paradigm for source separation in the last few years (Vincent et al., 2018, p. 443). Most deep learning source separation techniques use supervised learning with *separation-based training* (described in Section 2.1.3) thus formulating the source separation task holistically as minimizing the error of a mapping from mixtures to source signals (Vincent et al., 2018, p. 113). Deep learning-based methods currently show the best performance among source separation techniques (Rafii et al., 2018). In a recent evaluation campaign, SiSEC 2018, deep learning-based submissions were found to outperform all other methods for singing voice separation and multiple instrument separation (Stöter et al., 2018).

Due to the large number of publications and ongoing developments in this field, we cannot give an exhaustive survey here. We present several major types of methods and representative examples of each type: generic feed-forward neural networks, CNNs, recurrent neural networks, generative adversarial networks, and finally hybrid and other models. The reader is referred to Wang & Chen (2018) for a recent review of deep learning for speech separation and enhancement and to Rafii et al. (2018) and Cano et al. (2019) for extended overviews of music separation with sections on deep learning.

Feed-Forward Neural Networks

Some of the earliest deep learning source separation techniques used generic feed-forward neural networks (FNNs, described Section 2.2.1). Uhlich et al. (2015) approached multi-instrument separation using an FNN that is fed a mixture spectrogram frame, along with neighboring context frames, and trained to predict source spectrograms. The predicted source spectrograms are then combined with the mixture phase to reconstruct the source estimate.

Generally speaking, the specialized neural architectures described below, which enforce parameter sharing within the network, were found to be more effective than generic FNNs for source separation (Cano et al., 2019).

Convolutional Neural Networks

CNNs have been found effective for source separation (Wang & Chen, 2018) since they effectively model shift-invariant features using convolutional layers, where in each layer the same convolution kernel is applied to different parts of the layer's input (see Section 2.2.4 for details). Simpson et al. (2015) used a CNN for singing voice separation that receives the mixture spectrogram as input and outputs a binary mask for extracting the vocals.

Takahashi & Mitsufuji (2017) employed a CNN which is composed of "dense blocks" of layers. Every layer in a dense block receives as input not only the output of the previous layer, but the concatenated outputs of *all* the previous layers in the block. The model contained several interconnected dense blocks, with downsampling and upsampling layers interspersed in-between. The CNN processed a mixture spectrogram by splitting it into several frequency bands and processing each band separately, and then combining the computed feature maps for all bands. The authors showed that this technique outperformed all other techniques evaluated in the SiSEC 2016 evaluation campaign (Liutkus et al., 2017).

Recurrent Neural Networks

Recurrent neural networks (RNNs; Rumelhart et al., 1986) are particularly effective for audio processing and source separation in particular because they are designed to model sequences (Vincent et al., 2018, p. 118). In RNNs, every hidden or output unit can have *feedback connections* to that unit's values in previous time steps. The feedback connections remain fixed for the entire sequence, which is what creates the recurrent structure (Goodfellow et al., 2016, pp. 363–364).

Huang et al. (2015) proposed an RNN that processes spectrograms in short frames. The RNN incorporates relationships between processing frames over time using feedback connections. The network predicts soft masks which are applied to the original mixture to produce source spectra estimates.

Uhlich et al. (2017) continued their aforementioned previous work (Uhlich et al., 2015) by blending the FNN's outputs with outputs from an RNN that ran in parallel. The RNN was shown to outperform the FNN due to the RNN's ability to better take into account the context information from the mixture. Blending the output of the two networks led to even better results.

Generative Adversarial Networks

Generative adversarial networks (GANs; Goodfellow et al., 2014) combine two neural networks: a generator and a discriminator. The generator is designed to manufacture examples that resemble the training examples as closely as possible and the discriminator is trained to distinguish between the training examples and those produced by the generator. The generator is trained to maximize the probability of the discriminator making a mistake.

The use of GANs has achieved widespread popularity for image synthesis, and recently they have been employed for audio tasks including speech enhancement, sound synthesis, and source separation (Engel et al., 2019; Pascual et al., 2017; Wang & Chen, 2018). Fan et al. (2018) created a GAN for singing voice separation which operated on magnitude spectrograms. In order to bootstrap the generator network, they first trained it in a supervised fashion and then continued with standard GAN training.

Subakan & Smaragdis (2017) used a GAN as part of a *generative separation* formulation in which each source is modelled using a probability distribution that is conditioned on a latent variable, and the mixture distribution is conditioned on the sum of all sources. The GAN was trained to generate source spectra from the latent variable values. A mixture was then separated by finding latent variable values that maximize the probability of the mixture and using the GAN to predict source spectra from those values.

Stoller et al. (2018a) used GAN-like adversarial training, but combined supervised and unsupervised training by alternating between two training datasets: a labelled multi-track dataset and an unlabelled dataset containing unassociated sources and mixtures. The novelty of this method is that training is separation-based and yet it still takes advantage of unlabelled data, which is easier to find than multi-track data.

Hybrid and Other Methods

Luo et al. (2016) approached singing voice separation using a unique approach based on clustering: every time-frequency bin in the mixture spectrogram was transformed using an encoder network to an *embedding* vector, and bin embeddings were then grouped using k-means clustering to associate every bin to one estimated source. The idea of clustering spectrogram bins was originally used for speech separation (Hershey et al., 2016).

Nugraha et al. (2016) combined deep learning with classic signal processing techniques for singing voice separation in multi-channel mixtures. They used an FNN to predict source spectra and combined those spectra with spatial covariance matrices, which were estimated using an iterative expectation-maximization algorithm, to create a multi-channel filter.

Takahashi et al. (2018b) built an improved version of the aforementioned CNN with dense blocks (Takahashi & Mitsufuji, 2017) by combining it with an RNN. More specifically, RNN "blocks" were inserted into the network immediately following some dense CNN blocks. This was motivated by the effectiveness of CNNs in modelling local structure and the effectiveness of RNNs in modelling temporal relationships. This technique outperformed all other submissions in SiSEC 2018 (Stöter et al., 2018). Remarkably, it was even found to outperform an ideal binary mask, which is considered as an upper baseline when measuring separation performance.

Phase Estimation

The choice of model type is orthogonal to the choice of audio features, but the majority of methods use time-frequency features and more specifically magnitude spectrograms (Wang & Chen, 2018). When a network predicts only magnitude spectrograms, phase must be estimated separately.

As described above, some techniques simply use the mixture phase as is for the source estimates. Several other techniques have addressed the phase estimation problem. Lee et al. (2017) built a complex-valued neural network that processes the mixture's complex STFT without discarding phase information. Muth et al. (2018) proposed a fused approach combining two networks: the first network processes the magnitudes while the second processes the phase (or, more specifically, the derivative of the phase). The outputs of the two are then concatenated and fed into a fusion network that predicts the complex source spectra.

Takahashi et al. (2018a) devised a method for estimating phases only: a CNN is trained to predict the source phase given the mixture spectrum (phase and magnitude) and the source magnitude (which must be estimated by some other means). Interestingly, they took a classification-based approach by discretizing the phase into a predetermined number of classes.

Yet another way to deal with the problem of phase estimation is to process audio directly in the time domain, without using time-frequency features at all. In the next section we discuss a technique that does exactly that.

2.3.3 Wave-U-Net

In this section we describe a source separation method called Wave-U-Net. Wave-U-Net (Stoller et al., 2018b) is a CNN that was originally designed for two tasks: singing voice separation (voice and accompaniment) and multi-instrument separation (bass, drums, guitar, vocals, and "other"). In the SiSEC 2018 evaluation campaign (Stöter et al., 2018), Wave-U-Net received a high score compared to other state-of-the-art techniques.

In the following chapters of this thesis, we will use Wave-U-Net extensively for our experiments on source separation of choral music (see chapters 5 and 6). In preparation for this, we describe here the original Wave-U-Net technique, starting with its predecessor: U-Net.

U-Net for Image Segmentation

Wave-U-Net is based on U-Net (Ronneberger et al., 2015), a technique that was originally proposed for semantic segmentation of biomedical images. In image segmentation, the input is an image and the desired output is a classification of each pixel into a segment label. U-Net is designed to perform this pixel-by-pixel classification by using the information contained in the image on multiple scales. It does this by employing a hierarchical encoder-decoder architecture with skip connections. As seen in Figure 2.5, the schematic diagram of the neural network resembles the letter U and this is the reason for the name U-Net.

In U-Net's encoder, an input image is gradually scaled in a sequence of convolutional downsampling layers into a low-resolution feature map with a high number of channels. The decoder then scales the feature map back up to the original image's resolution in a sequence of convolutional upsampling layers. Crucially, the feature map that is output from each encoder layer is also directly connected to the input of the corresponding decoder layer. These *skip connections*, depicted by gray arrows in Figure 2.5, enable the network to combine high-resolution information from finer levels of processing with coarse-level features derived using the context of the entire image.

U-Net for Audio Source Separation

U-Net has first been used for source separation by Jansson et al. (2017). In this work, the authors essentially treated the source separation task as an image-to-image mapping. The mixture is first cut into short segments of fixed length and the Short-Time Fourier Transform is computed for each segment. The U-Net is applied on a mixture segment's magnitude spectrum (the "input image") and outputs a soft mask (the "output image"). The soft mask is applied to the mixture magnitude spectrum by element-wise multiplication, and the resulting magnitude spectrum is combined with the mixture's phase to give the final

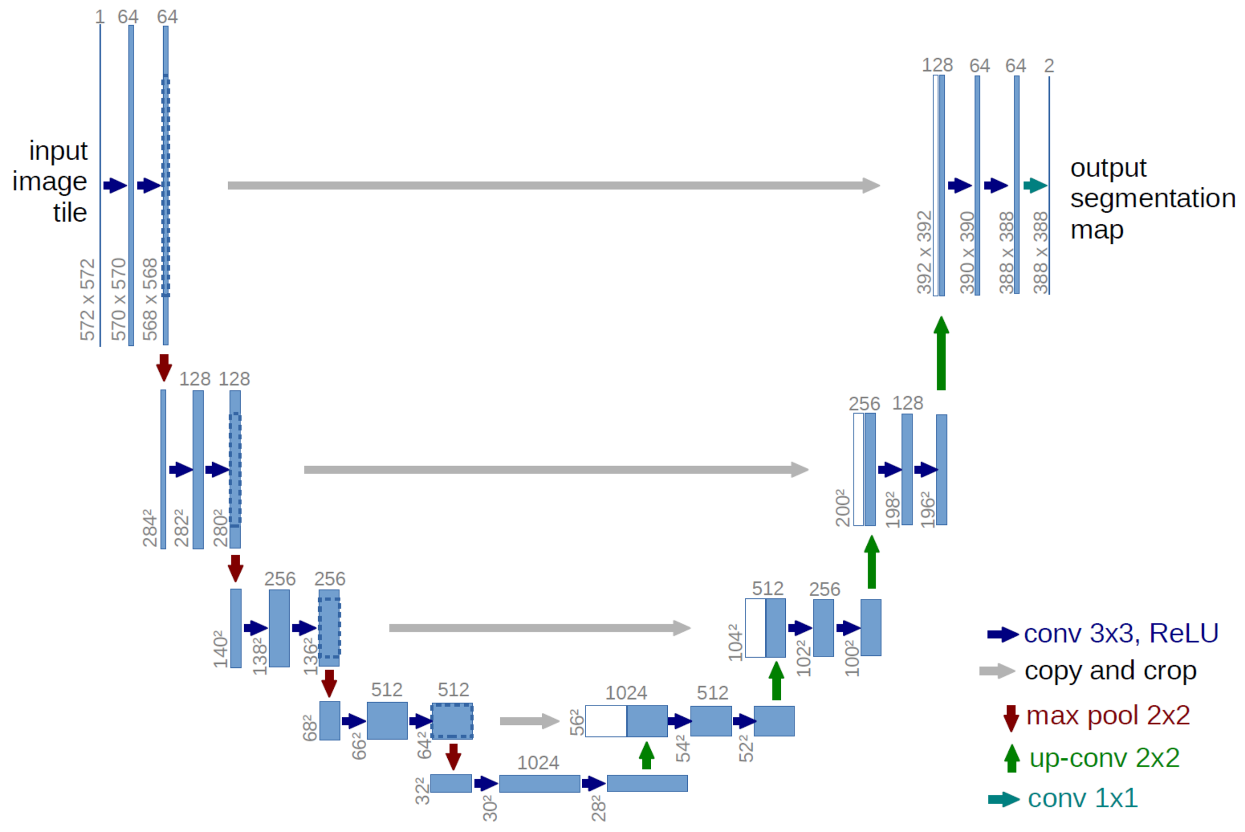


Figure 2.5 Original U-Net architecture. Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. Reproduced from Ronneberger et al. (2015) with permission.

predicted source spectrum.

The spectrogram U-Net achieved state-of-the-art performance on singing voice separation when it was published. The authors further demonstrated the effectiveness of the U-Net's skip connections by showing that removing them had a significant negative impact on the model's performance.

Wave-U-Net

Wave-U-Net is similar in architecture to the spectrogram U-Net. Its main contribution was to abolish completely the use of spectrograms and apply the U-Net architecture directly on the time-domain signal. The signal is cut into short segments and each segment is fed into the network to directly predict the separated sources. Since a waveform is one-dimensional, the network must now use one-dimensional convolutions rather than the two-dimensional convolutions used in image processing. According to the authors' experiments, Wave-U-Net yields separation results that are superior to those of the spectrogram U-Net. See Figure 2.6 for an illustration of the Wave-U-Net architecture.

The Wave-U-Net authors made several other important architectural changes. First, they observed that the transposed convolution operation used by the original U-Net in the upsampling layers causes aliasing artifacts. This was also previously observed by researchers working on image processing as 'checkerboard artifacts' (Odena et al., 2016). In order to avoid the aliasing artifacts, Wave-U-Net replaced the transposed convolutions with linear interpolation followed by normal convolution.

Another contribution made by Wave-U-Net authors is refraining from the use of zero-padded convolutions in downsampling layers. Zero-padding was originally used in downsampling layers to avoid shrinking the image after every layer, and to better preserve information at the borders of the image. However, Wave-U-Net authors found that using zero-padded convolutions created transient-like artifacts at the beginning and the end of the network's outputs. Since the network operates on short segments that are subsequently concatenated to each other, those artifacts created discontinuities at segment boundaries and their effect was intensified. As already mentioned, zero-padding is needed to preserve the dimensions of the input data; without it, the network's output will be shorter than the input. However, due to the segment-wise manner in which Wave-U-Net operates, this limitation was turned into an advantage. Instead of padding input segments with zeros, every segment is padded

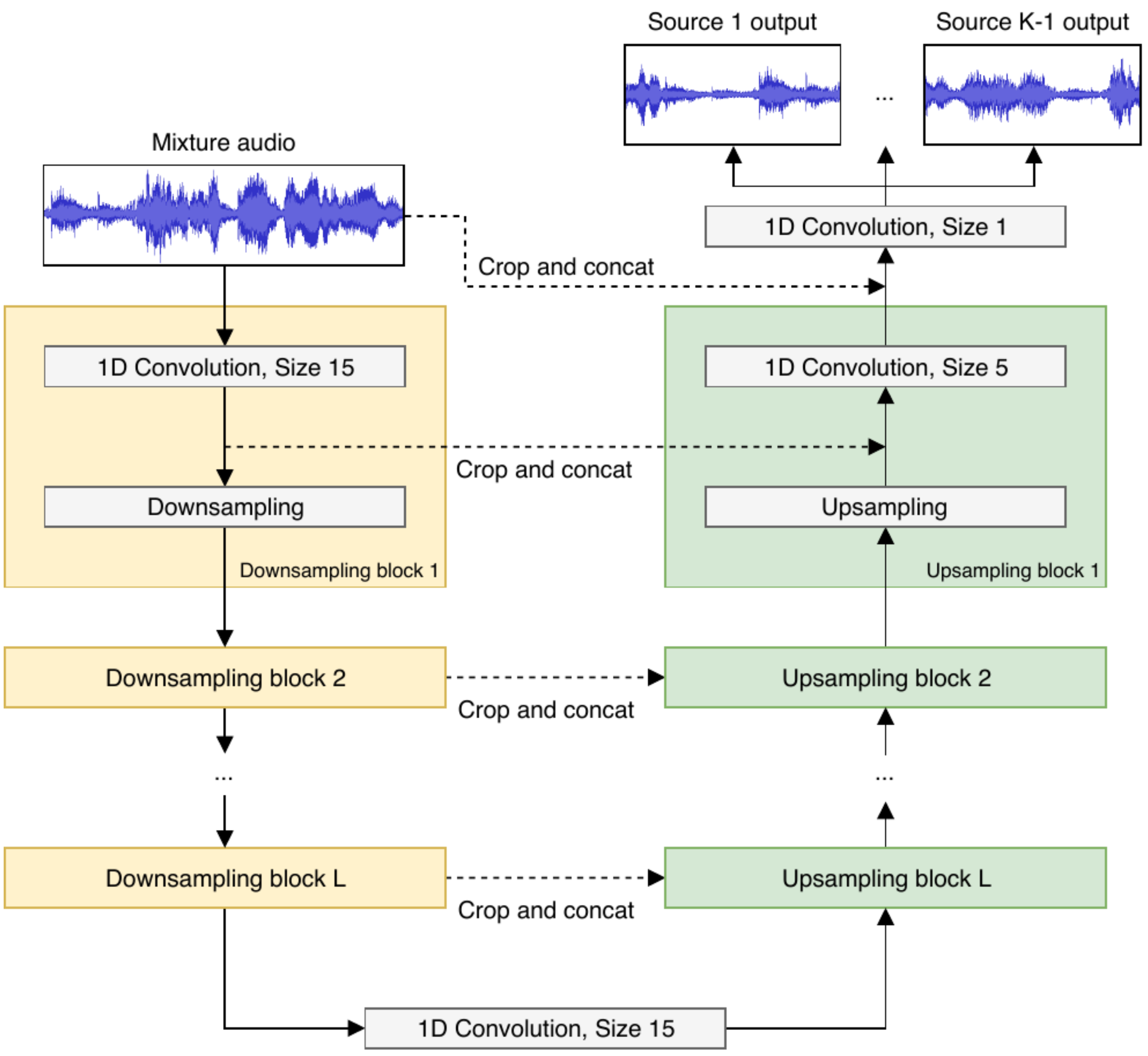


Figure 2.6 Schematic illustration of the Wave-U-Net model architecture. Copyright (c) 2018 Daniel Stoller, licensed under the MIT License.[7]

[7]`https://github.com/f90/Wave-U-Net/blob/master/LICENSE`

with the actual audio that precedes it and follows it. In other words, in order to process an audio track it is divided into *overlapping* segments, and each of those segments is fed to the network to predict a much shorter source segment (the input and output segments are centered on the same time point). In this way, the network is able to use the temporal context in the input signal to effectively predict the separated sources without artifacts at segment boundaries.

Wave-U-Net further improved separation quality by enforcing source additivity. Enforcing source additivity is based on the assumption that the mixture is a sum of all sources. Originally, Wave-U-Net had predicted every source independently of the others. The source additivity assumption means that in order to predict n desired sources, only the first $n-1$ sources need to be predicted independently. The last source is predicted simply as the difference between the mixture and the sum of all other source predictions.

Wave-U-Net can be trained to work with multi-channel audio inputs and outputs. The authors report that working with stereo input and output yielded better results than working with mono audio. This shows that Wave-U-Net is able to leverage inter-channel correlations to improve separation.

2.4 Choral Music

A choir is an organized group of singers who sing together. The music sung by a choir is called choral music. Choral music can be monophonic if all singers sing the same melody at the same time. However, choral music is often polyphonic in nature, that is, it is comprised of different parts that are sung simultaneously.

Until the 20th century, many researchers had believed that polyphonic choral music was invented by medieval Christian monks in the 9th century. However, ethnomusicological research has made it clear that traditions of polyphonic choir music could be found in many different cultures in various parts of the world long before the 9th century (Jordania, 2011). Nonetheless, in this thesis we focus on professional music ("art music") rather than traditional music. The term choral music will be used hereafter to refer specifically to the style of European professional choral music, and mainly to music composed during the Baroque, Classical, and Romantic eras (17th to 19th centuries).

2.4.1 History and Musical Styles

In many cultures, the origins of traditional polyphonic choir music are not well-understood due to the lack of historical documentation (Jordania, 2011). However, the course of development of European professional choir music is fairly well-known based on documents dating as early as the 9th century. *Musica enchiriadis* is a treatise from the 9th century that details contemporary singing practices in liturgical music of the Christian Church, including a polyphonic singing technique called *organum* (Erickson, 2001). The treatise was disseminated widely in Europe and is considered one of the most widely read medieval music theory treatises. Most researchers today believe that the polyphonic singing practices described in the treatise originated from the integration of traditional European polyphonic traditions into the rigid framework of monophonic church hymns that were the prominent form of liturgical music at the time (Jordania, 2015, p. 266). The fusion of these two contrasting styles led to the creation of the unique choral style known to us today as the style of classical choir music, influenced by dominance of the melody on the one hand and by independence of parts on the other hand.

Early *organum* consisted of parallel singing in consonant intervals only. Subsequent developments between the 9th and 12th centuries introduced more sophisticated types of interaction between parts such as contrary, oblique, and similar motion. Rhythmic modes were introduced as a creative principle to guide the rhythmic organization of notes in a melody. These developments and others lead to a gradual increase in the independence of parts in the polyphonic texture and paved the way for the development and formalization of techniques known as *counterpoint*. Counterpoint is a set of musical composition techniques employing simultaneously sounding musical parts (Sachs & Dahlhaus, 2001). The term was first used in the 14th century and the techniques continued to evolve throughout the following centuries. Early counterpoint techniques are considered to have culminated in the Renaissance era with the compositions of Giovanni Pierluigi da Palestrina in the 16th century. While early counterpoint techniques are characterized by strict rules, the 17th century saw an insurgence of a more free counterpoint style in which less limitations were put on the treatment of dissonant sonorities.

An important transition in musical style occurred in the 16th and 17th centuries. Counterpoint of the Renaissance was *modal*: parts were treated as individual lines and the main principle guiding part writing was melodic coherence; pitches in each part were organized using one of eight musical *modes* that determined the set of allowed pitches and the melodic relationships between them. Going into the 17th century which marks the beginning of the Baroque era, the musical basis for counterpoint moved gradually towards the newly developed concept of *tonal harmony.* In tonal harmony there is a crucial importance to sonorities created from the combination of simultaneous pitches and the interactions between these sonorities. In other words, rather than focusing on individual part melodies, writing was now increasingly guided by chords and chord progressions (Hyer, 2001). Pitches in tonal harmony are organized using one of only two modes, called major and minor, instead of the eight medieval modes. The complex imitation textures of Renaissance counterpoint were simplified into more homophonic textures that consist of one dominant melody and harmonic accompaniment.

In the ensuing Classical era (18th century to early 19th century), the use of tonal harmony continued to expand as a framework of organization for vocal and instrumental music alike. While in Baroque music some harmonic progressions were still motivated by counterpoint considerations of melodic motion, in the Classical era it was harmony and tonal functions that served as the main basis for musical organization. In Classical music, the harmonic rhythmic (the rate at which chords change) is considerably slower than in Baroque music. That is, chords were no longer a passing phenomenon, and harmony was now used to establish structure both locally and globally. Locally, chords fulfilled functions of tension and relief in relation to the tonic (the "home key") and were used for syntactical purposes to form phrases. Globally, harmony was used to create long-term structure and coherence (Cohn et al., 2001).

In the Romantic era (19th century to early 20th century), the developments in harmony can be seen as reflecting the general spirit of that era, a spirit of increased personal expression and emotions. New sonorities that extend the pure harmonies of the Classical era were valued as novel and inspirational. Music development seems to have stemmed from a "more of everything" kind of mentality: longer compositions, larger ensembles and orchestras, new musical instruments, and an extended harmonic language characterized by more frequent use of dissonances.

2.4.2 Choir Structure

Choirs that sing in the style of European professional choir music are generally structured by dividing the singers into four different parts according to their vocal range. A singer's vocal range is the range of notes that are comfortable for them to sing. Women singers are split into two parts: those with higher-pitched voices are called *soprano* while the lower-pitched are called *alto*. Men are similarly split into two parts: the higher men voices are called *tenor* and the lower *bass*. This common choir structure is dubbed SATB: soprano, alto, tenor, bass.

Most choir music from the Baroque, Classical, and Romantic periods is written for SATB choirs. Other common types of choirs include women's choirs, usually comprised of two soprano and two alto sections (SSAA) or two soprano sections and one alto section (SSA); male choirs, with two tenor sections and two bass sections (TTBB); and boy choirs, which are normally made up of boys whose voices have not yet changed and thus sing in the soprano and alto ranges.

Sometimes a choral part is split into two or more parts; this is called *divisi*. For example, the soprano part can be split into soprano 1 and soprano 2. Divisi can be requested by a composer for sections where more than four simultaneous pitches are desired.

A singer's assignment into one of the choir parts is determined first and foremost by their vocal range, but the decision is not always straightforward. It is influenced by many additional factors including the singer's vocal timbre, agility, ability to sing harmony (as opposed to melody), the blend of a singer's voice with other singers in the same section, and overall choir balance (Smith & Sataloff, 2013). Even the exact vocal ranges used for voice classification vary between sources, and are different between classification for solo singing and choral singing. In choral music, composers tend to write mostly pitches that lie in the comfortable range for most singers, contrary to solo singing (e.g., opera) where limits of the vocal range are often stretched.

Approximate standard choral voice ranges are given in Table 2.1. Throughout this thesis we indicate pitches using Scientific Pitch Notation (Holoman, 2014) that consists of a pitch name followed by an octave number, where C4 denotes the "middle C" note on a piano and A4 denotes the A above that middle C with the standardized fundamental frequency of 440 Hz.

Table 2.1: Standard vocal ranges in choral music according to Smith & Sataloff (2013, p. 234). Other sources give slightly different ranges, see for example The Harvard Dictionary of Music (Randel, 2003b). Frequencies are given according to a reference point of A4 = 440 Hz.

Voice	Pitch Range	Frequency Range (Hz)
Soprano 1	F4–A5	349.23–880.00
Soprano 2	C4–F5	261.63–698.46
Alto 1	A3–E5	220.00–659.26
Alto 2	F3–D5	174.61–587.33
Tenor 1	F3–A4	174.61–440.00
Tenor 2	C3–F4	130.81–349.23
Bass 1	G2–D4	98.00–293.66
Bass 2	E2–C4	82.41–261.63

Note the significant amount of overlap between the pitch ranges of the different voices. For example, a low bass and a low soprano are able to sing the same note, and the alto 2 range is completely contained in the tenor 1 range. However, it is standard practice for composers to maintain the natural ordering of the voices. That is, when all voices sing at the same time, soprano will always sing the highest note, followed by alto, tenor, and bass in that order. Sometimes, however, melodic, timbral, or other musical considerations cause composers to create a voice crossing in which the natural ordering is disrupted.

A choir is normally led by a *conductor*. The conductor is a person who uses his hand, face, and body gestures to lead the choir in song during performance. Furthermore, the conductor has the important role of directing choir rehearsals. A choir rehearsal normally starts with warm-up and voice exercises, followed by practice of the musical pieces that are to be performed. The conductor manages the practice by choosing what pieces and which sections are worked on. In some professional choirs, singers are able to sight-sing their parts from the musical score without prior learning of the parts. However, in difficult pieces and in amateur choirs, singers often need to sing their parts several times before mastering the pitches, the rhythms, and the text. Thus, a significant portion of choir rehearsals is devoted to part reading in order to achieve familiarity with the music, many times assisted by a rehearsal pianist doubling the choir parts on the piano. In order to enhance the musicality and expressivity of the performance, the conductor often adds their own personal interpretations on top of the score and makes choices regarding every aspect of the performance, including dynamics, tempo, articulation, phrasing, breaths, timbre, and diction.

The size of choirs varied widely across centuries and still varies today across choirs. In early medieval days of polyphonic vocal church music, it may well be that every part was sung by only a single singer; there is no evidence to support that multiple singers sang the same part. In the Renaissance, church choirs grew considerably: choirs of the 15th and 16th are estimated to have had an average of 20 to 30 singers (Randel, 2003a). In the 17th and 18th centuries choirs continued to grow up to 40 singers and more, and in special events choirs were sometimes combined to form bodies of several hundred singers. In Germany, however, the musical customs of the Protestant church were quite different. There is considerable amount of debate among scholars about the conventional size of choirs, specifically regarding choirs in the music of J. S. Bach. Some believe that Bach wrote for choirs with only one singer per part, with supporting singers (*ripienists*) who joined the main singers only for some movements (Parrott, 2010; Rifkin, 1982). Others believe that Bach's choirs were customarily 12 to 16 singers in size (Glöckner, 2010).

In the 19th century, choir sizes continued to grow; community choirs increased in popularity with the expanding availability of musical education in Europe and North America. Earlier pieces that were originally composed for small choirs were now being customarily performed by choirs of 40 singers and more. In large-scale pieces involving both a choir and an orchestra, very large choirs were required to rival the powerful sound of the increasingly large orchestras employed at the time. In the premiere of Beethoven's 9th symphony in 1824, for example, it is estimated that the choir consisted of between 80 and 120 singers (Kelly, 2001, p. 134). Mahler's 8th symphony ('Symphony of a Thousand') is an extreme example in the late Romantic period: the choir in this symphony's premiere in 1910 consisted of no less than 858 singers, with an orchestra of 171 instrumentalists (Floros & Wicker, 1995, p. 243).

Despite such extreme examples, smaller choirs have continued to exist and perform throughout the 19th, 20th, and 21st centuries. Today, choirs are popular in many cultures around the world not only as professional performance bodies, but also (and mainly) as amateur groups performing for the sake of enjoyment. For example, a recent survey of Canadian choirs and choir singers in 2017 has found that an estimated 10% of Canada's population sing in choirs (Hill, 2017). The most common types of choirs in Canada are church choirs, school choirs, community choirs, and professional choirs. According to the survey, the estimated total number of choirs in Canada is 28,000, the majority of which are church choirs (63%). The median number of singers in a choir was found to be 36.

2.4.3 Musical Scores

Musical scores are a key element of the work presented in this thesis. In the following chapters, we use musical scores to generate a synthetic training dataset for source separation (Section 3) and to guide source separation (Section 6). In this section we therefore present the characteristics of musical scores with a specific focus on choral scores.

A musical score is a graphical symbolic representation of a musical composition (Charlton & Whitney, 2001). The score is created by a composer to indicate to performers how the music should be performed. A score is similar to a recipe in this respect: it guides the performance but cannot completely describe the music (as a recording, for example, could); some room is always left for interpretation by the performers. The term score is used with different meanings: sometimes, it is used broadly to refer to all written or printed representations of music. Other times, it is used more narrowly to refer only to such representations that show the music of all the instruments and voices participating in a composition (as opposed to showing only one or some of the parts). We henceforth use the term score in the latter more narrow sense.

History and Characteristics of Music Notation

Scores are written using music notation. Many different music notation systems have developed in various cultures around the world. The most prevalent notation system in use today for Western art music is *common Western music notation*. This standardized notation system establishes a common language between composers and performers around the world, thus enabling performers to easily learn new pieces and granting composers the knowledge that pieces they write will be interpreted correctly.

The complex notation system we know today as common Western music notation has evolved gradually over many centuries through various innovations introduced by individual composers and theoreticians. The roots of this notation lie in a system of signs known as *neumes* that were used by monks to indicate the melody contours of church chants. The earliest examples of neume notation date back to the 9th century and were found in various places in Europe (Bent et al., 2001).

Between the 9th and 19th centuries, the rudimentary set of neume signs developed into a complex system for notating almost every aspect of music, including pitch, rhythm, meter, note grouping, key, dynamics, articulation, tempo, and instrument-specific playing techniques (Bent et al., 2001). Notable notation innovations throughout history include: the introduction of staff lines and clefs for precise indication of pitch, circa 1030 by Guido of Arezzo; the adoption of square notation in the 13th century shifting the focus from melodic lines and phrases to individual notes; the formulation in the 13th century of rhythmic modes as an abstraction of repeating rhythmic patterns; the development of mensural notation throughout the 13th, 14th, and 15th centuries for more precise indication of note duration; bar lines in the 16th century for a clear visual separation of time units; increased usage in the 16th and 17th centuries of beams and slurs to group and connect notes; fractional time signatures to indicate meter independently from tempo in the 18th century along with verbal indications of tempo; formalization of accidentals and key signatures in their modern sense in the 17th and 18th centuries to indicate raising or lowering of a note's pitch; dynamic markings in the 17th century to indicate loudness and hairpin symbols in the 18th century to indicate gradual change in loudness; the gradual adoption in the 17th to 19th centuries of scores with multiple staves for notating keyboard music and music with multiple instruments or voices (scores previously existed in the Middle Ages for polyphonic singing but were abandoned around the 13th century); and finally, the large expansion of the vocabulary of dynamics and articulation signs in the 19th century.

The long list of notation developments presented above is just a small fraction of the diverging historical trends; many other developments and proposed systems did not make it into the mainstream notation system that is widely taught today to musicians. See Bent et al. (2001) for an in-depth description of the history of Western music notation and a review of other notation systems.

Types of Scores

There are several types of musical scores. A *full score* contains the music for all instruments and voices participating in a piece. Each part is written on a separate staff and staves are laid out one under the other on each page. It is often not convenient for performers to play directly from a full score, as the music of other parts could distract them from playing their own part. This is especially true in music with many instruments such as orchestral music.

For this reason, in these cases performers play from individual *parts* showing only their own music. Conductors still use the full score during performances and rehearsals because they need to integrate information from all parts in order to conduct effectively.

Choral Music Scores

In choral music, the standard practice is that singers sing from the score rather than from individual parts (Butcher & Studebaker, 1985). Singing from the score encourages every singer to be more aware and attentive to the music sung by other voices. In pieces involving both a choir and an orchestra, it would be inconvenient for singers to sing from a full score with all orchestral parts in it; in these cases, singers use a *vocal score* that contains all vocal parts but reduces all instrumental parts to a single piano part. The piano part can then be used by a rehearsal pianist during choir rehearsals.

Figure 2.7 shows an example of a vocal score for a piece of choral music. The score is made up of five parts: soprano, alto, tenor, bass, and piano. The piano part is a reduction of all the original orchestra parts combined for use in rehearsals. Every part is written on a single staff, except for the piano part that is written across two staves (one staff for the left hand and one for the right hand). The score excerpt contains two lines of music, called 'systems', and each line contains three bars that are separated by bar lines. A time signature at the beginning of the excerpt (circled in blue) indicates the meter, $\substack{12\\8}$, meaning every bar is made up of 12 eighth notes (four beats of three eighth notes each). Every staff contains notes indicating pitch and duration. Some notes are connected using ties to extend their duration. Rests on the staff indicate the absence of any note. Slurs are used to mark several notes that are sung on the same syllable (this is a convention specific to vocal writing). Dynamic markings (circled in orange) indicate loudness: ***p*** stands for *piano* meaning soft, ***f*** (*forte*) means loud, and *cresc.* (*crescendo*) indicates a gradual increase in loudness. The tempo marking *Larghetto* (circled in green) indicates the speed and character in which this excerpt should be performed: 'fairly slowly'. A clef on each staff (circled in red) determines the pitch reference point for reading note pitches on that staff: some staves use the treble clef and others use the bass clef. A syllable is written under each note (underlined in pink) to indicate the text for singing, and dashes are used to connect individual syllables into words.

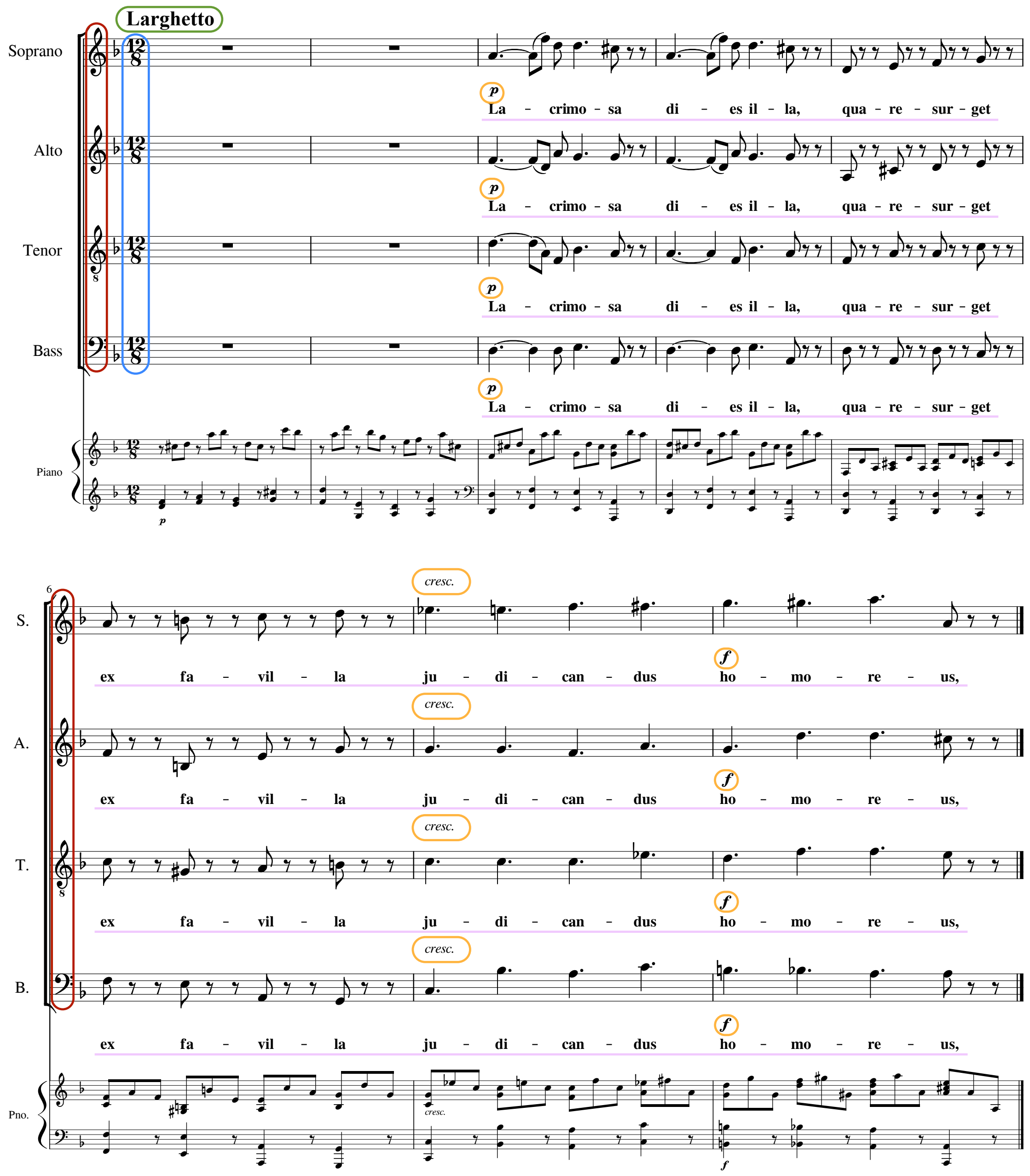

Larghetto
Soprano
Alto
Tenor
Bass
Piano
p
La - crimo - sa di - es il - la, qua - re - sur - get
6
S.
A.
T.
B.
Pno.
cresc.
f
ex fa - vil - la ju - di - can - dus ho - mo - re - us,

Figure 2.7 An example excerpt from a vocal score: the first phrase of the Lacrimosa movement from the Requiem by Wolfgang Amadeus Mozart. See text for an explanation of the colored markings. Score under the public domain, edited by Jes Wagner.[9]

[9]`http://www.cpdl.org/wiki/index.php/Requiem,_KV_626_(Wolfgang_Amadeus_Mozart)`

2.4.4 Acoustical Characteristics

Since our work in this thesis is concerned with processing recordings of choral music, it is crucial that we understand its acoustical properties. Before examining choirs specifically, we briefly explain the acoustics and physiology of the human singing voice, given that a choir is simply a group of singers.

The Singing Voice

The human singing voice can be described acoustically as a “voice source” generated by the vibrating vocal folds and a vocal tract “filter” that modifies it (Sundberg, 1987). Physiologically, the voice starts as a stream of air that exits the lungs and passes through the vocal folds and then through the vocal tract. Under the right conditions, the air flow created by the breathing system causes the vocal folds to vibrate and generate a sound called the *voice source*. The vibration of the vocal folds is periodic and so the voice source can be described as a harmonic series of partials. The partials’ amplitudes decrease at a rate of about 12 dB per octave; the exact spectral envelope of the voice source varies across voice types and people (Sundberg, 1987, pp. 63–65). The voice source subsequently passes through the vocal tract, which consists of the laryngeal cavity, the pharynx, the mouth cavity, and the nasal cavities. The vocal tract acts as a resonator, amplifying some frequencies more than others. The frequencies that are transmitted with the highest amplitude through the vocal tract are called the *formant frequencies* (Sundberg, 1987, p. 93). As the shape of the vocal tract changes, so do the formant frequencies. It is this change in formant frequencies that creates the distinct timbre of each vowel in speaking and in singing.

The timbre of the human singing voice varies considerably between groups of people. For example, women normally have a distinctly different voice timbre than men, and children have a different timbre than adults. Furthermore, the timbre varies among individuals within those groups: no two people have exactly the same voice. The factors that determine a person’s voice timbre are mainly morphological, that is, they are related to the shape and structure of the person’s voice production organs (Sundberg, 1987, pp. 1–2). The length, thickness, and viscosity of the vocal folds affect the voice’s pitch range. The dimensions of the vocal tract (the shape and size of the pharynx and mouth cavities) affect formant frequencies. Apart from these inherent morphological differences, a person’s speech and singing voice are greatly affected by the way the voice production organs are used. Naturally-acquired speech habits vary across geographical regions and across societies, and professional voice training can further modify the timbre and range of the singing voice.

Choral Singing

Acoustically, choral music can be seen simply as a superposition of several singing voices. There are, however, several acoustical characteristics that are unique to choral music. Ternström & Karna (2002, p. 272) give a holistic overview of topics in choir acoustics, including issues related to loudness and balance (diversity of singers' vocal power, effects of choir size and formation, and a singer's perception of their own voice compared to other singers), intonation (vibrato, vowel-related pitch perturbations, interval intonation between voices, unison intonation within voices, and the effect of amplitude on pitch perception), timbre (acoustical differences between voice types, trained vs. untrained voices, and pitch salience in relation to timbre), and inter-voice relationships (uniformity of vowels, ensemble timing, and blending of different voice types).

Ternström (2003) reports results of several empirical investigations into choral music acoustics. One study measured the effect of room acoustics on choir singing and found that singers from different choirs consistently adapted their voice usage and sound levels to the room acoustics. Another study found that intonation errors of a choir singer significantly increase when the individual singer's sound level is low compared to the combined sound level of the other singers. The paper also presents several studies that investigate the differences between solo singing and choral singing. In solo singing, singers were found to use vibrato with a larger extent compared to choral singing. Furthermore, singers asked to sing in solo singing style were found to use more power in a frequency region dubbed the *singer's formant* (2–3 kHz), while when singing in choral style they did not exhibit this phenomenon, and emphasized the lower partials instead. It is clear that solo singing and choir singing are fundamentally different tasks, and when singing in a choir singers tend to reduce the individuality of their timbre and strive to blend with the rest of the choir (Sundberg, 1987, p. 143).

Several studies have investigated intonation specifically. Jers & Ternström (2005) studied a multi-channel choir recording and reported several findings: ascending scales were more accurately intoned than descending scales; melodic big intervals (fifths and octaves) are enlarged; during note transitions, intonation within voices is more scattered (as is to be expected due to non-perfect transition synchronization); intonation is also more scattered during sections with a faster tempo; and vibrato seems to be at least partly synchronized between singers. Daffern (2017) studied vibrato in an SATB vocal quartet and found that singers attempt to control vibrato in order to improve blend by reducing its extent, and that in some cases singers synchronized their vibrato with each other. Dai & Dixon (2017) also investigated SATB vocal quartet singing under different conditions and found that when singers can hear each other, the pitch error of each individual singer increases but the harmonic interval error between the singers decreases; this can be interpreted as an attempt by singers to adjust their intonation to each other so that the overall ensemble tuning is improved. Cuesta et al. (2018) analyzed semi-professional choir recordings and found that pitch dispersion within a specific voice varied between 16 and 30 cents, and these results agree with those reported in Ternström (2003). In the same study, singers were found to use vibrato in less than half of the notes, but it is suggested that professional choir singers would employ vibrato more often.

When a choir sings a cappella (i.e., without instrumental accompaniment), pitch drift may occur due to the absence of a reference tone. When a choir experiences pitch drift, the intonation of intervals within the choir remains the same but all the pitches of all voices shift together with respect to an absolute reference pitch. The most common type of drift is gradual lowering of the pitch (Alldahl, 2008, p. 4). Howard (2007) investigated drift in a cappella SATB singing and found that the singers' tendency to non-equal temperament caused pitch drift when singing a piece containing certain key modulations. Devaney et al. (2012) made a similar study with three-part a cappella ensembles and found that a specially-crafted exercise caused a slight drift upwards in pitch. Interval intonation is just one possible cause for drift; Mauch et al. (2014) found that even solo singers drift slightly when singing without accompaniment.

Solo Singing and Choral Sound Characteristics

To illustrate the acoustical characteristics of vocal and choral music, Figure 2.8 shows a spectrogram of a short musical excerpt along with the corresponding score. The excerpt consists of a short musical phrase that is repeated twice: it is first sung by a bass soloist and then repeated by the choir (the sopranos repeat the melody sung by the soloist while the other three voices provide harmony). The spectrogram illustrates the main acoustical characteristics of vocal and choral music. In the solo phrase, partial trajectories are distinctly visible: the fundamental frequency (starting with a G3 note at around 196 Hz, marked with the letter A) and above it equally-spaced overtones up to about 3200 Hz. The singer's vibrato can be seen as oscillations in the partial trajectories, which have a larger extent in the higher partials (marked with the letter B) due to the nature of the harmonic series.

The sung vowels influence the sound's spectral envelope. For example, the onset of the Norwegian vowel /æ/ (for the word "er") is clearly marked by an increase in amplitude in frequencies around 700 Hz (marked with the letter C), which is the frequency of the first formant for that vowel. Wherever the consonant /s/ appears in the text, a vertical bar appears in the spectrogram (marked with the numbers 1–8 in the spectrogram and in the score) showing that the energy is spread across a large range of frequencies in the pronunciation of this fricative consonant. There are no partial trajectories during the pronunciation of /s/ because it is an unvoiced consonant. Due to room reverberations, the /s/ sound decays gradually rather than ending abruptly when the singer finishes to pronounce it.

In comparison to the solo half of the excerpt, the choral half may seem more disorganized. Partials appear more like "blobs" rather than clearly distinguished lines because the sound is a mixture of many voices, and the voices are neither perfectly synchronized nor perfectly in tune with each other, even though the recording was made by professional musicians and generally sounds very well-blended and in tune. Vibrato can barely be distinguished in the choral phrase: as discussed above, singers tend to use smaller vibrato when singing choral music, and the vibrato of different singers are for the most part unsynchronized. Even the sibilant consonants appear more "fuzzy" due to pronunciation and timing variations (numbers 5–8 compared to numbers 1–4). Since four parts are sung simultaneously, multiple fundamental frequencies can be observed (marked with the letter D) along with their corresponding overtones, which reach higher frequencies (marked with the letter E) due to the participation of the soprano, alto, and tenor singers.

In conclusion, this example illustrates that vocal music has complex acoustical characteristics, mainly due to smooth pitch changes (such as vibrato and note transitions), the combination of voiced and unvoiced elements in sung lyrics, and continuous variations of the spectral envelope. In choral music, these complexities are intensified by the fact that the choir sound is a superposition of multiple singers' voices, where each singer has a unique and variable timbre, and synchronization between singers is never perfect. Due to these characteristics, it is logical to assume that source separation of choral music would be more

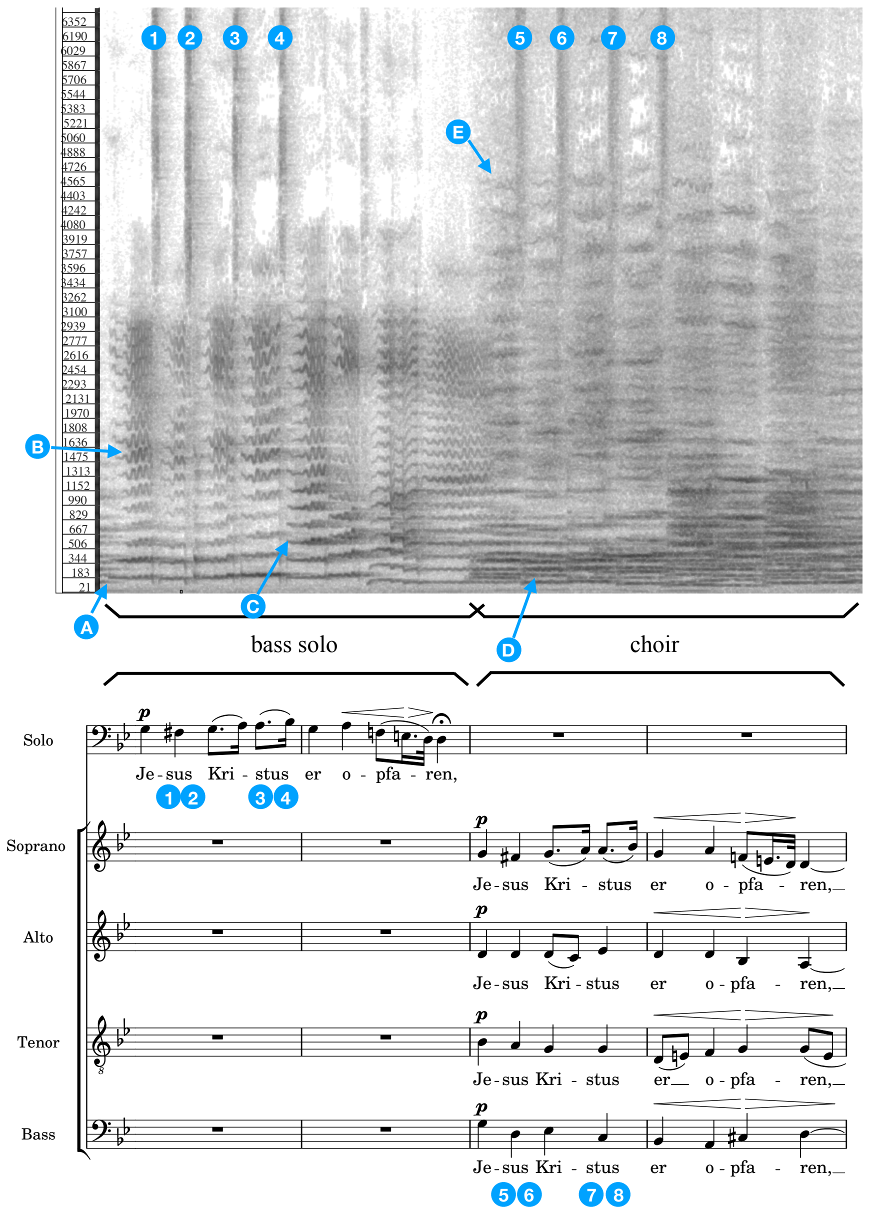


Figure 2.8 A spectrogram and a score showing solo singing and choral singing side by side. The spectrogram shows an excerpt from an unaccompanied choral music recording, 'Jesus Kristus er opfaren' (op. 74 no. 3) by Edvard Grieg performed by Grex Vocalis and Carl Høgset.[12] The excerpt's score is shown below the spectrogram. The score was transcribed by Peter Kaplan and is used with permission.[13] See text for an explanation of the colored markings.

[12]The performance can be found on YouTube: `https://youtu.be/9BxiCtbEpxQ`
[13]`http://www.cpdl.org/wiki/index.php/Jesus_Kristus_er_opfaren_(Edvard_Grieg)`

challenging than many other source separation tasks.

2.5 Score-Informed Source Separation

As mentioned in Section 2.1.2, blind source separation techniques are difficult to use for real-world musical applications. Most separation techniques use prior information of some kind to guide the separation process, for example, in the form of training data or signal models that characterize the acoustical properties of each source. When separation relies on prior information that is highly detailed, it is called *informed* separation. One such highly detailed source of information is the musical score, when available.

As discussed in Section 2.4.3, musical scores contain many types of information. At the most basic level, the score specifies which parts (instrumental or vocal) are contained in a composition. For each part, the score specifies a sequence of notes with pitch and timing information for every note. In addition, the score may contain instructions specifying meter (a recurring pattern of beats), tempo (speed and character), dynamics (relative loudness and gradual loudness changes), articulation (type of note attack or transition between notes), and in the case of vocal music, lyrics.

Score-informed source separation techniques use elements from the musical score to guide the separation process (Ewert et al., 2014). Normally, these techniques rely mostly on the part, pitch, and timing information from the score. The way in which this information is used varies across techniques.

2.5.1 Score Alignment

Unfortunately, there is no direct mapping between time in the score and time in a recording that corresponds to that score. This is due to the fact that the score contains only high-level timing information which is then interpreted by a human performer. Multiple recordings of the same score often have different timing both globally (due to different playing speeds) and locally (due to expressive interpretation and slight timing variations). As such, before employing a musical score for source separation it is necessary to align it to the recording.

Score alignment (also known as score-audio synchronization) can be done manually, but this is time-consuming and error-prone, so we wish to automate it. Automatic score alignment has been tackled in two different scenarios: online and offline (Dannenberg & Raphael, 2006). In online alignment, also known as score following, the score is aligned in real-time to an incoming music signal. This is used in applications such as automatic page turning and real-time computer accompaniment. For score-informed source separation, however, we are generally interested in the offline scenario, in which the complete recording is available when performing the alignment.

Score alignment is generally performed by extracting a sequence of features from both the audio and the score and then finding the optimal alignment between those two sequences (Ewert et al., 2014). Chroma-based features have been found to work well for this purpose as they can easily be extracted from both audio and scores (Hu et al., 2003). Recently, transposition-invariant features have been proposed to allow for matching of scores with recordings transposed to a different key (Arzt & Lattner, 2018).

For aligning the extracted feature sequences of the audio and the score, two commonly-used techniques are dynamic time warping (DTW) and hidden Markov models (HMMs). Both are general techniques that are applied to score alignment. DTW finds an optimal alignment between any two time series while allowing for stretching ("warping") along the time axis (Hu et al., 2003). HMMs can be used to model the musical performance as a stochastic process with multiple hidden states; they have been employed to align recordings of the singing voice (Cano et al., 1999).

2.5.2 Separation Techniques

Scores have been used as early as 1998 to guide musical source separation (Meron & Hirose, 1998). One popular way to use the score is synthesizing a signal that is similar to one of the sources and then using it in some way to guide separation. Meron & Hirose (1998) separated a singer from piano accompaniment by synthesizing the piano part from the musical score using note models built from a database of recordings, and then subtracting the synthesized piano part from the mixture to retrieve the separated singing voice. Raphael (2008) separated a solo instrument from orchestral accompaniment by creating a binary mask: every spectrogram bin is classified as belonging to one of the two sources. The classifier that creates the mask is trained on a dataset of synthesized recordings produced from a set of scores. Similarly, Ganseman et al. (2010) created source models from audio synthesized from each source's score, and then applied those source models to real mixtures to perform separation.

Another way to use the score is to create harmonicity-based constraints driven by the pitch and timing information contained in it. Ben-Shalom & Dubnov (2004) performed multiple instrument separation using optimal filtering, where the filter imposed a harmonicity constraint on the separated signals based on the pitches indicated by the score. Li et al. (2009) similarly employed harmonic masks for each source and initialized those masks based on MIDI (Musical Instrument Digital Interface) scores. They then used the masks with least-squares estimation to effectively separate overlapping harmonics. Duan & Pardo (2011) created a real-time separation system in which, again, time-frequency masking was used to assign each spectrogram bin to one of the sources based on the expected pitches in each source. Score following was used in order to align the score to the audio in real time.

Scores were also integrated into the framework of NMF as additional factorization constraints. Ewert & Müller (2012) developed a score-informed NMF technique to separate the left and right hand notes in piano recordings by initializing the NMF basis signals and activations matrices: basis signals were initialized to harmonic combs and activations were initialized as binary matrices resembling a piano roll representation of the score. We describe this technique in detail and present our own implementation in Section 4 below. Hennequin et al. (2011) employed a technique quite similar to Ewert & Müller but used a parametric harmonic model for source spectrograms rather than fixed basis signal initializations. Using a parametric model enabled the factorization to explicitly take vibrato into account.

Şimşekli & Cemgil (2012) performed separation using a method called generalized coupled tensor factorization, which is a generalization of NMF to simultaneously factorize multiple matrices or tensors. They proposed two models: in the first, activations are constrained according to the score (similar to the piano roll initializations described above) and basis signals are constrained using isolated note recordings. For the second model, the activation constraints are also initialized from a score but interestingly, the score does not have to be perfectly aligned to the audio and it can even belong to another piece. Instead of explicit activation constraints, the second model relies on relationships between notes in the score: which pitches tend to appear at the same time. The authors showed that using the second model, which only relied on an approximate score, sometimes surpassed the performance of the first model that required a perfectly aligned score.

Rodriguez-Serrano et al. (2015) also built on top of NMF with instrument timbre models. In their technique, every instrument is constrained using a multi-excitation source-filter model that is pre-trained on solo excerpts of that instrument. Separation is done in real time: the score is aligned to the audio, the pre-trained instrument models are updated to fit the audio according to the score, and finally the mixture is separated based on the updated models. Rodriguez-Serrano et al. (2016) focused on improving separation of overlapping partials by using a variant of NMF called complex matrix factorization (CMF) that takes into account the mixture's phase, as opposed to NMF which only factorizes the magnitude spectrogram. They used the same constraints on basis signals and activations as used by Ewert & Müller (2012), but additionally introduced shift-invariance to the basis signals in order to better account for vibrato in the separated sources.

Recently, score information has also been integrated into separation techniques based on deep learning. Ewert & Sandler (2017) take an interesting approach: they treat separation as an unsupervised learning task by training an autoencoder neural network on each mixture separately. An autoencoder is a network that transforms ("encodes") its input to a low-dimensional representation and is trained to reconstruct ("decode") the original input back from that latent representation. For score-informed separation, activity constraints are imposed on the latent representation during training so that each latent unit is associated to a single note in the score. Separation is then performed on a note-by-note level by altering the latent representation to contain only the desired notes and resynthesizing the corresponding audio using the decoder part of the network. The authors comment that this technique can be seen as a nonlinear extension of NMF because both techniques involve a parts-based representation of the mixture. Compared to a score-informed NMF technique described above (Ewert & Müller, 2012), this autoencoder-based approach was found to perform better on the same task of separating the two hands in piano recordings.

Miron et al. (2017b) created a CNN that operates on masked spectrograms. Instead of feeding the mixture spectrogram directly to the CNN, a filtered version of the spectrogram was created for each source. The "score-filtered" spectrograms were then all fed simultaneously into the CNN as multiple input channels, and the CNN predicted the final source spectrograms which were combined with the mixture's phase and resynthesized. The score-filtered spectrograms were created by applying a mask that kept only frequency bins that were expected to be associated with that source's notes, assuming all note spectra are shaped approximately as harmonic combs. All training data was synthesized from scores with varying human-like performance characteristics such as tempo, dynamics, timbre, and local timing deviations (Miron et al., 2017a). The CNN method was shown to outperform a score-informed NMF system (Miron et al., 2016).

Chapter 3

Synthesized Chorales Dataset

In this chapter, we describe the creation of a dataset that consists of synthesized renditions of chorale harmonizations composed by Johann Sebastian Bach. In the next chapters, we will use this dataset to train and evaluate source separation techniques. Since we wish to experiment with deep learning-based techniques, we require a large dataset of choir recordings in which every choir part is recorded on a separate track without instrumental accompaniment. Unfortunately, such a dataset proved hard to find, as choirs are usually recorded using a set of microphones that capture a mixture of all voices and instruments (Ihalainen, 2008).

Several studies have been performed using multi-track a cappella vocal and choral recordings (Devaney et al., 2012; Jers & Ternström, 2005) but these studies did not publish the actual recordings. The freely available Mixing Secrets multi-track dataset[1] contains three multi-microphone choral recordings, but each microphone picked up multiple voices, so the recordings are unsuitable for use as training examples for source separation.

Recently, a dataset of multi-track choral recordings was published in which every choir part was recorded separately (Cuesta et al., 2018). In order to ensure that all recordings are properly synchronized, singers were instructed to follow a pre-recorded video of the choir conductor. The dataset also contains a semi-synchronized MIDI file for each recording. Unfortunately, the dataset consists of only three songs, yet deep learning methods for source separation normally require a much larger amount of training data. For example, a recent dataset for music separation called MUSDB18 (Rafii et al., 2017) contains 150 songs with a total duration of about 10 hours.

[1] `http://www.cambridge-mt.com/ms-mtk.htm`

Several websites sell professionally-recorded choir practice tracks,[2,3] which are synchronized solo recordings of individual choir parts. However, we were unable to obtain authorization from the rights holders to conduct experiments using these tracks.

3.1 Choir Synthesis

In the absence of a choral music dataset that is suitable for deep learning-based source separation, we opt to synthesize our own dataset. Synthesizing a choir can be achieved by synthesizing multiple singing voices with slight timing, timbre, and pitch variations (Schnell et al., 2000). Overviews of methods for singing voice synthesis are given by Rodet (2002) and Sundberg (2006). Additional approaches include concatenative synthesis (Bonada & Serra, 2007) and deep neural networks (Gómez et al., 2018). Recently, a method for choir synthesis was proposed based on voice cloning: a model that is trained on recordings from multiple singers is adapted to individual target voices using small amounts of data (Blaauw et al., 2019).

Professional choir audio tracks are often produced using commercial sample libraries.[4,5] These sample libraries contain thousands of professionally recorded samples of choirs singing different pitches and syllables with various dynamics and articulations. These short samples can then be combined to produce realistic-sounding choir music. Unfortunately, these sample libraries are prohibitively expensive for use in our project.

In light of the high price and complexity of the aforementioned synthesis methods, we choose a much simpler sample-based approach for our dataset. We use the FluidSynth software synthesizer (Henningsson & Team, 2011), which converts MIDI messages to audio by using audio samples and synthesis rules stored in a SoundFont file. The `MuseScore_General` SoundFont[6], which is distributed with the MuseScore notation software, is free to use and contains decent choir samples. We use the SoundFont's 'Choir Aahs' preset, which contains 14 samples. Each sample is a short recording of a single choir section singing a sustained note on an /a/ vowel. Every sample corresponds to a single pitch. To synthesize a pitch that does not have an associated sample, FluidSynth simply pitch-shifts the sample that has

[2]ChoralPractice: `https://www.choralpractice.com`

[3]Choral Rehearsal Tracks: `https://choralrehearsaltracks.com`

[4]EastWest Hollywood Choirs: `http://www.soundsonline.com/hollywood-choirs`

[5]Vienna Choir: `https://www.vsl.co.at/en/Voices_Complete/Vienna_Choir`

[6]`https://musescore.org/en/handbook/3/soundfonts-and-sfz-files#soundfonts`

the closest pitch. This way, the entire pitch range of the four choral voices is covered. To synthesize a note that is longer than the corresponding sample, a predefined segment of the sample is looped.

This sample-based synthesis method is simple and effective, but it regrettably cannot synthesize sung lyrics. On the one hand, this is a disadvantage because it means our dataset is less representative of real-world recordings. On the other hand, the consistency of the dataset allows us to conduct highly controlled experiments. We believe that insights we gain from using the synthesized dataset will be applicable to real-world recordings as well.

3.2 Bach Chorale Harmonizations

In order to create our dataset, we require a suitable corpus of choral music that is available in MIDI format. We choose a well-known corpus of chorale harmonizations by Johann Sebastian Bach. A chorale is a Lutheran church hymn (Marshall & Leaver, 2001a). Bach harmonized around 400 chorales throughout his life; around half of them are part of large-scale vocal compositions such as oratorios, Passions, and cantatas, while the other half are found in shorter works and unidentified manuscripts (Marshall & Leaver, 2001b). Throughout this thesis, we use the chorale numbering scheme established by the Riemenschneider edition (Bach, 1941), which assigned each harmonization a number from 1 to 371.

Bach's chorale harmonizations possess several characteristics that make them good candidates to serve as a coherent dataset for source separation. They are all written for four voices in homorhythmic texture, that is, the rhythm is identical (or very similar) in all voices (Marshall & Leaver, 2001b). The writing is mostly syllabic (every chord matches one syllable in the text) and the rhythm consists mainly of quarter notes and eighth notes. The chorales are composed in the style of tonal harmony; modulations are relatively infrequent and occur only to closely related tonalities. Structurally, the chorales are built as a sequence of short phrases, each ending with a fermata (musical pause). Figure 3.1 shows an example chorale harmonization that exhibits these characteristics.

3.3 Synthesis Procedure

We have created a program to synthesize the corpus of Bach chorale harmonizations. Our program is made available as open-source software (see Appendix B). The program is written

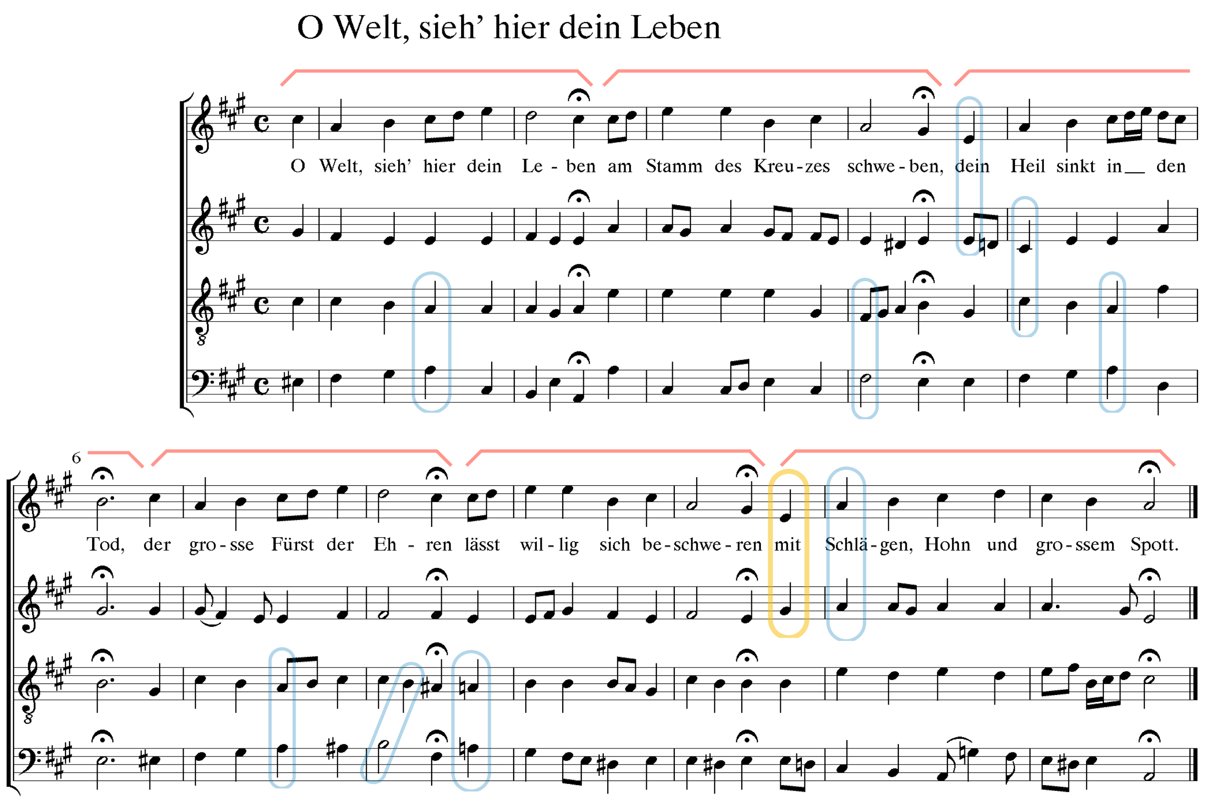

O Welt, sieh' hier dein Leben
O Welt, sieh' hier dein Le - ben am Stamm des Kreu-zes schwe-ben, dein Heil sinkt in den
6
Tod, der gro-sse Fürst der Eh - ren lässt wil-lig sich be-schwe-ren mit Schlä-gen, Hohn und gro-ssem Spott.

Figure 3.1 Chorale harmonization by J. S. Bach, BWV 393 (chorale number 295 in the Riemenschneider edition). The chorale is made up of 6 short phrases of 2 to 3 measures (phrases are marked in red). Each of the four voices maintains its standard range. There is one voice crossing between the soprano and the alto, marked in yellow. Several voices sing in unison on several occasions, marked in blue. Most chords are homorhythmic apart from a few passing tones and similar embellishments. The chorale is written in the key of A major with tonicizations of the 5th and 2nd scale degrees. This transcription by Center for Computer Assisted Research in the Humanities is licensed under CC BY-NC 3.0.[8]

[8]https://creativecommons.org/licenses/by-nc/3.0/

in Python and uses music21 (Cuthbert & Ariza, 2010), a software library for processing musical scores. Music21 contains a comprehensive corpus of Bach chorales in MusicXML format. Our program reads all of the 371 chorales in the Riemenschneider edition and omits chorales that contain instrumental parts, bringing the total number of chorales down to 351. See Appendix A for the full list of chorales. The program splits each chorale into four tracks, one track per voice. Each of the tracks is then exported to a MIDI file and synthesized using FluidSynth, as described above.

3.3.1 Higher-Variability Dataset

Real-world recordings possess many sources of variability that are absent from the synthesized chorales described above. In order to test our separation techniques on a dataset that more closely resembles real-world choir recordings, we create an altered version of the dataset with three added features: simulated breaths, random omitted notes, and tempo variations.

Real chorale performances contain breaths between phrases. Rather than meticulously going through each chorale and choosing musically-pleasing locations to insert breaths, we take a simpler approach: we simulate one-beat-long breaths by inserting short silences in all voices simultaneously. We do this by simply changing the note that occurs on every eighth beat into a rest.

Another issue we wish to deal with in this dataset is that in the chorales, all four voices sing at all times. In other types of choral music, there are often sections where one or more voices are silent while the other voices continue to sing. In order to simulate this situation, we further alter the dataset by randomly choosing 10% of the notes in each voice and changing them into rests.

In the original dataset, all chorales are synthesized at a fixed tempo of 90 beats per minute (BPM). In this version of the dataset, we synthesize each chorale at a random tempo between 70 and 100 BPM, in steps of 5 BPM. Human performers add many slight tempo changes and local note timing variations. However, simulating human-like timing variations is out of scope for our synthesized dataset and is left for future work.

3.3.2 Dataset Partitions

Following standard machine learning practices, each of the two datasets is divided into three partitions: training, validation, and test (Goodfellow et al., 2016, p. 110). The training set is used for iterative gradient-based model optimization (see Section 2.2.2). The test set is only used after training, to estimate the trained model's generalization error. The validation set is used during model development to determine optimal values for *hyperparameters* that control the learning process itself, such as the learning rate (Goodfellow et al., 2016, p. 120).

In our datasets, we use about 75% of the examples for the training set, 15% for the validation set, and the remaining 10% for the test set. A full listing of dataset partitions is given in Appendix A.

Chapter 4

Score-Informed NMF for Choral Music

Having created a dataset of synthesized Bach chorale harmonizations (described in Section 3), we proceed to test a score-informed source separation technique on it to establish a baseline for separation performance. Out of the several methods described in Section 2.5, we choose to focus on the score-informed NMF technique proposed by Ewert & Müller (2012), which we henceforth call SI-NMF.

SI-NMF is an extension of classic NMF (described in Section 2.1.3). The premise of SI-NMF is that classic NMF is difficult to control: the optimization process may produce a factorization that approximates the mixture well but has little to no correlation with the actual sound sources. Using the pitch and timing information contained in the musical score, the factorization can be constrained in both the time and frequency axes simultaneously in order to yield a more meaningful separation.

In classic NMF, the basis signals and activations matrices are typically initialized with random values before starting the optimization process. Alternatively, they could be initialized using a technique based on singular value decomposition to improve the factorization in some cases (Boutsidis & Gallopoulos, 2008). SI-NMF uses specially-crafted initializations of the basis signals and activations to impose score constraints. It does this by relying on a simple fact: any element that is initialized to zero will remain zero in the final factorization. This is an inherent property of the multiplicative update rules used to perform the optimization (Lee & Seung, 2000).

The SI-NMF initialization scheme works in the following manner. First, the number of basis signals is determined by counting the number of unique pitches that occur in the score. One basis signal is then initialized for each pitch as an approximation of a harmonic series with a fundamental frequency that corresponds to that pitch. The harmonic series is created with a set number of partials p and some tolerance ϕ (expressed in semitones) around each partial to allow for slight pitch deviations and non-harmonicities. p and ϕ are parameters that can be adjusted according to the application. Since the amount of energy in higher partials is normally smaller than in the fundamental frequency, each partial's frequency region is initialized to $\frac{1}{n^2}$, where n is the partial index ($n = 1$ being the fundamental). All frequency bins not corresponding to any partial are initialized to zero. The basis signal initialization is illustrated in Figure 4.1a.

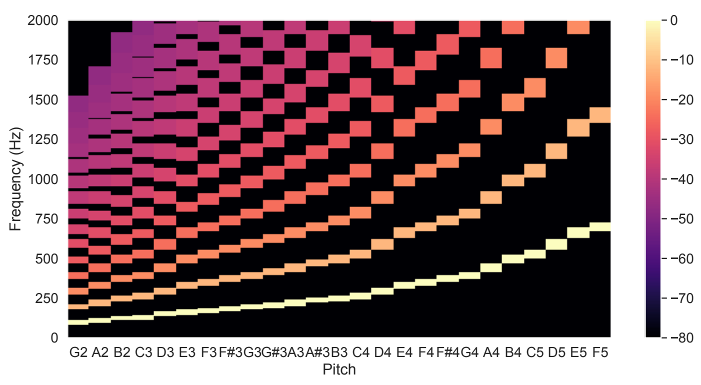
In order to make use of the timing information contained in the score, the activations matrix is initialized in a manner similar to a piano roll, as illustrated in Figure 4.1b. Recall that each column in the activations matrix initialization corresponds to one time frame. In each column, all elements corresponding to pitches that are active at that time frame are set to 1, while the rest are set to 0. In order to account for reverberations and slight misalignments of the score to the audio, the activations matrix is also initialized to 1 in frames preceding every note onset by up to t_{on} seconds, and following every note offset by up to t_{off} seconds. These tolerances are set according to the expected properties of the mixture and the score alignment.

As a result of this initialization scheme, the factorization process is forced to find basis signals $\boldsymbol{W}$ that are shaped approximately like harmonic combs with pitches corresponding to the notes in the score, and activations $\boldsymbol{H}$ that approximately match the expected note onsets and offsets. An example factorization is shown in Figures 4.1c and 4.1d.

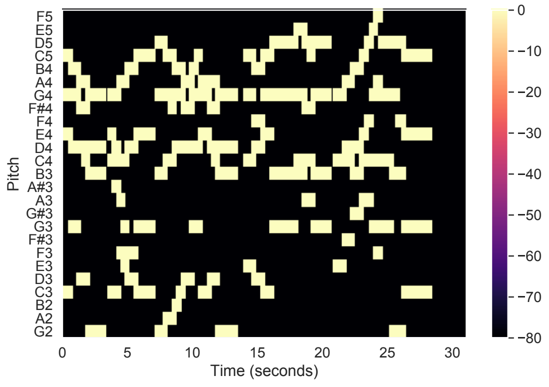
Following the spectrogram factorization, $\boldsymbol{H}$ is segregated according to the musical score into a set of per-source activation matrices $\boldsymbol{H}_i$, so that every $\boldsymbol{H}_i$ contains only the activations corresponding to notes that belong to source i. A ratio mask $\boldsymbol{M}_i$ is then computed for each source as follows:

$$\boldsymbol{M}_i = (\boldsymbol{W}\boldsymbol{H}_i) \oslash (\boldsymbol{W}\boldsymbol{H} + \epsilon),$$

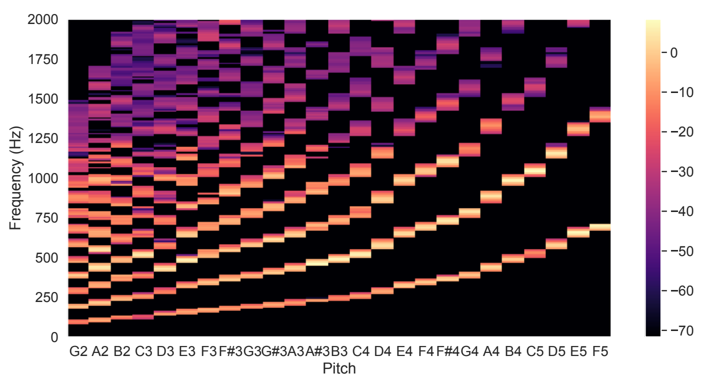
where $\oslash$ denotes element-wise division (Hadamard division) and ϵ is a small positive constant used to prevent division by zero. The mask for each source is then applied to the mixture



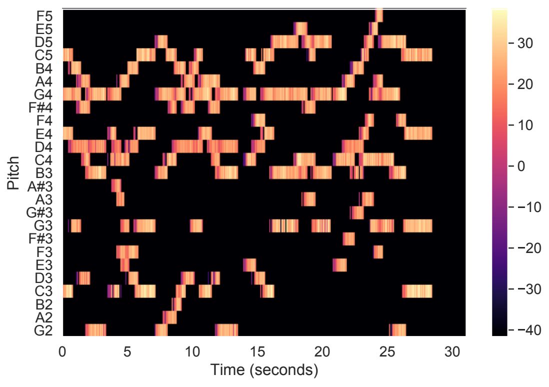
(a) Basis signal initializations



(b) Activation initializations



(c) Factorized basis signals



(d) Factorized activations

Figure 4.1 Initializations for score-informed NMF and the resulting factorization on an example Bach chorale. In this example, the parameter values are $p = 15$, $\phi = 0.5$, and $t_{\text{on}} = t_{\text{off}} = 0.2$. Colors indicate amplitude in dBFS.

spectrum $\boldsymbol{S}$ (which includes the mixture's phase) to retrieve the final source estimate, $\hat{\boldsymbol{S}}_i$:

$$\hat{\boldsymbol{S}}_i = \boldsymbol{M}_i \odot \boldsymbol{S},$$

where $\odot$ denotes element-wise multiplication (Hadamard product).

4.1 Experiments

We have implemented the SI-NMF technique described above using Python. Our code is made available as open-source software (see Appendix B). To perform the score-informed initialization, we first read the score corresponding to each reference source from a MIDI file. We then initialize the basis signals and activations using the scheme outlined above according to MIDI `note on` and `note off` messages from all sources combined. Next, we initialize per-source activation matrices $\boldsymbol{H}_i$ from each source's MIDI file separately. To perform the factorization, we use librosa (McFee et al., 2015) which in turn relies on scikit-learn's NMF implementation.[1]

We have run and evaluated our implementation on the synthesized chorales datasets described in Section 3. In order to better adapt the technique to our dataset, we tested various combinations of parameters in four experiments, listed in Table 4.1. These experiments and their results are presented below.

Table 4.1 Listing of NMF experiments

Experiment	Onset tolerance (seconds)	Offset tolerance (seconds)	Pitch tolerance (semitones)	STFT window size (samples)
A	0.2	1	1	2,048
B	0	0.2	1	2,048
C	0	0.2	0.4	2,048
D	0	0.2	0.4	4,096

[1]`https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html`

4.1.1 Experiment A: Original Parameter Values

In the first experiment, we use the same parameter values given in the original SI-NMF paper: $\phi = 1$, $t_{\text{on}} = 0.2$ and $t_{\text{off}} = 1$. The value of p is not mentioned in the original paper, so we assume the series of partials extends up to the Nyquist frequency. Also not indicated in the original paper are the parameters used for the STFT, so we assumed sensible defaults: a Hann window with length of 2,048 samples and a hop size of 512 samples. (The audio sample rate in our dataset is 22,050 Hz.)

A qualitative examination of this experiment's results reveals that separation was somewhat successful. The segregation of basis signals, as dictated by the score-informed initialization, was musically meaningful: every basis signal mostly corresponded to a single note. Furthermore, each estimated source mostly contained the right notes. However, the estimated sources exhibited several problems. First, they contained significant amplitude modulation in sustained notes, where such modulation did not exist in the reference sources. This was caused by the existence of vibrato in the reference sources: since SI-NMF only allocates one static spectral template per note, continuous changes in frequency cannot be accounted for.

Another problem that surfaced in the results is that notes sometimes appeared in the estimated sources when they should not have. This happened in places where the same pitch occurred in two different voices within a short time span. In those cases, there was no way to differentiate between the two occurrences because they appeared as overlapping in the activations matrix. This, in turn, was caused by overly permissive values of the activation tolerances t_{on} and t_{off}: any two notes less than $t_{\text{on}} + t_{\text{off}}$ seconds apart would "collide" in the activations matrix initialization and their association to the correct source would thus become ambiguous. We attempt to solve this problem in the next experiment by adjusting the activation tolerances.

4.1.2 Experiment B: Smaller Activation Tolerances

Since we fully control the synthesis method used to create our chorales dataset, we know that the score is in fact precisely aligned to the audio. As such, no tolerance is required for note onsets, and we set $t_{\text{on}} = 0$. For note offsets, however, some tolerance is still required because notes do not end abruptly after a `note off` MIDI message is received. Rather, notes decay over a time span determined by the SoundFont used for synthesis. Examining our SoundFont (specified in Section 3) we determine that the note decay time is 0.2 seconds. Hence, we set $t_{\text{on}} = 0.2$.

Listening to separation results with these modified parameters, we notice an improvement compared to Experiment A: notes do not overlap in the estimated sources anymore. We note, however, that this improvement relies on our intimate knowledge with the test dataset. Such fine tunings are less feasible when working with real-world recordings performed under unknown conditions, and when the score alignment is not known to be perfect.

In the results of this experiment we notice that each note still contains some interferences. This is especially pronounced in the bass estimates which contain many interferences in the higher part of the spectrum. This may be caused by the frequency tolerance ϕ: since ϕ is specified in semitones, its effect is larger on higher partials. For low fundamental frequencies and high values of ϕ, higher partials start to overlap each other and eventually become one contiguous strip instead of a harmonic comb. We attempt to solve this issue in the next experiment.

4.1.3 Experiment C: Smaller Frequency Tolerance

In order to reduce interferences for low notes in the higher part of the spectrum, we test reducing the value of the frequency tolerance ϕ, which was set to 1 in the previous experiments. We find that reducing it to 0.1 is too restrictive: extracted sources sound too "thin" and the distinctiveness of the vocal timbre is reduced. On the other hand, using a value of 0.8 allows too many interferences in the bass estimates. We find that $\phi = 0.4$ strikes the best balance for our dataset between preserving the original sources and preventing interferences.

As in the previous experiment, we note here that this reduction of the tolerance works well due to the nature of our synthesized dataset, in which pitches are always perfectly in tune. In real choir music there are often mistunings and significant pitch drifts, which would cause complete failure of SI-NMF with a low value of ϕ.

4.1.4 Experiment D: Larger STFT Window

Since NMF operates on the result of the STFT, any changes in the STFT parameters are bound to affect separation results. Examining the default STFT parameters used in the previous experiments, we notice that the size of one frequency bin is 10.77 Hz. And yet, one semitone in the bass range (the difference between A#2 and A2) is equivalent to 6.54

Hz. Hence, two bass fundamental frequencies could belong in the same frequency bin, and separating them using NMF would be more difficult.

We can increase the frequency resolution of the STFT by increasing its window size. We found the optimal window size for our dataset to be 4,096 samples. With this window size the bass estimates contained less interferences between notes, presumably because each basis signal was more accurately separated. Increasing the window size further to 8,192, we found that note onsets and offsets became “smeared” due to the decreased time resolution of the STFT.

4.1.5 Quantitative Comparison of Experiments

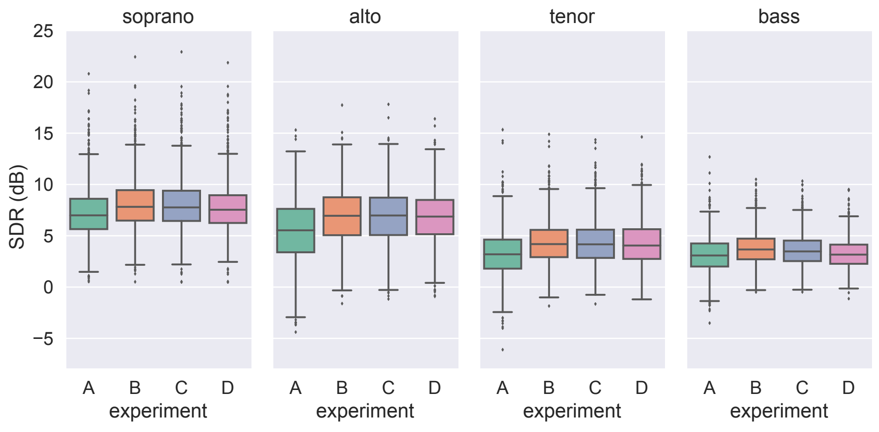
To quantitatively compare between the experiments described above, we use the evaluation metrics provided by the BSS Eval library version 4 (Stöter et al., 2018) which was discussed in Section 2.1.4. The library computes several evaluation metrics by splitting the signal into short frames and comparing the reference sources from the test dataset to the estimated sources generated by the separation technique. We use the default setting of 1-second non-overlapping evaluation frames for consistency with the SiSEC 2018 evaluation campaign (Stöter et al., 2018).

Out of the four metrics computed by BSS Eval, in presenting our evaluations we use mainly the source-to-distortion (SDR) metric which represents the total separation error. When comparing SDR across techniques, we prefer to use the median and interquartile range (visualized using box plots) rather than mean and standard deviation. This, again, is in line with the SiSEC 2018 evaluation campaign and is motivated by the tendency of the SDR metric to yield extreme outlier evaluations in specific situations. We discuss the limitations of SDR further in Section 6.3.8 below.

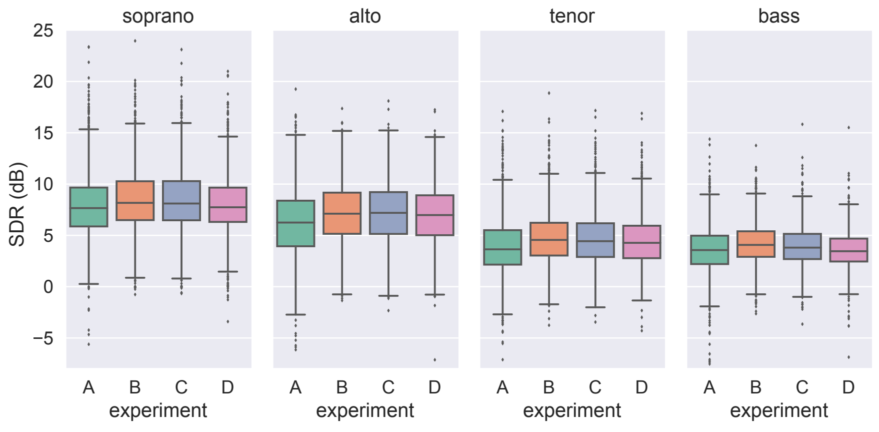
We evaluated Experiments A–D on the 31 synthesized Bach chorales belonging to the test partition in our two datasets: the normal dataset and the higher-variability dataset, described in Section 3. The evaluation results are given in Figure 4.2.

As can be seen in the figure, the best separation performance was achieved on the soprano voice, followed by the alto, tenor, and bass in that order. The mean SDR in Experiment A is lower than in the other experiments for all voices. Overall there seems to be little difference in SDR between Experiments B-D, unlike our qualitative observations may have suggested.

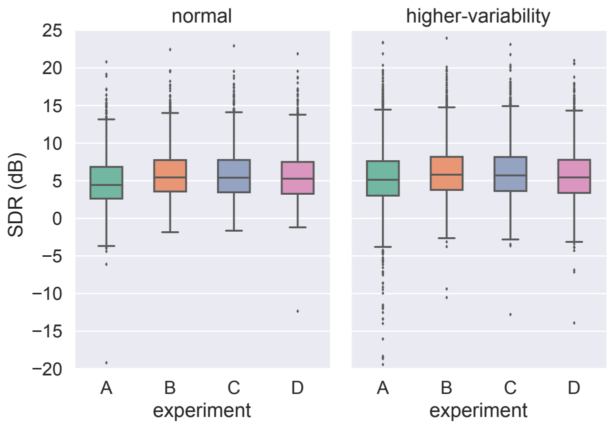
In all experiments, the evaluation on the higher-variability dataset produced many out-



(a) Results on the normal chorales dataset, by voice



(b) Results on the higher-variability chorales dataset, by voice



(c) Combined results for all voices, by dataset

Figure 4.2 SDR evaluation results of Experiments A–D

liers: evaluation frames in which the SDR was particularly low compared to the overall distribution. A closer examination of some of these frames reveals that they often occur when the evaluated source is silent while the others continue to sing. Since in the higher-variability dataset some notes are randomly omitted, this situation is more common in it compared to the normal dataset. When one voice was silent, SI-NMF sometimes mistakenly attributed to that voice partials that belong to other voices in the mixture. This caused a particularly low SDR due to reasons that will be discussed in Section 6.3.8 below.

4.1.6 Failed Experiments

Apart from Experiments A–D, we tried several other combinations of parameters that did not improve results. We nonetheless report those failed attempts here because we believe they shed light on the limitations of employing SI-NMF on our dataset.

In the original SI-NMF paper, the authors reported achieving an improvement in separation quality by using two basis signals per note: one signal for the onset and one for the sustained part. This was motivated by the difference in spectral characteristics: piano note onsets are transient-like with energy spread across a broad frequency range, while the sustained part of each note is (nearly) harmonic. In our synthesized dataset there are no distinct note onsets, but we did try using two basis signals per pitch by simply initializing both with the same spectral and temporal constraints. We hypothesized that this might improve the handling of the smooth temporal evolution of each note. However, it did not work as expected: the two basis signals belonging to the same pitch were always factorized identically (presumably due to the way the multiplicative updates work), so the second basis signal did not supply any added value.

We also tried initializing only the activations while leaving the basis signals randomly initialized, so that NMF is left solely with the temporal constraints. This degraded separation performance significantly compared to explicit spectral constraints, matching the findings in the original SI-NMF paper, so we did not pursue it further. The degradation occurred mainly because certain note combinations (such as thirds and octaves) often occur simultaneously in our mixtures, so factorized basis signals often contained a mix of two notes.

Since the number of partials p in the spectral initializations was unspecified in the original paper, and high partials overlapped across basis signal initializations due to the tolerance ϕ, we attempted to limit the number of partials. We found, however, that any limit on p

degrades separation quality, probably because the amount of energy in the high frequencies, despite being sometimes very low, never actually reaches zero in any source.

4.2 Conclusions

All in all, SI-NMF achieved decent separation results on our synthesized dataset of Bach chorales. Interferences between estimated sources were very low or nonexistent due to the hard constraints imposed using the score. However, these hard constraints rely on strong assumptions that are valid on our synthesized dataset but less so on real choir recordings. First, the score must be perfectly aligned to the audio: recall that separation quality was significantly reduced when we attempted to use a higher temporal tolerance. Second, the basis signal initializations can only capture harmonic components, while real choir music normally has lyrics that contain many consonants that are inharmonic.

Most importantly, SI-NMF only uses static spectral templates. This works well for piano recordings but is bound to be less effective for vocal music because the latter is characterized by continuous evolution of spectral parameters. Pitches change continuously due to vibrato, note transitions, and drift. Spectral envelopes also evolve continuously due to changes in formant frequencies when transitioning between vowels. Hence, sources estimated in our experiments contained noticeable amplitude modulation artifacts caused by SI-NMF's inability to model vibrato and other variation factors in the synthesized chorales.

The limitations of SI-NMF could be alleviated by employing one of the more powerful signal models we discussed when reviewing score-informed separation techniques in Section 2.5.2. Specifically, integrating source-filter signal models into NMF proved rather successful in separating the singing voice and instruments with vibrato (Durrieu et al., 2009; Heittola et al., 2009; Nakamura & Kameoka, 2016; Smaragdis et al., 2014). It is unclear, however, whether source-filter models would be effective for choral music separation. As we have shown in Section 2.4.4, the spectral characteristics of choral music are highly varied because in fact, every choir singer is an individual sound source with a distinctive timbre, pitch, amplitude, and timing.

These conclusions left us unsure whether NMF is the right tool for choral music separation. In the next chapters, we explore a completely different approach that is based on deep learning. After conducting several experiments with the deep learning-based technique, we finally (in Section 5.5.6) compare the new results to those obtained with SI-NMF.

Chapter 5

Wave-U-Net for Choral Music

Following our experiments with score-informed NMF, we concluded in the previous chapter that classic NMF, even with added score constraints, is not sufficient for the task of choral music source separation. When considering in which direction to proceed, we examined the results of SiSEC 2018, an evaluation campaign for source separation techniques (Stöter et al., 2018). The results of that campaign showed that for the task of singing voice separation and multi-instrument separation, the best performing techniques were those based on deep learning. One of the leading submissions in the campaign was Wave-U-Net, described in Section 2.3.3. Since the source code of Wave-U-Net was made publicly available by its authors[1], we decided to use it to bootstrap our experimentation process without having to reinvent the wheel.

This chapter details our experiments with Wave-U-Net on source separation of choral music. In these experiments, we do not modify the Wave-U-Net architecture in any way (chapter 6 will discuss our modifications). We simply train Wave-U-Net on our own dataset of synthesized Bach chorales presented in chapter 3 and measure its performance.

5.1 Training Procedure

We use a training procedure that is identical to the procedure used for the original Wave-U-Net. We describe the procedure here for completeness, as some details were omitted from the original Wave-U-Net paper and documentation.

[1]Wave-U-Net code: `https://github.com/f90/Wave-U-Net`

For training, each chorale in the training partition of the dataset is divided into segments with length matching the input size of the neural network. Segments are extracted at random positions, and they may overlap. All segments from all chorales are randomly shuffled and collected into batches of 16 segments each. Training is performed on these batches using the Adam optimizer (Kingma & Ba, 2014) with an initial learning rate of 0.0001. The loss function is the mean squared error (see Section 2.2.2) between the predicted source and the reference source, averaged over all sources and all segments in each batch.

The validation partition of the dataset is used to adjust the learning rate and decide when training should be stopped. In this context, an epoch is defined as 2,000 training iterations. After each epoch, the loss is evaluated on the entire validation set. When there is no improvement in the minimum validation set loss for 20 consecutive epochs, a "fine-tuning" stage is entered: the learning rate is reduced to 0.00001 and the batch size is doubled to 32. When, again, there is no improvement in the validation set loss for 20 consecutive epochs, training is stopped. When training is finished, model performance is evaluated on the entire test partition of the dataset.

5.2 Training Infrastructure

The deep learning software library used to implement Wave-U-Net, TensorFlow (Abadi et al., 2016), allows training neural networks on CPUs (central processing units) and certain types of GPUs (graphical processing units). Using a GPU speeds up training significantly because GPUs are optimized for parallel execution of linear algebra computations of the kind used in neural networks. We used GPUs provided by Compute Canada[2] to run our training tasks. Compute Canada is an organization that provides high performance computing infrastructure for Canadian researchers. We ran a simple benchmark to compare the training speed on our private laptop's CPU (Intel Core i5 2.7 GHz) and on a GPU provided by Compute Canada (NVIDIA P100) and found that on the GPU training speed increased by a factor of about 30. To illustrate, this means that a one-day training task on Compute Canada would take a whole month on our private laptop. For this reason, the availability of GPUs on Compute Canada significantly increased our ability to experiment with different model configurations.

Compute Canada uses a scheduling mechanism to execute jobs. Every job requests a

[2]`https://www.computecanada.ca`

certain running time and a set of resources, and is assigned to available matching nodes on large compute clusters. We ran our training jobs on the Graham cluster[3]. For each job we requested an allocation of a single GPU, 6 CPU cores, and 32 gigabytes of random access memory. The CPUs are used to load and prepare training data while the GPUs perform the gradient-based optimization. The configuration for every training task was saved in a JSON file (JavaScript Object Notation, a structured text format) for reproducibility.

5.3 Reproducing Results on Singing Voice

In order to verify that our training environment is set up correctly, we first set out to reproduce the original Wave-U-Net training procedure and check that we can obtain results comparable to the pre-trained models shared by the authors. We acquired the original datasets for training: CCMixter (Liutkus et al., 2015) and MUSDB (Rafii et al., 2017).

We trained two of the published model variants for the singing voice separation task. The variants are dubbed M1 and M5-HighSR. M1 is the baseline separation model. M5-HighSR is the best-performing model variant according to the authors (it was added to the code after the paper was published). M5-HighSR includes the following improvements over M1: it enforces source additivity rather than predicting each source individually; it uses temporal input context; it uses stereo input and output rather than mono; it uses learned upsampling parameters rather than naive linear interpolation; and finally, it uses a sample rate of 44.1 kHz rather than the baseline of 22.05 kHz.

After completing training we obtained separation results for several songs from the test dataset (that is, songs that were not used during training). Our trained models obtained results with comparable quality to the pre-trained models. As the Wave-U-Net authors reported, results from M1 contained audible artifacts at segment boundaries (approximately every 750 milliseconds). M5-HighSR, on the other hand, was able to produce separation results that were almost free of artifacts.

5.4 Experiments on Synthesized Bach Chorales

After having successfully reproduced Wave-U-Net results on the original singing voice separation dataset, we commence with experiments on the synthesized Bach chorales dataset

[3]`https://docs.computecanada.ca/wiki/Graham`

presented in Section 3. We describe the methodology for Experiments 1–4, followed by their results.

5.4.1 Experiment 1: Bass and Soprano Mixtures

We first train the model to separate the soprano and bass voices from a mixture of the two. That is, the chorale mixtures in this experiment do not contain all four voices; they contain only the soprano and bass, the so-called *outer voices*. We choose to conduct our first experiment on bass and soprano only because we hypothesized that they would be the easiest for the model to separate. Several factors support this hypothesis. First, the normal pitch ranges for these voices have almost no overlap: soprano range is normally C4 to A5, while bass range is E2 to D4 (see Table 2.1). Furthermore, the voices are different in their melodic content: soprano carries the chorale melody and thus tends to contain stepwise motion, while the bass has a higher frequency of leaps due to its harmonic function. In comparison, the two inner voices (tenor and alto) have overlapping ranges and similar melodic characteristics, possibly making them harder for the model to distinguish without additional information.

5.4.2 Experiment 2: Extract SATB

We proceed by training the model to separate all four voices from a mixture containing all four voices. We anticipate that in this experiment, results for the soprano and bass voices would be worse than Experiment 1 because of the confounding factors introduced by having two additional voices in the mixture. Furthermore, we expect that results for alto and tenor would be worse than results for soprano and bass in this experiment due to the overlap in pitch ranges and similarity in melodic characteristics between alto and tenor.

5.4.3 Experiment 3: Extract Single Voice

In this experiment, we train the model to extract a single voice from a mixture of all four voices. We hypothesize that this experiment would generate an improvement over Experiment 2 because in this experiment, the model training could optimize an almost equal number of parameters for a task that is simpler. The learned feature map could be geared specifically for the extracted voice, rather than having to be generic enough to enable extraction of all voices.

The disadvantage of this approach is that we would have to train four different models in order to extract all four voices. These four models would together use four times the number of parameters compared to the single model from Experiment 2 that was trained to extract all four voices. This is only a real disadvantage in an execution environment with limited memory or compute resources or where prediction time must remain short. Such environments are not part of our research goals.

5.4.4 Experiment 4: Higher-Variability Dataset, Extract Single Voice

In this experiment we train and evaluate models on the higher-variability dataset described in Section 3.3.1 in order to test Wave-U-Net on mixtures that more closely resemble real-world choir music. Specifically, we saw that the models in Experiments 1–3 were unable to cope well with silences in the mixture (this will be explained in Section 5.5.2 below). We anticipate that in this experiment, the model's performance in such cases would improve due to the inclusion of silences in the training dataset. Still, we reason that results in this experiment would be slightly worse than previous experiments due to the added variability in the dataset. In particular, we assume that in some cases the model would confuse between sources. For example, in a case where a soprano note is omitted but the alto sings a note that falls in the soprano range, the model might confuse the alto note as belonging to the soprano.

Like in Experiment 3, in this experiment we train four models. Every model is trained to extract a single voice from the SATB mixture.

5.5 Results

In this section we present quantitative results from Experiments 1–4. A listing of these experiments is given in Table 5.1. As in the evaluation of score-informed NMF (see Section 4.1.5), we use the SDR metric provided by the BSS Eval library with its default settings.

5.5.1 Experiment 1: Bass and Soprano Mixtures

Figure 5.1 shows the results of Experiment 1. Recall that the model in this experiment was trained on separating the soprano and bass only. The figure shows that performance on the soprano was significantly better than on the bass. Note that these results are significantly

Table 5.1 Listing of Wave-U-Net experiments. HV stands for higher-variability.

Experiment	Dataset	Mixture Voices	Extracted Voices	Model Type
1	normal	SB	SB	one model for all voices
2	normal	SATB	SATB	one model for all voices
3	normal	SATB	SATB	one model per voice
4	HV	SATB	SATB	one model for all voices

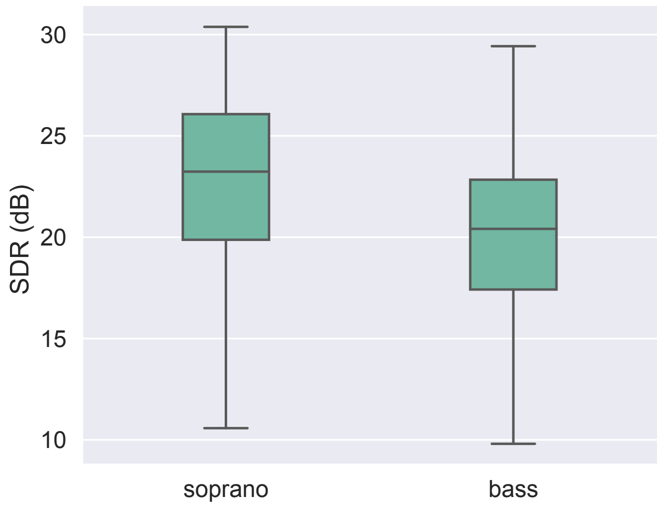


Figure 5.1 SDR evaluations of Experiment 1 results by voice

better than those achieved by techniques evaluated in SiSEC 2018 on singing voice separation. Leading submissions in SiSEC 2018 achieved median SDR of 5–10 dB for vocals and 10–15 dB for accompaniment (Stöter et al., 2018). This difference is probably due to the fact that here the task is simpler since we use synthesized mixtures as opposed to real-world recordings.

5.5.2 Experiment 2: Extract SATB

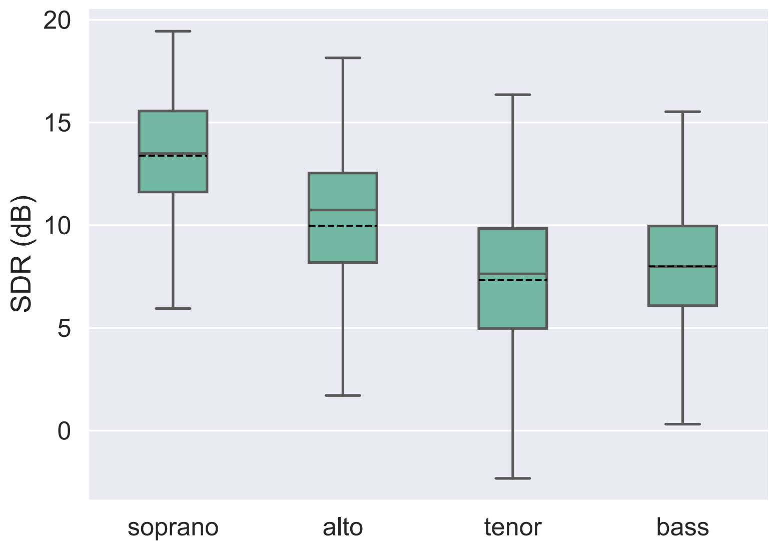


Figure 5.2 SDR evaluations of Experiment 2 results by voice. The dashed lines indicate the mean.

The results of this experiment, illustrated in Figure 5.2, show that median performance was best on the soprano voice, followed by the alto, bass, and tenor in that order. Importantly, the figure shows that the interquartile range for the inner voices (alto and tenor) is larger than that of the outer voices (soprano and bass). Also, the mean for the inner voices is noticeably lower than the median due to outliers. This indicates that separation performance on the inner voices was less consistent.

In order to investigate this inconsistency, we examine the ten frames with the lowest SDR in this experiment (see Table 5.2). Negative SDR generally indicates bad separation quality. In the two frames with the lowest SDR, it seems the failure was caused by a silent section in the alto part in chorale 358. Figure 5.4 shows the score for this section: note the silences in alto and tenor. Silences are very unusual for Bach chorale harmonizations and so the training dataset contained very little of them. For this reason, the model did not learn to predict silences well: its predictions on silent segments contained a large amount of interference from other voices. Note also that the SDR for these two near-silent frames is extremely low (below -40 dB). This extremity is caused by the way the SDR metric is designed. We discuss this further in Section 6.3.8 below.

The bad performance on all the other 8 frames in Table 5.2 was caused by voice crossings between the alto and the tenor: segments in which the alto goes lower than the tenor (see Section 2.4.2 for an explanation). For example, the third frame in the table (chorale 358 frame 46) corresponds to the voice crossing shown in Figure 5.4. Voice crossings pose a challenge for the model because it has evidently learned to rely on the standard SATB ordering of the voices. In Section 6 below, we explore using the musical score to improve separation in such challenging cases.

5.5.3 Comparison of Experiments 1-2

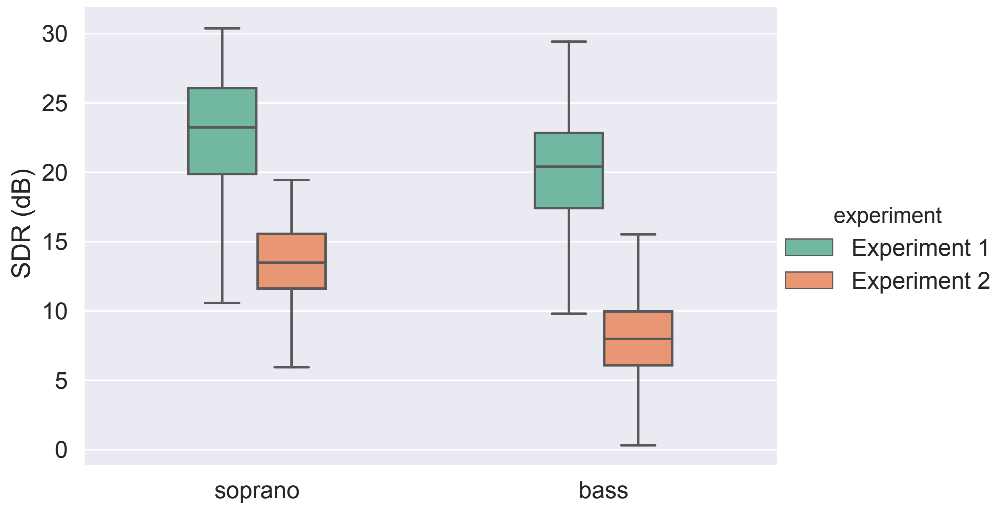


Figure 5.3 SDR evaluation of Experiment 1 results compared to Experiment 2 results by voice. Only soprano and bass are compared because they are the sources common to both models.

Figure 5.3 shows that when extracting from an SATB mixture (as opposed to an SB mixture as in Experiment 1), the performance on the soprano and bass voices degrades

Table 5.2: The ten evaluation frames with the lowest SDR in Experiment 2. The last column shows the reason for the bad performance, as explained in the text.

Source	Chorale	Frame	SDR	Reason
alto	358	39	-45.28	Silent
alto	358	38	-39.70	Silent
alto	358	46	-5.33	Crossing
alto	341	19	-5.25	Crossing
tenor	367	3	-5.14	Crossing
tenor	367	4	-4.68	Crossing
alto	366	20	-4.04	Crossing
tenor	358	45	-3.64	Crossing
tenor	358	48	-3.22	Crossing
tenor	358	44	-3.19	Crossing

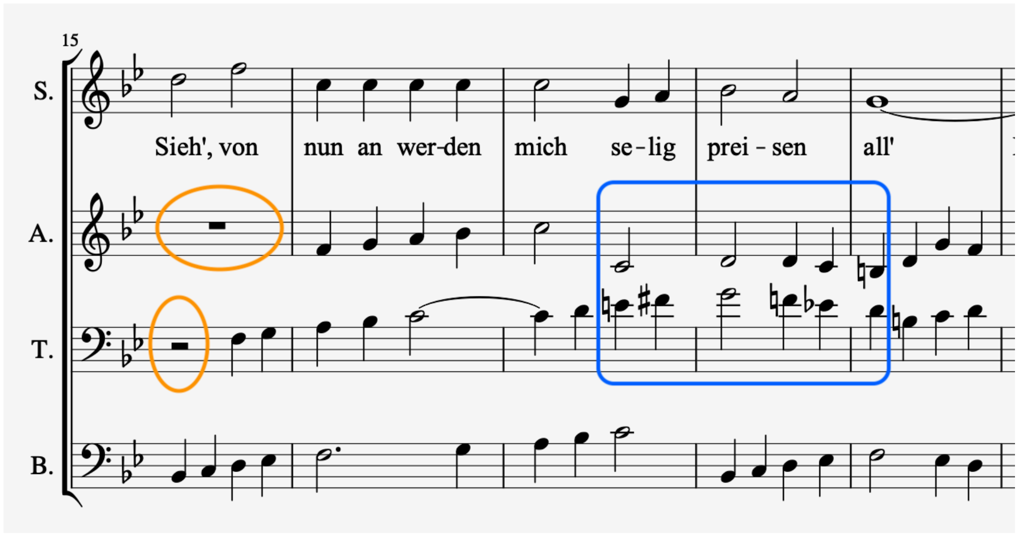


Figure 5.4 The final phrase of chorale 358. Unusually for Bach chorale harmonizations, the alto and tenor parts contain silences in measure 15 (marked in orange). Furthermore, there is a voice crossing between the alto and tenor parts that lasts for 7 beats (marked in blue).

significantly. This degradation is to be expected, given that the task in Experiment 2 is more difficult yet the number of model parameters remained the same.

5.5.4 Experiment 3: Extract Single Voice

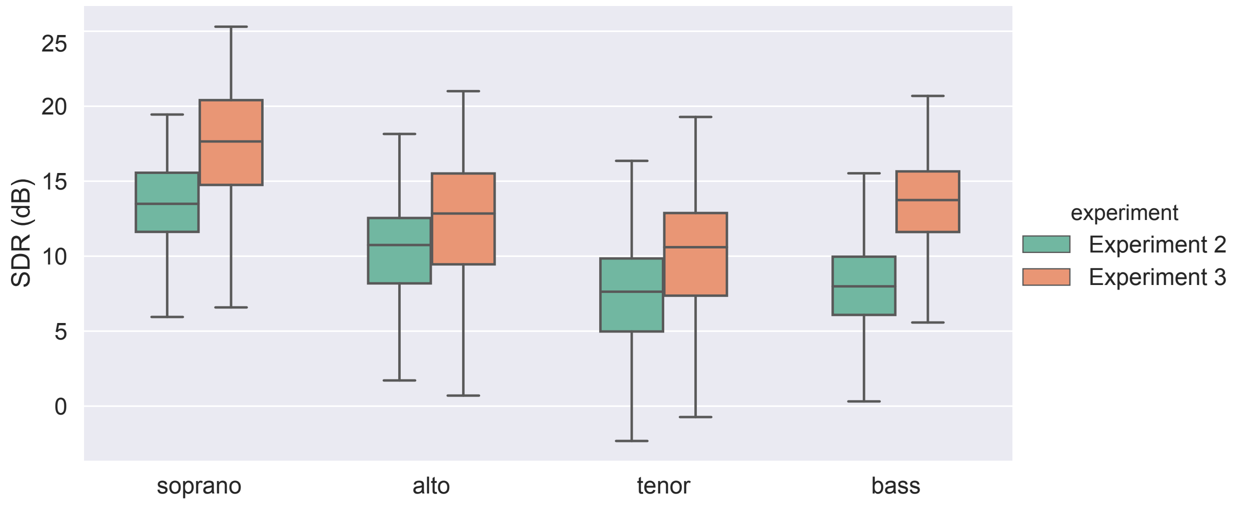


Figure 5.5 SDR evaluations for Experiment 3 by voice, with Experiment 2 results shown for reference.

Results for Experiment 3 are shown in Figure 5.5. Separation in this experiment is significantly better than Experiment 2 on all voices. This confirms our hypothesis that optimizing a model for extracting a single voice yields better performance than extracting all voices at once. The results show that models perform significantly better on the outer voices (soprano and bass) than on the inner voices.

5.5.5 Experiment 4: Higher-Variability Dataset, Extract Single Voice

Results for Experiment 4 are shown in Figure 5.6. Like in Experiment 3, performance on the outer voices is superior to performance on the inner voices. The comparison between Experiment 4 and Experiment 3 shows that as we hypothesized, using a dataset with higher variability leads to inferior separation results. This can be seen as an indication that training on real-world recordings (rather than a synthesized dataset) would lead to an even larger degradation, since real-world music has many additional variation factors that are not included in this dataset, such as lyrics, choir timbre, choir size, use of divisi, room acoustics,

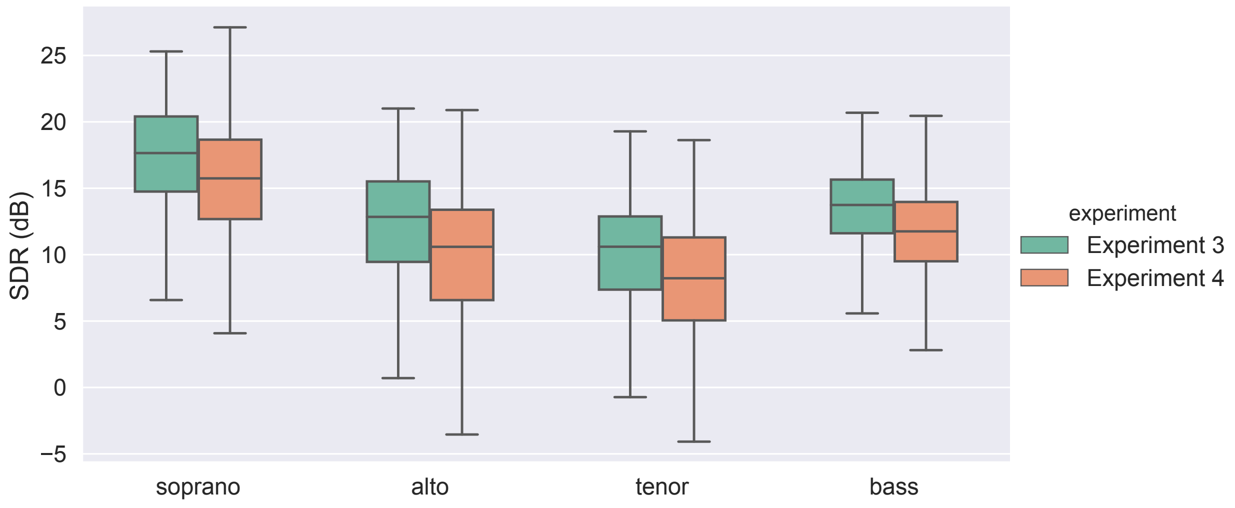


Figure 5.6 SDR evaluations for Experiment 4 by voice, with Experiment 3 results shown for reference.

musical textures, and recording quality.

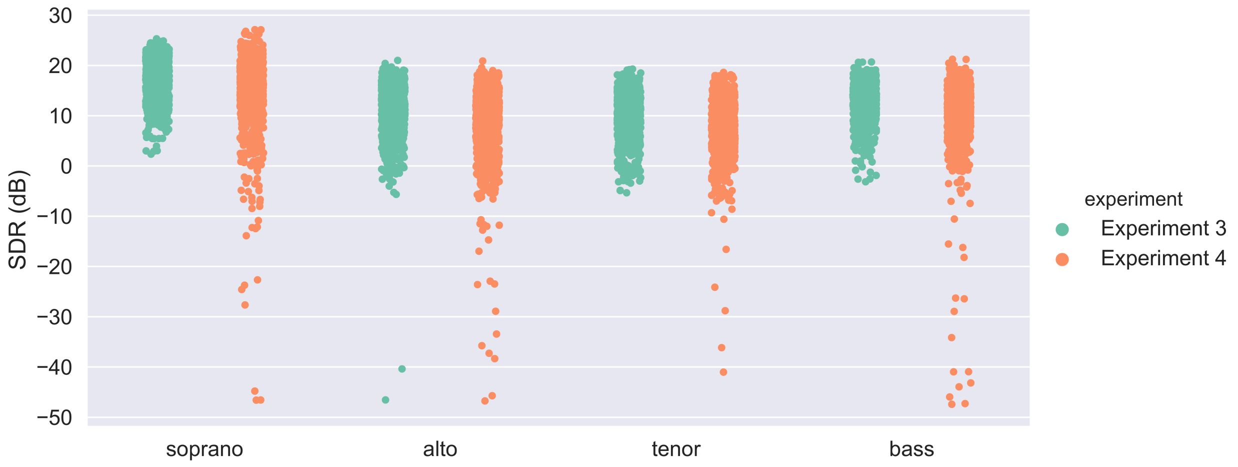


Figure 5.7 Full distribution of SDR evaluations in Experiments 3–4 by voice.

Figure 5.7 shows that results in this experiment contain a large number of outlier frames with low SDR compared to Experiment 3. Investigation of these outliers reveals that many of them are caused by segments in which one voice is silent while the other voices continue to sing. In such segments it was difficult for the model to 'guess' which voice is silent due to the overlap between voice ranges, so the model sometimes conflated one voice with another.

The score for one such example is shown in Figure 5.8.

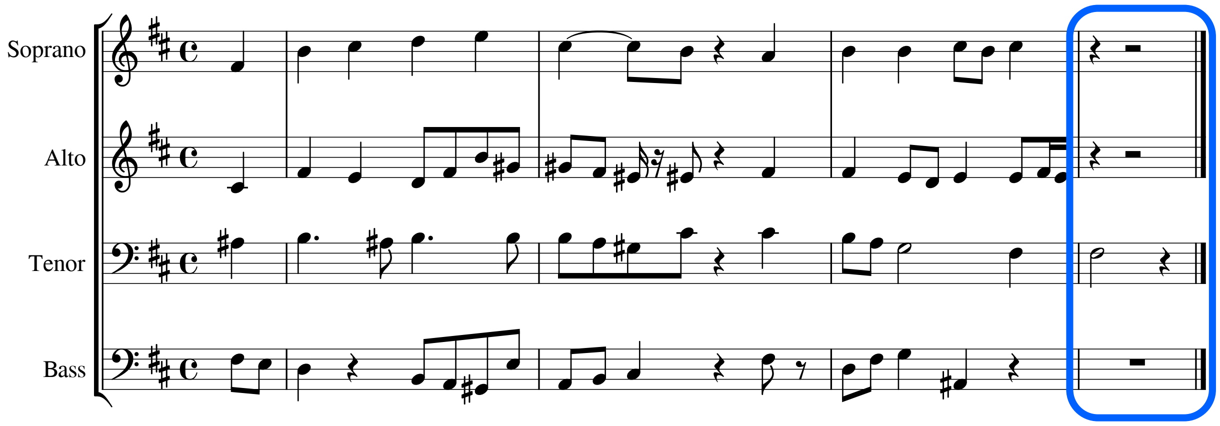


Figure 5.8 The score corresponding to an evaluation frame that achieved a very low SDR in Experiment 4 (chorale 364, frame 10). Some notes from the original Bach harmonization are omitted in the score because it is part of the higher-variability dataset. The SDR for the frame corresponding to the score segment marked in blue was low because the model mistook the long tenor note for a bass note. This is an 'understandable' mistake because this note (F#3) lies close to the middle of bass range.

5.5.6 Comparison between Experiments 1–4 and NMF

For an overall comparison, Figure 5.9 shows the results of Experiments 1–4 side by side, along with the best results achieved with NMF as described in Section 4. As expected, Experiment 1 had the best results for soprano and bass, due to the fact that the mixtures in that experiment consisted only of these two voices. All Wave-U-Net models outperformed the best results achieved with NMF by a large margin. Interestingly, Experiment 4 models achieved a higher median SDR compared to Experiment 2, even though Experiment 4 was evaluated on the higher-variability dataset. This improvement was likely caused by the use of one model per voice as opposed to one model for all voices. Still, Experiment 4 shows a high number of outliers mostly due to misclassified notes, as explained earlier. In the next section, we explore integrating the musical score into the separation process in order to reduce errors that are caused by misclassified notes.

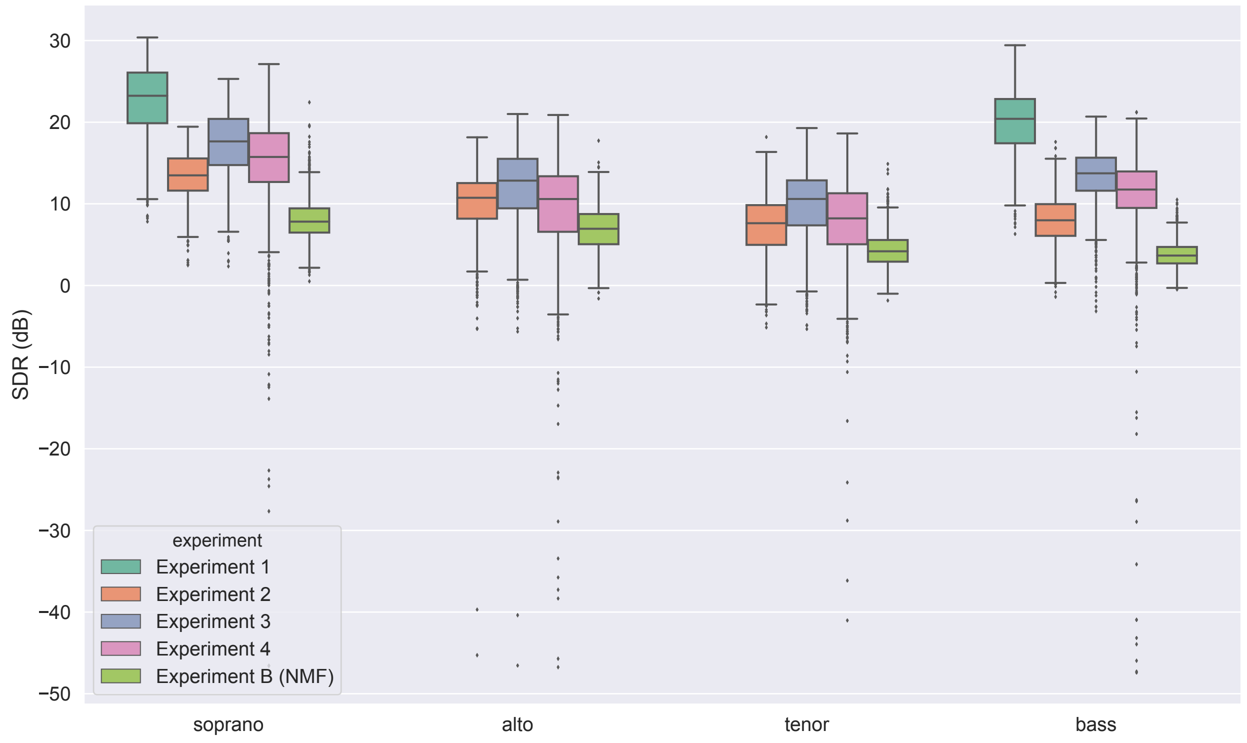


Figure 5.9 SDR results for Experiments 1–4 by voice, with outlier frames shown as points. Results from Experiment B (the score-informed NMF experiment that achieved the highest median SDR) are shown for comparison.

Chapter 6

Score-Informed Wave-U-Net

One of the goals in choral music performance is achieving a blend of the different voices (see Section 2.4). For this reason, in some cases even expert human listeners cannot identify individual parts in a choral recording. In these difficult cases where parts are well blended, the score (see Section 2.4.3) may prove useful to human listeners in identifying individual parts, as it provides information on what pitches to expect and at what times. If the score helps human listeners to identify parts, it might similarly assist a machine learning separation model. We therefore investigate in this section the integration of score information into the separation process.

Timbre is generally a useful differentiating factor that could be used for source separation. In choral music, however, the timbres of the women's voices (soprano and alto) are similar to each other, and so are the men's (tenor and bass). If not timbre, separation of choral music could be guided by the pitch ranges of each choir part, but unfortunately the ranges have considerable overlap. For example, an F4 note can easily belong to the soprano, alto, or tenor; in rare cases it could even be sung by the bass (see Section 2.4.2). Another option would be to rely on the ordering of the voices: normally, when all four voices sing at the same time, the lowest note would belong to the bass and the highest to the soprano, with tenor and alto in-between. However, there are many cases in which not all voices sing at the same time, and even when they do, they sometimes cross over each other (see Section 2.4.2). Hence, we conclude that in pitch ranges that are shared between two or more choir parts, the musical score might be the only way to associate notes to a specific voice.

We publish our score-informed version of Wave-U-Net, including all the code related to the experiments described in this section, as open-source software (see Appendix B).

6.1 Conditioning Wave-U-Net on Scores

Having decided that we wish to integrate information from the score into the separation process, the question arises how to feed that information into Wave-U-Net. There are three main questions to be answered: what information do we extract from the score, how do we represent that information, and in what way do we feed it into the neural network.

6.1.1 Choosing Information to Extract from the Score

As discussed in Section 2.5, many types of information can be extracted from a score. For our model, we choose to extract only timing and pitch information from the score. We reason that this information would be the most crucial to guide separation of each note to the voice it originated from. In choral music, every choir part sings at most one note at a time (as opposed to some instruments that can play several notes at the same time). There are cases of *divisi* in which one part is divided into several parts (for example, soprano 1 and soprano 2); we simply choose to treat these *divisi* as distinct parts, so that our initial statement always holds: for every point in time a part can have at most one active note. Consequently, in order to represent a four-part choral piece we simply need four sequences, where each sequence is comprised of notes each having a pitch and a duration, and of rests that have only duration. We discard any other information from the score.

As discussed in Section 2.5.1, score alignment is normally a prerequisite for score-informed separation. Fortunately, for our experiments on synthesized chorales, we already have a perfectly aligned score for every recording, as our recordings are deterministically generated directly from the score.

6.1.2 Feeding the Score into Wave-U-Net

Wave-U-Net is designed to handle audio inputs only. Here we desire to feed an additional input of a different modality: a musical score, and we wish that the model incorporates information from both input modalities when generating the output. In machine learning, this general idea is referred to as *conditioning*: predicting outputs for a certain input using information extracted from an auxiliary input (Perez et al., 2018).

Finding methods for conditioning neural networks is an open research problem. Dumoulin et al. (2018) describe three methods: *concatenation-based conditioning*, in which the conditioning vector is concatenated to the input vector; *conditional biasing*, in which the conditioning vector is added to the input vector; and *conditional scaling*, in which the input vector is multiplied by the conditioning vector. These methods are illustrated in Figure 6.1. The authors point out that all three conditioning methods may seem natural and intuitive. For our experiments, we used the straightforward approach of concatenation; we leave it to future work to explore other conditioning methods. Interestingly, Dumoulin et al. (2018) point out that concatenation-based conditioning is equivalent to conditional biasing with a linear transformation applied to the conditioning vector.

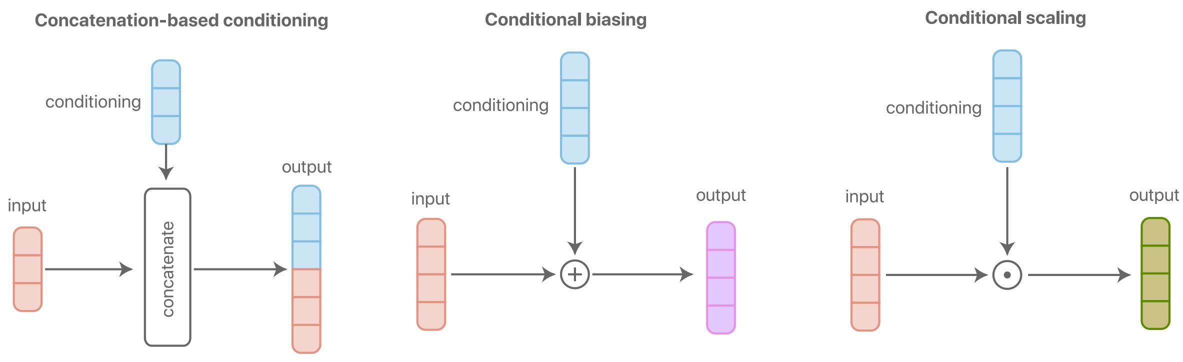


Figure 6.1 Conditioning methods for neural networks. Adapted from Dumoulin et al. (2018), licensed under CC-BY 4.0.[2]

Once we have decided to concatenate the score to the audio, we have to decide where in the network to concatenate it: we could concatenate it directly to the audio input that is sent to the first layer of the encoder; or we could concatenate it to the low-resolution feature map at the bottleneck between the encoder and the decoder; or we could concatenate it after the last layer of the decoder just before generating the output audio. It is also possible, of course, to condition the network in multiple locations at the same time. See Figure 6.2 for an illustration of conditioning locations.

In Section 6.2 we investigate concatenative score conditioning in three locations: input conditioning, output conditioning, and input-output conditioning. Input-output conditioning simply means feeding the score at both the input and output locations. We did not

[2]`https://creativecommons.org/licenses/by/4.0/`

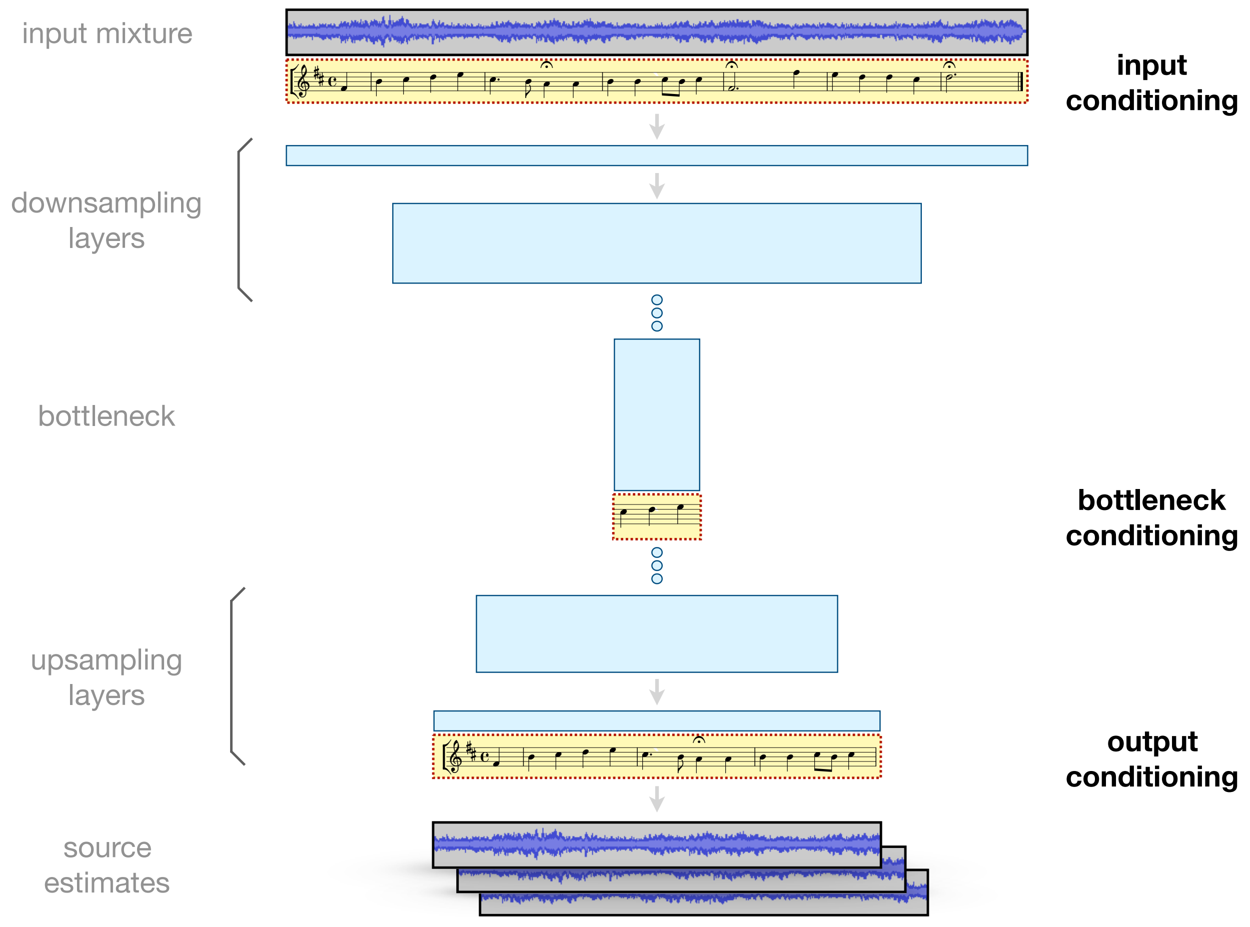


Figure 6.2 Score conditioning locations.

investigate conditioning at the bottleneck. Concatenative conditioning at the bottleneck would necessitate resampling the score information to the bottleneck's much lower temporal resolution; that way, we would lose important timing information. When we condition at the input or output of the network, we can use the same sampling rate for the audio and the score.

Different conditioning methods of the same network architecture could apply to different tasks; compare, for example, Slizovskaia et al. (2019), who used multiplicative conditioning in the Wave-U-Net bottleneck to condition the separation on instrument labels. In that work, the conditioning vector has no temporal dimension: it applies to the whole audio segment; it thus makes sense to apply it at the bottleneck where temporal resolution is low and the amount of channels in the feature map space is high.

Intuitively, when we concatenate the score to the audio input, we treat the score simply as an additional channel of input to the neural network. Recall that the input of the original non-score-informed Wave-U-Net is an audio segment: a matrix of size $c \times n$, where n is the number of time instants (16,384 in the basic model variant) and c is the number of audio channels (1 for the basic mono model variant and 2 for stereo). Since the score is a temporally-evolving structure just like the audio, and they are temporally aligned to each other, we simply concatenate the score to the audio input along the rows axis. Consequently, a column in the input matrix continues to represent a single time instant. Note that this means that we must use the same sampling rate for score representation as for the audio input.

6.1.3 Score Representations

Before feeding the score information into the neural network, we must decide how to represent it. The choice of score representation might significantly affect the performance of the neural network. There are no conceptually right or wrong representations; the best representation is the one that maximizes the model's performance. Note that following the discussion in the previous section, we are interested in a representation for the music of a single part only (as opposed to a score of all parts combined).

The first score representation we propose is *MIDI pitch*. In this representation, a score is a vector of n integers (n is the input segment length). Values from 1 to 127 indicate that a note with the corresponding MIDI note number is active (the MIDI note number is a standardized way to specify a note's pitch). The value 0 has a special meaning in this representation: it is reserved to indicate that there is no active pitch at that time.

A possible problem with the *MIDI pitch* representation is that the range of the score data (0 to 127) is radically different from the range of audio data (-1 to 1). This difference might cause the gradient-based training process to work inefficiently. In fact, it is generally a recommended practice to normalize any neural network inputs by shifting and scaling them so that their average is zero and they all have the same covariance (LeCun et al., 1998). We thus propose a second representation, *normalized pitch*, in which we normalize the score to be in the same range as the audio data.

Given a MIDI note number M, the normalized pitch is computed as:

$$M_{\text{normalized}} = \begin{cases} -1, & M = 0 \\ \frac{M - M_{\min}}{M_{\max} - M_{\min}}, & \text{otherwise} \end{cases},$$

where $M_{\min}$ and $M_{\max}$ are constants expressing the minimum and maximum expected note pitches, respectively. We set $M_{\min} = 36$ and $M_{\max} = 84$, based on the knowledge that pitches in choral music normally range from C2 (very low bass note) to C6 (very high soprano note); in Bach chorales there are no notes outside of this range. Note that in this representation values are real numbers rather than integers.

$M_{\text{normalized}}$ receives the value -1 when no note is active, and a value from 0 to 1 when a note is active. The rationale in leaving values between -1 and 0 unused is that there should be a clear differentiation between the case where a note is active and the case where no note is active. This differentiation is admittedly somewhat artificial, but given that we are limiting the representation to a single value per time point, it seemed like the most logical choice.

The problem of representing silences in the previous two representations leads to the following representation. Rather than representing pitch only, we introduce a two-channel representation: *pitch and amplitude*. In this representation, the score is a matrix of size $2 \times n$, in which the first row represents the pitch and the second one represents the amplitude. The full range of pitches is normalized to the range $[-1, 1]$:

$$P = -1 + 2 \cdot \frac{M - M_{\min}}{M_{\max} - M_{\min}}.$$

The second row, representing the amplitude, is actually boolean: its value is 1 when a note is active and 0 when no note is active. Unfortunately, this does not completely solve the issue with representing silences: the pitch channel must still be assigned a value when no note is active, a value which has no musical meaning; we chose this value arbitrarily to be -1.

As an attempt to resolve issues in the above representations, we propose a *piano roll* representation. Here, the score is represented as a matrix of size $p \times n$ where p is the number of available pitches ($p = M_{\max} - M_{\min} + 1$). This representation is similar to a piano roll or to the interface in which MIDI notes are entered in a digital audio workstation: each pitch is represented as a row and columns represent time instants. The matrix element at row p_i and column n_j is set to 1 if a note with pitch p_i is active at time p_j. Otherwise, the element is set to 0. Since a vocal part can only sing at most one note at a time, this is a *one-hot* representation: any column can only contain at most *one* value of 1.

Lastly, we propose an additional score representation: *pure tone*. Since our model is a convolutional neural network that excels at processing audio, we reason that feeding the score into the model as audio might work well for guiding the separation. For simplicity, we do not deal with smooth note transitions; any note onset will result in a transient in the audio-like score. We define the score as a piecewise sine function where the frequency is controlled by the active note's pitch. The pure tone frequency f is determined by the standard MIDI note number to frequency mapping:

$$f = 440 \cdot 2^{\frac{M-69}{12}}.$$

Whenever there is no active note, f is set to 0. The score vector of length n samples then receives the following value at each sample index i:

$$S_i = \sin\left(2\pi f * \frac{i}{F_s}\right),$$

where F_s is the sample rate of the model's audio input (for consistency).

It is important to note a shortcoming in all of the above representations. These representations do not differentiate between a sustained note and a repeated note. In other words, two consecutive notes with the same pitch are represented exactly the same as one note with a longer duration. In contrast, in practice there is a big audible difference between these two cases due to the attack of the second note. It might prove beneficial to devise a score representation that does encode this difference. One possibility that is left for future work is refining the *pitch and amplitude* representation so that the amplitude channel emulates an attack-decay-sustain-release envelope for every note.

6.2 Score-Informed Training on Synthesized Bach Chorales

In this section we describe the experiments we conducted on training score-informed variants of Wave-U-Net. For score-informed training, we generated a variant of the synthesized chorales dataset described in Section 3. In this score-informed dataset, every example contains the mixture audio, each part's audio, and each part's score information. Following the observations in Section 6.1, we only keep timing and pitch information from the score and discard all the rest. In order to create the dataset, we follow the same procedure outlined in Section 3. After we synthesize each part's audio from a MIDI file, we also use the same MIDI file to extract score information for that part by converting the MIDI `note on` and `note off` events into a vector containing the active MIDI note number (or 0 if no note is active) for every time instant. This vector uses the same sample rate as the model's audio input.

The experiments presented in this section are designed to check whether introducing the score into the separation process could improve results for separating SATB mixtures (Experiment 2–4). Given that non-score-informed Wave-U-Net performed very well on two-voice mixtures (see Section 5.5.1), it seemed that introducing the scores for this task is unnecessary. We investigate five score types (defined in Section 6.1.3): MIDI pitch, normalized pitch, pitch and amplitude, piano roll, and pure tone. We also independently investigate three score conditioning locations (defined in Section 6.1.2): input conditioning, output conditioning, and input-output conditioning.

6.2.1 Experiment 5: MIDI Pitch, Extract SATB

This is a preliminary experiment to check the viability of score-informed training. Like in Experiment 2, in this experiment we train the model to separate all four voices from an SATB mixture. We use the simplest score representation: MIDI pitch. We hypothesized that the model would be able to use the score to improve separation quality, or in the worst case, it would just learn to ignore the score and achieve results comparable to Experiment

2. In this experiment, we use input-output conditioning. Input-output conditioning makes sense intuitively because we feed the score into the model in the same locations as the audio: recall that in the Wave-U-Net architecture the original audio input is concatenated to the computed feature map just before the output layer (see Section 2.3.3).

6.2.2 Experiment 6: Normalized Pitch, Extract Single Voice

Since extracting only a single voice from an SATB mixture turned out to improve results for non-score-informed separation (see Experiment 3), we wished to check whether it would have the same effect on score-informed separation. Hence, in this experiment we train four models, one per voice. As mentioned in Section 5.5.2, the worst model performance was encountered in segments containing voice crossings. We hypothesize that the models in this experiment will be able to utilize the pitch and timing information contained in the score to disambiguate these cases. For this reason, we expect the largest improvement in this experiment (compared to Experiment 3) to be in the alto and tenor voices.

In the analysis of results from Experiment 5, we found that conditioning on the MIDI pitch score representation does not work well (see Section 6.3.1 below). For this reason, in this experiment we switch to the normalized pitch score representation. We continue to use input-output conditioning in this experiment.

6.2.3 Experiment 7: Multi-Source Training

Instead of training the network to separate specific voices, in this experiment we train it to separate any of the four voices given that voice's score. In this model variant, the score is the only indication as to which voice should be extracted from the mixture. In this sense, the separation is not only score-informed, it is *score-guided*. Whereas in previous score-informed variants the model *could* use the score to improve separation results, in this variant the model *must* make use of the score. This mode of operation also gives greater flexibility to users of the model: here, one could choose individual notes to extract from a recording, possibly alternating between voices, rather than only being able to extract voices as a whole. Furthermore, multi-source training potentially enables the model to be used on recordings with less or more than four voices.

To achieve this mode of operation, we train the model on all four voices without specifying which training examples belong to which voice. That is, for each example in the original training dataset we generate four separate training examples, one for each choir voice.

We continue to use the same score conditioning as in Experiment 6: normalized pitch score with input-output conditioning.

6.2.4 Experiment 8: Compare Conditioning Methods, Extract SATB

In this experiment we examine the effects of the different score representations and score conditioning locations on separation performance. Following the insights gained from Experiment 4, we use the higher-variability dataset for the present experiment. The higher-variability dataset contains many cases of ambiguity in associating notes to their proper voice, and this would enable us to evaluate the way in which the different score conditioning methods help to disambiguate these cases.

We run a total of 12 tests to compare all possible combinations of four score representations (normalized pitch, pitch and amplitude, piano roll, and pure tone) and three conditioning locations (input, output, and input-output). Models in this experiment are trained to extract all four voices, like in Experiment 2.

6.2.5 Experiment 9: Compare Conditioning Methods, Extract Single Voice

In order to further isolate the effect of score conditioning method, we test all combinations of score type and score conditioning locations on extraction of only a single voice (similar to Experiment 3). We choose to test specifically on the tenor because it achieved the worst median SDR in Experiments 2–4, due in part to being frequently conflated with the alto and bass. For this reason we expect tenor extraction to serve as a good benchmark case for comparing the different score conditioning methods.

Like in Experiment 8, in this experiment we run 12 tests (4 score representations times 3 conditioning locations). All models are trained to extract the tenor only from the higher-variability dataset.

6.2.6 Experiment 10: Compare Conditioning Methods, Multi-Source Training

In this experiment we examine the effect of score conditioning method (similar to Experiments 8–9) in a multi-source training setting (as defined in Experiment 7) on the higher-variability dataset. To this end, we again run tests on all combinations of score representation and score conditioning location, with multi-source training to extract any of the four voices in a score-guided manner.

6.3 Results

In this section we present quantitative evaluation results of Experiments 5–10. Table 6.1 lists these experiments for reference.

Table 6.1: Listing of score-informed Wave-U-Net experiments. HV stands for higher-variability.

Experiment	Dataset	Voices	Score Conditioning	Model Type
5	normal	SATB	MIDI pitch, input-output	one model for all voices
6	normal	SATB	normalized pitch, input-output	one model per voice
7	normal	SATB	normalized pitch, input-output	multi-source
8	HV	SATB	multiple[3]	one model for all voices
9	HV	tenor	multiple[3]	one model per voice
10	HV	SATB	multiple[3]	multi-source

6.3.1 Experiment 5: MIDI Pitch, Extract SATB

This experiment using the MIDI pitch score representation did not yield the expected results. During training, we noticed that after several iterations the loss value stopped improving and got "stuck" around a very high mean squared error of about 0.5, even when left to train for over 100,000 iterations (see Figure 6.3). We are not completely sure what caused this to occur. Failure of the neural network optimization process can have many causes: an inadequate choice of learning rate (Goodfellow et al., 2016, pp. 82–87); local saddle points or large flat regions in the loss function landscape (Goodfellow et al., 2016, p. 278); and the so-called exploding and vanishing gradient problems, in which the norm of the gradient increases or decreases rapidly (Bengio et al., 1994; Pascanu et al., 2013). In our case, we suspect that the failure was caused by the large values of the score conditioning, as the range

[3]All 12 combinations of four score representations (normalized pitch, pitch and amplitude, piano roll, and pure tone) and three conditioning locations (input, output, and input-output).

of MIDI pitch values (0 to 127) is radically different from the range of the audio data that flows through the network (-1 to 1). This is elaborated in Section 6.1.3. A simple possible solution is scaling the score so that it lies in the same range as the audio data, and this is what we investigate in the next experiment.

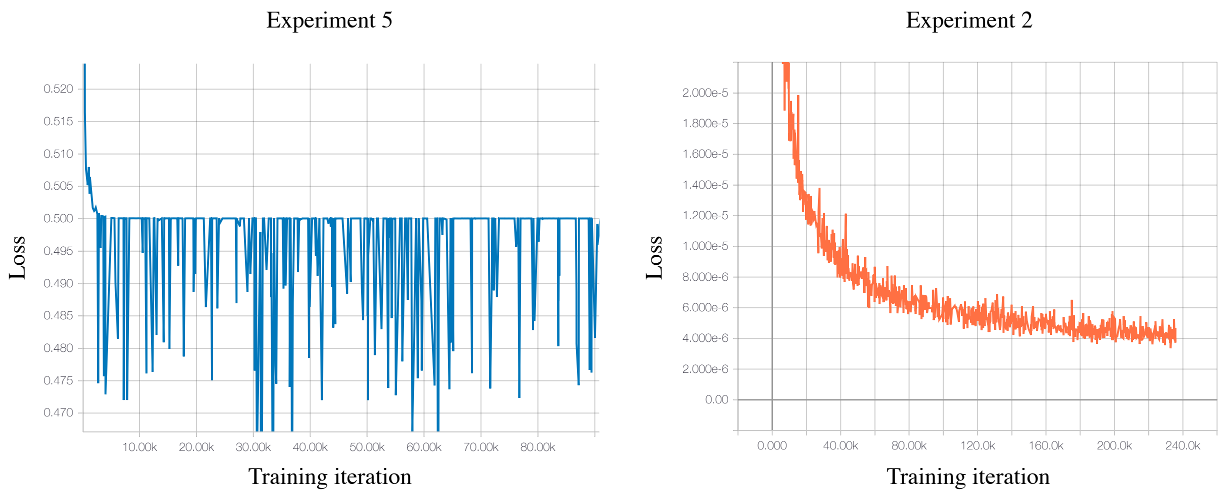


Figure 6.3 Left: the loss value during training in Experiment 5, plotted against training iteration. Right: the loss value during training in Experiment 2, for comparison. Note the orders of magnitude difference in the scale of the y axis.

6.3.2 Experiment 6: Normalized Pitch, Extract Single Voice

Figure 6.4 shows the results of this experiment, with the results of Experiment 3 (the non-score-informed counterpart of this experiment) shown for reference. As we expected, adding the score information improves separation performance specifically on the alto and tenor voices. In order to verify that the score was actually employed by the model, we inspect the performance on frames involving voice crossings between the alto and tenor. Recall that these frames were seen to cause trouble in Experiments 2–4. Table 6.2 shows that in all the frames we examined, SDR indeed improved substantially in this experiment. This indicates that the score was employed to disambiguate voice crossings.

For the soprano and bass voices, Figure 6.4 shows a slight degradation in performance in this experiment compared to its non-score-informed counterpart. This degradation may have occurred because the score-informed model must learn to use the score without any

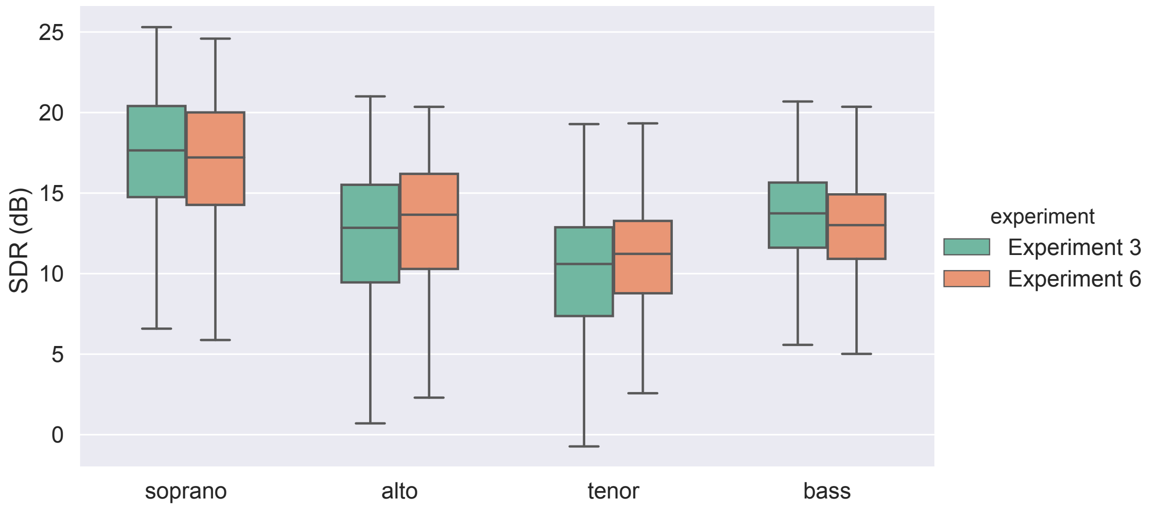


Figure 6.4 SDR evaluations by voice in Experiment 6. Experiment 3 is shown for comparison.

Table 6.2 Comparison between Experiments 2, 3, and 6 of performance on frames with voice crossings (frames taken from Experiment 2, see Table 5.2). A clear improvement is achieved in Experiment 6 using the score.

Source	Chorale	Frame	Ex. 2 SDR	Ex. 3 SDR	Ex. 6 SDR
alto	358	46	-5.25	-2.99	2.30
alto	341	19	-5.33	-2.58	8.11
tenor	367	3	-3.19	-3.41	5.54
tenor	367	4	-3.64	-5.34	10.44
alto	366	20	-3.22	-4.86	8.21
tenor	358	45	-4.04	-5.26	7.37
tenor	358	48	-5.14	-5.65	8.47
tenor	358	44	-4.68	-4.01	6.88

increase in the number of parameters. It is possible that increasing the number of model parameters would eliminate this regression.

6.3.3 Experiment 7: Multi-Source Training

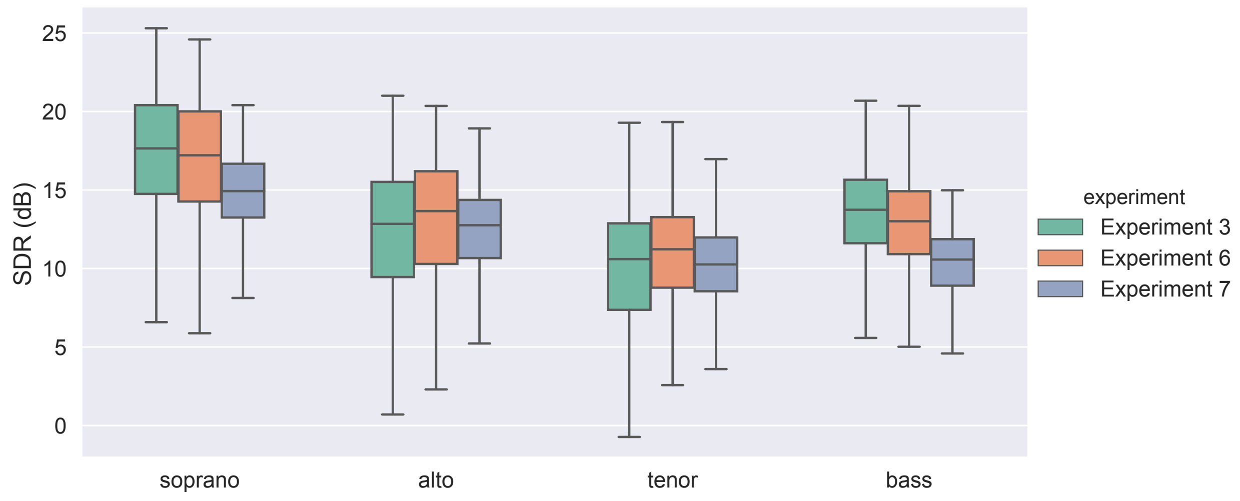


Figure 6.5 SDR evaluations by voice in Experiment 7, compared to Experiments 3 and 6.

Figure 6.5 shows the results of multi-source training (this experiment) compared to non-score-informed training (Experiment 3) and single-source score-informed training (Experiment 6). Interestingly, this experiment achieves SDR results that are inferior to Experiments 3 and 6 in terms of median, but it can be seen that in all voices the SDR distribution is more concentrated in this experiment (the interquartile range is smaller). This indicates that separation performance is more consistent using multi-source training. Most importantly, we note that Experiment 7 uses a *quarter* of the parameters compared to Experiments 3 and 6 (because it separates all four voices using a single model), yet it achieves comparable separation performance on the inner voices.

6.3.4 Experiment 8: Compare Conditioning Methods, Extract SATB

Figure 6.6 shows the results of this experiment comparing score types and conditioning locations. Comparing the different score types, normalized pitch and pure tone seem to have about the same performance, with pitch and amplitude and piano roll performing slightly

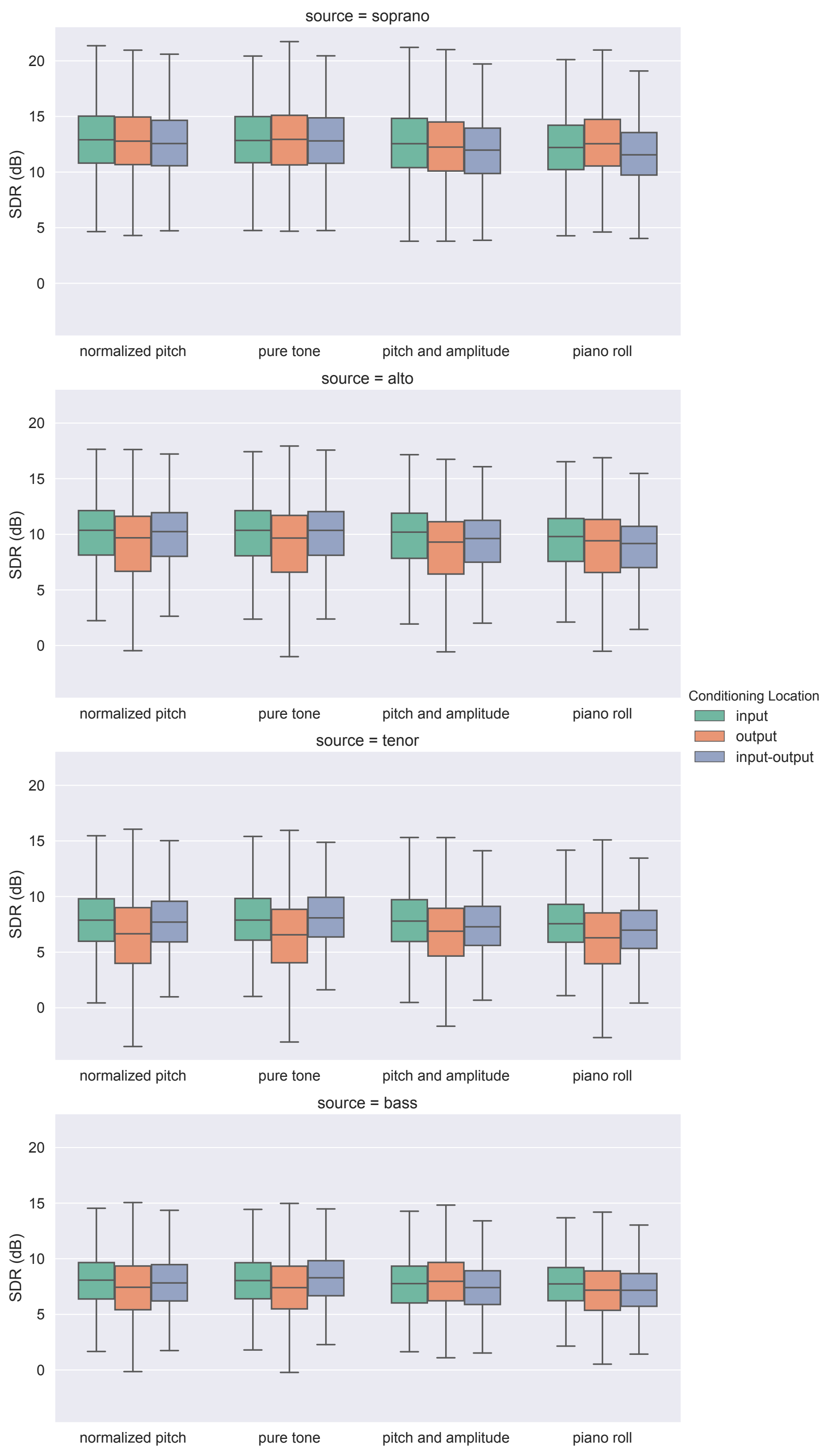


Figure 6.6 SDR evaluations by voice, score type, and conditioning location in Experiment 8.

worse. Interestingly, the conditioning location has no consistent effect on soprano and bass separation performance. For alto and tenor, however, where the score is most important, the results seem to indicate that output conditioning is significantly worse than both input and input-output conditioning, in terms of median and interquartile range for all score types.

We suspect that conditioning at the output does not perform well because of the structure of Wave-U-Net's output layer. The output is computed sample-by-sample as a dot product of the output layer's weights and a vector consisting of: one score sample, one input audio sample, and one sample from the computed feature map. Hence, in the case of output-only conditioning, the score can only have a "shallow" effect as it is only involved in this final dot product and has no effect on the feature map that is computed in multiple convolutional layers.

6.3.5 Experiment 9: Compare Conditioning Methods, Extract Single Voice

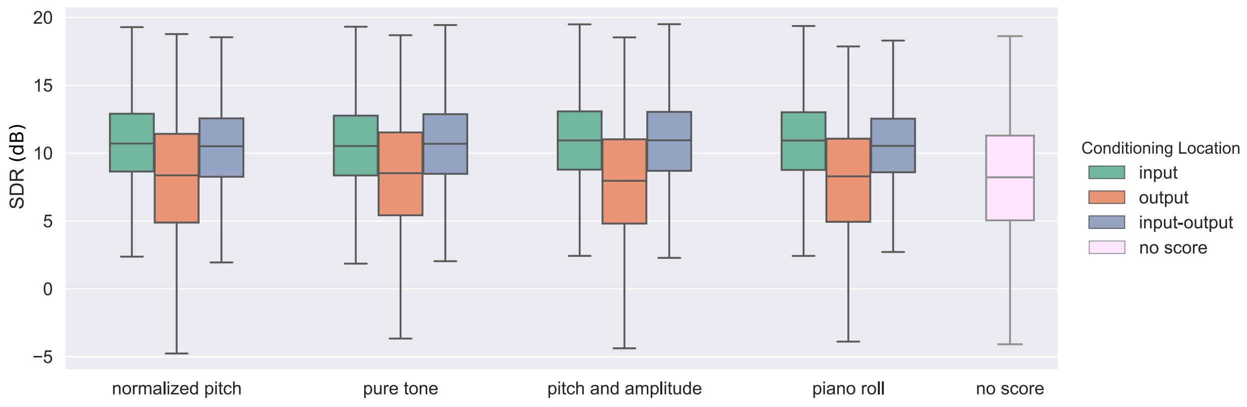


Figure 6.7 A comparison of SDR evaluations by score type and conditioning location in Experiment 9. We also show the 'no score' condition for comparison, taken from the tenor model trained in Experiment 4 on the same dataset.

The results of this experiment which tested the effect of score conditioning method on tenor extraction are shown in Figure 6.7. As in Experiment 8, it can be seen that output is the worst conditioning location. In fact, performance with output conditioning is almost identical to performance without a score at all. This leads us to suspect that the models conditioned at the output have learned to simply ignore the score.

There is almost no difference in separation performance between input and input-output conditioning. All combinations of conditioning locations (other than output) and score types perform similarly and considerably better than no score at all, which confirms our findings from Experiment 8: tenor separation benefits greatly from score conditioning.

6.3.6 Experiment 10: Compare Conditioning Methods, Multi-Source Training

This experiment tested the effect of score conditioning method on multi-source training. The training of 4 models out of 12 in this experiment ended prematurely. The failed models are those that used output conditioning (which performed badly also in Experiment 9). Like in Experiment 5, the training loss in the failed models got stuck at a relatively high value, although in this experiment the actual value was significantly lower than in Experiment 5. Figure 6.8 illustrates the failed training process.

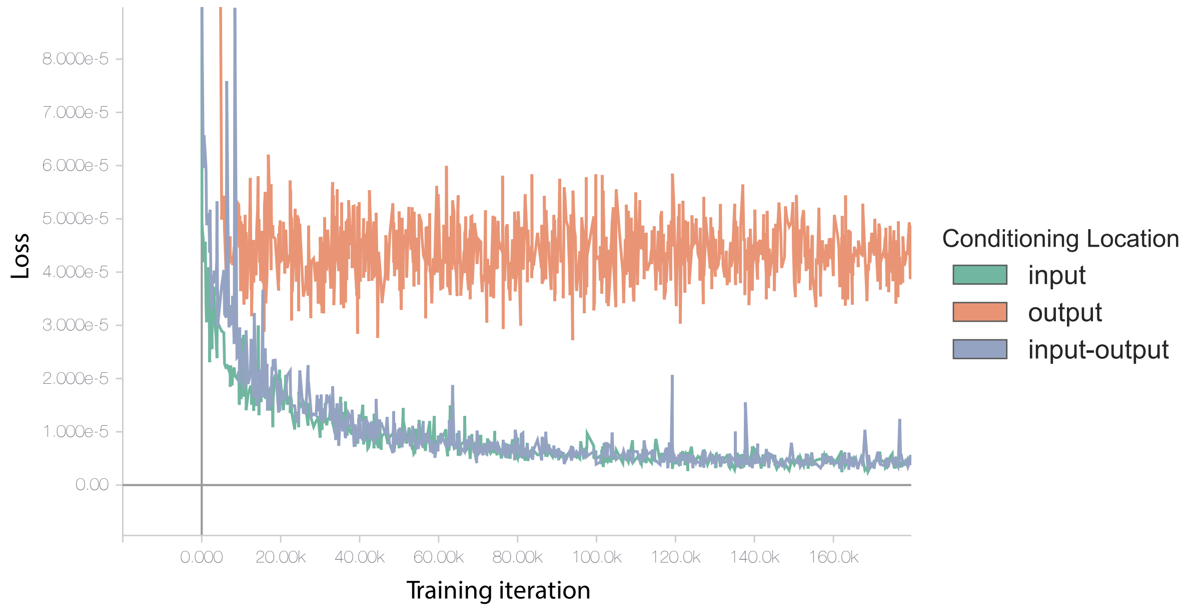


Figure 6.8 The training loss evolution of three models in Experiment 10. All three models used the pitch and amplitude score representation, and differed in the score concatenation location.

Excluding the models that failed to train we are left with 8 models. The per-source results for these models are shown in Figure 6.9. Recall that every one of the 8 trained models is able to separate all four choir voices, guided by the score alone. In this experiment, score type and conditioning location do not seem to have a significant effect on separation quality. Interestingly, the pure tone score type achieves the best median SDR (by a small margin) for soprano and alto, but for tenor and bass it ranks among the lowest.

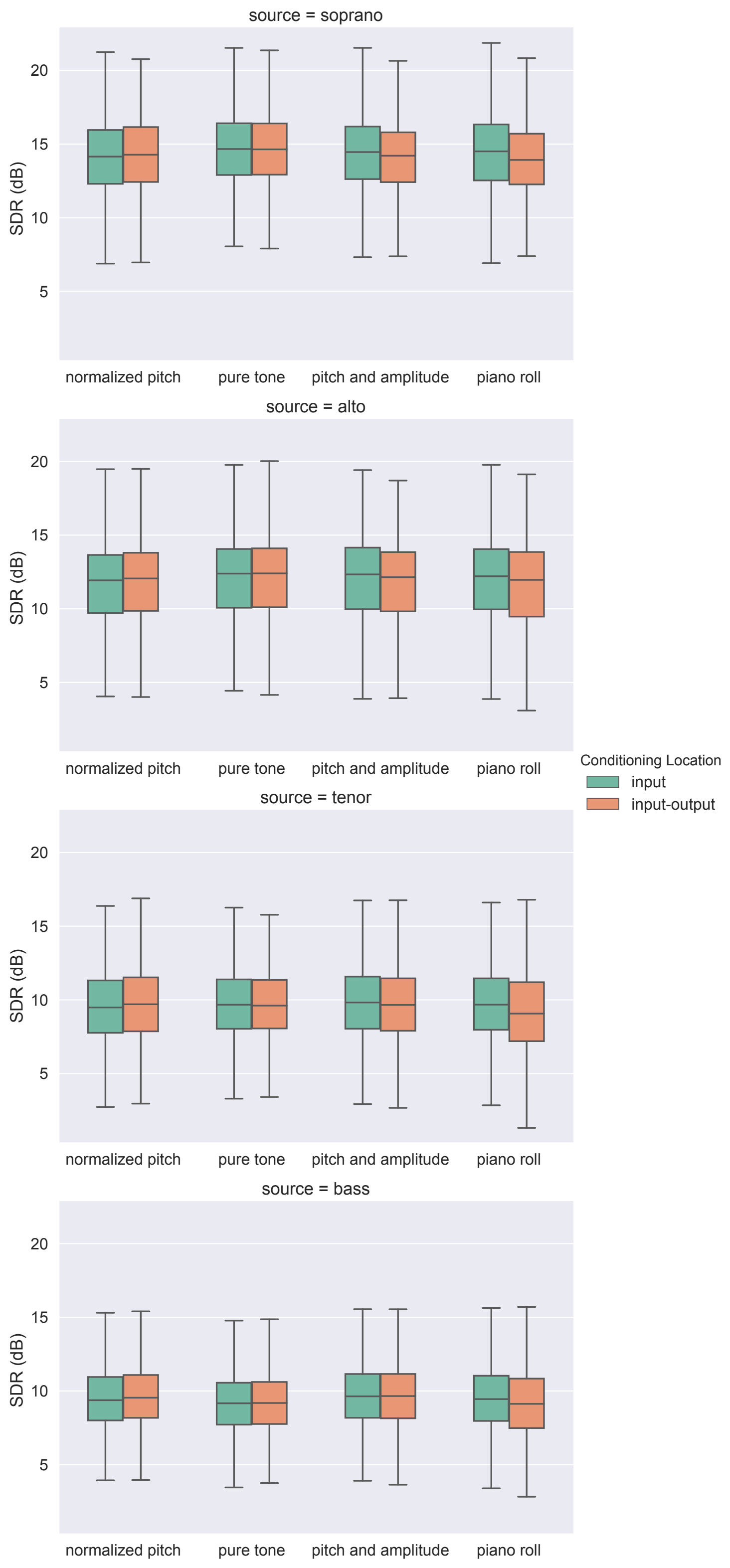


Figure 6.9 A comparison of SDR evaluations by score type and conditioning location in Experiment 10, for each voice separately.

6.3.7 Comparison: Does Using the Score Improve Separation Performance?

After having run the experiments described above and analyzing their results, we finally come to the bottom line that we set out to investigate: Can the musical score be used to improve the quality of source separation of choral music? To answer this question we compare results from Experiments 4, 8, 9, and 10. All these experiments were trained on the higher-variability dataset, which was more representative of real-world choir music compared to the initial dataset of unaltered Bach chorales, and was also shown in Experiment 4 to be more challenging for our models. We choose to compare performance on the tenor voice specifically because it is challenging, as discussed above.

Figure 6.10 shows a comparison of the best models from Experiments 8–10 along with their non-score-informed counterparts. The comparison confirms that using the score significantly improves separation performance in our models. Surprisingly, multi-source training achieves the highest median SDR, even though it extracts all four choir voices with a single model, guided only by the score. However, the interquartile range in multi-source training is large compared to single-voice training and even SATB training. This may indicate that the multi-source model requires more parameters.

6.3.8 Limitations of SDR

In the above sections we have used SDR for evaluating separation quality. SDR is the accepted standard used by a prominent source separation evaluation campaign, SiSEC 2018 (Stöter et al., 2018). However, when performing our evaluations we noticed a problem with the use of SDR. In certain cases, the ranking of evaluations determined by SDR did not match our subjective ranking. Specifically, some frames in which the reference source was near-silent and the estimated source was very quiet received SDR of -20 to -10 dB, even though perceptually the error did not seem significant. And yet, other frames in which the estimated source contained a wrong note or omitted a certain note were evaluated with a higher SDR of -5 to 0 dB.

The SDR is designed such that the energy ratio between the reference source and the residual (separation error) determines the score (Vincent et al., 2006). With $\boldsymbol{s}$ being the reference source frame and $\hat{\boldsymbol{s}}$ the estimated source frame, the SDR is defined by:

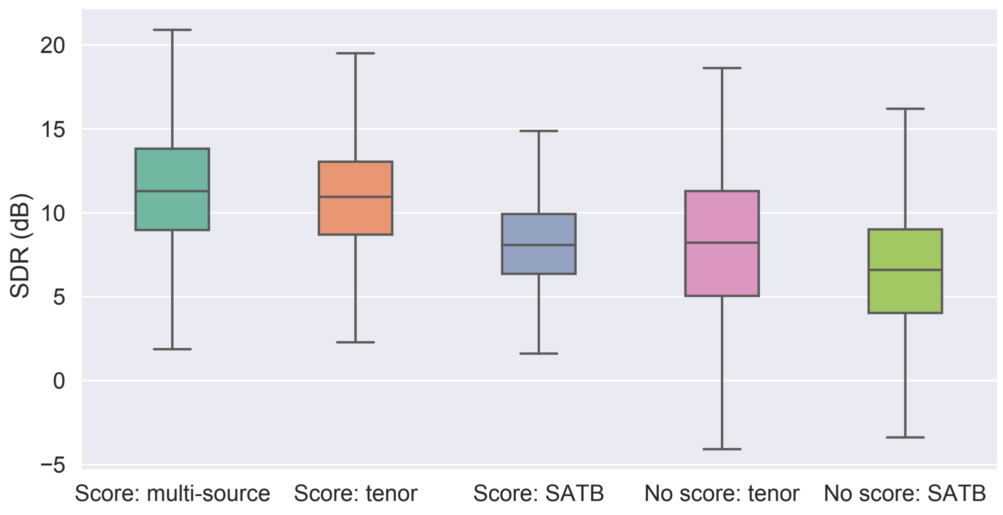


Figure 6.10 Comparison between tenor separation performance on the higher-variability dataset in five scenarios: score-informed multi-source training (the best model from Experiment 10), score-informed single-voice training (the best model from Experiment 9), score-informed SATB training (the best model from Experiment 8), non-score-informed single-voice training (Experiment 4), and non-score-informed SATB (like Experiment 2 but re-run on the higher-variability dataset).

$$\mathrm{SDR} = 10 \log_{10} \left(\frac{\|\boldsymbol{s}\|^2}{\|\boldsymbol{s} - \hat{\boldsymbol{s}}\|^2} \right)$$

This works well in the general case, but in cases where the reference source is completely silent the SDR is undefined (log(0)). Furthermore, in cases where the reference source is nearly silent (due to dithering or a reverb tail of a note from a previous frame, for example), the SDR could be misleadingly low even if the estimated signal is relatively quiet (but not completely silent).

To investigate this further, we checked whether there is a correlation between frame energy and SDR in our experiment evaluations. As illustrated in Figure 6.11, we found that frames with lower energy tended to get a lower SDR evaluation. Furthermore, extremely quiet frames were the only ones with extremely low SDR values.

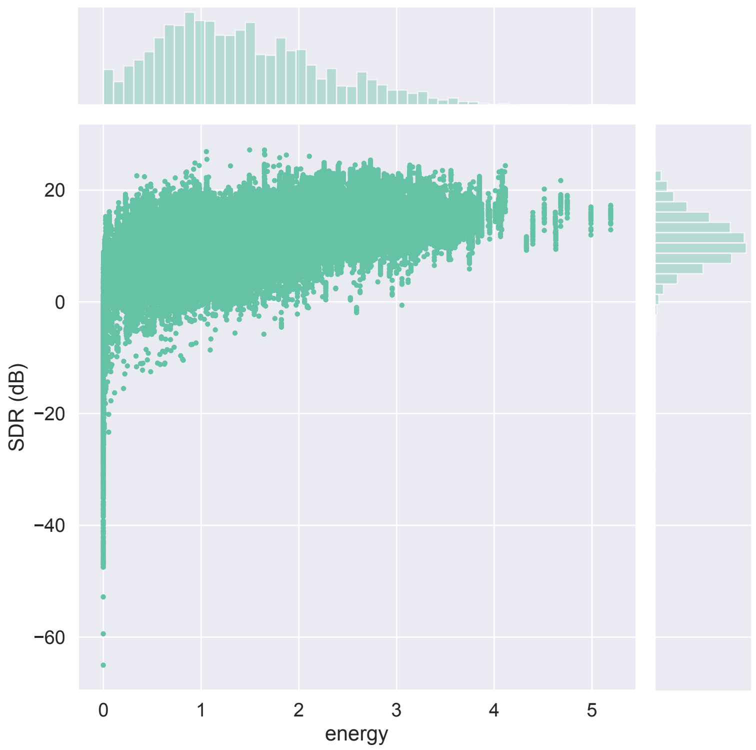


Figure 6.11 Frame SDR plotted against frame energy, showing all evaluation frames from Experiment 4.

Stoller et al. (2018b) have also recently commented on this limitation of the SDR, and have circumvented it by using the median rather than the mean so that the effect of outliers is reduced. We followed the same practice in our evaluations, as did the SiSEC 2018 evaluation campaign.

Using the median does not fix the underlying problem, however: SDR does not correlate with subjective rankings for near-silent frames. In order to address this issue, we propose adding a small positive regularization parameter ϵ to the standard definition of the SDR:

$$\mathrm{SDR_{reg}} = 10\log_{10}\left(\frac{\|\boldsymbol{s}\|^2 + \epsilon}{\|\boldsymbol{s} - \hat{\boldsymbol{s}}\|^2}\right)$$

Regularization is a common technique for controlling the behavior of objective functions (Tikhonov et al., 1995). The effect of ϵ on $\mathrm{SDR_{reg}}$ increases as the energy in $\boldsymbol{s}$ decreases. The value of ϵ should be chosen on a case-by-case basis according to the nature of the dataset being evaluated, so that it boosts the rating of frames that are near-silent but does not significantly alter the ratings of other frames. On our dataset, we found that using $\epsilon = 0.0001$ had the desired effect. For reference, the mean frame energy in our dataset is 1.40 (for 1-second frames and a sample rate of 22,050 Hz). Figure 6.12 shows that $\mathrm{SDR_{reg}}$ remained almost unchanged compared to SDR for most frames, but was boosted for some extremely quiet frames.

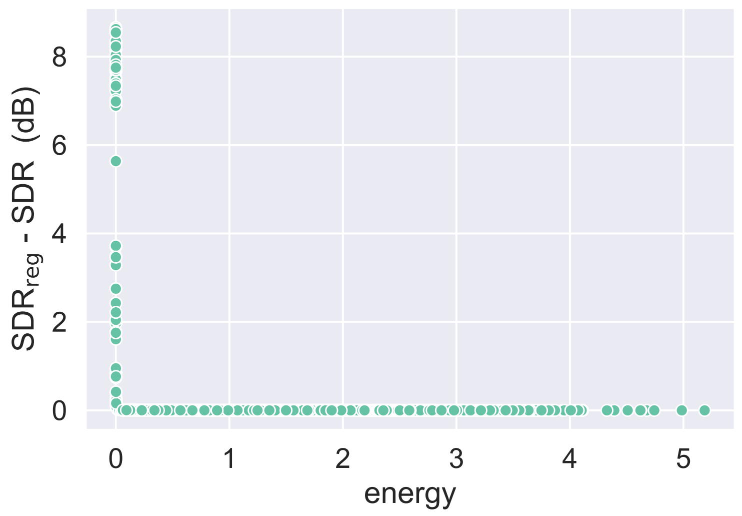


Figure 6.12 The difference between regularized SDR ($\epsilon = 0.0001$) and standard SDR, plotted against frame energy. Regularization has a significant effect only on near-silent frames.

Regularization is a simple yet effective method to improve the correlation between SDR and perceptual rankings of quiet segments in the reference sources. When using regularization one should be careful to note the chosen value of ϵ. In order to remain consistent with other evaluations and enable meaningful comparisons with other studies, we did not use regularization when evaluating the results of our experiments.

6.3.9 Failed Experiments and Lessons Learned

We have run several additional experiments that failed for various reasons. One important lesson we have learned in the process is that when training a neural network, the quality of the training dataset is crucial to the success of the trained model. In early stages of our experimentation, we generated a dataset of all 371 chorales. Unknown to us, a small number of chorales in this dataset were badly labeled because we had forgotten to filter out chorales with instrumental parts. For example, in one chorale in this early version of the dataset, the ground truth for the soprano part was actually the trumpet part, and in the same chorale the ground truth for the alto part was the soprano part. The existence of bad examples in the training data significantly degraded the trained model's performance, even though the number of bad examples was very small compared to the overall number of examples.

After a careful inspection of our dataset we discovered some more errors. For example, in chorales that contained grace notes (short notes that are notated differently than normal notes) sometimes the voices in a chorale would get out of sync. This was caused by a bug in music21, the library that we used to convert the chorales from MusicXML files to MIDI files for synthesis. Fixing that bug led to an improvement in model performance.

We made several experiments varying the configuration of the Wave-U-Net model. Specifically, we tried to train the model with and without temporal context in the input. As reported in the original Wave-U-Net paper, we found that training without temporal context creates click artifacts at segment boundaries. As such, we discarded these experiments. However, training with temporal context just to minimize these artifacts may seem like an overkill. It introduces a large increase in the number of model parameters: an input segment of over 5 seconds is used to predict an output segment of less than 1 second. This increase in the number of parameters leads to a large increase in training time and prediction time, and it is not clear whether these are strictly necessary. More experimentation is needed in order to determine the optimal amount of temporal context for Wave-U-Net.

Chapter 7

Conclusions

The experiments presented in this thesis are, to our knowledge, the first to investigate source separation of choral music. In Section 3, we developed a **dataset of synthesized Bach chorales** on which we conduct our experiments. We proceeded to establish a baseline for separation performance using a **score-informed NMF** method in Section 4. Our experiments showed that score-informed NMF is fairly effective on our dataset, but we found several limitations that suggested that it would be less effective on real-world recordings.

Following the recent success in using deep learning methods for source separation, in Section 5 we investigated a technique called Wave-U-Net. Initial tests on separation of soprano and bass mixtures showed excellent results, but separating four-voice mixtures proved more difficult. We introduced a simple change to Wave-U-Net: instead of training one neural network to separate all sources at once, we trained a **separate network for each source**. This led to a significant improvement and to results that are comparable with the state of the art in separation of vocals and accompaniment.

In order to test separation of recordings that more closely resemble real-world choral music, we created a higher-variability version of the dataset. We found that Wave-U-Net's performance significantly degraded on this dataset. In an attempt to improve performance, in Section 6 we introduced **score-informed Wave-U-Net**. We investigated the effects of introducing the musical score into the separation process by experimenting with various score conditioning methods on all four choir voices in our two datasets. In total, we ran more than a hundred neural network training sessions, each session lasting 24 hours. Our experiments showed that using the score significantly improves separation quality.

Compared to the score-informed NMF baseline, score-informed Wave-U-Net attained significantly better separation results. NMF, however, has the advantage of simplicity: it is easier to implement, faster to produce estimates, and does not necessitate a lengthy training stage.

In Section 6.2.3, we introduced **multi-source training**, in which we trained Wave-U-Net to separate any of the four choir voices using only the score as a guide. This idea enables a mode of operation in which the user extracts individual notes from a recording by simply indicating the desired pitches and times. We found that multi-source training performs comparably to single-source training, even though it can extract any of the four voices using a single model.

Unfortunately, we were not able to test Wave-U-Net on real choir recordings due to the absence of suitable training data. We do believe, however, that score-informed Wave-U-Net has the potential to work well on such recordings, due to its success in singing voice separation and following our experiments on synthesized data. A major challenge remains, then, to compile a multi-track dataset of choral music that is of sufficient size and quality for neural network training. Until such a dataset is created, another possible avenue of research is to employ better choir synthesis methods, particularly ones that can synthesize sung lyrics.

It remains to be seen whether score-informed Wave-U-Net would work well with scores that are not perfectly aligned to the recording. It would be interesting to try and incorporate score alignment directly into the separation process, so that Wave-U-Net could be conditioned using only a symbolic, non-aligned score.

Objective evaluation of source separation methods remains an open challenge. As we have reported in Section 6.3.8, we found a widely used evaluation metric to be inadequate in certain cases. Other researchers have recently reached similar conclusions. We proposed regularization as a possible remedy, but a widely applicable metric that is better correlated with subjective ratings is yet to be found.

In conclusion, we have found that Wave-U-Net's multi-scale convolutional structure, built on the strong foundations of deep learning, allows it to effectively handle the complexities of choral music separation. We developed a score-informed variant of Wave-U-Net that significantly improved separation performance on a synthesized dataset of Bach chorales. We believe that score-informed Wave-U-Net could be applicable to other musical source separation tasks, such as lead and accompaniment separation, in cases where the musical score is available.

Appendix A

Datasets

The datasets described in Section 3 are built from Bach chorale harmonizations collected and published in the Riemenschneider edition (Bach, 1941). The edition contains a total of 371 harmonizations, out of which we exclude 20 because they contain instrumental parts or more than four vocal parts. The following chorales are excluded: 11, 43, 46, 51, 116, 150, 270, 298, 313, 323, 327, 329, 330, 331, 344, 347, 348, 353, 362, 368. The 351 remaining chorales are sorted by their number in the Riemenschneider edition and split into three partitions, as listed in Table A.1.

Table A.1 List of dataset partitions

	number of chorales	total duration	avg. chorale duration
normal dataset			
training	270	02:45:28	00:00:36
validation	50	00:28:58	00:00:34
test	31	00:18:09	00:00:35
total	351	03:32:36	00:00:36
higher-variability dataset			
training	270	02:56:48	00:00:39
validation	50	00:32:14	00:00:38
test	31	00:19:33	00:00:37
total	351	03:48:36	00:00:39

Appendix B

Supplemental Material

We publish the following audio and code to accompany this thesis. All software is released under an open-source license on GitHub.

- Audio examples for all of our experiments:
  `https://www.matangover.com/choir-separation`
- Code used to create the synthesized chorales dataset (described in Section 3):
  `https://github.com/matangover/synthesize-chorales`
- Score-informed NMF code (described in Section 4):
  `https://github.com/matangover/score-informed-nmf`
- Score-informed Wave-U-Net code (described in Section 6):
  `https://github.com/matangover/score-informed-Wave-U-Net`
- Code used to analyze experiment results and generate figures:
  `https://github.com/matangover/thesis-results-analysis`

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., … Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation* (pp. 265–283). Savannah, GA: USENIX Association.

Alldahl, P.-G. (2008). *Choral intonation*. Stockholm, Sweden: Gehrmans Musikförlag.

Arzt, A., & Lattner, S. (2018). Audio-to-score alignment using transposition-invariant features. In *Proceedings of the International Society for Music Information Retrieval Conference* (pp. 592–599). Paris, France.

Bach, J. S. (1941). *371 harmonized chorales and 69 chorale melodies with figured bass* (A. Riemenschneider, Ed.). New York, NY: G. Schirmer, Inc.

Barker, J., Marxer, R., Vincent, E., & Watanabe, S. (2015). The third "CHiME" speech separation and recognition challenge: Dataset, task and baselines. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (pp. 504–511). `https://doi.org/10.1109/ASRU.2015.7404837`

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, *5*(2), 157–166. `https://doi.org/10.1109/72.279181`

Ben-Shalom, A., & Dubnov, S. (2004). Optimal filtering of an instrument sound in a mixed recording given approximate pitch prior. In *Proceedings of the International Computer Music Conference (ICMC)*. San Francisco, CA: International Computer Music Association.

Bent, I. D., Hughes, D. W., Provine, R. C., Rastall, R., Kilmer, A., Hiley, D., … Chew, G. (2001). Notation. In *Grove Music Online*. Oxford, UK: Oxford University Press.

Blaauw, M., Bonada, J., & Daido, R. (2019). Data efficient voice cloning for neural singing synthesis. In *IEEE International Conference on Acoustics, Speech and Signal Processing*

(ICASSP) (pp. 6840–6844). Brighton, UK.

Bonada, J., & Serra, X. (2007). Synthesis of the singing voice by performance sampling and spectral models. *IEEE Signal Processing Magazine*, *24*(2), 67–79. https://doi.org/10.1109/MSP.2007.323266

Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Y. Lechevallier & G. Saporta (Eds.), *Proceedings of COMPSTAT'2010* (pp. 177–186). Heidelberg, Germany: Physica-Verlag HD.

Boutsidis, C., & Gallopoulos, E. (2008). SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, *41*(4), 1350–1362. https://doi.org/10.1016/j.patcog.2007.09.010

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. Cambridge, UK: Cambridge University Press.

Bregman, A. S. (1990). *Auditory scene analysis: The perceptual organization of sound*. Cambridge, MA: The MIT Press.

Brown, G. J., & Cooke, M. (1994). Computational auditory scene analysis. *Computer Speech & Language*, *8*(4), 297–336. https://doi.org/10.1006/csla.1994.1016

Butcher, K., & Studebaker, D. (1985). Choral part-books then and now. *Choral Journal*, *26*(2), 19–21.

Cano, E., FitzGerald, D., Liutkus, A., Plumbley, M. D., & Stöter, F. (2019). Musical source separation: An introduction. *IEEE Signal Processing Magazine*, *36*(1), 31–40. https://doi.org/10.1109/MSP.2018.2874719

Cano, P., Loscos, A., & Bonada, J. (1999). Score-performance matching using HMMs. In *Proceedings of the 1999 International Computer Music Conference (ICMC)* (pp. 441–444). Beijing, China: Michigan Publishing.

Cardoso, J. (1998). Blind signal separation: Statistical principles. *Proceedings of the IEEE*, *86*(10), 2009–2025. https://doi.org/10.1109/5.720250

Casey, M. A., & Westner, A. (2000). Separation of mixed audio sources by independent subspace analysis. In *Proceedings of the 2000 International Computer Music Conference (ICMC)* (Vol. 2000). Berlin, Germany: Michigan Publishing.

Charlton, D., & Whitney, K. (2001). Score (i). In *Grove Music Online*. Oxford, UK: Oxford University Press.

Cherry, E. C. (1953). Some experiments on the recognition of speech, with one and with two ears. *Journal of the Acoustical Society of America*, *25*, 975–979. https://doi.org/

10.1121/1.1907229

Cichocki, A., & Phan, A.-H. (2009). Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *92*(3), 708–721.

Cichocki, A., Zdunek, R., & Amari, S.-i. (2006). New algorithms for non-negative matrix factorization in applications to blind source separation. In *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings* (Vol. 5, pp. 621–624). Toulouse, France: IEEE.

Cohn, R., Hyer, B., Dahlhaus, C., Anderson, J., & Wilson, C. (2001). Harmony. In *Grove Music Online*. Oxford, UK: Oxford University Press.

Comon, P. (1994). Independent component analysis, a new concept? *Signal Processing*, *36*(3), 287–314. https://doi.org/10.1016/0165-1684(94)90029-9

Cuesta, H., Gómez, E., Martorell, A., & Loáiciga, F. (2018). Analysis of intonation in unison choir singing. In *Proceedings of the International Conference on Music Perception and Cognition (ICMPC)*. Graz, Austria.

Cuthbert, M. S., & Ariza, C. (2010). Music21: A toolkit for computer-aided musicology and symbolic music data. In J. S. Downie & R. C. Veltkamp (Eds.), *Proceedings of the International Society for Music Information Retrieval Conference* (pp. 637–642). Utrecht, Netherlands.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*, 303–314. https://doi.org/10.1007/BF02551274

Daffern, H. (2017). Blend in singing ensemble performance: Vibrato production in a vocal quartet. *Journal of Voice*, *31*(3), 385.e23–385.e29. https://doi.org/10.1016/j.jvoice.2016.09.007

Dai, J., & Dixon, S. (2017). Analysis of interactive intonation in unaccompanied SATB ensembles. In X. Hu, S. J. Cunningham, D. Turnbull, & Z. Duan (Eds.), *Proceedings of the International Society for Music Information Retrieval Conference*. Suzhou, China.

Dannenberg, R. B., & Raphael, C. (2006). Music score alignment and computer accompaniment. *Communications of the ACM*, *49*(8), 38–43. https://doi.org/10.1145/1145287.1145311

Devaney, J., Mandel, M. I., & Fujinaga, I. (2012). A study of intonation in three-part singing using the Automatic Music Performance Analysis and Comparison Toolkit (AMPACT). In F. Gouyon, P. Herrera, L. G. Martins, & M. Müller (Eds.), *Proceedings of the International*

Society for Music Information Retrieval Conference (pp. 511–516). Porto, Portugal: FEUP Edições.

Duan, Z., & Pardo, B. (2011). Soundprism: An online system for score-informed source separation of music audio. *IEEE Journal of Selected Topics in Signal Processing*, *5*(6), 1205–1215. `https://doi.org/10.1109/JSTSP.2011.2159701`

Dumoulin, V., Perez, E., Schucher, N., Strub, F., Vries, H. de, Courville, A., & Bengio, Y. (2018). Feature-wise transformations. *Distill*, *3*(7), e11. `https://doi.org/10.23915/distill.00011`

Dumoulin, V., & Visin, F. (2016). *A guide to convolution arithmetic for deep learning.* Retrieved from `http://arxiv.org/abs/1603.07285`

Durrieu, J., Ozerov, A., Févotte, C., Richard, G., & David, B. (2009). Main instrument separation from stereophonic audio signals using a source/filter model. In *2009 17th European Signal Processing Conference* (pp. 15–19). Glasgow, UK: IEEE.

Emiya, V., Vincent, E., Harlander, N., & Hohmann, V. (2011). Subjective and objective quality assessment of audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, *19*(7), 2046–2057. `https://doi.org/10.1109/TASL.2011.2109381`

Engel, J., Agrawal, K. K., Chen, S., Gulrajani, I., Donahue, C., & Roberts, A. (2019). GANSynth: Adversarial neural audio synthesis. In *Proceedings of the International Conference on Learning Representations (ICLR).* New Orleans, LA.

Erickson, R. (2001). Musica enchiriadis, Scolica enchiriadis. In *Grove Music Online.* Oxford, UK: Oxford University Press.

Ewert, S., & Müller, M. (2012). Using score-informed constraints for NMF-based source separation. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 129–132). `https://doi.org/10.1109/ICASSP.2012.6287834`

Ewert, S., Pardo, B., Mueller, M., & Plumbley, M. D. (2014). Score-informed source separation for musical audio recordings: An overview. *IEEE Signal Processing Magazine*, *31*(3), 116–124. `https://doi.org/10.1109/MSP.2013.2296076`

Ewert, S., & Sandler, M. B. (2017). Structured dropout for weak label and multi-instance learning and its application to score-informed source separation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2277–2281). New Orleans, LA: IEEE.

Fan, Z., Lai, Y., & Jang, J. R. (2018). SVSGAN: Singing voice separation via generative adversarial network. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 726–730). `https://doi.org/10.1109/ICASSP.2018.8462091`

Févotte, C., Bertin, N., & Durrieu, J.-L. (2009). Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis. *Neural Computation*, *21*(3), 793–830.

Févotte, C., Gribonval, R., & Vincent, E. (2005). *BSS_EVAL Toolbox User Guide – Revision 2.0* (Technical Report No. 1706). Retrieved from Institut de Recherche en Informatique et Systèmes Aléatoires website: `https://hal.inria.fr/inria-00564760`

Févotte, C., & Idier, J. (2011). Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation*, *23*(9), 2421–2456.

Finesso, L., & Spreij, P. (2006). Nonnegative matrix factorization and I-divergence alternating minimization. *Linear Algebra and Its Applications*, *416*(2), 270–287. `https://doi.org/10.1016/j.laa.2005.11.012`

Floros, C., & Wicker, V. (1995). *Gustav Mahler: The symphonies.* Portland, OR: Scolar Press.

Furui, S. (1986). Speaker-independent isolated word recognition based on emphasized spectral dynamics. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 1991–1994). `https://doi.org/10.1109/ICASSP.1986.1168654`

Ganseman, J., Mysore, G. J., Abel, J. S., & Scheunders, P. (2010). Source separation by score synthesis. In *Proceedings of the International Computer Music Conference (ICMC)* (pp. 462–465). New York, NY.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 315–323). Fort Lauderdale, FL.

Glöckner, A. (2010). On the performing forces of Johann Sebastian Bach's Leipzig church music. *Early Music*, *38*(2), 215–222.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* Cambridge, MA: MIT Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 27* (pp. 2672–2680). Curran Associates, Inc.

Gómez, E., Blaauw, M., Bonada, J., Chandna, P., & Cuesta, H. (2018). *Deep learning for singing processing: Achievements, challenges and impact on singers and listeners.* Keynote speech presented at the 2018 Joint Workshop on Machine Learning for Music, Stockholm, Sweden.

Griffin, D., & Lim, J. (1984). Signal estimation from modified short-time Fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, *32*(2), 236–243. https://doi.org/10.1109/TASSP.1984.1164317

Heittola, T., Klapuri, A., & Virtanen, T. (2009). Musical instrument recognition in polyphonic audio using source-filter model for sound separation. In *Proceedings of the International Society for Music Information Retrieval Conference* (pp. 327–332). Kobe, Japan.

Hennequin, R., Badeau, R., & David, B. (2010). Time-dependent parametric and harmonic templates in non-negative matrix factorization. In *Proceedings of the 13th International Conference on Digital Audio Effects (DAFx).* Graz, Austria.

Hennequin, R., David, B., & Badeau, R. (2011). Score informed audio source separation using a parametric model of non-negative spectrogram. In *Proc. Of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (pp. 45–48). Prague, Czech Republic: IEEE.

Henningsson, D., & Team, F. D. (2011). FluidSynth real-time and thread safety challenges. In *Proceedings of the 9th International Linux Audio Conference* (pp. 123–128). Maynooth, Ireland.

Hershey, J. R., Chen, Z., Roux, J. L., & Watanabe, S. (2016). Deep clustering: Discriminative embeddings for segmentation and separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 31–35). https://doi.org/10.1109/ICASSP.2016.7471631

Hill, K. (2017). *Choral singing, choral attendance, and the situation of choirs in Canada* [Report]. Retrieved from https://hillstrategies.com/resource/choral-singing-choral-attendance-and-the-situation-of-choirs-in-canada/

Holoman, D. K. (2014). *Writing about music: A style sheet.* Berkley, CA: University of California Press.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366. https://doi.org/10.1016/0893-6080(89)90020-8

Howard, D. M. (2007). Intonation drift in a cappella soprano, alto, tenor, bass quartet singing with key modulation. *Journal of Voice*, *21*(3), 300–315. https://doi.org/10.1016/j.jvoice.2005.12.005

Hu, N., Dannenberg, R. B., & Tzanetakis, G. (2003). Polyphonic audio matching and alignment for music retrieval. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (pp. 185–188). https://doi.org/10.1109/ASPAA.2003.1285862

Huang, P.-S., Kim, M., Hasegawa-Johnson, M., & Smaragdis, P. (2015). Joint optimization of masks and deep recurrent neural networks for monaural source separation. *IEEE/ACM Transactions on Audio, Speech and Language Processing*, *23*(12), 2136–2147. https://doi.org/10.1109/TASLP.2015.2468583

Hyer, B. (2001). Tonality. In *Grove Music Online*. Oxford, UK: Oxford University Press.

Hyvärinen, A., & Oja, E. (2000). Independent component analysis: Algorithms and applications. *Neural Networks*, *13*(4), 411–430. https://doi.org/10.1016/S0893-6080(00)00026-5

Ihalainen, K. (2008). *Methods of choir recording for an audio engineer* (Bachelor's thesis, Tampere University of Applied Sciences). Retrieved from http://urn.fi/URN:NBN:fi:amk-201003065302

Jansson, A., Humphrey, E., Montecchio, N., Bittner, R., Kumar, A., & Weyde, T. (2017). Singing voice separation with deep U-Net convolutional networks. In *Proceedings of the International Society for Music Information Retrieval Conference*. Suzhou, China.

Jers, H., & Ternström, S. (2005). *Intonation analysis of a multi-channel choir recording* [Quarterly Progress and Status Report]. Stockholm, Sweden: KTH Computer Science and Communication, Dept. for Speech, Music and Hearing.

Jordania, J. (2011). *Why do people sing? Music in human evolution* (A. Jordania, Ed.). Tbilisi, Georgia: Logos.

Jordania, J. (2015). *Choral singing in human culture and evolution.* Lambert Academic Publishing.

Kelly, T. F. (2001). *First nights: Five musical premieres.* New Haven, CT: Yale University Press.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. San Diego, USA.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in Neural Information Processing Systems 25* (pp. 1097–1105). Curran Associates, Inc.

Lecun, Y. (1989). Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, & L. Steels (Eds.), *Connectionism in perspective.* Zürich, Switzerland: Elsevier.

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K.-R. (1998). Efficient BackProp. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade* (Second Edition, pp. 9–48). `https://doi.org/10.1007/978-3-642-35289-8_3`

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. `https://doi.org/10.1038/nature14539`

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In D. Touretzky (Ed.), *Advances in Neural Information Processing Systems (NIPS 1989)* (Vol. 2). Denver, CO: Morgan Kaufmann.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755), 788–791. `https://doi.org/10.1038/44565`

Lee, D. D., & Seung, H. S. (2000). Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems* (pp. 535–541). Cambridge, MA: MIT Press.

Lee, Y., Wang, C., Wang, S., Wang, J., & Wu, C. (2017). Fully complex deep neural network for phase-incorporating monaural source separation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 281–285). `https://doi.org/10.1109/ICASSP.2017.7952162`

Lefèvre, A., Bach, F., & Févotte, C. (2011). Itakura-Saito nonnegative matrix factorization with group sparsity. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 21–24). `https://doi.org/10.1109/ICASSP.2011.5946318`

Lewicki, M. S. (2002). Efficient coding of natural sounds. *Nature Neuroscience*, *5*(4), 356. `https://doi.org/10.1038/nn831`

Li, Y., Woodruff, J., & Wang, D. (2009). Monaural musical sound separation based on pitch and common amplitude modulation. *IEEE Transactions on Audio, Speech, and Language Processing*, *17*(7), 1361–1371. `https://doi.org/10.1109/TASL.2009.2020886`

Liutkus, A., Fitzgerald, D., & Rafii, Z. (2015). Scalable audio separation with light Kernel

Additive Modelling. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 76–80). `https://doi.org/10.1109/ICASSP.2015.7177935`

Liutkus, A., Stöter, F.-R., Rafii, Z., Kitamura, D., Rivet, B., Ito, N., … Fontecave, J. (2017). The 2016 Signal Separation Evaluation Campaign. In P. Tichavský, M. Babaie-Zadeh, O. J. J. Michel, & N. Thirion-Moreau (Eds.), *Latent Variable Analysis and Signal Separation (LVA/ICA 2017)* (pp. 323–332). Cham, Switzerland: Springer International Publishing.

Luo, Y., Chen, Z., Hershey, J. R., Roux, J. L., & Mesgarani, N. (2016). Deep clustering and conventional networks for music separation: Stronger together. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 61–65). `https://doi.org/10.1109/ICASSP.2017.7952118`

Marshall, R. L., & Leaver, R. A. (2001a). Chorale. In *Grove Music Online*. Oxford, UK: Oxford University Press.

Marshall, R. L., & Leaver, R. A. (2001b). Chorale settings. In *Grove Music Online*. Oxford, UK: Oxford University Press.

Mauch, M., Frieler, K., & Dixon, S. (2014). Intonation in unaccompanied singing: Accuracy, drift, and a model of reference pitch memory. *The Journal of the Acoustical Society of America*, *136*(1), 401–411. `https://doi.org/10.1121/1.4881915`

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, *5*(4), 115–133. `https://doi.org/10.1007/BF02478259`

McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015). librosa: Audio and music signal analysis in Python. In *Proceedings of the 14th Python in Science Conference* (pp. 18–24). Austin, Texas.

Meron, Y., & Hirose, K. (1998). Separation of singing and piano sounds. In *5th International Conference on Spoken Language Processing*. Sydney, Australia.

Miron, M., Carabias-Orti, J. J., Bosch, J. J., Gómez, E., & Janer, J. (2016). Score-informed source separation for multichannel orchestral recordings. *Journal of Electrical and Computer Engineering*, *2016*. `https://doi.org/10.1155/2016/8363507`

Miron, M., Janer, J., & Gómez, E. (2017a). Generating data to train convolutional neural networks for classical music source separation. In *Proceedings of the International Society for Music Information Retrieval Conference* (pp. 227–233). Suzhou, China.

Miron, M., Janer, J., & Gómez, E. (2017b). Monaural score-informed source separation for classical music using convolutional neural networks. In *Proceedings of the International Society for Music Information Retrieval Conference* (pp. 55–62). Suzhou, China.

Mitianoudis, N., & Davies, M. E. (2003). Audio source separation of convolutive mixtures. *IEEE Transactions on Speech and Audio Processing*, *11*(5), 489–497. `https://doi.org/10.1109/TSA.2003.815820`

Muth, J., Uhlich, S., Perraudin, N., Kemp, T., Cardinaux, F., & Mitsufuji, Y. (2018, July 7). *Improving DNN-based music source separation using phase features.* Presented at the Joint Workshop on Machine Learning for Music at ICML, IJCAI/ECAI and AAMAS. Retrieved from `http://arxiv.org/abs/1807.02710`

Nakamura, T., & Kameoka, H. (2016). Shifted and convolutive source-filter non-negative matrix factorization for monaural audio source separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 489–493). `https://doi.org/10.1109/ICASSP.2016.7471723`

Nugraha, A. A., Liutkus, A., & Vincent, E. (2016). Multichannel music separation with deep neural networks. In *2016 24th European Signal Processing Conference (EUSIPCO)* (pp. 1748–1752). `https://doi.org/10.1109/EUSIPCO.2016.7760548`

Odena, A., Dumoulin, V., & Olah, C. (2016). Deconvolution and checkerboard artifacts. *Distill*, *1*(10), e3. `https://doi.org/10.23915/distill.00003`

Oord, A. van den, Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., … Kavukcuoglu, K. (2016). WaveNet: A generative model for raw audio. In *9th ISCA Speech Synthesis Workshop*. Sunnyvale, CA.

Orfanidis, S. J. (1995). *Introduction to signal processing.* Upper Saddle River, NJ: Prentice-Hall, Inc.

Parrott, A. (2010). Bach's chorus: The Leipzig line. A response to Andreas Glöckner. *Early Music*, *38*(2), 223–235.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning* (Vol. 28, pp. 1310–1318). Atlanta, GA: JMLR.org.

Pascual, S., Bonafonte, A., & Serrà, J. (2017). SEGAN: Speech enhancement generative adversarial network. In *Proceedings of Interspeech 2017* (pp. 3642–3646). `https://doi.org/10.21437/Interspeech.2017-1428`

Patterson, R., Nimmo-Smith, I., Holdsworth, J., & Rice, P. (1987). *An efficient auditory filterbank based on the gammatone function.* Presented at the Meeting of the Institute of

Acoustics on Auditory Modelling, RSRE, Malvern, UK.

Perez, E., Strub, F., Vries, H. de, Dumoulin, V., & Courville, A. (2018). FiLM: Visual reasoning with a general conditioning layer. In *32nd Conference on Artificial Intelligence (AAAI-18)*. New Orleans, LA.

Pons, J., Lidy, T., & Serra, X. (2016). Experimenting with musically motivated convolutional neural networks. In *14th International Workshop on Content-Based Multimedia Indexing (CBMI)* (pp. 1–6). `https://doi.org/10.1109/CBMI.2016.7500246`

Purwins, H., Li, B., Virtanen, T., Schlüter, J., Chang, S.-y., & Sainath, T. (2019). Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 206–219. `https://doi.org/10.1109/JSTSP.2019.2908700`

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., & Bittner, R. (2017). *MUSDB18 – a corpus for music separation* [Data set]. `https://doi.org/10.5281/zenodo.1117372`

Rafii, Z., Liutkus, A., Stöter, F.-R., Mimilakis, S. I., FitzGerald, D., & Pardo, B. (2018). An overview of lead and accompaniment separation in music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *26*(8), 1307–1335. `https://doi.org/10.1109/TASLP.2018.2825440`

Randel, D. M. (Ed.). (2003a). Chorus. In *The Harvard Dictionary of Music* (4th edition). Cambridge, MA: Harvard University Press.

Randel, D. M. (Ed.). (2003b). Voice. In *The Harvard Dictionary of Music* (4th edition). Cambridge, MA: Harvard University Press.

Ranzato, M., Huang, F. J., Boureau, Y. L., & LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'07)*. `https://doi.org/10.1109/CVPR.2007.383157`

Raphael, C. (2008). A classifier-based approach to score-guided source separation of musical audio. *Computer Music Journal*, *32*(1), 51–59. `https://doi.org/10.1162/comj.2008.32.1.51`

Ravanelli, M., & Bengio, Y. (2018). Speaker recognition from raw waveform with SincNet. In *2018 IEEE Spoken Language Technology Workshop (SLT)* (pp. 1021–1028). `https://doi.org/10.1109/SLT.2018.8639585`

Rifkin, J. (1982). Bach's chorus: A preliminary report. *The Musical Times*, *123*(1677), 747–754. `https://doi.org/10.2307/961592`

Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of*

Mathematical Statistics, *22*(3), 400–407.

Rodet, X. (2002). Synthesis and processing of the singing voice. In *Proceedings of the 1st IEEE Benelux Workshop on Model-based Processing and Coding of Audio (MPCA-2002)* (pp. 15–31). Leuven, Belgium.

Rodriguez-Serrano, F. J., Duan, Z., Vera-Candeas, P., Pardo, B., & Carabias-Orti, J. J. (2015). Online score-informed source separation with adaptive instrument models. *Journal of New Music Research*, *44*(2), 83–96. `https://doi.org/10.1080/09298215.2014.989174`

Rodriguez-Serrano, F. J., Ewert, S., Vera-Candeas, P., & Sandler, M. (2016). A score-informed shift-invariant extension of complex matrix factorization for improving the separation of overlapped partials in music recordings. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 61–65). `https://doi.org/10.1109/ICASSP.2016.7471637`

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI 2015)* (pp. 234–241). Cham, Switzerland: Springer.

Roweis, S. T. (2000). One microphone source separation. In *Advances in Neural Information Processing Systems 13* (pp. 793–799). Cambridge, MA: MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533. `https://doi.org/10.1038/323533a0`

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252. `https://doi.org/10.1007/s11263-015-0816-y`

Sachs, K.-J., & Dahlhaus, C. (2001). Counterpoint. In *Grove Music Online*. Oxford, UK: Oxford University Press.

Scherer, D., Müller, A., & Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks* (pp. 92–101). Berlin, Heidelberg: Springer-Verlag.

Schnell, N., Peeters, G., Lemouton, S., Manoury, P., & Rodet, X. (2000). Synthesizing a choir in real-time using pitch synchronous overlap add (PSOLA). In *Proceedings of the International Computer Music Conference (ICMC)*. Berlin, Germany.

Shen, J., Pang, R., Weiss, R. J., Schuster, M., Jaitly, N., Yang, Z., … Wu, Y. (2018). Natural TTS synthesis by conditioning WaveNet on mel spectrogram predictions. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 4779–4783). `https://doi.org/10.1109/ICASSP.2018.8461368`

Simpson, A. J. R., Roma, G., & Plumbley, M. D. (2015). Deep karaoke: Extracting vocals from musical mixtures using a convolutional deep neural network. In E. Vincent, A. Yeredor, Z. Koldovský, & P. Tichavský (Eds.), *12th International Conference on Latent Variable Analysis and Signal Separation* (pp. 429–436). Liberec, Czech Republic: Springer.

Slizovskaia, O., Kim, L., Haro, G., & Gómez, E. (2019). End-to-end sound source separation conditioned on instrument labels. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 306–310). `https://doi.org/10.1109/ICASSP.2019.8683800`

Smaragdis, P., Fevotte, C., Mysore, G. J., Mohammadiha, N., & Hoffman, M. (2014). Static and dynamic source separation using nonnegative factorizations: A unified view. *IEEE Signal Processing Magazine*, *31*(3), 66–75. `https://doi.org/10.1109/MSP.2013.2297715`

Smith, B., & Sataloff, R. T. (2013). *Choral pedagogy* (3rd edition). Plural Publishing.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(1), 1929–1958.

Stoller, D., Ewert, S., & Dixon, S. (2018a). Adversarial semi-supervised audio source separation applied to singing voice extraction. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2391–2395). `https://doi.org/10.1109/ICASSP.2018.8461722`

Stoller, D., Ewert, S., & Dixon, S. (2018b). Wave-U-Net: A multi-scale neural network for end-to-end audio source separation. In *Proceedings of the International Society for Music Information Retrieval Conference* (pp. 334–340). Paris, France.

Stöter, F.-R., Liutkus, A., & Ito, N. (2018). The 2018 signal separation evaluation campaign. In Y. Deville, S. Gannot, R. Mason, M. D. Plumbley, & D. Ward (Eds.), *14th International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA 2018)* (pp. 293–305). Guildford, UK: Springer.

Subakan, C., & Smaragdis, P. (2017). Generative adversarial source separation. In *2018*

IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 26–30). Calgary, AB, Canada: IEEE.

Sundberg, J. (1987). *Science of the singing voice.* DeKalb, IL: Northern Illinois University Press.

Sundberg, J. (2006). The KTH synthesis of singing. *Advances in Cognitive Psychology*, *2*(2), 131–143.

Şimşekli, U., & Cemgil, A. T. (2012). Score guided musical source separation using Generalized Coupled Tensor Factorization. In *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)* (pp. 2639–2643). Bucharest, Romania.

Takahashi, N., Agrawal, P., Goswami, N., & Mitsufuji, Y. (2018a). PhaseNet: Discretized phase modeling with deep neural networks for audio source separation. In *Proceedings of Interspeech 2018*. `https://doi.org/10.21437/Interspeech.2018-1773`

Takahashi, N., Goswami, N., & Mitsufuji, Y. (2018b). MMDenseLSTM: An efficient combination of convolutional and recurrent neural networks for audio source separation. In *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)* (pp. 106–110). `https://doi.org/10.1109/IWAENC.2018.8521383`

Takahashi, N., & Mitsufuji, Y. (2017). Multi-scale Multi-band DenseNets for Audio Source Separation. In *2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (pp. 21–25). New Paltz, NY: IEEE.

Ternström, S. (2003). Choir acoustics: An overview of scientific research published to date. *International Journal of Research in Choral Singing*, *1*(1), 3–12.

Ternström, S., & Karna, D. R. (2002). Choir. In R. Parncutt & G. McPherson (Eds.), *The science & psychology of music performance: Creative strategies for teaching and learning.* Oxford, UK: Oxford University Press.

Tikhonov, A. N., Goncharsky, A., Stepanov, V. V., & Yagola, A. G. (1995). *Numerical methods for the solution of ill-posed problems.* In *Mathematics and Its Applications.* Springer Netherlands.

Uhlich, S., Giron, F., & Mitsufuji, Y. (2015). Deep neural network based instrument extraction from music. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 2135–2139). `https://doi.org/10.1109/ICASSP.2015.7178348`

Uhlich, S., Porcu, M., Giron, F., Enenkl, M., Kemp, T., Takahashi, N., & Mitsufuji, Y. (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)* (pp. 261–265). https://doi.org/10.1109/ICASSP.2017.7952158

Van Trees, H. L. (2002). *Optimum array processing.* In *Detection, Estimation, and Modulation Theory*: *Vol. IV*. New York, NY: Wiley-Interscience.

Vincent, E., Gribonval, R., & Fevotte, C. (2006). Performance measurement in blind audio source separation. *IEEE Transactions on Audio, Speech, and Language Processing*, *14*(4), 1462–1469. https://doi.org/10.1109/TSA.2005.858005

Vincent, E., Virtanen, T., & Gannot, S. (2018). *Audio source separation and speech enhancement.* John Wiley & Sons.

Wang, D., & Brown, G. J. (2006). *Computational auditory scene analysis: Principles, algorithms, and applications.* Wiley-IEEE Press.

Wang, D., & Chen, J. (2018). Supervised speech separation based on deep learning: An overview. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, *26*(10), 1702–1726. https://doi.org/10.1109/TASLP.2018.2842159

Ward, D., Mason, R. D., Kim, R. C., Stöter, F.-R., Liutkus, A., & Plumbley, M. D. (2018a). SiSEC 2018: State of the art in musical audio source separation – subjective selection of the best algorithm. In *Proceedings of the 4th Workshop on Intelligent Music Production.* Huddersfield, UK: University of Huddersfield.

Ward, D., Wierstorf, H., Mason, R. D., Grais, E. M., & Plumbley, M. D. (2018b). BSS Eval or PEASS? Predicting the perception of singing-voice separation. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 596–600). https://doi.org/10.1109/ICASSP.2018.8462194

Zhao, H., Gallo, O., Frosio, I., & Kautz, J. (2017). Loss functions for image restoration with neural networks. *IEEE Transactions on Computational Imaging*, *3*(1), 47–57. https://doi.org/10.1109/TCI.2016.2644865