NORTHWESTERN UNIVERSITY

Score-Informed and Hierarchical Methods for Computational Musical Scene Analysis

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science & Communication

By

Ethan Manilow

EVANSTON, ILLINOIS

June 2022

ABSTRACT

Score-Informed and Hierarchical Methods
for Computational Musical Scene Analysis

Ethan Manilow

Imagine sitting in a room listening to some friends play a song. Perhaps one friend is playing guitar, another playing bass, and a third is playing drums. The musical content in this scene is extraordinarily complex, yet it contains many types of structure that is easy for us to comprehend. For instance, even though we only hear the mixture of all the musicians playing together, we have a good idea of what each of the instruments sounds like by itself. Furthermore, we are able to understand the notes that each instrument is playing, giving us information about musical events are happening and when they occur. While it might be trivial for our human brains to decipher this musical scene and extract all of these structures, it is a difficult problem to make a machine do a similar analysis.

These are just some of the problems that fall under the umbrella of Musical Scene Analysis, a subset of machine learning that specifically targets the analysis of music as raw audio data. The goal of this field is to identify and locate the key elements of a musical scene that humans attend to when we listen to songs. As a matter of fact, the structures I mentioned in the previous paragraph both have corresponding Musical Scene Analysis tasks. Estimating what each of the instrument sources sounds like in isolation is a task called *source separation*, and is a fundamental task in Musical Scene Analysis. Determining what notes musicians play and when they occur is called *Automatic Music Transcription (AMT)* and is also a fundamental problem in the field. Making progress in solving these two tasks can enable a wide array of applications, ranging from search, retrieval, and analyses of large scale musical corpora (e.g., Spotify, Apple Music, YouTube) to creative uses for helping musicians.

In this dissertation, I extend the capabilities of Musical Scene Analysis systems by creating source separation and AMT systems that are (a) able to support more instruments, and (b) are more controllable than prior work. I do this by introducing **Score-Informed** (i.e., using musical score data as a conditioning input or as a training target) and **hierarchical** methods to analyze musical scenes.

Making systems that are able to support *more instruments* is important because many musical scenes contain multiple instruments simultaneously. Yet, most prior AMT systems are trained on isolated piano recordings, and thus cannot transcribe multiple instruments in a mixture. Furthermore, many prior separation systems only support a small number of fixed source types, modeling each source independently. Here, I extend the capabilities of existing separation and transcription systems. I do so, first, by producing a combined separation and AMT system that is able to simultaneously separate and transcribe up to five instruments in a mixture–many more instruments than most AMT systems typically consider. Additionally, I reframe the source separation problem as hierarchical, showing how a system is able to learn relationships between different source types to efficaciously separate much more fine-grained source types than most previous systems.

To make systems *controllable* for a given input example, the goal is to create separation systems whose output can be altered without costly process of retraining it from scratch. In other words, I want to make source separation systems that are steerable at inference time. In this dissertation, I accomplish this in two ways. The first way is through the hierarchical lens, where I introduce a Query-by-Example mechanism that enables it to change which source it separates at inference time, giving an end-user control knob which source gets separated. The second method of control emerges from the notion that the Musical Scene Analysis tools we build will never be perfect, therefore it is important to have the ability to correct these systems if they make mistakes. From this idea, I revive the paradigm of Score-Informed separation–or using musical score data as a conditioning input to the separator–and rejuvenate it for the deep learning era. I show how a Deep Score-Informed Separation system allows making note-level edits to the source estimates at inference time, proving the ability to make fine-grained edits than previous deep learning-based separation systems.

For two of the three projects in this document, I provide mockups of user interfaces built atop of the techniques proposed here, as a means of envisioning how these systems might work in the hands of musicians and artists. I hope the work in this dissertation paves the way for a new generation of work that emphasizes flexibility and controllability, for the sake of making systems that can impact the workflow of end-users.

Acknowledgments

The first person I want to thank is my advisor, Bryan Pardo. In my time at Northwestern, I have felt fortunate to call him my teacher, mentor, collaborator, and friend. He has deftly guided me through the highs and lows of PhD life and taught me immensely throughout the journey. His tireless efforts have helped me grow from an aspiring young scientist to a productive member of the research community. It is hard to overstate how much of an impact Bryan has had on my life. I will forever be grateful that he decided to take a chance on me many years ago. I look forward to many more collaborations, conversations, (in-person) conferences, lunches, and much more! Thank you.

I want to thank the rest of my thesis committee: Jessica Hullman, Doug Downey, and Jesse Engel. I deeply appreciate the time and effort that each of them have spent giving me feedback throughout the process of creating this dissertation. Furthermore, I want to thank Darren Gergle for his mentorship throughout the years and Mike Horn for his time spent on the first iteration of this dissertation. Finally, I would like to thank Nick Diakopoulos, who generously helped me find a middle ground for this thesis when my research topic strayed from the usual TSB area.

Additionally, I have to thank the mentors who I have been lucky to learn from at my internships. An enormous thank you to Jonathan Le Roux and Gordon Wichern, who nurtured my curious mind at MERL and provided me with many patient answers to my questions. I will always look back at my time at MERL as a time of great personal and professional growth. Jonathan and Gordon's mentorship led to the work in second chapter of this dissertation. Also, a gigantic thank you to my mentors at Google, Jesse Engel and Curtis "Fjord" Hawthorne. Even through a pandemic, Jesse and Fjord helped me stabilize a sometimes-meandering research agenda, which ultimately ended up being the last chapter of this dissertation. Also, thanks to the rest of the Magenta team (Anna, Ian, Josh, Adam, Sehmon, Yusong, Rigel, Gautham, Vibert, Hallie, Ryan, and more) for teaching me and welcoming me into the team under such stressful global circumstances.

I would like to thank my current and former labmates: Mark Cartwright, Bongjun Kim, Prem Seetharaman, Fatemeh Pishdadian, Max Morrison, Hugo Flores García, and Patrick O'Reilly. Thank you for the countless conversations about research, music, and life. I feel extremely lucky to count many of you as

collaborators and all of you as friends. Getting through the trials of graduate school would have been impossible without having your friendship to rely on. Also, thank you to the many undergraduate and Master's students that I have had the privilege of working with throughout the years: Corey Grief, Ben Kalish, Daniel Felix-Kim, Nathan Shelly, Andreas Bugler, Aldo Aguilar, Noah Schaffer, Boaz Cogan, and Erika Rumbold.

I would be remiss not to thank my parents. Thank you for your undying love and support through all my years on this blue marble. Thank you for schlepping me to the distant reaches of suburbia for guitar lessons multiple times a week in high school. Thank you for providing me with an excellent education and supporting my crazy decisions to double major in undergrad, then purposely move to Texas, and then go back for more school! It is hard to envision my life turning out half as wonderful as it did without your encouragement.

I also want to thank the rest of my family. I am grateful to have a large support system consisting of my sister and many, many terrific cousins, aunts, and uncles. I would like to especially thank my late grandparents, Marshall, Sally, Elaine, and Floyd. I would not be here without the many gifts you have given me; I wish you could all be here to witness this moment!

I feel fortunate to have a long list of terrific friends that I need to thank. In no particular order, thank you to: Jared, Jeff, Woody, John, Allie, Noam, Jenn, Erin, Joe, Megan, Alex, Veronica, Ben, BenJo, Charlie, Brad, Max, Derek, Maranda, Rachel, Becky, Maria, Sam, Ali, Jake, Chris, Jaren, Matt, Wyatt, Madhav, Ryan, Scott, Christine, and Jeremy. Thank you for being an endless source of laughter, curiosity, and fun. I feel so lucky to have all of you in my life. I look forward to many more years of friendship!

The work described in this dissertation was generously supported by Mitsubishi Electric Research Labs (MERL), National Science Foundation award CRI-1730689 via the use of Mystic (Programmable Systems Research Testbed to Explore a Stack-WIde Adaptive System fabriC) at the Illinois Institute of Technology, and Google Research.

Finally, thank you to my partner, Rebecca, for tagging along with me on this wild journey through grad school and life. Your love, support, and understanding have been everything to me. Thank you for being my cheerleader, comedian, ballast, confidant, and everything in between. Thank you for being my best friend. I cannot wait to go on many new adventures with you! #BOTY2022

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Music is a fundamental aspect of the human experience. Every human culture on Earth makes music [188] and some researchers believe that musicality is embedded within us at a biological level [120], some going as far as stating that music might be a crucial element in the evolution of abstract thought [40, 102, 187]. Our relationship with music can be profoundly personal, emotional, spiritual, and autobiographical. In modern life, music accompanies us in recreation and in work, in solitude and in company, in religious ceremonies and on everyday trips to the grocery store.

Our history shows that music is deeply intertwined with our development of technology. In fact, some of the key pieces of material evidence that our paleolithic ancestors were musical are flutes carved from animal bones that date back tens of thousands of years [38]. While these flutes are perhaps among the earliest instruments, we have since created a rich lineage of musical tools. We have stretched animal skins to make drums, tensioned strings across wood to make lutes and violins, devised sophisticated mechanical systems to make the pianoforte play loud and soft, invented intricate tubing for saxophones and trumpets, experimented with electricity to record, amplify, and affect all of the noises we make, and coerced electrical signals to synthesize nearly mathematically perfect waves starting with the Moog synthesizers of the 1960s. The story of humans as musicians is enmeshed with the story of humans as technologists.

Our technological creations are more than the sum of their physical components, though. Each new instrument concept brings with it new conceptualizations of music itself [122, 171, 172]. We are able to create anew within the unique frameworks that each new musical instrument embodies. New instruments enable entirely new modes of musical expression; many of the instruments listed above play an indispensable role in forging the identity of musical genres. It is hard to imagine a symphony without a violin, a marching band without horns, or heavy metal music without distorted guitars. These instruments are embedded into these these genres at their core, and they provide the foundational color that gives the resulting music its character.

The development of recorded sound fundamentally altered our relationship with music. While the initial delight of recording was that it enabled a performance to be captured and then re-experienced by a listener,

music producers discovered how to use the act of recording as a means of creativity its own right [18, 68, 238, 299, 309]. The recording studio *itself* had became an instrument that could be played and manipulated, bringing with it a reconceptualization of music-making that was drastically more powerful than any of our previous instruments.

Of the many hallmarks in the history of recording technology, one of the more significant advances is the computerization of recording. Digital Audio Workstations (DAWs) have not only democratized the use of sophisticated recording techniques, but they have also enabled completely new ways to manipulate recorded sounds [6, 119, 189]. Early experimental recording techniques pioneered by Pierre Schaeffer [237] (and more familiarly, The Beatles [310] and The Beach Boys [93]), morphed into new techniques centered around sampling (i.e., reusing a portion of an old song to make a new song) [193, 227] and remixing [271]. Sampling and remixing have become mainstay techniques used in many popular music genres, like hip hop [45, 245]. As more technologies for making music become available, we find more and more ways to exploit these technologies to enable new modes of creative expression.

One way we can imagine music-making is as a conversation. Musicians often feel as though they are communicating when they play with other musicians [235], but there is also a conversation happening with their instrument [90, 132, 158, 171, 205]. With analogue instruments the musician-instrument conversation is very one sided; a skilled musician has an intimate relationship with the instrument, knowing how it will respond to every slight variation, however the instrument is never talking back of its own volition. The instrument is static in the hands of a skilled performer, rarely diverging from a known path of interaction. Recording can be seen as an extension to this conversation, whereby a performance is reflected back to the musician. Just as a conversation occurs when they play with another person, recording enables the musician to have a conversation with (a past version of) themself.[1] The recording studio enables musicians to add edits and effects, perhaps creating a 'fun house mirror' reflection of themselves, to extend the reflection metaphor. The recording studio is still static in this sense; it reflects in whatever way you tell it to but it does not understand the content you ask it to reflect. Just as recording creates a less static conversation than a instrument alone, computing further enhances the dynamics of the conversation. Current DAWs expand these fun house reflection capabilities of analogue recording, making many things easier and adding

[1]The great pianist Bill Evans noted as much on his album *Conversations with Myself* where he recorded three overdubbed versions of himself playing jazz standards. Recommended listening while reading this dissertation!

new levels of interactivity. However, this technology still cannot have an in-depth musical conversation in the same way that another musician can.

To enable a new kind of music-making, machines are missing a key ingredient: the ability to *listen*. Humans have a remarkable ability to listen to and understand music, being able to innately parse the myriad structures in a musical scene. For instance, imagine listening to our friends play a song. Imagine a guitar, a piano, and a drummer. Even though we cannot hear the guitar by itself, we have a good idea about what the guitar sounds like by itself if the piano and drums were muted or silent. Similarly, it is trivial for us to attend to the notes that each of the instrument plays; the notes are what make up the sound of the instrument. Not to mention our ability to understand higher-level qualities about the song: things like tempos, beats, song forms, genres, moods, etc, are all related to these lower-level characteristics like the sounds of instrument and note events. Our brains are able to do all of this advanced processing even when many overlapping things happen simultaneously! When listening to our friends, we do not just hear the guitar, but all three instruments. Furthermore, we are able to adapt to whatever acoustic environment that we are in; we still recognize the music as our friends' song whether they perform in a concert hall or in a drab basement, but the acoustics of these places are vastly different. Our ability to parse all these things is second nature, and being able quickly understand them is part of what drives our ability to have sophisticated musical conversations with other musicians.

These analogous listening processes are not so straightforward for machines. To do them, machines must operate starting from a raw audio signal. Audio signals are extremely high dimensional (e.g., a 3 min song has ≈8M samples), the relevant structures that span many orders of magnitude (i.e., note onsets occur at the order of milliseconds and musical forms (e.g., AABA, sonata form, verses, chorus, etc) can unfold over a few minutes), they often contain multiple relevant, overlapping sounds heard simultaneously (e.g., each of our individual friends in the band), and are highly sensitive to the particularities of the recording environment (e.g., room reverberation). Making machines that can emulate these listening processes is a foundational step towards making machines that can contribute to a musical conversation in a worthwhile manner. To do this, we need machines that can analyze musical scenes.

Fittingly, Musical Scene Analysis is the name for the field of artificial intelligence research that aims to identify and locate some of these key musical structures that humans attend to when they listen to a musical scene. Musical Scene Analysis can be viewed as a subfield of Machine Listening, an area that focuses on developing machine learning systems that can solve a wide range of problems related to sound [13, 14, 229].

Musical Scene Analysis, often overlapping with the area of Music Information Retrieval (MIR) [54], has been the subject of many decades of research, with recent advances coming in the form of deep learning and neural network research. Examples of the tasks that Musical Scene Analysis systems attempt to solve include fundamental frequency estimation [10, 67, 140], beat tracking and estimation [42, 62], melody extraction [230, 231], query by humming [41, 88, 150], genre classification [236, 277], mood classification [152], cover song identification [242], musical instrument identification [76], repairing damaged musical signals [37, 177], and more. Successful Musical Scene Analysis systems are already beginning to power the next generation of music production [2, 3, 125], education [258, 308], and recommendation tools [206, 281], and are enabling new means of expression, learning, and discovery for musicians and music listeners.

Musical Scene Analysis has the potential to create musical tools that can listen to and understand the music that we are making. Instead of being passive participants in the music-making conversation as before, a new generation of Musical Scene Analysis-laden tools could act as engaged interlocutors, being an instrument that acts more as another musician rather than just another tool. If handled with care, the machine's ability to listen could make these new systems a fun and powerful new addition in humankind's ever growing arsenal of musical tools, which in turn could be a new engine of musical expression.

In this dissertation, I will focus on two Musical Scene Analysis tasks in particular, specifically Musical Source Separation [174, 181, 284], and Automatic Music Transcription [7, 8].

Briefly, Musical Source Separation is the task of isolating individual musical instruments (i.e., sources) from a mixture, effectively muting unwanted sources (see Section 1.2). As I will discuss in the next section, source separation has the potential to help many downstream tasks [80, 100, 101, 108, 121, 126, 191, 246, 262, 294], but also is a useful technology in its own right, helping artist and musicians remix and clean up recordings [36, 60, 190, 199, 305]. The open problems with existing separation systems that I tackle in this dissertation are the fact that (a) most systems only work for a limited, fixed set of predefined sources, and (b) making alterations to the output of current systems is extremely tedious.

Automatic Music Transcription is concerned with converting a raw audio signal into a human-readable symbolic music format (see Section 1.3). The resulting symbolic format is a much smaller, more distilled representation of the musical content from the raw input audio data. Because of this, AMT systems have a broad set of potential use cases [5, 6, 9, 73, 129, 143, 159, 173, 200, 210, 217, 226, 242–244, 276, 286, 290]. One issue with most existing AMT systems is that they are only able to transcribe audio recordings

that contain one instrument (usually piano), and omit the case where a recording might have multiple instruments playing simultaneously. In Chapter 3 of this dissertation, I approach this problem by drawing an analogy between multi-instrument AMT and source separation. While all of the projects in this dissertation make contributions to source separation, only Chapter 3 makes direct contributions to Automatic Music Transcription, while Chapter 4 uses on the work of Chapter 3 to extend source separation systems.

Throughout the rest of this chapter I will devote many pages to explaining these tasks in greater detail, but what is important to know now is that these two tasks are fundamental problems in the area of Musical Scene Analysis. They are at the heart of Musical Scene Analysis because many recordings contain multiple instruments sounding simultaneously (Musical Source Separation) and converting musical audio to symbolic representation (Automatic Music Transcription) makes raw audio data more human-readable and parsimonious. Both of these tasks together provide a route for creating a rich understanding of musical scenes. Modern approaches to these tasks include a blend of techniques from the fields of audio signal processing and machine learning. Because Musical Source Separation and Automatic Music Transcription are core problems in Musical Scene understanding, solving them is a crucial step towards making machines that can be more active participants in our music-making processes.

1.1. Contributions of this Dissertation

This dissertation will make three main contributions to the Musical Scene Analysis literature:

- **In Chapter 2, I will demonstrate the first audio source separation system that exploits the hierarchical taxonomic relationships between different classes of sound sources.** The resulting system is uses the hierarchy to separate many more source types than are typically considered, and can maintain most of its performance when training on only 10% of available leaf data. The hierarchical framing, in turn, enables a fundamentally new approach for end users to direct the system to focus on separating a desired source at inference time via a query-by-example interaction paradigm.
- **In Chapter 3, I will demonstrate the first combined source separation and automatic music transcription system, which simultaneously separates and transcribes multiple instruments in an input mixture.** The proposed jointly-trained multitask system produces better results than dedicated single-task systems and analyzes a musical scene in a more holistic manner. It begins to solve a

longstanding issue with AMT systems by transcribing multiple concurrent instruments in a single recording. Importantly, it is a demonstration that these two tasks can be symbiotic and that considering them jointly can be beneficial.

- **In Chapter 4, I will demonstrate the first score-informed separation system of the deep learning era.** This system conditioned on musical score data when it makes source estimates. In doing so, it yields better separation performance than a baseline system. More importantly, I will show how the score-informed setup enables note-level editing of separation estimates at inference time, which provides another avenue for end users to make adjustments to the system's output post-hoc.

1.2. Musical Source Separation

In general, Source Separation is the task of decomposing an auditory scene that contains multiple sonic elements into a set of isolated constituent elements, called *sources*, such that each item in the set only contains sound for the desired source [174, 181, 284]. An example of source separation in a general auditory scene might be to remove an errant car honk from a radio interview. Put succinctly, the goal of Source Separation is to effectively *"mute"* everything except for the desired source(s).

Source Separation is a fundamental problem in Machine Listening, which, once solved, can enable more robust downstream tasks. Within music, the goal of a source separation system is to isolate the audio of a (set of) relevant musical instrument(s) from other instruments that might be heard in an auditory mixture. An example of musical source separation would be isolating a bass guitar from the rest of a rock band or isolating a violin soloist from the rest of the orchestra. It is extremely common that musical recordings have more than one instrument source playing simultaneously, and, thus, musical source separation is crucial in enabling a laundry list of downstream music information retrieval (MIR) tasks that are facilitated by listening to sources in isolation. Specifically, music source separation has successfully been used in conjunction with: lyric and music alignment [80], musical instrument identification [108], lyric recognition [191], automatic singer identification [121, 246, 294], vocal activity detection [262], audio repair [177], fundamental frequency estimation [126], and understanding the predictions of black-box audio models [100, 101]. Some recent systems for remixing [305] and instrument manipulation [36] have used source separation implicitly as means to adjust a single source in a mix, forgoing an explicit step that outputs sources directly. Source separation's usefulness makes sense intuitively; many of these tasks become easier if the musical scene is first split into

constituent parts because by "muting" extraneous sound sources the downstream algorithm can better focus on the specific audio most relevant for its task.

Additionally, creative uses in video and music are both important direct applications for source separation. Music source separation is used within creative settings to isolate individual instruments for remixing music in the case that a user does not have access to isolated recordings of each instrument, which is in many scenarios. In multimedia editing and post-processing, source separation is used for upmixing or remixing vintage or imperfectly recorded movies or audio to higher quality [199]. Notably, source separation technology was the invisible hero of Peter Jackson's 2021 documentary *The Beatles: Get Back*. Built from scrapped archival materials from their 1969 recording sessions, the Jackson documentary depicts a fly-on-the-wall view of the The Beatles as they recorded *Let It Be*. Much of the audio recorded in the 1969 archives was recorded with a single microphone, making it difficult to hear some instruments clearly, and begriming important conversations with the cacophony of errant guitar, bass, and drum noodling. As such, Jackson's team turned to source separation to isolate each source and remix the audio, proving to be a crucial part of the restoration process of the old material [60, 190]. In the process, the documentary became a high-profile demonstration of how source separation can directly enable a better listening experience.[2] Given its many potential direct and downstream applications, music source separation is a crucial aspect of Musical Scene Analysis and a worthwhile endeavor in its own right.

1.2.1. A Primer on Audio Signal Processing and Source Separation

In this section, I will provide a more in-depth view of the mechanics of source separation. As I do so, I will also introduce all of the audio signal processing basics required to understand current source separation approaches and the rest of this dissertation.

In the digital domain, monophonic (i.e., single channel) audio can be thought of as a one dimensional array of discrete-valued numbers $x(t)$, where the value of x at time t is the amplitude of the signal at that instant in time. When digital audio recordings are made, audio signals are sampled at a uniform rate, meaning the absolute time difference between two neighboring indices t and $t+1$ is constant for any value of t. This is called the *sample rate*. Similarly, the values of $x(t)$ are quantized to the nearest value within a range of linearly uniform discrete steps. This is called the *bit depth*. A standard value for the sample rate are 44.1 kilohertz (kHz), or 44,100 times a second and a standard value for bit depth is 16 bits. Audio encoded

[2] See a short video of the results here: `https://youtu.be/sBR5VNWJZmw`

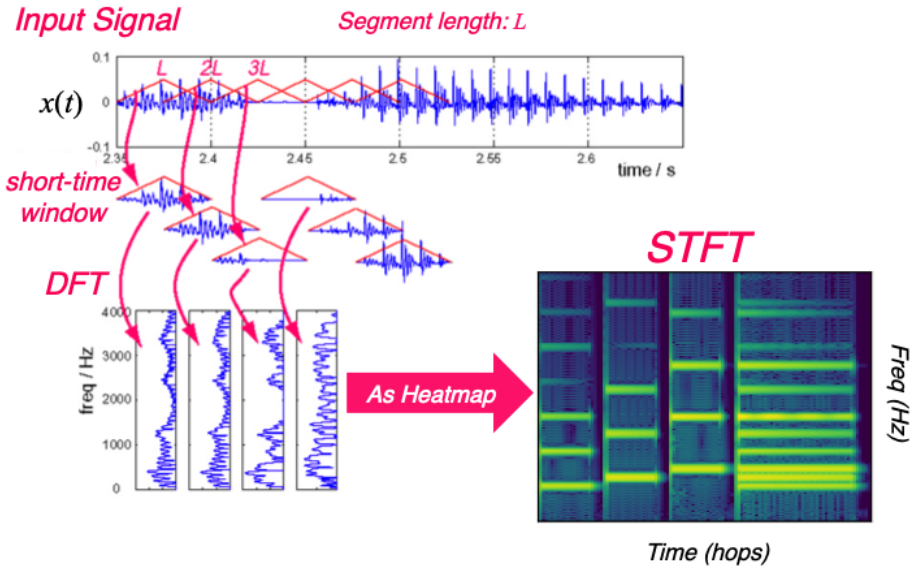


Figure 1.1. Calculation of a Short-Time Fourier Spectrum. An input signal $x(t)$ is split into overlapping segments, called *short-time windows*, and a discrete Fourier transform (DFT) is computed on each of these windows, and these DFTs are stacked to make a 2D complex-valued array with frequency and phase information at for each window. Phase information is omitted here for clarity. Image adapted from [208].

at 44.1 kHz in 16 bits is called "Compact Disc Quality." There are many other ways to represent audio digitally (including expanding this definition to include stereophonic audio), but this definition is suitable for understanding source separation. Audio signals like this are sometimes called "time series" signals or "waveforms" (shown in the top blue plot labeled "Input Signal" in Figure 1.1).

An audio mixture is a signal that is a composite of multiple individual signals. It may be a natural mix, such as the result of using just one microphone to record multiple sound sources (e.g., a single-microphone recording of The Beatles, described above). It may also be an artificial one, made in a studio where multiple individually-recorded tracks are mixed together in digital audio software. A common way to formulate source separation is to assume that an audio mixture signal $x(t) \in \mathbb{R}^{1\times T}$, for a signal of duration T, contains N sound sources $s_1(t), s_2(t), ..., s_N(t)$, such that:

$$x(t) = \sum_{i=1}^{N} s_i(t). \tag{1.1}$$

Addition here is element-wise and every source has is the same length as the mix, $s_n(t) \in \mathbb{R}^{1\times T} \forall n$. The goal of source separation is to make every source estimate, $\tilde{s_n}(t)$, as close as possible to the corresponding ground truth source, $s_n(t)$.

A source separation system is called a "blind" system if it is expected to estimate the sources $s_n(t)$ given no other information other than the mixture, $x(t)$. On the other hand, source separation systems that have access to additional information, such as a musical score or the spatial locations of the microphones, are referred to as "informed" separation systems. Relevant to this work is a class of source separation systems that use musical scores as additional input for separation, which are called *score-informed* separation systems. Although score-informed separation systems have fallen out of fashion with the popularization of deep learning-based methods[3] for source separation, in this dissertation I will reintroduce the Score-Informed concept with deep learning. I will situate the work in this dissertation within this historical context in Introduction Section 1.4.2 and in Chapter 4.

Historically, many source separation approaches compute a complex-valued Short-Time Fourier Transform (STFT) from the mixture, $x(t)$, as a preprocessing step. An STFT is used to model the frequency and phase content of the audio signal as they change over time. An STFT is calculated by computing a discrete Fourier transform (DFT) of a short segment of the audio clip with length l (in samples), and then shifting forward in time ("hopping") by a fraction of the window length, before calculating the next DFT. These short segments have length l and are referred to as "short-time" windows (these are the "Short-Time" in the name "Short-Time Fourier Transform"). The DFT of each short-time window calculates the frequencies active in the audio *in that window*. The process of segmenting, computing a DFT, and hopping is repeated for the length of the time series audio signal. By collecting the set of resultant DFT parameters and keeping track of when they occur in the audio signal, we can stack the DFT outputs sorted in time to make an STFT. These steps are shown in Figure 1.1. The result, $\text{STFT}(x) = X(f,t) \in \mathbb{C}^{F\times T}$, is a complex-valued, two dimensional matrix with frequency bins f and time bins t. The "hops" that shift the short-time window are usually fractions of the length of the window, usually $\frac{1}{2}l$ or $\frac{1}{4}l$. The length of the short-time window, l, is in samples, and typically has values of 512, 1024, or 2048 samples, which are on the order of tens of milliseconds. Note that while t is still time, in the STFT, $X(f,t)$, t is indexed in terms of hops (i.e., fractions of l, called "frames") rather than in samples when the signal is a time-series. STFTs can be converted back

[3]Whether supervised neural network source separation systems are considered "blind" or "informed" is a matter for discussion. A purist might argue that the supervised training set disqualifies neural network systems from being truly "blind."

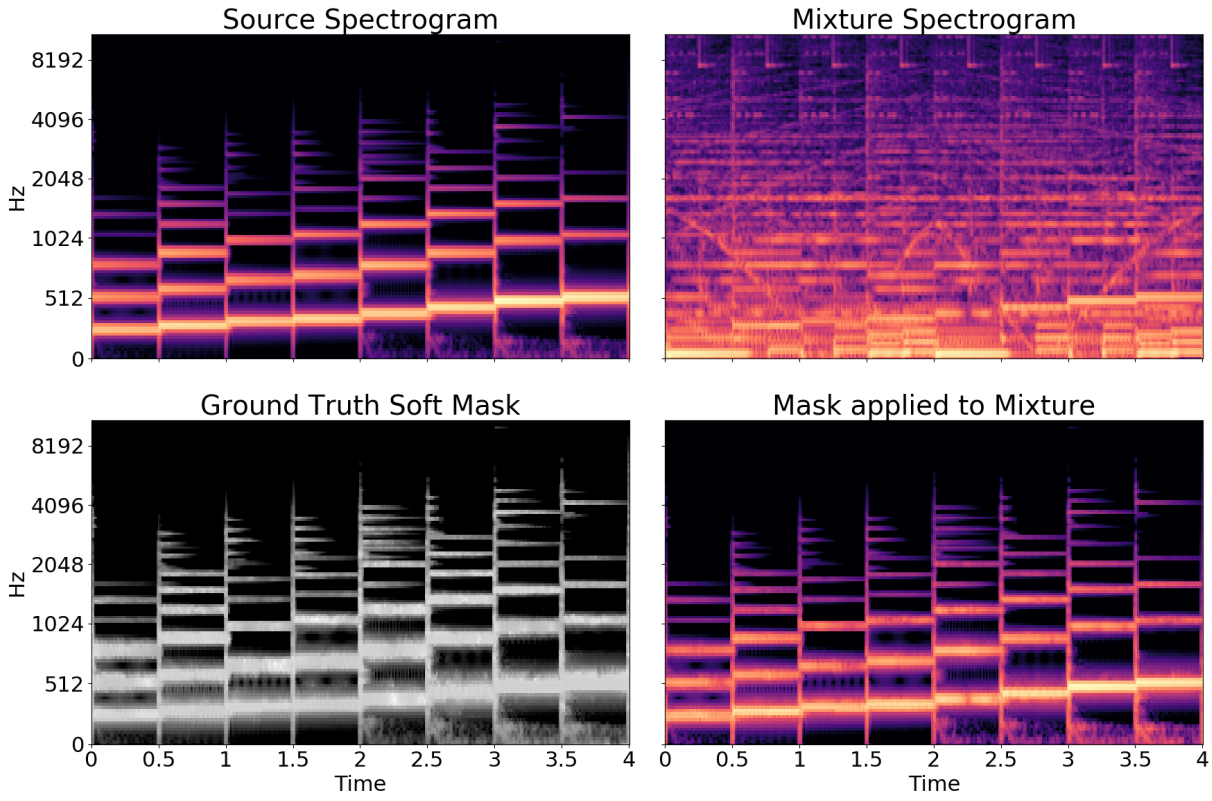


Figure 1.2. Applying a mask to a spectrogram. The top left figure shows a spectrogram of an isolated source (e.g. a piano). When that same source is mixed in with many other instruments (top right), it becomes extremely difficult to discern on a spectrogram. However, getting access to a high-quality mask of the desired source (bottom left) can produce a high-quality source estimate when the mask is element-wise multiplied by the mixture spectrogram (bottom right). The mask is a real-valued 2D array that has values in the interval [0.0, 1.0] such that when element-wise multiplied by a mixture time-frequency representation, the result is an estimation of a source. The louder a source is at a particular (t, f) point, the closer that value is to 1.0 and vice versa. In this case, the mask of the piano (bottom left) was known a priori from the ground truth piano spectrogram (top right), but in the general case of mask-based source separation, the mask must be estimated.

to a time-series representation losslessly by taking an inverse STFT (iSTFT). STFTs are frequently referred to as "time-frequency" representations, and individual elements of the 2D matrix are referred to as "time-frequency" bins or (t, f) bins for short. It is common to denote time-series signals with lower case letters (e.g., $x(t)$) and time-frequency with capital letters (e.g., $X(t, f)$).

The STFT is commonly used because frequency is an important component of human hearing; the STFT is a way to explicitly encode frequency information. Other time-frequency representations often used are the *magnitude spectrogram*, $||X||$, *power-spectrogram*, $||X||^2$, and *log-spectrogram*, $\log(||X||)$, where $||\cdot||$, $||\cdot||^2$, $\log(||\cdot||)$ refer to applying an absolute value, squaring, or logarithm to each element in the matrix,

respectively. Most commonly used for processing and visualization is the log-spectrogram, shown at the bottom right of Figure 1.1. The log spectrogram represents the signal magnitude at each time frequency point on a logarithmic scale. This is because most sounds contain large differences in amplitude and human volume perception is, itself, logarithmic. Another commonly used used variant is the *log-mel spectrogram*, which applies an element-wise log, but the frequency axis is spaced semi-logarithmically, rather than linearly. Having logarithmically spaced frequency bins mirrors human frequency perception, which is also logarithmic. A note on terminology: the term "spectrogram" is used as a short-hand for the magnitude spectrogram, power-spectrogram, log-spectrogram, or log-mel spectrogram depending on the context (usually only one of the above is used in a given context).

Given these additional formulations, a common approach to source separation is to find a set of estimated "masks", $\tilde{M}_1, \tilde{M}_2, ..., \tilde{M}_N$ for $\tilde{M}_n \in [0.0, 1.0]^{F \times T}$ that can be applied to the mixture STFT to get source estimates. Each mask corresponds with only one source, so in order to separate N sources, we need to make N masks. These masks are real-valued 2D arrays that are the same shape as the STFT, where each element is between 0.0 and 1.0. The elements in the mask distribute energy from the mixture at a given time-frequency (t, f) bin to each source. A louder source at a given (t, f) bin is assigned a value closer to 1.0, and a softer source at the same bin is assigned a value closer to 0.0. There is one mask per source and each mask corresponds to a desired source such that estimated source spectrogram $|\tilde{X}_i|$ is calculated like

$$|\tilde{X}_i| = |X| \odot \tilde{M}_i, \tag{1.2}$$

where $\odot$ means element-wise multiplication [288]. This process is shown in Figure 1.2, where the a mask (bottom left) is calculated from the known ground truth source spectrogram (top left) and the mask is applied to the mixture (top right) to get an estimate of the source (bottom right). In its most distilled conceptualization, the task of source separation becomes the task of trying to estimate a mask for each desired source.

One detail that I have omitted in the last paragraph is that while a spectrogram, $|X| \in \mathbb{R}^{F \times T}$, is real-valued, the STFT, $X(f, t) \in \mathbb{C}^{F \times T}$, is complex-valued, i.e., each (t, f) bin has a magnitude and phase component. Both magnitude and phase are required to convert an STFT back into a waveform (and a waveform is needed to actually *hear* the signal). However, the masks discussed above, $\tilde{M}_i \in [0.0, 1.0]^{F \times T}$ are real-valued, and thus are only applied to the magnitude components of an STFT, e.g. $|X| \in \mathbb{R}^{F \times T}$.

Phase is complicated[4] for many reasons; when we listen to audio, some phase errors stick out (like phase discontinuities) and some are imperceptible (like initial phase offsets) [66, 86]. In order to mitigate the issues involved with phase when dealing with real-valued masks being applied to real-valued spectrograms, we can employ a trick: use the phase from the mixture's complex-valued STFT. In other words, the phase of each source estimate is added as a post-processing step by copying it directly from the mixture STFT [86]. All of the source separation models in this dissertation will produce masks on real-valued mixture spectrograms and will use this trick of simply copying over the mixture phase for the model's source estimates.

Although copying the phase from the mix can potentially lead to perceptual quality issues [138], in many situations the performance of the systems are bottlenecked by producing high quality (magnitude) masks. In other words, it might be more impactful on perceptual quality if the model produced a better mask than if it estimated the phase perfectly (of course, doing *both* is most desirable). Regardless, this simple trick is usually good enough to produce high quality estimates and is used by many state-of-the-art separation systems from the past few years [111, 265, 268, 269]. A budding trend is to have a network predict the source's phase alongside the magnitude, which has lead to state-of-the-art performance in some recent systems [34, 35, 43, 148]. It is also possible, though not as popular, to approximate the phase as a post-processing step using various signal processing algorithms [95, 96] or even incorporate these so-called phase estimation algorithms into the network's training and inference procedures [153, 293, 298].

It has also become popular for neural network systems to forego time-frequency representations altogether, inputting and outputting waveforms directly. These systems are sometimes referred to as "end-to-end" systems. Instead of explicitly calculating an STFT (or similar) as a preprocessing step, these systems use the waveform $x(t) \in \mathbb{R}^{1\times T}$ directly as input to the network and output a set of estimated sources directly as waveforms, $\tilde{s_1}(t), ..., \tilde{s_N}(t)$ for $\tilde{s_i}(t) \in \mathbb{R}^{1\times T}$. The first layers of such systems implicitly learn a transformation of the waveform data in a similar manner to the STFT, however the parameters of this transform are not fixed before-hand as in the STFT, but learned from the data.[5] Whereas mask-based spectrogram models must employ special tricks to represent phase post-hoc, as in the previous paragraph, these end-to-end models learn how to represent phase implicitly such that no special post-processing phase tricks are required.

[4]See `https://source-separation.github.io/tutorial/basics/phase.html`.

[5]The DFT and inverse DFT (iDFT), part of the STFT/iSTFT as described above, define a particular *filter bank* used to transform the signal to and from a time-frequency representation [256]. Filter banks encode an input signal as multiple frequency components (usually 512, 1024, 2048, etc), with each centered around a single frequency of interest. These components make up the frequency axis in an STFT (or spectrogram). The process of computing both the DFT and iDFT is differentiable, enabling both processes to be incorporated into neural network training [32, 33, 35, 153, 298]. Recent work has thus generalized the special DFT/iDFT filter banks either by learning a set of parameterized filter banks [48, 209], or by using a convolutional layer learn the filter bank transformations directly from the waveform [169, 170, 278].

Recent work in speech separation has shown that end-to-end systems are capable of surpassing the theoretical maximum performance of mask-based spectrogram techniques (evaluated on canonical speech separation datasets according to objective measures of quality, discussed in Section 1.2.3) [29, 167, 169, 266]. However, this impressive feat has yet to be replicated with end-to-end models trained on music, and, therefore, the set of current state-of-the-art music separation networks contains both mask-based spectrogram models and end-to-end waveform models.

1.2.2. Datasets for Source Separation

Data is an important component of any machine learning system. Thus, in order to use machine learning to do source separation, we require audio data consisting of mixtures paired with the isolated recordings of the sources that were combined to produce that mixture. For most recent systems, the de-facto standard dataset for benchmarking is the MUSDB18 [224] dataset, which is composed of 150 full-length songs that contain mixtures of live musicians playing their instruments, recorded and mixed together in a studio setting.

Despite being the most common dataset used for source separation over the last few years, MUSDB18 has some shortcomings that limit its usefulness for the goals of this dissertation. The first is that MUSDB18 defines four source categories: "Voice," "Bass,", "Drums," and "Other." The latter source, "Other" is a catch-all category for any source that does not easily fall into the first three categories. This means that guitars, pianos, saxophones, violins, synthesizers, didgeridoos, zithers, and any other non-bass/voice/drum source type gets assigned to be a part of "Other." Because these are all lumped into one source type, this also means that you cannot train a system to separate any one "Other" instrument from a second "Other" instrument by training a system using MUSDB18; it is impossible to use a system trained on MUSDB18 to separate a mixture of a guitar and piano. As I will discuss in Chapter 2, using musical instrument hierarchies can help neural networks produce better separation estimates, but an important part of using such a hierarchy is having enough instrument types to meaningfully populate it. MUSDB18's four source types are not sufficient in this regard.

The second issue is that, because MUSDB18 contains recordings of live musicians, it does not come with aligned transcription data. I will delve into Automatic Music Transcription further in Section 1.3, but, briefly, transcription data gives the precise timing of note events (e.g., note onsets, note pitches, instrument ID, etc) such that it can be aligned with the corresponding audio signal of that performance. Having transcription

data is important for training multi-task separation and transcription systems like Cerberus in Chapter 3, as well as score-informed separation systems as I do in Chapter 4.

Therefore, to accomplish the goals set forth in this dissertation, I will use the Slakh2100 dataset [184]. In 2019, I created the Slakh2100 (or Slakh) dataset for use with musical source separation and/or automatic music transcription systems. There are some key differences between Slakh and MUSDB18. The first is that, rather than originating from recordings of live musicians, the audio in Slakh is all synthesized from symbolic note data (specifically, MIDI data from the Lakh MIDI Dataset [221], a dataset of ≈180,000 MIDI files was scraped from the web). This is beneficial in some aspects and detrimental in others.

The benefits of Slakh are that because the audio data is synthesized from such a diverse wellspring of MIDI data, it is easy to generate an ample amount of data that can match a wide spectrum of needs. For instance, has a lot more source types than MUSDB18; Slakh has 34 instrument sources (i.e., no "Other" source) to MUSDB18's 4. This diversity of source types will enable the hierarchical separation models described in Chapter 2. Additionally, because Slakh was synthesized from note data, the audio data is accompanied by perfectly aligned transcription data. Slakh's aligned transcription data will be used in Chapters 3 and 4 for multi-task separation and transcription and score-informed separation, respectively.

However, because Slakh data is synthesized it does not sound like live musicians. Even though when creating Slakh we used professional-grade sample-base synthesizers (the current gold standard for realistic sounding instrument synthesis), the audio still sounds very different than recordings of live musicians. For instance, the synthesizers are never out of tune, whereas the tuning of live musicians is less precise. Slakh data is still easily recognizable as music, though systems trained on it might have a hard time generalizing to recordings of live musicians. For source separation, we investigated generalization in the paper introducing Slakh [184], but further work is needed to bridge the gap between synthesized data and real data (i.e., sim2real).

Every chapter in this dissertation has experiments that use Slakh. Slakh is made up of audio signals containing musical mixtures with accompanying audio for every source, and aligned transcription data in the MIDI format. It contains 145 hours of mixture data with separate audio files and aligned transcription data for each source. The audio is synthesized using 187 different synthesizer patches (i.e., instrument sounds) arranged into 34 instrument source types (e.g., there are 9 different electric bass synthesizer patches that all get mapped to the "Electric Bass" source type. See them all at `http://www.slakh.com/`). All of the audio is CD Quality (i.e., 44.1 kHz, 16 bit). Slakh totals 1710 total mixtures, with 1289 used for training, 270

for validation, and 151 withheld as a test set. By default, each Slakh mixture is guaranteed to contain at least one guitar, one piano, one bass, and one drum source; they might contain other sources as well (e.g., strings), but they all have at least those four sources. In Chapter 2, I will use the mixes as they come in Slakh (i.e., using all of the 4+ sources), but in Chapters 3 and 4 I will create mixes from just 4-5 of the sources in each Slakh mix instead of using the mixes with the additional instrument sources. Further details about how Slakh is used for experiments will be provided in their respective chapters.

1.2.3. Evaluating Source Separation Systems

The evaluation of source separation systems is a difficult problem that has been debated for many years. The goal of source separation is to produce the audio for an isolated source from a mixture, and, as such, determining if a system did well naturally leads to considering whether the resulting output *sounds* like the desired isolated source (e.g., "Does this guitar estimate sound like it would if the rest of the band were muted?"). In general, determining how something sounds introduces numerous complexities, many of which stem from the nuanced and multifaceted nature of human auditory perception [12, 98]. The other subtext here is that, of course, perception is highly subjective, so what we really want to know is "Do *most people* think that the source estimate sounds like what the desired source should sound like in isolation?" Given the myriad intricacies of human perception, the conceptually simplest answer to evaluate source separation is to just have people listen to the output and rate how good it sounds [65, 124]. In fact, the gold standard method of evaluation involves doing listener studies with highly-trained participants in an acoustically treated room. However, this is expensive and time consuming, leading to small sample sizes. It is, thus, rarely done. As an alternative, researchers have proposed ways of crowdsourcing similar types of evaluations over the web, substituting a small number of ratings from highly-trained experts with a large number of ratings by lightly-trained non-experts [23, 24, 130, 151, 239]. The hope of this strategy is that the law of large numbers will wash away any variation caused by the diverse listening experiences of participants. Still, crowdsourcing costs money, takes time and it is non-trivial to ensure trustworthy results. Perhaps most importantly, listener studies are difficult to fully automate, and are very inefficient when trying to evaluate hundreds or thousands of hours of source separation output. Evaluating hundreds or thousands of hours audio is common when designing separation systems, so a scalable evaluation method is oftentimes the most desirable.

With that in mind, the most common way to do source separation evaluation is via automated processes. When doing automated evaluations of source separation systems, we need to shift the evaluation question

from asking how good the output separation sounds (as might happen in a listener study), to instead trying to quantify how similar an estimated source signal is to a known reference source signal. This tactic has the benefit of being cheap to run, easy to scale, and has a smaller, more well-defined scope than determining if a source estimate "sounds good." However, the trade-off is that automated similarity measures can be sensitive to things that people are *in*sensitive to (e.g., phase offsets) and vice versa. Despite this, the practical benefits of automated similarity measures has made them the de-facto yardstick that researchers reach for to measure their separation systems. Typically this similarity is measured using signal processing techniques. Quite a few of these have been proposed that are specifically targeted to evaluate the quality of speech signals [117, 185, 186, 212, 225, 267, 273], but for music separation the most common of these are a trio of measures called Source-to-Distortion Ratio (SDR), Source-to-Artifacts Ratio (SAR), and Source-to-Interference Ratio (SIR) [283].[6] As is standard, I will only report SDR in this dissertation, which is generally thought of as a measure of the "overall quality" of a separation estimate. There are multiple implementations of SDR described in the literature [154, 222, 264], however in this dissertation I will use the following definition:

$$\text{SDR}(\hat{s}, s) := 10 \log_{10} \left(\frac{\|s\|^2}{\|s - \hat{s}\|^2} \right) \tag{1.3}$$

given some reference source signal s and an estimate source signal $\hat{s}$. When the absolute difference between the reference and estimated source signals, $\|s - \hat{s}\|$ is small, the denominator is much bigger than the the numerator, yielding a large SDR value and indicating that the source estimate is more similar to the reference signal. In other words, higher values indicate source estimates closer to the reference, i.e., better separation performance.

There are two additional variations on this SDR equation that I will use in this dissertation. The first is the so-called Scale-Invariant Source-to-Distortion Ratio (SI-SDR), which introduces the idea that the loudness of the signal should not effect the SDR score ("scale" here refers to a signal's amplitude). The SI-SDR [154] adds a scaling factor, α, to the target signal to add match the amplitude of the estimated signal (this is the "scale-invariance"). SI-SDR is defined as

$$\text{SI-SDR}(\hat{s}, s) := 10 \log_{10} \left(\frac{\|\alpha s\|^2}{\|\alpha s - \hat{s}\|^2} \right) = 10 \log_{10} \left(\frac{\| \frac{\hat{s}^T s}{\|s\|^2} s \|^2}{\| \frac{\hat{s}^T s}{\|s\|^2} s - \hat{s} \|^2} \right) \tag{1.4}$$

[6] These also go by the name `BSSEval` measures, due their the name of one of the early widely-circulated MATLAB implementations, named Blind Source Separation Evaluation.

where $\alpha := \frac{\hat{s}^T s}{\|s\|^2}$ is the scaling factor that ensures that the estimated signal $\hat{s}$ and the reference s have the same scale, with $\hat{s}^T s$ indicating the vector dot product between $\hat{s}$ and s.

The second variation on SDR is measuring the improvement in SDR over the mixture. The idea here is to measure the SDR before doing any kind of processing (i.e., using the mixture as the "source estimate") and compare that number to the SDR using a source estimate from the separation algorithm in question. Applying this variation to the Scale-Invariant SDR yields the following:

$$\text{SI-SDRi}(\hat{s}, m, s) := \text{SI-SDR}(\hat{s}, s) - \text{SI-SDR}(m, s), \tag{1.5}$$

which is called the SI-SDR Improvement, or SI-SDRi.

In this dissertation, I will report separation results using SDR and SI-SDRi. This is due to the fact that SDR was the standard up until SI-SDRi had become standardized, which occurred between the work in Chapters 3 (Cerberus) and 2 (Hierarchical Separation). (Note that the chapters do not appear in chronological order, so the work in Chapter 3 predates the work in Chapter 2.) In each chapter, I will be explicit about which version of SDR I am using.

While it is true that objective measures (including SDR and its variants) are no substitute for human listener studies,[7] they are the standard for measuring separation performance and it can act as a (noisy) proxy for true separation quality as judged by a human. As I touched on briefly above, its real appeal is the ability to easily measure separation performance on thousands of hours of audio, as I do in this dissertation and which would be prohibitively expensive if doing listener studies. A fruitful direction for future work would be the development of a method that can cheaply evaluate audio quality in a way that is strongly correlated to human perception. However, this idea is beyond the scope of this document.

1.3. Automatic Music Transcription

The goal of an Automatic Music Transcription (AMT) system is to convert an auditory musical signal into some form of a musical score, such as a piano roll or Western European notation [7, 8, 143–145] (this will be discussed further in Section 1.3.1). The generated musical score, also called a *transcription*, is a symbolic representation of the content of a musical piece. A transcription can be thought of as a set of instructions to produce the audio content of the music; these instructions come in the form of note events, which can

[7] They also require ground truth sources [180], which can be hard to come by.

determine the start, stop, loudness, and pitch of a note, as well as any other characteristics relevant for representing the performance (e.g., in some cases it might be important to notate vibrato or dynamics). The process of music transcription is the musical equivalent to transcribing a recording of a person's speech into a text document. The process of *transcribing* a musical piece can be thought of as the opposite process of *playing* a musical piece; playing a piece turns notation into sound, whereas transcription turns sound into notation.

In general, transcription data is much more parsimonious than an audio recording; instead of the millions of data points that make up the samples of a digital audio recording, a transcription of the same piece might consist of a few hundred or a few thousand note events that specify the musically-relevant structural aspects of the musical audio.

An effective AMT system could have broad applications, enabling interactions for music education [9, 73, 290] (e.g., analyzing and providing feedback on a student's performance from a recording), and music creation [6] (e.g., indexing musical ideas by rhythm, or melodic or harmonic structure). Manual transcription is a notoriously labor-intensive process [143] and, so, automating transcription could enable the transcription of large music corpora (e.g., Spotify, YouTube, Apple Music, etc), which could possibly power recommendation engines that have knowledge of the structural components of the music. It could also facilitate large-scale musicological analyses, revealing insights about how musical compositions vary across time periods, geographic locations, genres, etc [5, 129, 200, 210, 217, 243, 244, 286]. Additionally, an important problem for rights holders is the ability to determine when a new rendition of an existing song (i.e., a "cover" song) appears so that they can provide royalties to the original authors. This task is called cover song ID, and is made difficult because new versions often have different tempos, key signatures, and instrumentation compared to the original [242]. A system that can transcribe musical mixtures could power cover song ID applications that are invariant to these confounding factors [159, 173, 226, 276].

Although the majority of work in this dissertation is centered on source separation, AMT plays a crucial, albeit smaller, role in the latter two projects, Chapters 3 and 4. As such, it is important to discuss so that I can contextualize the impact of the contributions of Chapters 3, and understand how AMT fits into the work of Chapter 4.

```
[Verse 1]
C   G   D          A               E           E
Hey Joe, where you goin' with that gun in your hand?
C     G   D       A                          E                E
  Hey Joe, I said where you goin' with that gun in your hand,
```

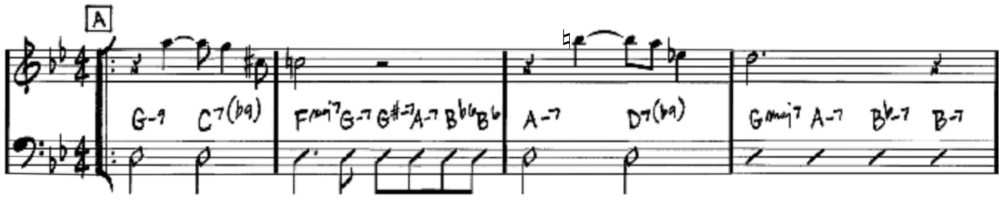
Figure 1.3. A transcription can be quite sparse. This one has only lyrics and chords (the letters `C`, `G`, `D`, `A`, `E` above the lyrics), but omits other pertinent info that is expected to be interpreted by a musician, like the melody or the timing of the chords. Song: "Hey Joe" recorded by various artists, & famously Jimi Hendrix.

1.3.1. Representing Data for Automatic Transcription

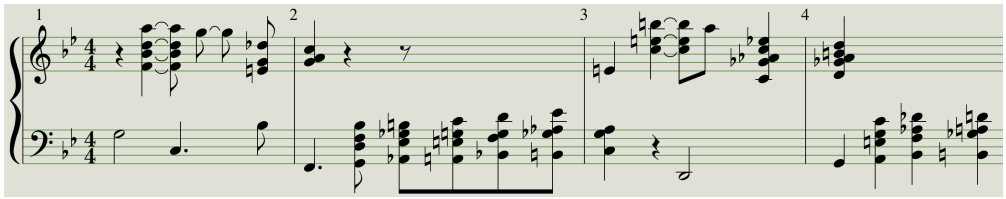
The level of detail and structure of a musical score or transcription is dependent on its use case. For instance, in searching for how to play a popular song on the internet, you might find a website containing the lyrics and chords of the song (like in Figure 1.3), but no information on the melody, timing of notes, instrumentation, loudness, articulation, tempo, rhythm, chord voicings (i.e., what specific notes to play for each chord), and so on.

Contrast this to Western European music notation (Figure 1.4b), which might have markings for some of the things that the lyrics and chords website lacked, but might contain relative or vague information on loudness and exact timing of notes. In either case, such information might be left out intentionally, with the details expected to be filled in according to the interpretation of a musician. As is visible by comparing three different incarnations of the same musical score in Figure 1.4, the amount of detail captured in the transcription is related to how it is represented.

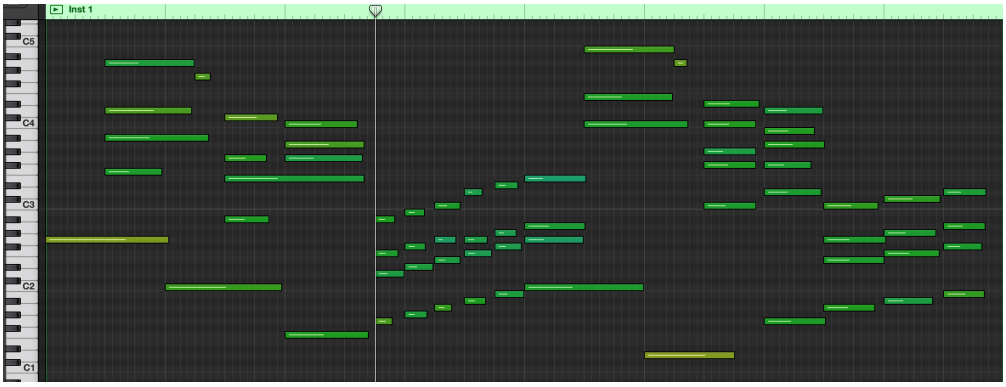
The input to an automatic music transcription system is an audio signal and the output is a transcription. More precisely, almost all modern systems use a spectrogram or one of the aforementioned spectrogram variants (i.e., log-mel spectrogram) as input to the network so that frequency information is represented explicitly. Typically, the desired output is a transcription containing data on every single note such as the pitch, loudness, absolute timing information (i.e., precise start and end), and instrument type. To this end, most AMT systems create transcriptions in the piano roll format (shown in Figure 1.4c), which can be represented as a 2D matrix with pitch represented along one axis, time represented along the other, and values corresponding to volume of the notes. The name "piano roll" derives from the 1800's, when player pianos read and played music from paper rolls that had score information inscribed as perforations. Today, piano rolls are commonly found in modern audio software, such as digital audio workstations (DAWs), where, despite their name, piano rolls can represent music from any type of instrument. A piano roll usually only



(a) Jazz "lead" sheet.



(b) Western European music notation.



(c) Piano roll.

Figure 1.4. Three types of musical scores representing the same piece of music, any of which could be the output of a Automatic Music Transcription (AMT) system. Song: "Ruby, My Dear" by Thelonious Monk.

represents one instrument at a time; to represent multiple instruments at once, multiple piano rolls are required, one for each instrument. Typically, the pitch resolution corresponds with the notes of the twelve-tone equal temperament chromatic scale, as is visually represented by a piano along the y-axis in the figure above, and the time resolution is typically on the order of a few milliseconds. The limitations of the piano roll revolve around the fact both that time and pitch are quantized, meaning that some occasionally-occurring musical expressions like glissando (i.e., smoothly gliding from one pitch to another) are hard or impossible to represent. Regardless, piano rolls have been the output format of choice for almost all AMT systems.

1.3.2. Datasets and Evaluation Measures for Automatic Music Transcription

As I will discuss in more detail in Section 1.4.2.1, most of the effort in building AMT systems been centered around transcribing recordings of solo piano [11, 53, 63, 103, 135, 136, 139, 204, 219, 247, 307]. Because of this, the standard datasets for AMT, like MAPS [64] and MAESTRO [106], exclusively contain recordings of solo piano performances. Relatively less attention has been given to automatically transcribing other instruments individually; nonetheless a few datasets exist which support transcription of solo guitar recordings (GuitarSet [303]) or solo drum recordings (E-Groove MIDI Dataset [19, 89]).

One of the goals of this dissertation is to develop systems that can transcribe multiple instruments simultaneously (i.e., multi-track transcription). I am aware of only two datasets that support such a goal: MusicNet [272], and Slakh2100 [184]. MusicNet is a dataset of 330 freely-licensed classical recordings with post-hoc transcription annotations. These annotations were created through an error-prone automated alignment process. MusicNet does not provide recordings of each isolated instrument, precluding doing any type of simultaneous separation and transcription (as done in Chapters 3). Slakh2100 (or Slakh), as discussed above in Section 1.2.2, is a dataset containing many audio tracks synthesized from MIDI data (MIDI is a standard format for transmitting and storing symbolic note data, and a MIDI file is often the actual output artifact of an AMT system that an end user sees). Because the audio is synthesized, we can use the MIDI that the audio was synthesized from as our transcription annotations, with the huge benefit that we are guaranteed that the audio and transcription data are well aligned. Just as before, a potential drawback to using Slakh is that a transcription system trained on synthesized data might have a harder time generalizing to data that comes from recordings of live musicians. However, this distribution shift seems to be less of a worry in AMT than it is in source separation, as hinted at by the MT3 transcription model [84], which leans on Slakh data to achieve state-of-the-art AMT performance across a wide range of datasets. Therefore, for transcription tasks in this dissertation, I will use the Slakh dataset.

Slakh has 145 hours of mixture audio data, where the mixes are composed of a set of single-instrument audio clips that each have aligned transcription data. Further details are in Section 1.2.2. When using Slakh for automatic transcription in this dissertation, a system will input a mixture and output a separate transcription for each individual instrument track (up to 5, shown in Chapter 3).

The evaluation of AMT systems is more straightforward than evaluating source separation systems. The common measures that researchers use are precision, recall, and F1-score estimated between the estimated transcription and the ground truth transcription. There are two common ways to calculate these scores:

Frame Scores The ground truth and estimated transcriptions are converted to a (binarized) piano roll representation (See Figure 1.4c) with a certain frame rate (e.g., 32.5 Hz) and precision, recall, and F1-scores are calculated, matching corresponding entries in the piano rolls, element-wise. In other words, for every pitch and every time frame in the piano we determine if the estimated transcription value at that entry is a True positive, False positive, or False negative with respect to the ground truth transcription value at the same entry. We then aggregate these into Precision, Recall, and F1-scores as per usual. Practically, this is done by flattening both piano rolls to 1D arrays, and calling `scikit-learn`'s [214] `sklearn.metrics.f1_score()`.[8]

This is perhaps a naïve way of evaluating transcription because it treats every time-pitch entry in the piano roll with equal weight. However, not every time-pitch bin should carry the same magnitude. For example, imagine that the ground truth transcription contains a sustained note over 1 second, whereas the estimated transcription turns on an off that note ten times in that second. The Frame Score will count the gaps when the note is off in the piano roll as False negatives of that singular note, but otherwise the frames where the piano roll is active count as True positives. However, when the note is turned on and of ten times in the estimate piano roll, that transcription says there are ten notes! Those additional notes should be considered not as missing notes as the Frame Score would say (i.e., False negatives), but as extraneous notes (i.e., False positives). This leads to the second way to evaluate transcription systems.

Note Onset Score & Note Onset/Offset Scores Here, instead of calculating a score directly from the piano roll frames, we extract information about the note onsets for the estimated and reference transcriptions. We then count how many estimated notes match the ground truth notes and use those counts to calculate precision, recall, and F1-scores [55, 56, 222]. An estimated note onset is deemed a True positive if the estimated onset is within 50ms of the ground truth onset and it has the same pitch value. Estimated and ground truth notes are matched together using bipartite graph matching [222]. For the Note Onset Score, only the start of the note is considered in this matching process, and in the Note Onset/Offset Score, both the start and end of the note are considered in the matching process. The Note Onset and Note Onset/Offset scores are in general much stricter than the Frame Score, and mitigates the problem of incorrectly weighing

[8] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html`

every frame in a piano roll. In this dissertation all of the transcription scores will be Note Onset/Offset Scores.

1.4. Open Problems with Recent Approaches to Source Separation and AMT

Recently, researchers have demonstrated impressive results on the tasks of Source Separation and AMT by introducing deep learning methods. Within music source separation, the 2021 Music Demixing Challenge (MDX)[9] was a recent competition that saw teams vie to be crowned the "best separation system." Every single entry in the 2021 MDX Challenge was a deep network [198]. The story is similar with AMT; all state-of-the-art results are due to novel uses of neural networks [84, 103, 105]. Although deep networks are the current state-of-the-art solutions to both tasks, these solutions still have major shortcomings. In this section, I will describe open issues in existing work with regard to both of these tasks that I will address in this dissertation.

1.4.1. Open Problems with Music Source Separation

There are two major issues with existing source separation systems. The first has to do with the problem definition itself: in almost all modern music separation systems, a fixed set of sources are defined in a rigid, closed-world setting. This is extremely burdensome if the source you want to separate is left out of the problem definition, which is the case if your source is not one of Vocals, Bass, or Drums (see Section 1.2.2. The second is that the output of current source separation systems is not easily modifiable. In other words, once a model is trained, there is no way to alter its output if there are any mistakes, posing a usability issue for end-users.

1.4.1.1. Fixed Source Types. As just mentioned, the first issue with music source separation is with how sources in music separation systems are all defined in a closed-world setting, with a limited set of fixed source types. For example, the vast majority of recent music source separation systems [33, 35, 43, 44, 111, 141, 148, 161, 198, 233, 265, 268, 269] are trained using the source types defined by the popular MUSDB18 dataset [223]: "Voice," "Bass," "Drums," and "Other." The first three source types have relatively well-defined scopes, but the "Other" However, part of the beauty of music is in its diverse array of musical instruments, ranging from Tuvan throat singers to church choirs, grand pianos to thumb pianos, and so on. The vast diversity of musical instrument sources is a clear example of an open-world setting. There is a clear mismatch

[9]MDX was a continuation of the biennial music source separation competition, formerly named SiSEC [163, 264]. The most recent SiSECs all had deep network-based winners as well.

between musical sources as they exist in the world (i.e., open-world), and how they are defined for the task of music source separation (i.e., closed-world).

Given that most systems are only able to separate a fixed set of source types, it is extremely hard for an end-user to separate a source omitted by the system's training data. This can be a common situation when considering the small number of sources included in the MUSDB18 dataset described above. Most systems have no way to add source types other than retraining the whole model from scratch with new data that matches the source type. However, because deep learning models are exceedingly data-hungry, gathering enough clean data is so out of reach for most end-users that this option is infeasible, not to mention the exorbitant computational costs. **It would thus be desirable to ameliorate the established requirement for a large amount of training data for every possible source type.**

1.4.1.2. Changing a Model's Output Post-hoc. Making adjustments to the output of a source separation model is extremely important for making this technology accessible to end-users, and, yet, making adjustments to current systems is extraordinarily difficult. Source separation is a way to edit and manipulate audio, and the vast majority of other tools for editing audio enable users to make adjustments as they are working. However, modern source separation systems do not allow for these types of interactions, which an end-user might expect. With existing systems the user's only means of editing its output is editing the resultant audio signal itself. Yet, audio signals are notoriously high-dimensional; a 30-second audio clip can have millions of data points and many thousands must be altered to make a realistic-sounding change. So edits like this are not really a feasible option for an end-user. **Altering the output of a source separation model after it has been trained is a near-impossible process for end-users.**

For instance, one common issue a user might have is that they might want to separate a new instrument sound. If an end-user wants a completely different source separated, the options they have are to edit the thousands of data points or to train a whole new system from scratch.

In Chapter 2, I will propose a solution to this problem. By leveraging the long-established musicological ideas of classifying musical instruments based on sonic qualities [287], I will show that is possible to explicitly model the relationships between source types in a hierarchical fashion. This enables systems to take advantage of their experience with similar-sounding sources when encountering new source types, which opens up source separation systems to separate more types of sources, enables users to control the type of source the system outputs, and eases the substantial data requirements of current systems.

Instead of switching which instrument source gets separated, what if a user wants to make a more minor edit like changing a few incorrect notes? In other words, what if the sound of a piano note is heard in the guitar source estimate? In this case, it is even more burdensome for a user to edit note-level errors with current systems, i.e., issues where the audio of a single note is put into the incorrect source estimate. There are no existing deep learning systems that support this interaction, so a user would have to use an older technology (e.g., NMF [156]) that does allow this interaction but might introduce other artifacts in the separated audio. Other than that, the user's only recourse is to edit tens of thousands or hundreds of thousands of data points of the output audio to fix the sound of the stray note. What would be best is if the user had a way to affect the separation output by making changes in a representation that was less information dense that a raw source output.

In Chapter 4, I will propose a solution to this problem a modelling approach to source separation that will support an interface to let an end-user to make note-level changes to the separation output by editing transcription data (i.e., note data).

1.4.2. Problems with Considering Source Separation and Automatic Music Transcription as Distinct Tasks

Transcribing multiple musical instruments in a mixture has some similarities to the problem that source separation systems are designed for–namely, in both tasks, the systems must analyze multiple overlapping sounds in a musical mixture. The link between these two tasks has long been noted; early overview papers describing a plethora of AMT systems speculated about its usefulness in source separation [8, 144, 145, 218]. Contemporaneously, Non-Negative Matrix Factorization (NMF) [156] was proposed as a solution for AMT [253], but ultimately ended up becoming the workhorse algorithm for source separation for many years [75, 252, 285]. Right up until the popularization of deep learning, there was a small, but thriving community of researchers that coalesced around score-informed source separation, which investigated the usefulness of steering source separation algorithms using transcription data [59, 71].

However, considering both of these tasks simultaneously has fallen somewhat out of favor in the era of deep learning. Nine of the top ten most cited deep learning transcription papers only consider instruments in isolation [11, 53, 103, 106, 135, 136, 139, 247, 261]. Similarly, none of the $\tilde{5}0$ music source separation systems in the Music Demixing Challenge [198] and none of the 16 separation systems on Papers With Code [203] addresses transcription in any form (either as input conditioning or as an output). **Given the**

similarity of source separation and multi-instrument AMT, considering these problems independently is a missed opportunity.

For the rest this section, I will outline the open problems in the recent literature that arise from treating Automatic Music Transcription and Source Separation as separate tasks. The first major problem is simple: modern AMT systems are not designed to transcribe more than one instrument simultaneously, which means that they will not work in for most musical recordings (because most recordings contain more than one source at a time). Second, as a means to analyze a given musical scene, only separating the sources or only transcribing the instruments therein produces a less complete analysis of the scene than if *both* tasks are performed. Finally, I will make the argument that a transcription, as provided by a multi-instrument AMT system, can a be a useful way to guide a separation system post-hoc, as I discussed above.

1.4.2.1. Existing Automatic Music Transcription Systems Only Work for Isolated Instruments. Recent Automatic Music Transcription (AMT) systems have shown great advancements with the introduction of deep learning, with results on benchmark datasets steadily improving year after year. However, most contemporary AMT systems are limited by a huge practical issue: **they assume that only one instrument is present in an input recording.** Historically, most systems have been specifically designed to transcribe solo piano recordings [11, 53, 63, 64, 103, 105, 106, 135, 136, 139, 149, 204, 219, 247, 307], and a small amount are designed to transcribe drum recordings [19, 213, 261, 301]. In other words, these systems are not Automatic *Music* Transcription systems, they're really Automatic *Piano* Transcription systems; they are not expected to work if the input audio contains a full rock band or a horn section. **Few systems assume that anything other than a piano is present in the audio.** In spite of the fact that solo piano recordings are a tiny subset of all recorded music, piano transcription has dominated the AMT space.

The sonic qualities of the piano make it an easy instrument to focus on: piano notes have a clearly defined onset, each note decays similarly, and the fundamental frequency of each note is fixed for its duration (i.e., no vibrato or glissando). The piano's sonic qualities also make authentic-sounding recordings easy to produce using high quality synthesizers based solely on signal processing methods. This means that large, realistic datasets with ground truth transcription labels for training machine learning models are relatively easy to produce [64, 106]. Those same sonic qualities make it easier for machine learning methods to model piano notes, so much so that some systems are designed around the specific acoustic properties of piano notes [103, 135, 149]. A similar story holds for drum set transcription; the drum set's "easy-to-produce" sonic qualities make it well suited for synthesizing realistic-sounding audio using traditional signal processing methods, and,

again, this property of the instrument has been exploited to produce high-quality training data for drum transcription models [21]. However, many musical instruments do not have these "easy-to-produce" sonic qualities that make it convincingly real audio relatively simple to produce. For example, a violin note might have none of the qualities that make piano and drum notes easy to transcribe: the violinist has complete control over the volume of the note for its duration. This means that the onset could be unclear (i.e., starting very quiet) and each note's decay could be different. The fundamental frequency of a violin note could also change over the duration of the note either from vibrato or glissando. Given that violins and other musical instruments might not have all of the nice sonic qualities that drums or pianos might have, it is natural to ask if synthesized audio is suitable to use to train AMT systems. Prior work has left it unclear as to whether we can still use synthesized audio data for transcription when its realism cannot be taken for granted.

Furthermore, individual instruments often do not occur in isolation, but rather they are heard with other instruments in a mixture. A main issue with the current state of Automatic Music Transcription research is its myopic focus on the individual transcription of isolated single-instrument recordings. **Many musical recordings have multiple instruments simultaneously and therefore most Automatic Music Transcription systems will fail on these recordings.**

Although a few counterexamples existed [272, 274], the world of AMT systems only working for solo piano recordings was largely the story for the years that led up to my work in Chapter 3 of this dissertation. That work was originally published in late 2019, and, in the time since, a handful of multi-instrument AMT systems have appeared [57, 84, 112, 160, 270, 306]. Chapter 3 outlines my strategy for multi-instrument transcription, which is one of the first deep learning systems that is able overcome the issues that prior single-instrument transcription systems had.

1.4.2.2. A More Complete Musical Scene Analysis. This dissertation is focused on the tasks of musical source separation and automatic music transcription. Both of these tasks provide a different, yet complementary mode of analyzing a musical scene–source separation estimates the audio data of each instrument source in isolation from the others, and music transcription estimates the symbolic note data for each instrument source. The audio data and note data are different means of viewing of the same underlying musical data in the auditory scene. Audio source data and symbolic note data exist at different levels of abstraction; the source data is a waveform, containing millions of data samples that precisely specify the source audio, while transcription data is a score, which compresses a waveform into a few hundred relevant data points, but it is not readily able to be heard without an external synthesizer. For example, representing the guitar part for a

30 second excerpt for the The Beatles' song "Oh, Darling!" as a waveform consists of 1.3 million samples,[10] whereas the guitar only plays a few dozen notes. The source audio provides low-level information which requires effectively no extra effort to listen to, but from which higher level structure must be inferred (e.g., harmony, melody, rhythm, tempo, or a transcription, etc). Whereas the transcription data readily provides information about high-level structures, but offers little information about how to realize the low-level details crucial for complete reconstruction of the artifact that is being represented. Each type of data provides its own perspective of the scene, and obtaining both enables a more holistic understanding of the musical scene at different levels of abstraction. **Thus, a system that only produces source audio or transcription data provides a less complete description of a musical scene–either omitting high-level structures or low-level details–than a system which produces both.**

In Chapter 3, I will describe a system that produces both separation and transcription estimates simultaneously, delivering high-level structures and low-level details about the musical content in the musical scene. Additionally, I will show how we can leverage high-level note data to make edits to low-level source estimates in Chapter 4.

1.4.2.3. Multi-Instrument AMT can be used to Guide Separation. As I discussed a few pages ago in Section 1.4.1.2, changing the output of a separation system after it has been trained can be a daunting task. In that section, I made the argument that editing the transcribed notes in the musical scene is a way to alleviate the tedium of changing the source data directly. Well, a prerequisite to enabling that interaction is actually having those transcribed notes. Of course, a user could transcribe the notes themselves, but manual transcription is also a notoriously laborious process and is really only an option for trained musicians, as music transcription is not an everyday skill.

Therefore, to enable a score-informed separation system as I propose in Chapter 4, having a multi-instrument AMT system as a pre-processor goes a long way towards making it readily usable for most people. In Chapter 4, I use a simpler variant of the multi-instrument system I describe in Chapter 3. However, without *any* multi-instrument transcription data, the note-editing interaction becomes impossible to those who are unable or unwilling to manually transcribe a musical scene. Multi-instrument AMT is an important piece of the puzzle that makes the score-informed separation system in Chapter 4 work.

[10] A typical sample rate is 44.1 kHz: 44100 samples/second $\times$ 30 seconds = 1,323,000 samples.

1.5. Broader Impacts

As is the case in many areas, modern machine learning systems have the potential to make a big impact on how we make and consume music. For instance, in 2020, OpenAI released Jukebox [47], generative audio model trained on 1.2 million songs and is able to synthesize uncannily realistic-sounding songs based on input conditioning from lyrics and metadata such as artist names, genre, year of release, and additional tags like "mood" or playlist keywords associated with songs from their dataset (e.g., it generates a country song, given the prompt: "Country, in the style of Alan Jackson"). One of the many predicaments about music machine learning systems that Jukebox embodies is one of controlling the system. Without a certain level of control over a system's output, how can a user feel like they are still a part of the songwriting process? Furthermore, without control is it even possible for a user to use it to write their own songs, or will the system make whatever it fancies without any regard for the user's desires? Hearkening back to the beginning of this chapter, I think it is helpful to frame these music machine learning systems as instruments being in conversation with a musicians. We know that humans are innately able to analyze the many complex structures (e.g., overlapping sources, note events) in musical scenes. So when thinking about how a machine learning can fit into conversation with a musician, the machine learning system that cannot analyze these structures in music is at a disadvantage when to "speaking" to the human musician. While there is some evidence that Jukebox has learned useful representations for analysis tasks [25] (including for source separation [175]), having explicit avenues for control that align with the important structures in musical scenes (e.g., sources, notes) will be paramount to making future systems that users want to interact with.

The work in this dissertation demonstrates ways to control musical scene analysis systems; for example, Chapter 2 and Chapter 4 both describe different ways of making controllable separation systems, where the majority of prior separation work rarely dealt with the problem of controlling a system's output. Furthermore, the analysis systems described here are much more capable than prior systems in important ways (i.e., scaling separation to many more source types like in Chapter 2 or being able to transcribe multiple overlapping instruments as in Chapter 3). The expanded output of the analysis systems I describe here could be used as conditioning to control the next generation of music machine learning models. For instance, if a future Jukebox-style large scale generative audio model were conditioned on transcription data–produced by a multi-instrument AMT system–then the system would not just generate *any* country song, but could instead generate *my* country song by being given a transcription of my playing.

CHAPTER 2

Hierarchical Source Separation

In this chapter, I introduce the concept of hierarchical source separation. Typical source separation systems make very few–if any–assumptions about the structure of the acoustic scenes that they attempt to analyze. In contrast, here I develop a set of strategies for source separation based on the notion that source types in an acoustic scene can be decomposed hierarchically. As a prerequisite, this involves formulating an understanding of how acoustic scenes can naturally be seen as being constructed hierarchically. It is from this perspective that we can understand not only how an acoustic scene is assembled hierarchically, but how that same scene can be disassembled hierarchically using a hierarchical source separation system.

In Section 2.2 I will identify how acoustic scenes can naturally be thought of as hierarchical, and I will outline principles for grouping sources according to a hierarchy. These grouping principles will guide us as I define a set of four hierarchical source separation strategies, developed in Section 2.3 and evaluated in Sections 2.4 and 2.5. Finally, in Section 2.6 I will propose ideas for designing interactive separation systems built on a back-end hierarchical separation system, and show how hierarchical separation can lead to more flexible systems that are able to separate many more source types than most other separation systems. In this work, I restrict the discussion to musical scenes. As such, I am able exploit existing work that establishes *taxonomic relationships* between different types of sources via hierarchical classification systems [287]. The resulting separation systems explicitly model sources in a hierarchical manner as a means to analyze a musical scene hierarchically.

Viewing a musical scene hierarchically is a potential solution to a widespread and unspoken problem with modern source separation systems: most systems presume a single fixed segmentation of the auditory scene, which means that dividing the scene in any new way requires retraining a system from scratch. As a whole, the field of source separation has seen notable performance improvements with the introduction of deep learning techniques, most notably in the areas of speech enhancement [69, 289, 295, 304], speech separation [115, 147, 168, 278, 279, 292, 297], and music separation [33, 35, 43, 44, 111, 127, 141, 148, 161, 166, 198, 233, 265, 268, 269]. These techniques succeed in cases where the notion of a source is well-defined and limited in scope. In the case of speech enhancement or separation, the target source never changes: it

Figure 2.1. A common auditory scene (left): people are having a conversation with the radio on in the background, and the radio has a band playing. There are many sensible ways to determine what groupings of interest are in this scene, e.g., the radio vs. the talkers, or the woman's voice vs. the man's voice vs the radio, etc. However, one flexible way to understand this scene is by splitting it up hierarchically (right). This chapter discusses strategies for hierarchical source separation.

is always defined as the speech of a single speaker. However, in real-world scenarios we might have a more complicated understanding of a source. Consider the case where a band is playing on the radio while two people are having a conversation, shown in Figure 2.1. What are the "sources" in this scene? Is the radio one source and the overlapped speech a second source? Or are each of the interlocutors considered a separate source? Are each of the instruments in the band on the radio their own source as well? Clearly, there are many correct answers to this question, but one flexible way to understand this auditory scene is to apply a hierarchical structure to its parts.

In this chapter, I do just that. I reframe the source separation problem as hierarchical, based on the premise that sound sources can be organized in a hierarchical structure. These hierarchies are assumed to be constructed such that they capture taxonomic relationships between different sources, *a la* the Hornbostel-Sachs taxonomy of musical instruments [287]. Here, I investigate musical scenes although this work can be extended to any type of auditory scene (e.g. environmental sounds).

2.1. Contributions of this Chapter

The contributions of this chapter are the following:

- I provide a theoretical framework for analyzing the sources in a musical scene in a hierarchical fashion. I propose four general strategies for realizing hierarchical source separation systems. These strategies vary along two axes: the first axis describes whether the system is designed to separate just one instrument (i.e., Single-instrument), or multiple instruments at once (i.e., Multi-instrument); the second axis describes whether the system is designed to separate at just one

hierarchical level (i.e., Single-level), or multiple hierarchical levels at once (i.e., Multi-level). As I develop these strategies, I will highlight how certain design decisions (e.g., Multi-level Multi-instrument designs) can lead to neural networks that are able to scale to separating larger numbers of sources than naïve separation techniques. I will show that a hierarchical Multi-level Query-by-Example is able to separate many more fine-grained sources than typical separation systems usually consider.

- I implement these strategies by modifying a common source separation architecture and conduct a set of experiments to test the effectiveness of all four strategies. I find that Multi-level systems always outperform Single-level systems, with the largest performance boost coming at the most specific (i.e., finest-grained) source types, which are more difficult to separate. I also find that Multi-level systems are able to retain up to 90% of these performance benefits when trained on only 10% of this finest-grained data.
- I propose an imagined user interface for hierarchical separation systems, built around the fact that such systems are much more flexible than existing systems, with respect to the number of sources they are able to separate.

2.2. Auditory Hierarchies

2.2.1. Background

Framing a musical scene as hierarchical has precedent in fields that study human audition. Evidence shows that human auditory perception has many hierarchical characteristics [74, 211, 215, 296]. In his seminal text on human auditory perception, *Auditory Scene Analysis* [12], Albert S. Bregman notes how auditory scenes are organized hierarchically. In analyzing an auditory scene containing two musicians playing a duet, he concludes: "It makes sense, therefore, to think of the auditory perceptual organization of [a musical] duet as having a hierarchical structure [...]. This argument implies that there are levels of perceptual belongingness intermediate between 'the same thing' and 'unrelated things."' While human perceptual auditory hierarchies come in many forms (e.g., timing hierarchies, timbre hierarchies, rhythmic hierarchies, etc), in this chapter I focus on how sound sources can be organized hierarchically. Specifically, I focus on the task of building hierarchical source separation systems using a musical instrument hierarchy.

The study of the classification of musical instruments falls within the field of musicology, specifically a subfield called organology [92, 275], which itself overlaps with ethnomusicology and musical acoustics. Many

traditions within organology taxonomize musical instruments hierarchically. Almost all human cultures throughout history have created musical instrument classification systems [134], many of which are inherently hierarchical. Some of the earliest systems appear over 4000 years ago in ancient China [99], which group together musical instruments by the physical materials that they are made of. One prominent Western example is the Hornbostel-Sachs system [287], which classifies musical instruments by their sound production mechanisms in a hierarchical manner similar to the Dewey Decimal System [46]. Another system widely used in Western music classifies instruments by their musical pitch range, with terms named after singing voice classifications: e.g., *soprano*, *alto*, *tenor*, *baritone*, and *bass* (from highest pitched register to lowest). Each of these systems captures some notion of a taxonomic relationship between musical instruments. Prior work has shown that machine listening systems have been able to successfully leverage hierarchical taxonomic relationships for the task of musical instrument recognition (i.e., determining the identity of an instrument from a recording) [83].

Of particular note is the fact that there is an element of a musical instrument hierarchy inherent in the process of creating modern-day musical recordings. In the recording studio, each track is assumed to be an isolated recording of a single instrument (e.g., a full drum set), or part of one instrument (e.g., just the kick drum from a full drum set). At the mixing board, a sound engineer can mix together multiple tracks into a *submix* (called a "send" or "bus" track), which acts as a single unit in the recording session, having effects and other signal routing configurations be specific to the submix rather than the individual tracks therein [207]. The submixes are then manipulated alongside other tracks, which may contain only a single instrument. For example, a standard practice is to record every separate piece of a drum kit with a single microphone (e.g. the snare, the kick drum, the toms, and each of the cymbals has its own dedicated microphone) and then combine all of them into a drum submix. An example of this is shown in Figure 2.2. This configuration is *hierarchical*; the engineer can choose to manipulate all of the drum sounds (the drum submix) or manipulate individual drum tracks within it.

The work in this chapter takes inspiration from hierarchical musicological taxonomies and the hierarchies found in modern recording practices. Here, I leverage the hierarchical relationships between instruments as a useful cue for source separation systems.

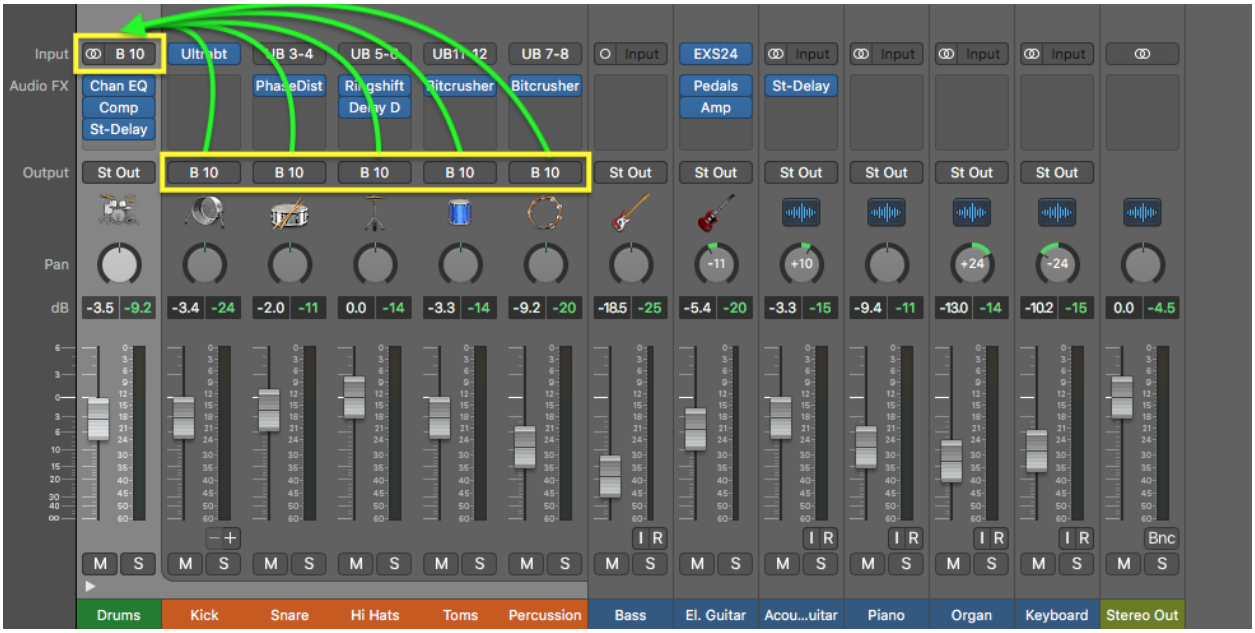


Figure 2.2. An annotated screenshot of Logic Pro, a popular Digital Audio Workstation (DAW), showing how to create a "mix bus" that combines multiple individually recorded tracks into one editable unit, called a "bus." Use of this technique is very common. The individually recorded tracks are visualized as vertical strips, named and color coded along the bottom (e.g., "Kick", "Bass", "Piano", etc). Each track has their own set of controls to manipulate the sound of that instrument track. The audio from all of the individual drum tracks (labeled with orange: "Kick", "Snare", "Hi-Hat", "Toms", "Percussion"), is routed to a super-track (i.e., the bus track), which is the leftmost track, annotated in green and titled "Drums." The Drums bus has its own set of controls that can manipulate all of its constituent tracks. The DAW enables an end user is able to manipulate *all* of the drum tracks as a unit, or they can edit each track *individually*. This indicates that there can be a clear hierarchical structure between musical instruments when music is created. This chapter investigates how we can leverage hierarchical structures for deconstructing and analyzing musical signals via source separation. Image from iZotope [52].

2.2.2. The Structure of Auditory Hierarchies

As a general formulation, we can define an auditory hierarchy such that sources at higher levels in the hierarchy are composed of mixtures of sources at lower levels of the hierarchy. At the top of the hierarchy is the root node, which is sole source type that contains all source types, and at the bottom of the hierarchy is a set of leaf sources that cannot be further decomposed. Other than the root node and the leaf nodes, every other source[1] node can be further separated into *child* sources and combined with its *siblings* to create *parent* sources. Note that a child node can only be mixed to create exactly one parent node; in other words, multiple inheritance is precluded in this definition. This setup is shown in Figure 2.3, and although Figure 2.3 shows

[1]A quick note on terminology: here, I will only use the term "source" to mean a sound source, and I will use the terms "root node" to mean the node that has no parents and "leaf node" to mean a node that has no children. To avoid a collision, I will *not* be using "source" and "sink" terminology to refer to nodes in a hierarchy.

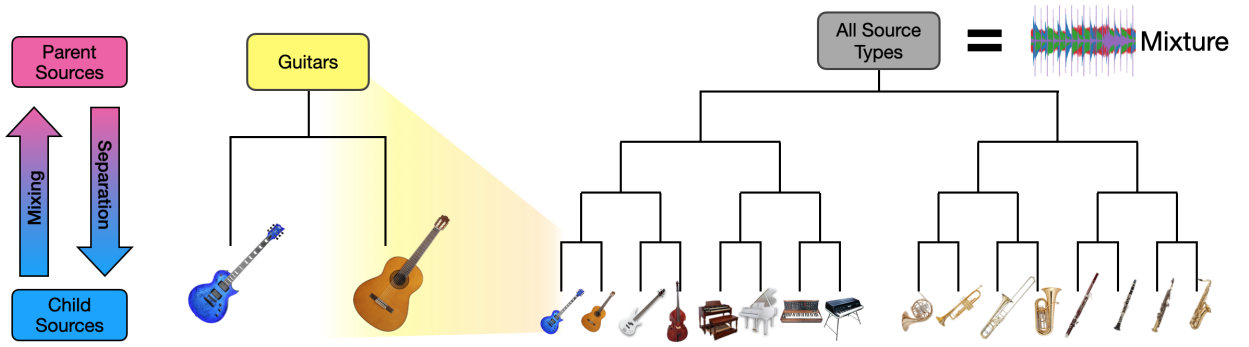


Figure 2.3. Sources can be mixed together to make parent sources and separated to make child sources (left). As such, a set of sources makes a hierarchical tree, with specific source types (like acoustic guitars) at the leaf nodes and broad source types ("All Source Types") at the root. We assume that a given audio mixture has sources that lie within this hierarchical structure.

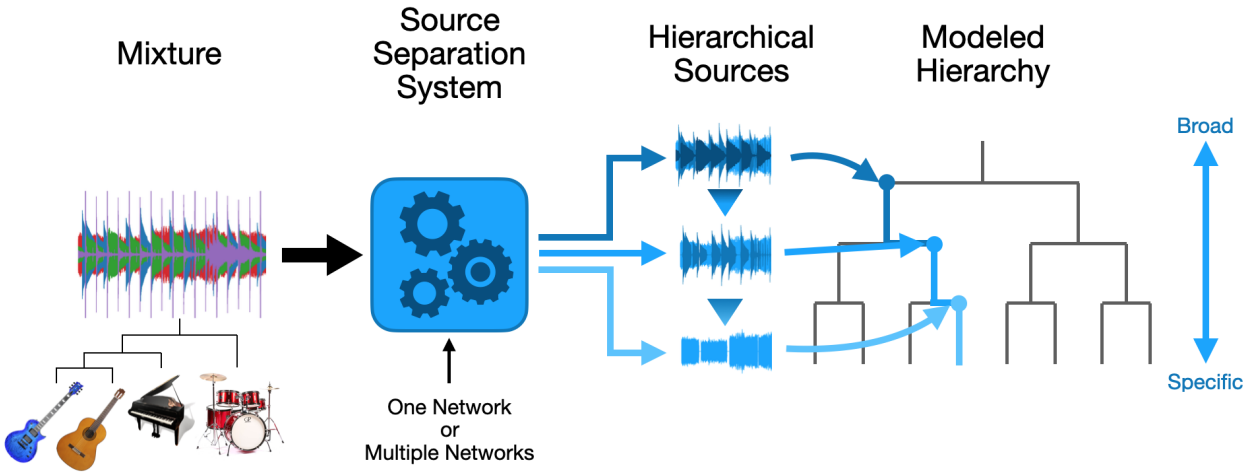


Figure 2.4. Sources in a mixture are assumed to exist within some hierarchical tree. The hierarchy at the bottom left is based on a musical instrument taxonomic scheme [287] that puts pairs of similar-sounding sources (e.g., an acoustic and electric guitar) closer together than pairs of sources that sound dissimilar (e.g., an acoustic guitar and a drum set). A hierarchical source separation is able to produce a set of hierarchical sources, which are assumed to correspond to multiple levels of the modeled hierarchy (right).

musical instruments, an auditory hierarchy can be populated by any arbitrary source types. Notice that the source types at higher levels of the hierarchy are more broadly defined (e.g., "stringed instruments") than the source types represented by lower levels (e.g., "electric guitar"). In this work, I focus on musical instrument hierarchies because of the strong precedent in both organological research and in the studio practices of recording engineers.

Importantly, I define a *hierarchical path* down a hierarchy as a sequence of source types from a source node to a child node at a lower level. The hierarchical path includes all sources from the higher node to the

lower node, e.g. [strings/keys] → [guitars] → [acoustic guitars]. In this project, I focus on source types that are explicitly chosen to correspond with established hierarchical musical instrument taxonomies (i.e., the Hornbostel-Sachs system [287]), although this hierarchical formulation can be applied to auditory mixtures with any type of source content (e.g., environmental sounds).

For a given auditory scene, I assume that the sources therein correspond to parts of this hierarchy; the root node is the mixture, which can be viewed as hierarchically composed of all sources, with each source component corresponding to its specific place in the hierarchical tree. I also define a "submix", which is the result of mixing together a set of child sources of the same parent. Given an audio mixture, we can say a separation is hierarchical if it contains a set of sources and submixes along a hierarchical path. For instance, separating out all guitars (acoustic and electric) from a mixture that includes electric guitar, acoustic guitar, piano, and drums.

An important observation about the structure of auditory hierarchies is that we can say that the amount of acoustic energy in a parent node must be greater than or equal to the amount of energy in a child node. [2] Recall that each parent node is defined as a mixture of the collection of all its child nodes. Because acoustic signals are additive[3], we can say that a parent node is a superset of any child node. Concretely, if we look at signals corresponding to two nodes, a parent node n_l and one of its child nodes n_{l-1} that exist along the same hierarchical path, then

$$E(n_l) \geq E(n_{l-1}), \tag{2.1}$$

for some measure of overall acoustic energy, $E(\cdot)$. This is because as we travel up along a hierarchical path (i.e., closer to the root), each node will be composed of more ancestor nodes. In other words, the parent node n_l contains not just the child node n_{l-1}, but the parent also contains all of the signals corresponding to the siblings of the child node (i.e., children of n_l from different paths). Each non-silent child of the parent contributes its own acoustic energy to the parent node. Therefore, we expect that as we travel up a hierarchical path that the nodes contain more acoustic energy overall. I will leverage this fact about auditory hierarchies when I design hierarchical separation systems in the next section (Section 2.3.4.2).

[2] I use the term "acoustic energy" here as a bit of shorthand for our common understanding of the "loudness" or "softness" of a sound. The loudness of a sound–a description of our perception–is related to but not equivalent to acoustic energy.
[3] Ignoring destructive interference, which is negligible for current purposes.

2.3. Hierarchical Source Separation

Recall that a standard source separation model should produce a set of desired source estimates given an input mixture. The input mixture and output sources can be represented as waveforms or spectrograms. If spectrograms are output, this typically means that the model will directly output a mask that is applied to the mix to get an estimate for each source. In this chapter, all of the models I consider use spectrograms when inputting the mixture, and directly output masks that get applied to the mixture spectrogram to produce source estimates.

Most existing source separation neural networks are not expected to be able to estimate more than one type of source at a time (i.e., a network for each source is trained independently, specializing in separating a particular source [110, 128, 148, 184, 194, 220, 232, 257, 265, 268, 269, 280]). This specialization negates the possibility that a network could learn hierarchical relationships between sources. In the cases that separation systems do make estimates for more than one source [44, 157, 250], the sources are not expected to have relationship to each other, especially relationships that might leverage known musical instrument hierarchies.

In this chapter, I will present a set of general strategies for hierarchical source separation. These strategies assume that sources in an auditory scene can be mapped onto a hierarchical tree according to the principles of auditory hierarchies outlined in the previous section (Section 2.2). Furthermore, assuming that the hierarchies meet the definition of a rooted tree, these strategies are agnostic to structure of the hierarchical tree that a separation system is modeling. All hierarchies in this chapter satisfy the definition of an m-ary tree [39, 146], i.e., a rooted tree where each leaf node has the same depth and where each non-leaf node has a maximum of m children, for some arbitrarily settable value, m (see Figure 2.13 for the full hierarchy used for the experiments in this chapter). I define a separation as *hierarchical* if a set of source estimates are related to each other such that each source in the set corresponds to one node along a hierarchical path.

There is one overarching limitation that the hierarchical strategies that I present will all have, and that is separating instances of sources which belong to the same node. For example, if someone wanted to separate a recording of two acoustic guitars (i.e., both instances of the same node) then a hierarchical separation system would be the wrong tool. In this instance, a user is better off using a separation system that is designed for multi-instance separation of the same source type, of which there are a plethora of options: this ends up being the same problem solved by speech separation systems [115, 147].

Hierarchical separation, however, solves an orthogonal problem: how do we create separation systems that can pragmatically scale to many more source types with a minimal burden on computing resources and additional data? As I will outline in this chapter, scaling source separation systems such that they can support large numbers of sources is a non-trivial task. Naïve solutions might scale linearly with respect to the number of sources, however each new source requires its own independently trained network, which comes at a substantial cost. The promise of hierarchy is that, when a system encounters a new source, it can leverage knowledge about similar sources it has seen in the past, bypassing the need to retrain a new network for every new source and massively improving the potential for scaling to support more source types.

One thing to notice about the structure of hierarchies that is important for hierarchical source separation: nodes closer to the root have more available data than nodes closer to the leaf nodes. That is to say, nodes higher in the tree are defined more broadly, and therefore more sources fit into that broader source definition. The opposite is true, as well: nodes closer to the leaves have narrower source definitions, and thus less sources fit their more limited criteria. For instance, many instrument sources (e.g., Guitars, Violins, Violas, Mandolins, etc) might be eligible to belong within a broadly-defined "Stringed Instruments" source node that resides high up in the hierarchy. However, a lower node might have a tighter definition like "Electric Guitars," and thus only a subset of the "Stringed Instrument" sources meet this stricter definition. This is important because these narrower definitions at lower hierarchical levels mean that, practically, there is less data to draw from at these lower levels. In spite of this, the hierarchy also tells us that any given leaf node is related to its parent and sibling nodes, the hope being that a network can leverage these relationships at the leaf nodes where data is sparse.

Here, I will demonstrate hierarchical source separation strategies through the domain of music, which has a long-established history of hierarchical taxonomies of musical instruments (e.g., Hornbostel-Sachs [287]), although other taxonomies exist for more general domains of sound [85]. I will instantiate these strategies by adding a series of modifications to a Mask Inference source separation network [69], all of which I will experimentally validate in Section 2.4.

2.3.1. Strategies for Hierarchical Source Separation

With an understanding of the structure of auditory hierarchies in mind, I will demonstrate four different strategies for hierarchical source separation in this dissertation. These strategies vary along two axes, namely:

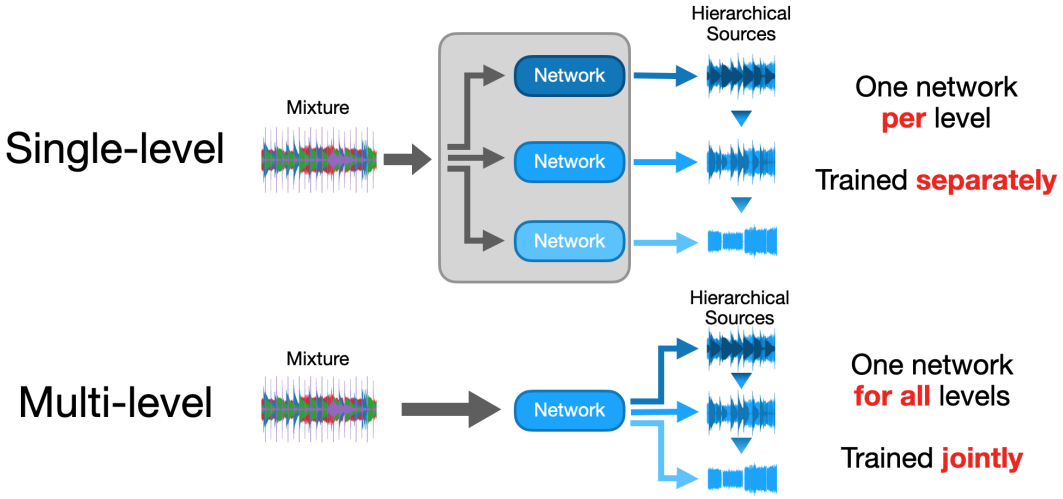


Figure 2.5. For hierarchical source separation, systems can be Single-level or Multi-level.

2.3.1.1. Single-level or Multi-level. Single-level Separation Systems have one model per *each* level in the hierarchy, trained on each level independently, leading to *a set* of independently trained models. I refer to these as "Single-level" because each model is only responsible for outputting source estimates for one level at a time. While collection of Single-level models is able to separate multiple levels of the hierarchy, I refer to the collection as a "Single-level Separation System" because each model only operates for one level.

In contrast, Multi-level Separation Systems only contain one model for *all* levels in the hierarchy, trained on all levels *at once*. In this case, just one model can cover all levels of the hierarchy, thus I refer to the system as Multi-level. See the Figure 2.5 for an example of Single-level (top) and Multi-level (bottom) hierarchical separation systems with a hierarchy containing three levels.

2.3.1.2. Single-instrument or Multi-instrument. Single-instrument Separation Systems are systems where the constituent models are specialized for separating just one source or down one hierarchical path. In this case, the hierarchical path that the system can separate must be decided upon prior to training. Recall that a path from the root to the leaf node can be uniquely defined if given only the leaf node. Therefore, in this case a Single-instrument Separation System is specialized to separating just one type of instrument, as defined by the leaf node, and it produces separation estimates at many levels of the hierarchical path. As such, in this chapter I refer to such models as Source-Specific Separation (SSS) models.

On the other hand, Multi-instrument Separation Systems are where models are able to separate multiple paths in a hierarchy. They are not specialized to any particular hierarchical path, and are able to flexibly switch which hierarchical path is returned by the system at inference time. One naïve strategy to switch

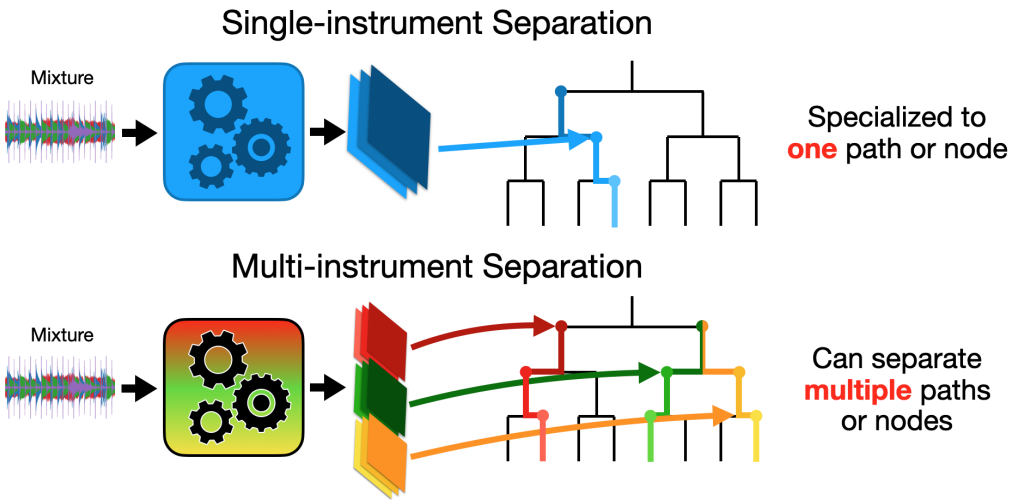


Figure 2.6. For hierarchical source separation, systems can be Single-instrument or Multi-instrument.

which source is returned is to separate every leaf node (and therefore every path) and let the user choose their source, post-hoc. In the upcoming sections, I will describe why this strategy is suboptimal. Instead, it is more practical to have a control mechanism for what the system separates. With this in mind, I instantiate the Multi-instrument strategy using Query-by-Example (QBE) networks, which use an audio query to determine which hierarchical path to separate along. See Figure 2.6 for examples of Single-instrument (top) and Multi-instrument (bottom) systems with respect to a hierarchy.

In total, these two axes yield the following four strategies: **Single-instrument Single-level**, **Single-instrument Multi-level**, **Multi-instrument Single-level**, and **Multi-instrument Multi-level**. These strategies are shown in Figure 2.7. The four strategies I will outline in this chapter are a hierarchical generalization of existing non-hierarchical separation strategies that the research community has been developing for years. Researchers have primarily focused on solutions for the **Single-instrument Single-level** strategy [35, 127, 169, 263, 265, 268, 269]. However, as I will discuss, solutions for this strategy do not scale to large numbers of source types, and I will argue that hierarchical extensions to the **Single-instrument Single-level** (i.e., the other three strategies I develop here) are a viable option for achieving this desired scalability. In Section 2.4.0.2 I will experimentally verify these claims.

These four strategies are intentionally broad; they are flexible by design, omitting many implementation details such that there could be many possible solutions. Here I will provide but *one* set of solutions by modifying a standard source separation network architecture [69] (i.e., my "backbone" separation system, which is technically a **Single-instrument Single-level** system for a hierarchy that has only one node; in

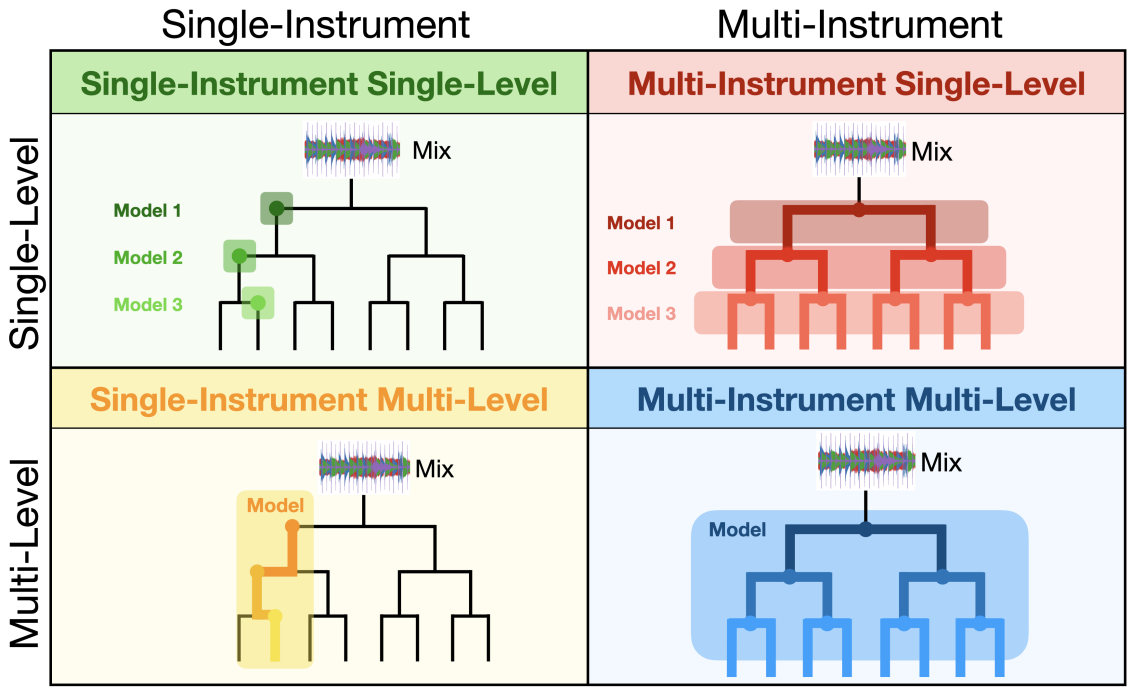


Figure 2.7. The four strategies for Hierarchical Source Separation vary along two axes: whether the system can separate multiple instrument (Multi-Instrument) or only one instrument (Single-Instrument), and whether the system can separate at multiple levels (Multi-Level) or just one level (Single-Level).

other words, a system designed to separate only one node), and discuss how to avoid naïve pitfalls towards the ultimate goal of a hierarchical system that can scale to separating dozens of sources. None of the hierarchical strategies in this chapter are specific to my choice of neural network architecture; here I reframe the *problem* while making a minimal set of choices about the neural network architecture as possible. Thus, the proposed hierarchical separation strategies are compatible with any separation architecture that is able to produce multiple source estimates at once (e.g., U-Nets [34, 35, 111, 127, 133, 148], other spectrogram-based separation systems [265, 268, 269], or even waveform-based separation systems [44, 232]). While I believe that the lessons sketched out here are widely applicable to other neural network architectures, the demonstration of these techniques in that context is left for future work. In the rest of this section I will demonstrate four ways that this standard architecture can be modified to adhere to each of the four hierarchical strategies respectively.

2.3.2. Backbone Separation System

Before detailing these four strategies, I will cover the separation system that serves as a backbone separation system for the rest of this chapter. Throughout the next few sections, I will build upon this backbone system, tailoring it to the hierarchical problem through the different strategies outlined above.

The backbone separation system is based off a now-standard source separation neural network architecture [69], which works as follows: a mixture spectrogram is input into the network and it is trained to produce a mask that is element-wise multiplied to the spectrogram of the input mixture (see the section on Masking, Section 1.2.1). The network itself is a stack of Bidirectional Long Short-Term Memory (BLSTM) layers [94, 118] (see Appendix Section 6.1 for more details about BLSTMs), with a final Fully Connected layer with a sigmoid activation function that produces the mask (a sigmoid compresses input values into the range $[0.0, 1.0]$, which is the same range that we need to make a mask). Implementation and hyperparameter specifics will be provided when I discuss the experimental design in Section 2.4.2.

Specifically, source separation networks based on mask inference typically attempt to estimate a real-valued mask $\hat{M}_c \in \mathbb{R}^{F \times T}$ for a single target source type c by minimizing an objective function between a ground truth, real-valued magnitude source spectrogram, $|S_c|$, and the source estimate obtained by applying the estimated mask to the mixture spectrogram, $|\hat{X}_c| = \hat{M}_c \odot |X|$. Here, $X \in \mathbb{C}^{F \times T}$ represents the complex-valued STFT of the input mixture, and the operators $|Y|$ and $\angle Y$ denote getting the magnitude and phase components of a complex-valued STFT, $Y \in \mathbb{C}^{F \times T}$, respectively. For an overview of mask-based source separation, see Section 1.2.1. A simple loss function we can use is either L1 ($\mathcal{L}_{\text{L1}}$), or MSE ($\mathcal{L}_{\text{MSE}}$) loss between an estimated spectrogram, $|\hat{X}_c| = \hat{M}_c \odot |X|$, and ground truth spectrogram, $|X_c|$ for some source c:

$$\mathcal{L}_{\text{L1}} = \left\| \hat{M}_c \odot |X| - |S_c| \right\|_1, \qquad \mathcal{L}_{\text{MSE}} = \left\| \hat{M}_c \odot |X| - |S_c| \right\|_2^2, \tag{2.2}$$

where the network's spectrogram estimate for source c is given by $\tilde{S}_c = \hat{M}_c \odot |X|$, and $\odot$ denotes element-wise product.

Another commonly used objective function, used in this work, is the truncated phase sensitive approximation (tPSA) objective [69], which is a variant of the basic L1 loss between the estimated spectrogram, $|\hat{X}_c| = \hat{M}_c \odot |X|$, and ground truth spectrogram, $|X_c|$ for some source c. This loss function, $\mathcal{L}_{\text{tPSA}}$, is defined

like:

$$\mathcal{L}_{\text{tPSA}} = \left\| \hat{M}_c \odot |X| - \text{clamp}\left(|S_c| \odot \cos(\angle S_c - \angle X)\right) \right\|_1, \tag{2.3}$$

where $\text{clamp}(a) = \min(\max(a, 0.0), |X|)$ is a truncation function ensuring that some input value, a, falls within the range $[0.0, 1.0]$, which is the same range as the sigmoid activation function, the non-linearity applied to the last layer of the network. The terms inside the clamp function work as follows: $\angle S_c - \angle X$ is the phase difference between the ground truth source S_c and the input mixture X at every time frequency bin. This phase difference is wrapped in a cosine before being element-wise multiplied by the ground truth spectrogram. This means that if the phase of the ground truth source and the mixture are perfectly aligned for a given (t, f) bin (i.e., exactly 0° phase difference) the loss for that bin is given a weight of exactly 1.0. The opposite is also true, if the phases are 180° offset, then the loss is scaled by -1.0, however the truncation function converts negative values to 0.0. So in the case that the phases are 180° offset, then that (t, f) bin contributes 0.0 to the loss. This is why this loss is called phase *sensitive*: it accounts of phase offsets that occur between the ground truth source and mixture.

At inference time, the estimated mask $\hat{M}_c$ is element-wise multiplied with the original mixture spectrogram X to obtain an estimate for the target source S_c. As is common, the phase from the mixture spectrogram $\angle X$ is then combined with the estimated source spectrogram $\hat{X}_c = S_c \odot e^{j \cdot \angle X}$ to get a complex-valued Short-time Fourier Transform (STFT), which can in turn be converted to a waveform via an inverse STFT. The resulting waveform can then be played back through speakers for human listeners.

With a minimal modifications, the backbone Mask Inference system will cover the requirements needed to create Single-instrument separation systems, in both the Single-level and Multi-level cases. With further modifications of this backbone system, I will also develop a Query-by-Example (QBE) system for the Multi-Instrument strategies. As mentioned previously, many neural network-based source separation systems could be used as a backbone system for this work. I specifically chose the Mask Inference architecture as a means to demonstrate the hierarchical separation strategies because of its relative simplicity as compared to more modern source separation systems.

2.3.3. Single-instrument Single-level: Source-Specific Separation (SSS) systems

Up until a few years ago, almost all neural network-based source separation systems were specialized for separating only a single type of musical instrument source. In practical terms, this could mean that a network might output just one real-valued mask $\hat{M}_c \in \mathbb{R}^{F \times T}$ in an effort to estimate just one single target source type c. From a hierarchical perspective, one might say that these systems are responsible for a single node (i.e., one node at one level) in a hierarchy. Because such systems are typically specialized for a specific source type, I call them Source-Specific Separation (SSS) system.

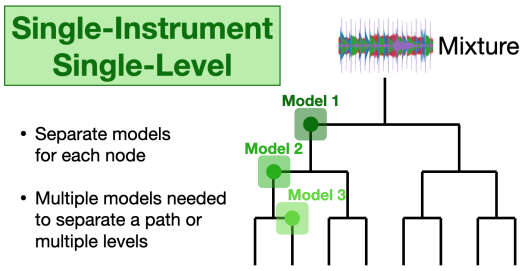


Figure 2.8. Single-instrument Single-level separation designates one model per node in a hierarchy. Source-Specific Separation (SSS) systems are an instantiation of this strategy.

However, I aim to investigate how we might reap the benefits that a hierarchical perspective affords us. In this section, I set out to answer how we can imbue existing SSS system with a sense of hierarchy.

As a first naïve strategy for building hierarchical SSS networks, it is possible to train a set of single networks, each of which output a single node at a given level of the hierarchy. Each such single-level network can be trained to minimize the tPSA objective above (Equation 2.3), where the target source is the component corresponding to the targeted source type in the hierarchy within the mixture. Each of these networks outputs a single mask for its targeted source type, and they are trained independently of each other. In order to cover all nodes in the hierarchy, this system would require the same number of networks as there are nodes.

This naïve strategy has a number of serious drawbacks, which all stem from the fact that the networks are all independent. The huge impracticality of training each network independently notwithstanding, this strategy does not actual benefit from hierarchy. Because each node is the responsibility of one independent network, the networks cannot leverage shared knowledge of related source types. In other words, the resulting networks would have no awareness of parent or child nodes along the same hierarchical path of a given node (i.e., moving vertically in the tree like moving from the "acoustic guitars" node to the "stringed instruments" node) nor do they have any knowledge of siblings in other hierarchical paths (i.e., moving horizontally in the

tree like moving from the "acoustic guitars" node to the "electric guitars" node). To implement this strategy using a system like our Mask Inference backbone system, we would need to train dozens of networks to cover just the *leaf nodes* of the Hornbastel-Sachs taxonomy, and furthermore, adding a new source type requires training a new network.[4] Of the benefits that we expect a hierarchical separation to provide–namely, the ability to scale to many source types and being able to flexibly separate new source types–this naïve strategy has none.

Regardless of its issues, this strategy makes up the first of the proposed source separation strategies: the **Single-instrument Single-level** strategy. So named because each network is responsible for separating at just one level, for just one instrument. Situating this chapter in the broader source separation literature, this **Single-instrument Single-level** strategy is far and wide the prevailing strategy for training source separation systems [110, 128, 148, 184, 194, 220, 232, 257, 265, 268, 269, 280].

While the **Single-instrument Single-level** strategy does have downsides, it might be reasonable to expect that, because each network is specialized to separate exactly one node, its separation performance is better compared to a system that is designed to separate multiple nodes. However, later in this chapter, I will show results that counter this intuition; providing hierarchical cues to a network can actually *improve* its separation performance over the **Single-instrument Single-level** strategy.

2.3.3.1. Modifying the Backbone System for Single-instrument Single-level. Because this strategy is conceptually the simplest, it therefore requires minimal changes to the backbone system. The only practical difference is that instead of training one system, we train multiple systems as described in Section 2.3.2. As we develop more sophisticated ways of taking advantage of the hierarchy, the changes to the backbone system will become more substantial.

[4]Certain speech separation network architectures may not have the same rigidity; see, for example, embedding space separation architectures like Deep Clustering [113] (discussed in Chapter 3) or Deep Attractor Networks (DANs) [31]. Though these networks are designed to separate two instances of the same source, e.g., two speakers, there has been some work applying Deep Clustering to music [17, 164, 179, 241]. However, once you train these nets to separate multiple instruments at once, they cease to be Source-Specific Separation systems, which is the topic of this section.

2.3.4. Single-instrument Multi-level: Hierarchical Source-Specific Separation

As just discussed, the **Single-instrument Single-level** strategy involves *independently* training a set of networks to cover all nodes in a hierarchy, each network only responsible for a single node. This strategy does not leverage any knowledge about how auditory hierarchies are defined. A trivial way to remedy this would be to *jointly* train a set of models for all of the leaf nodes. Such a system could recompose or "re-mix" the leaf sources according to the hierarchy, training the networks by combining loss functions for all nodes in the hierarchy. However, any sufficiently realistic hierarchy likely contains dozens or hundreds of leaf nodes, leading to memory and computation issues, as before, as well as difficulties balancing the contributions of all the losses.

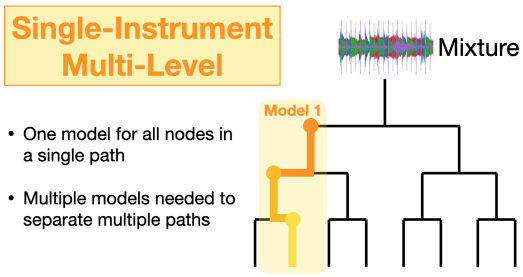


Figure 2.9. **Single-instrument Multi-level** separation designates one model that is charged with separating at multiple nodes down a single hierarchical path. Hierarchical Source-Specific Separation systems are an instantiation of this strategy.

In order to develop more feasible strategies for hierarchical separation, in this section I will restrict the conversation to Hierarchical *Source-Specific* separation strategies before discussing how to scale the lessons from these strategies to all paths in a hierarchy in the next section. In other words, in this section the discussion is limited to the hierarchical separation of a single path down a hierarchy.

As a means to slightly mitigate our scaling issues, consider instead a single network that outputs L source estimates, one estimate for each of the L levels along a single path down the hierarchy. This reduces the number of networks we have to train to cover the whole hierarchy from one per node to one per *leaf* node. This strategy enables us to train a single network for a whole path down the hierarchy (e.g., [strings/keys] $\rightarrow$ [guitars] $\rightarrow$ [acoustic guitars]), instead of one per each node. Here, our system now makes estimates for all of the sources in the hierarchical path in one fell swoop, from the broadest to narrowest source types.

While at first glance this may seem like a negligible reduction in the number of networks we must train to cover a whole hierarchy, the actual reduction is dependent on the branching factor of the hierarchy we wish to separate. In fact, many musical instrument hierarchies that we care about have a branching factor that produces a material reduction in the amount of networks that we must train when actualizing a **Single-instrument Multi-level** strategy to cover the whole hierarchy. For example, the musical instrument hierarchy

used in this chapter covers most Western instruments (See Figure 2.13) and it has 28 nodes, but only 17 leaf nodes, leading to a 39% reduction in the number of networks we need to train (and therefore computing resources needed, e.g., training time, memory, # parameters, etc) by following a **Single-instrument Multi-level** strategy. Even with these reductions, training one network per leaf node is a considerable undertaking if we want to cover the whole hierarchy.

However, when we again restrict the conversation to only considering a single path in a hierarchy the reduction between the **Single-instrument Single-level** strategy and **Single-instrument Multi-level** strategy is even more substantial: because we now only require one model for a whole path, this leads to $\frac{1}{L}$ the number of networks (compute/training time/parameters), for a hierarchy with L levels. In my experiments $L = 3$, which leads to only using a third of the resources for hierarchical separation.

Most importantly though, is that the **Single-instrument Multi-level** strategy is the first time that a system can take advantage of hierarchy during training. By having it output multiple levels along a hierarchical path, the system is given a notion of auditory hierarchies. Whereas the independently trained networks have no incentive to learn anything about any other nodes in a hierarchy, this joint training fundamentally enables it to "see" more than one node in its learning process. While this benefit may seem more theoretical than the reduction in resources described above, it is actually a key step in our goal to make flexible, scalable source separation systems. In fact, these benefits are more than theoretical; I will experimentally demonstrate that inducing hierarchical training like this leads to a considerable gain in separation performance, especially if we wish to separate very fine grained sources where we might have a limited amount of training data. Leveraging our understanding of auditory hierarchies during training is a foundational step in our narrative as we build up to being able to separate more source types.

2.3.4.1. Modifying the Backbone System for Single-instrument Multi-level Separation. Practically, the backbone Mask Inference system now outputs L real-valued masks $\hat{M}_l \in \mathbb{R}^{F \times T}$ for levels $l = 1 \dots L$ of a hierarchical path. With this simple change, the network is in charge of separating a source along with its parents and/or children. The network can now potentially leverage its learned knowledge for mask $\hat{M}_a$ when predicting $\hat{M}_b$ (for $a \neq b$), all for the negligible[5] cost of a larger output size (assuming that the number of levels L is relatively small, e.g., $L = 3$ in this chapter).

The resulting network is still considered a Single-instrument system because its specialized to separating along just one hierarchical path, however it is now responsible for multiple levels of a hierarchical path

[5]In general, size(`mask_output_layer`) << size(`rest_of_network`).

instead of just one at a time as with the previous strategy. As such, I refer to this strategy as a **Single-instrument Multi-level** strategy. Systems set up in this way are still specialist systems to an extent–that is, yes, they are designed to separate exactly one source type as before, but now the definition of that source type changes as one moves along the hierarchical path. Instead of just separating an acoustic guitar, this system makes estimates at all levels of the hierarchical path: [strings/keys] → [guitars] → [acoustic guitars]. Still, we would need many of these systems to get complete coverage of any sophisticated musical instrument hierarchy. Prior to addressing the hierarchy coverage problem, though, I will first discuss one additional technique for biasing source separation systems toward hierarchical sources.

2.3.4.2. Constraints on Hierarchical Masks. Assuming the components of a mixture exist in some hierarchy, it is possible leverage knowledge about the hierarchical structures of auditory scenes to impart constraints on the network. For instance, we can use the relationships defined in Section 2.2 on auditory hierarchies to require the set of masks produced by a multi-level hierarchical network to follow the same structure as the hierarchy, namely that masks at higher levels be composed of masks at lower levels. As before, this would require us to output masks for every node in the hierarchy, which, again, is infeasible for any sufficiently realistic hierarchy.

Instead, we consider imposing a hierarchical constraint that does not depend on knowledge of the whole hierarchy and instead hinges on another aspect of auditory hierarchies. The proposed hierarchical constraint leverages the fact that there should be more acoustic energy in broader source types (Equation 2.1) and requires that masks at higher levels in the hierarchy must apportion at least the same amount of energy as masks at lower levels. More precisely, the mask at level l is set as

$$\hat{M}_l = \max_{TF}(\hat{M}'_l, \hat{M}_{l-1}), \tag{2.4}$$

where $\max_{TF}$ is applied element-wise to every time-frequency bin, and $\hat{M}'_l$ is the mask estimate output by the network for level l for $l > 1$. Intuitively, this constraint ensures that there is the same or more acoustic energy [6] in mask estimates at more coarse levels of the hierarchy than there is at more specific levels. This reflects our knowledge of the structure of auditory hierarchies discussed in Section 2.2.2 that nodes closer to the root of the hierarchy correspond to more broadly defined source types (e.g., stringed instruments) and

[6] A note on this: it is very common in studio recordings to apply an audio effect known as Dynamic Range Compression to either individual instruments or the whole mix or both. This effect typically makes the quiet parts of a recording much louder, essentially reducing the volume between the quietest and loudest parts of a signal. Although the experiments presented in this section do not account for the usage of Dynamic Range Compression (i.e., it is not used in any recordings I study), how this effect will interact with the proposed Hierarchical Constraint is a question for future work.

therefore must have more or exactly the same amount of acoustic energy than children of that node (e.g., Acoustic Guitars).

2.3.5. Multi-instrument Single-level: Query-by-Example Networks

The approaches described above are problematic if one wants to separate many different instrument families in an instrument hierarchy: the trivial solutions in the previous section involve many networks and do not scale feasibly, so in order to obtain workable solutions we restricted our discussion to single-family instrument sources, or Source Specific Separation. In other words, Source Specific Separation strategies can only learn a *single* path down an instrument hierarchy at a time, but how do we develop strategies that learn *multiple* paths at once?

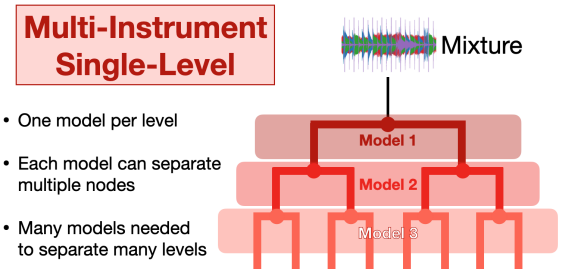


Figure 2.10. Multi-instrument Single-level separation can separate any node in the hierarchy with requiring multiple models, each responsible for only one level. Query-by-Example networks (without any hierarchical extensions) are an instantiation of this strategy.

Whereas the previous Source-Specific Separation systems embodied the Single-instrument strategy for learning hierarchy, in this section I explore ways actualize the Multi-instrument strategy for learning hierarchy. I posit that capturing multiple paths with one system (i.e., Multi-instrument) is crucial for enabling the system to learn relationships between different instrument families (e.g., the relationship between bowed strings (a violin) and plucked strings (a ukulele)).

So, how can a Multi-instrument separation system work? To simplify, let us restrict this current discussion to just single-level systems. As before, we begin by imagining a naïve solution to this problem: a system that outputs a source estimate for every source node at the current level. For a small number (e.g., 4) of source types this is completely feasible; of the handful of multi-instrument source separation systems that do exist [44, 178, 234], this is the solution they employ. Yet, as we scale this system to dozens of source types, we are bound to encounter pragmatic issues with respect to memory and computational resources. Even if we are able to palliate those problems, trying to find the correct way to balance the losses for dozens of sources during training poses a formidable challenge: in practice, few recordings have dozens of taxonomically distinct instruments playing simultaneously which leads to many training examples with silence for

most of the output sources. A trivial solution with the lowest loss that a network might find would be to always output silence, thus not separating anything. This naïve solution poses dilemma after dilemma.

Another viable option is to not expand the output dimensionality of our network for all possible sources, but instead repurpose a single output such that it can flexibly output any source. In other words, instead of having a dedicated output slot for each source type, only have one output slot and have a way to control which source that output slot separates. To accomplish this, we can use a conditioning signal to control the separation model. This input conditioning effectively acts as a control knob, allowing us to change the output of a network even if the other inputs do not change. Resulting networks usually make adaptations to their architecture to accommodate the additional conditioning information, e.g., a conditioning signal is fed to FiLM layers [216] that learns how to alter the neural net's layers according to the conditioning. In previous source separation work, researchers have explored making this conditioning input an n-way switch (represented as a 1-hot vector) determining a choice between a fixed number of source types (typically 4 in the literature) [34, 192, 232, 241, 251].

In the interest of flexibility, it would be nice if the control that a conditioning knob offered us was not fixed to a predetermined number of sources. Rather, in pursuit of our goals, it would be preferable if such a system could incorporate new source types (after it has been trained). Perhaps the most flexible control knob possible would be one that is in the same parameter space that the mixture is in, that is to say, the audio space. To that end, I propose a Query-by-Example (QBE) network architecture. QBE networks ingest a mixture and an audio example (i.e., the query) of the desired source to separate from the mixture [87, 157, 160, 183]. The query is an additional audio input to the system that used to determine which source will be separated from the mixture. Therefore, it is expected that the content of the query be from the same instrument family as the source we desire to separate, and that the query has only content from the desired source and no content from any other source; in other words, the query should be a recording of an isolated instrument source. The audio content in the query does not have to come from the mixture; in fact, for ease of use, it most likely comes from a completely different recording of the instrument targeted for separation (because if we already had access to an isolated recording from within the mixture, we would not need source separation at all!).

For example, given a recording of a mixture that contained an acoustic guitar, a bass guitar, and a drum set, if a QBE separation system were given a different recording of a different acoustic guitar as a query, the QBE system would be expected to produce a separation of the acoustic guitar as heard within the mixture.

Similarly, if the QBE system were given the same mix, but instead given a different recording of a drumset as a query, we would expect a separation of the drums from the original mixture. The query input guides the system and tells it which source it must separate from the mixture.

Recall that despite attempting to scale to multiple instruments, this examination so far has been restricted to Single-level separation, i.e., **Multi-instrument Single-level** (this discussion will widen to Multi-level separation in the following section). This means that the output level for every **Multi-instrument Single-level** system must be predetermined when training it, and therefore to cover multiple levels of a hierarchy we would need to train multiple Single-level QBE separation systems. All previously proposed conditioned separation systems (1-hot [34, 192, 232, 251] or QBE [87, 157, 160]) operate at a single level, even if not explicitly declared as so. It is also worth mentioning that when we *do* view the target source and queries as part of a hierarchy (even in Single-level systems), it is possible that the query input could come from a different hierarchical level than the expected source output. This could lead to ambiguities if the query audio comes from a broader level of the hierarchy than the desired target source. For example, consider the case where the query audio is from the [all guitars] node but the expected source output is [acoustic guitars], which is a more specific source type than the query. In this case, the target is not well defined given the query (the target source could be *any* guitar sound corresponding to a child node of [all guitars], not just [acoustic guitars]). Perhaps there are cases where this ambiguity is desired, but for the current work I ensure that all queries come from nodes that are at or below target source nodes in the hierarchy. In this chapter, I set all queries such that they originate from the the leaf nodes of the hierarchy (the leaf nodes I use are labeled "Level 1 - Child" in Figure 2.13) regardless of the expected output level of the target source. The leaf nodes implicitly define a unique path from the root to the leaf, clearing up any ambiguities with queries.

This marks an important step in the journey so far: we have further reduced the number of networks we need to train to separate sources from every node in a hierarchy compared to our first naïve **Single-instrument Single-level** strategy of training a network for each node, and even compared to the previous **Single-instrument Multi-level** strategy where we would need to train a network for each leaf node. Following the current **Multi-instrument Single-level** strategy, we now only require one QBE network for each *level*. Because it is safe to assume that the number of levels is considerably smaller than the total number of nodes, this leads to a much more lightweight system. At this point many of the high-level considerations about constructing a **Multi-instrument Single-level** system have been laid out. The rest of this section will be

devoted to implementation details about how to modify the backbone separation system such as to make it a QBE separation system.

2.3.5.1. Modifying the Backbone System for Multi-instrument Single-level Separation. As mentioned, a QBE system should be designed such that the query input produces a conditioning vector to the rest of the network that communicates which source to separate. The query audio could be either in the time domain (i.e., a waveform) or in the time-frequency domain (i.e., a spectrogram), however, given that the backbone network used here–a Mask Inference system–operates in the time-frequency domain, I therefore design a proposed QBE system such that the query input is also in the time-frequency domain.

The proposed QBE separation system builds on the backbone separation system by adding a query network to the backbone Mask Inference network. The job of the query net is to calculate a query vector, called the "query anchor", $A_q \in \mathbb{R}^k$ in a k-dimensional embedding space given a real-valued input query spectrogram $Q \in \mathbb{R}^{F \times T_q}$, of shape F frequency bins and T_q time steps. The size of the dimension k is a settable hyperparameter. The query anchor, A_q is a weighted sum of the query embeddings, $V_{q,i} \in mathbbR^{k \times F \times T_q}$, produced by the query network at each time-frequency bin $i = (f, t)$ of the query spectrogram space:

$$A_q = \frac{\sum_i P_{q,i} V_{q,i}}{\sum_i P_{q,i}}, \tag{2.5}$$

where $P_q \in \mathbb{R}^{FT_q}$ is a query presence vector for query Q, defined such that $P_{q,i} = 1$ if the magnitude at time-frequency bin $i = (f, t)$ is above a loudness threshold (set to -60 dB from the maximum here), and 0 otherwise. The intuition behind the query presence vector P_q is to ensure that quiet parts of the query spectrogram do not influence the query anchor; for a given isolated instrument recording there might be many more quiet time-frequency bins than loud ones.

The calculated query anchor A_q is then combined with the mixture spectrogram and both are input into the masking network, which produces a *single* mask, $\hat{M} \in \mathbb{R}^{F \times T_m}$, that is applied to the mixture spectrogram, $X \in \mathbb{R}^{F \times T_m}$, to create an estimate for a *single* desired source, according to the procedure described in Section 2.3.2. Specifically, the query anchor A_q is concatenated along the frequency dimension of the mixture for each time frame $t = 1 \ldots T_m$ in the mixture spectrogram, $X \in \mathbb{R}^{F \times T_m}$. This is shown in Figure 2.15. It is this combined anchor/mixture spectrogram matrix that is input to the Mask Inference network to produce a mask. The final loss function is the tPSA loss from Equation 2.3. Some prior work has applied a loss function directly to a query embedding space (or "query anchors" in this work) [157], however I will show that the proposed system learns a meaningful embedding space without a specific loss on the

query anchors. Also, note that the length of the time dimension for the mixture spectrogram T_m does not have to equal the length of the time dimension for the query spectrogram T_q, i.e., the mix and query can be of different lengths. In this work, the network architecture for the query network mirrors the architecture for the masking network: the query net is a smaller set of Bidirectional Long Short-term Memory (BLSTM) layers followed by a Fully Connected layer with no activation function that produces the k-dimensional query anchor A_q. The specifics of these networks will be provided in the Experiments section of this chapter.

2.3.6. Multi-instrument Multi-level: Hierarchical Query-by-Example

Despite the motivation for Query-by-Example (QBE) networks being that we can learn the relationships between different families of sources (i.e., Multi-instrument systems), thus far the discussion of QBE networks has precluded any talk of hierarchy. At this point, we have all of the puzzle pieces ready to construct the a system that embodies a solution for the Multi-instrument Multi-level strategy. In this section, I will develop the previous discussion of Single-level QBE networks to include the hierarchical case, following the same line of reasoning I employed when moving from the Single-instrument Single-level to the Single-instrument Multi-level strategies.

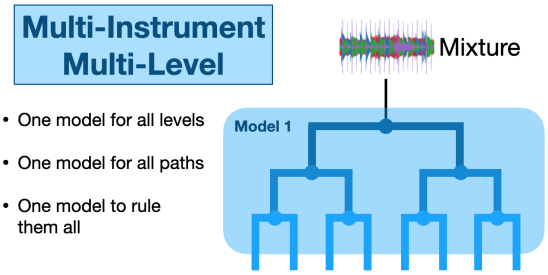


Figure 2.11. Multi-Instrument Multi-Level separation contains only one model that can separate a node at any level along any hierarchical path. Hierarchical Query-by-Example networks are an instantiation of this strategy.

Just like how Single-instrument Multi-level systems extended a Single-instrument Single-level system to leverage a hierarchy during training, so will the Multi-instrument Multi-level strategy extend the Multi-instrument Single-level strategy. In practice, this involves extending the Single-level QBE system to output multiple source estimates along a path hierarchy as determined by a query input. Because this system is a Multi-level system, it outputs source estimates for all nodes down the path specified by the query input. To get an estimate for just one of specific source node from a path defined by the query, one simply ignores the extraneous estimates. Just as with the Multi-instrument Single-level QBE system, the queries are always

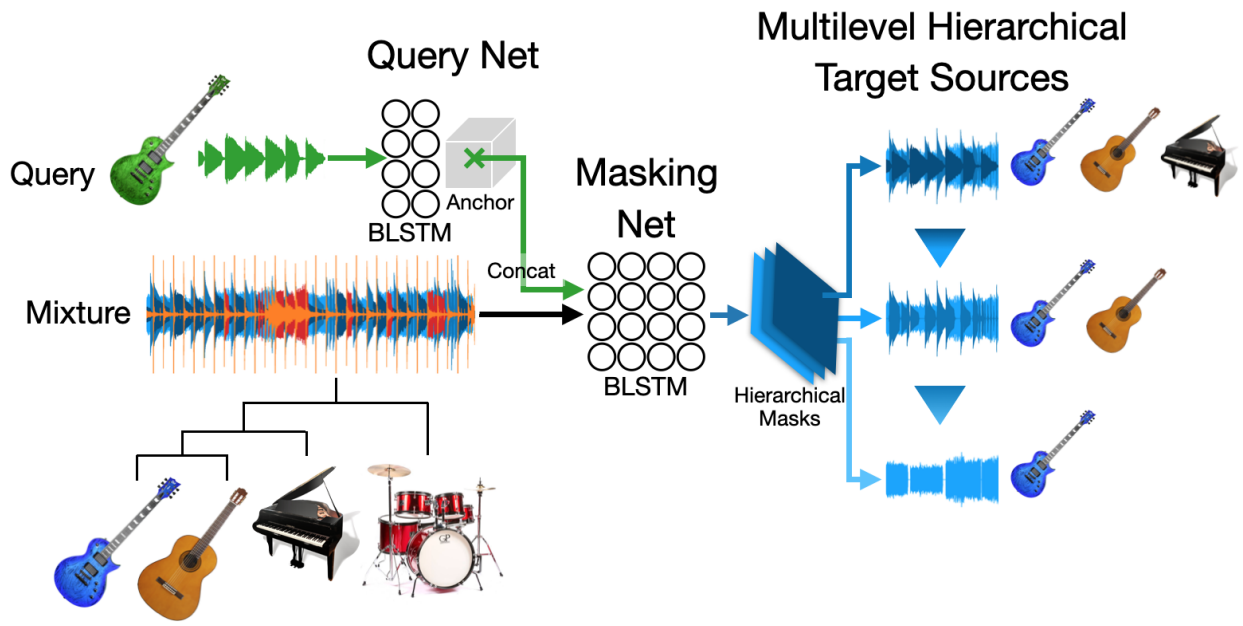


Figure 2.12. A proposed Hierarchical Source Separation system, a Query-by-Example system, is able to separate sources down any path of the modelled hierarchy based on an audio query input. Sources in the input mixture ("Mixture" left, middle) are assumed to have a hierarchical taxonomic relationship to each other (bottom left). If given audio of the green electric guitar as a query ("Query", top left), the system will produce a hierarchical set of separation estimates (right) that includes the sources that are the closest match to the green electric guitar according to the hierarchy. In this case, the separation includes the piano and both guitars at the broadest level of separation (top right), and only the blue electric guitar at the most granular level (bottom right).

expected to be from the leaf nodes to avoid any ambiguities. Finally, as discussed previously with the **Single-instrument Multi-level** strategy, we can enforce a hierarchical constraint ensuring that there is at least the same or more energy in source estimates that correspond to more coarse-grained nodes (as in Section 2.3.4.2).

This is the culmination of all of the hierarchical discussion so far. With this final piece we now have a separation system consisting of just one neural network that can separate any node in a hierarchy. Whereas in the first **Single-instrument Single-level** strategy, we required one network per node, now the **Multi-instrument Multi-level** strategy enables us to use one network for every node in the hierarchy! Any source type can be separated with one network by following the **Multi-instrument Multi-level** strategy. We have achieved peak flexibility!

2.3.6.1. Modifying the Backbone System for Multi-instrument Multi-level Separation. As with Source-Specific Separation using Mask Inference networks, extending QBE networks hierarchically boils down to two essential strategies: Single-level and Multi-level. In the Single-level case the masking subnetwork of the QBE

system produces just *one* mask, $\hat{M}$ for one specific level of a hierarchy, l. As before, *multiple* Single-level QBE networks are required to separate down a hierarchical path. In the Multi-level case, the masking subnet of the QBE system produces multiple masks, $\hat{M}_l$, one for each level $l = i \dots L$ of the hierarchy. Thus, only *one* Multi-level QBE is needed to separate down a hierarchical path. As before, because the Multi-level QBE system outputs multiple masks at once, we can apply the hierarchical constraint described in Section 2.3.4.2. QBE networks also have an additional required input, namely the query. For hierarchical QBE in both the Single-level and Multi-level case, we assert that the query input is always a leaf node in the hierarchy, because a leaf node is the only node that unambiguously determines a path from the root node (i.e., the mix). The full Multi-level Hierarchical QBE system is depicted in Figure 2.12.

2.4. Experimental Design

I designed and executed a set of experiments to determine the validity of the proceeding strategies and respectively proposed systems for hierarchical source separation. The goal of these experiments is to understand how well these systems work in a hierarchical scenario. As with all of the work in this chapter, I specifically look at hierarchies of musical instruments.

Specifically, these experiments are intended to answer the following questions:

(1) Is the hierarchical constraint helpful or harmful when doing Multi-level separation?

(2) Which hierarchical strategy performs better: Single-level or Multi-level systems?

(3) Can hierarchical separation strategies ameliorate the increased need for data for the most-specific source types?

For each of these three questions, I designed a corresponding experiment. In these experiments, I present answers to these questions for both Single-instrument and Multi-instrument separation. For the Single-instrument results, I pick one hierarchical path and fix it for all experiments (i.e., the leaf node is unchanging and determined ahead of time for these results), whereas on the other hand the Multi-instrument results cover all nodes of the hierarchy.

I want to take special care to emphasize that because all four hierarchical strategies in this chapter represent a new framing of the source separation problem, there is no prior work against which I can directly compare. As such, I will instead compare the Multi-level and Single-level strategies with each other in these experiments.

2.4.0.1. Does the Hierarchical Constraint help or hurt Multi-level separation systems? This question is restricted to Multi-level systems specifically, and concerns whether the hierarchical constraint outlined in Section 2.3.4.2 enables a model to learn to separate with better performance (according to SI-SDR). For this experiment, I look specifically at Multi-level systems and fix every aspect about the model architectures and training procedures except whether or not the hierarchical constraint is used. This experimental question will produce pragmatic information about how to best design Multi-level separation systems. It also hints at how we can best provide more types of inductive biases for hierarchical source separation.

2.4.0.2. Do Single-level or Multi-level systems separate better? To answer this question, I consider a standard source separation architecture and fix every aspect of training except whether the system was trained via Single-level or Multi-level strategy. In practice, the Single-level systems are composed of multiple models that each output sources at only one level, and the Multi-level systems consist of one model that outputs source estimates for all levels considered. I conduct a set of parallel experiments for the Single-instrument and Multi-instrument cases.

The overarching argument of this chapter is that Multi-level systems have a lot of benefits which stem from designing them around our knowledge of auditory hierarchies. Furthermore, the implicit hypothesis is that Multi-level systems will in fact separate better than their Single-level counterparts. If this hypothesis is demonstrated to be correct, then that indicates that imbuing a system with an inductive bias toward our knowledge of auditory hierarchies has benefit.

2.4.0.3. Can Hierarchical separation help with data sparsity at the most-specific source types? Neural network-based separation systems require a lot of data to train effectively. The more sources we desire to separate, the more data we need to train these systems. As discussed earlier, it becomes increasingly difficult to acquire the requisite data when our desired source becomes more specifically defined (e.g., string instruments vs. nylon-stringed acoustic guitars built in California from 1968-1971). This problem naturally arises with our hierarchical setup: as we move further from the root node (i.e., the mix node), the source types that the nodes represent are increasingly specific.

While this may initially seem at odds with the promises of flexibility that hierarchical separation provides, hierarchical separation also promises to learn the taxonomic relationships between different source types, especially those source types with a parent-child relationship. Therefore we might also paradoxically expect that a system that is able to learn these types of relationships could use its learned knowledge to reduce the required amount of training data for very specific source types (e.g., it could use some generalized

knowledge of string instruments to separate the specialized California nylon-stringed acoustic guitars). This experimental question seeks to answer whether which of these two effects is more dominant.

To answer it, I reduce the number of training examples that Multi-level models see in one of two ways: either by fully removing the example from the training set (i.e., all levels removed), or by removing just the leaf-level data from training set. Fully removing the example from the data means that there is less data in the training set, but to remove just the leaf node data from the training set, the network is still able to see nodes at higher levels of the hierarchy. Here, the leaf data is data from Level 1 of the hierarchy, or sometimes referred to as the "Child." I evaluate the resultant systems performance (according to SI-SDR) on separating leaf nodes and compare these results to training on the full dataset (i.e., no data removed). Fully removing the example provides a baseline for how a system would do with a fully reduced training set, whereas only removing the leaf indicates how well the system is able to leverage what it learned by separating the other levels. All training and architectural details are fixed except for the means of removing data and what percentage of data is removed.

How well a Multi-level system is still able to separate in the leaf-only removal training setup will indicate how well a system is able to learn about the structure of auditory hierarchies and how well the system is able to leverage that learned knowledge.

Before I analyze the results from these experiments, I will first discuss the details about the data, evaluation, and model configurations.

2.4.1. Dataset and Evaluation

To test the proposed hierarchical separation strategies, I required a large dataset with isolated sources of many instruments that could be combined in a hierarchical way. Specifically, I required a dataset with a wide variety of granular source labels, i.e., not only "guitars", but "acoustic guitars", "electric guitars", "effected guitars", and so on for every instrument in the dataset. Because of this, I chose Slakh2100 [184], which contains 2,100 musical mixtures along with isolated sources. More details about Slakh2100 are provided in Section 1.2.2. This dataset has 145 hours of mixture data split into 34 instrument categories, which is sufficient to answer the experimental questions

As part of the process for preparing the dataset for the experiments, I created a musical instrument hierarchy from Slakh's included instrument categories, as shown in its entirety in Figure 2.13. For the hierarchical separation experiments, I used a hierarchy with three levels, as shown by the colored sections

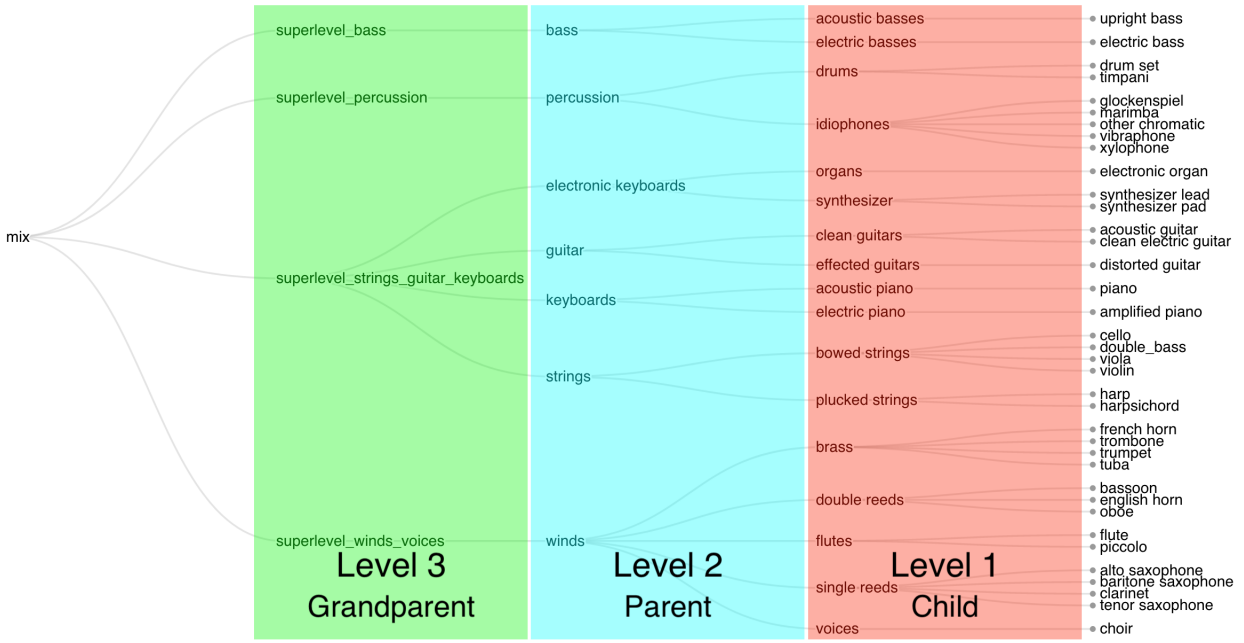


Figure 2.13. Musical instrument hierarchy used for these experiments. The instrument labels for the leaf nodes are taken from the Slakh2100 [184] dataset and aggregated in a manner inspired by the Hornbostel-Sachs system [287] taxonomy. The leftmost level is the root node, or mixture, which is cannot be combined into additional sources, by definition. The middle levels are used for separation in this work, and are highlighted in green, blue, and red. The rightmost nodes are omitted for separation in this work, but included in this image for context.

in Figure 2.13. The top level contained four categories: mid-range strings and keys (guitars, keyboards, and orchestral strings), bass instruments (acoustic and electric basses), winds (flutes, reeds, and brass), and percussion (drum sets and chromatic percussion). The middle level had seven categories (e.g., from mid-range strings: orchestral strings, guitars, keyboards, and electric keyboards), and the lowest level had eighteen categories (e.g., from guitars: clean guitars, and effected guitars). These three levels *exclude* the "Mix" node (the leftmost node in Figure 2.13), which is the trivial level consisting of the mixtures of all sources. Also omitted were the most fine-grained instrument classes at the rightmost of Figure 2.13, which are shown here to contextualize the other levels. These nodes were intentionally left out due to the fact that as the nodes get more specific, it is harder to get enough examples to effectively train a separation system. I note that this is just one of many possible hierarchies I could have used. Also of note is that almost all of the instruments described here (other than "bass" in Level 2 and "drums" in Level 1) would be categorized into the "Other" in MUSDB18 [224] (see Section 1.2.2 for more on MUSDB18).

To create examples with this dataset, I select paired audio segments containing an input mix and target sources. The target sources are in fact a set of instantaneous submixes of the sources, corresponding to

Level	Level Name	SSS Hierarchical Path
3	Grandparent	Keyboards, guitars, and orchestral strings
2	Parent	All guitars (both clean and effected)
1	Child	Only clean guitars (both electric and acoustic)

Table 2.1. Contents of each hierarchical level used for training and testing the hierarchical Source-Specific Separation (SSS) networks. Hierarchical SSS is Single-Instrument strategy, meaning it can only learn one path down the hierarchy at a time.

the different levels of the hierarchy. As an example illustrated in Table 2.1, at the highest level, all pianos, guitars, and strings make up the source class, while at the next level all guitars are the next source class, and at the lowest level only clean guitars are considered the final source class. The duration of the mixture and source segments is 10 seconds. Each song in the training set is chopped into 10 second segments, with a hop between adjacent segments of 2.5 seconds. For each segment, I compute the saliency of each hierarchical target source in the mixture. If the desired source has energy above -30 dBFS in that segment, then the segment is considered salient and included in the training set. Otherwise, the 10 second segment is discarded. For the experiments involving multiple levels, I ensure that for a given example, its parent (or grandparent) has energy from child nodes other than itself. In other words, I want to make sure that a parent is not exactly the same as the child, meaning that some of the child node's siblings or cousins are also salient.

For these experiments, I downsampled all audio to 16 kHz. I did the mixing on the fly and select segments randomly from the pool of salient examples for the specific experiment. For training, the networks saw 20,000 examples per epoch (≈55.5 h), and I used 3,000 examples (≈8.3 h) for the validation and test sets. The training, validation, and test sets were composed of distinct songs, and therefore none of these partitions have any audio examples in common. To demonstrate the Single-instrument strategies, I chose only one path down the hierarchy (shown in Table 2.1); this path is taken from the larger hierarchy used for the Multi-instrument experiments (Figure 2.13). For the QBE systems (i.e., both **Multi-instrument Single-level** and **Multi-instrument Multi-level** QBE systems), I separated every instrument type in the hierarchy shown in Figure 2.13. Query segments were selected from the set of salient segments such that they are always leaf nodes along the same path as the target regardless of the target level, but originate from different tracks.

For all experiments, I used the scale-invariant source-to-distortion ratio (SI-SDR) [155] to determine the output quality of our models (see more about SI-SDR in Section 1.2.3). For reference, in some cases I will also report the SI-SDR when doing no processing on the mixes. For all values in this chapter I will

Strategy	Model Name	Sources	# Networks	# Output Levels/Net
Single-instrument Single-level	SSS	Guitar	3	1
Single-instrument Multi-level	SSS	Guitar	1	3
Multi-instrument Single-level	QBE	All	3	1
Multi-instrument Multi-level	QBE	All	1	3

Table 2.2. The models used in these experiments and which hierarchical strategy they follow. SSS stands for Source-Specific Separation networks, which are Single-instrument systems specialized to one path in the hierarchy. In this case, these networks learn a path for Guitars shown in Table 2.1. The Query-by-Example (QBE) networks are able to separate all sources in the dataset, depending on an input audio query. The Single-level systems (rows 1 & 3) contain 3 networks, one for each level of the hierarchy. The Multi-level systems (rows 2 & 4) contain only 1 network that is responsible for separating all 3 levels of the hierarchy.

report the "SI-SDR improvement" or SI-SDRi (defined in Introduction Equation 1.5), which is the difference between the SI-SDR of the estimated source signal and the SI-SDR of the unprocessed mixture signal. Higher values indicate better separation performance, with values close to 0.0 dB indicating that the estimate is no better than the unprocessed mixture and values *below* 0.0 dB indicating that the separation performance is actually *worse* than if we had not done any separation in the first place and used the mixture as our "separated source."

As I have mentioned previously, all current automated measures of separation quality–SI-SDRi included–have a tenuous correlation with human perception [20, 23, 24, 77]. SI-SDRi does, however, measure how close a system's source estimate is to the ground truth source. It is also able to be quickly automated, which enables collecting evaluations for hundreds of thousands of source estimates, as done in this chapter. Finally, because the hierarchical reframing of the source separation problem is new, the separation quality numbers that I will present are provided as an existence proof of the technology.

2.4.2. Model Configurations

Recall that all networks for experiments in this chapter are adaptations of the backbone Mask Inference [69] system described in Section 2.3.2. This architecture is modified such that the resulting networks follow the **Single-instrument Single-level**, **Single-instrument Multi-level**, **Multi-instrument Single-level**, and **Multi-instrument Multi-level** strategies. The results of these modifications are shown in Table 2.2.

As discussed, the **Single-instrument Single-level** and **Single-instrument Multi-level** strategies require minimal changes. I refer to these systems as Source-Specific Separation (SSS) systems. These SSS systems. The SSS models were composed of 4 bidirectional long short-term memory (BLSTM) [118] layers with 600

hidden units in each direction and dropout [260] of 0.3 applied to all but the last layer, followed by a fully connected layer with sigmoid activation function that outputs a mask. This mask was applied to the mixture magnitude spectrogram and combined with the mixture's phase to create a source estimate. The only difference between the **Single-instrument Single-level** SSS system and the **Single-instrument Multi-level** SSS system was that the Single-level system consists of *three* networks that each output only one mask (for each level), whereas the Multi-level system consisted of *one* network outputting three masks.

As described in Section 2.3.6, a Query-by-Example (QBE) model was used to embody the **Multi-instrument Single-level** and **Multi-instrument Multi-level** strategies. These QBE models were composed of two sub-networks, a query net and a masking net. The query net was composed of 2 BLSTM layers with 600 nodes in each direction and dropout of 0.3 applied to the first layer, followed by a fully-connected layer with linear activation that mapped each time-frequency bin to an embedding space with 20 dimensions according to Equation 2.5. The masking net was the same as the SSS models above, with a larger input feature vector to accommodate the concatenated query anchor. The masking network produced the mask, which was applied to the mix spectrogram to get source estimates. Just as above, for **Multi-instrument Single-level** the QBE setup consisted of *three* Single-level model outputting one source per model and the **Multi-instrument Multi-level** QBE system is *one* model outputting sources for all three levels at once.

All models were trained with the Adam [142] optimizer at a learning rate of 1e-4 for 100 epochs and a batch size of 25. All networks were trained to optimize the tPSA loss (Eq. 2.3) described in Section 2.3.2. The learning rate was halved if the loss on the validation set did not decrease for 5 straight epochs. The gradient was clipped to the 10^{th} percentile of historical gradient norms if the norm of the minibatch was above that value [240].

2.5. Results

In this section, I will discuss the results of the experiments I conducted in an effort to answer the three questions I posed about hierarchical separation systems.

2.5.1. Does the Hierarchical Constraint help or hurt Multi-level separation systems?

The first results I will discuss are in regards to the experimental question about whether or not the hierarchical constraint posed in Section 2.4.0.1 leads to better separation performance. I choose to discuss these results first because they will inform the design of the Multi-level systems for the next two experiments. These

results are shown in Table 2.3, where the values per level indicate the mean SI-SDRi over the whole test set and the column marked "HC" determines whether the hierarchical constraint was applied during training or not.

Model	HC	Level 3	Level 2	Level 1
SSS (Guitar)		3.5	4.0	4.0
SSS (Guitar)	✓	3.2	3.6	3.8
QBE		3.2	2.4	0.2
QBE	✓	3.3	2.1	1.6

Table 2.3. Mean SI-SDRi (dB) for hierarchical Source-Specific Separation (SSS) and Query-by-Example (QBE) models. All models in this experiment are Multi-level models. Each Multi-level model is trained either with the hierarchical constraint (HC) described in Section 2.3.4.2 or with no constraints on the masks produced for sources at different levels of granularity.

In Table 2.3, we can observe that, for the source-specific separation network (recall that this system only separates guitars), the hierarchical constraint slightly diminishes performance at all levels according to the mean SI-SDRi results in the table, which is an indication that Source-Specific Separation (SSS) models are able to learn the specific hierarchical relationship for a single source (in this case, guitar) at different levels without the additional help offered by the hierarchical constraint. That is, the SSS networks are so specialized to the source they are designed to separate, that adding an additional inductive bias in the form of the hierarchical constraint actually hinders the SSS network's ability to produce the correct masks for its target source.

For the Query-by-Example (QBE) networks, which separate all types of instruments, the hierarchical constraint marginally hinders performance at Level 2, but helps considerably for the leaf node (Level 1). I hypothesize that QBE networks benefit more because they are unable to learn the specific mask "shapes" of any individual source, and thus need the additional help offered by the hierarchical constraint. The results I report in Table 2.3 are the mean of SI-SDRi over the entire test set, so the small differences (< 0.5 dB) between different experimental conditions might not hold up to further scrutiny. Despite this, in all subsequent experiments I will use the hierarchical constraint for QBE networks, and omit it for the SSS networks.

Strategy	Model	All Levels	Level 3	Level 2	Level 1
Single-instrument Single-level	SSS (Guitar)	1.8	3.2	2.7	−0.7
Single-instrument Multi-level	SSS (Guitar)	3.9	3.4	4.0	4.0
Multi-instrument Single-level	QBE	1.0	3.3	1.4	−1.9
Multi-instrument Multi-level	QBE	2.3	3.3	2.1	1.6

Table 2.4. Source-Specific Separation (SSS) and Query-by-Example (QBE) model results in terms of mean SI-SDRi (dB), where higher values are better. SSS networks are only trained to separate sources in the hierarchical path containing clean guitars (See Table 2.1), whereas QBE networks separate any source in the hierarchy. Here I compare Single-level networks to Multi-level networks. There is only one multi-level network for all three levels, but three single-level networks, i.e., one for each level (see Table 2.2). The 'All Levels' column is the aggregate of the latter three columns, which show each level individually. The Multi-level systems (rows 2 & 4) handily outperform the Single-level systems (rows 1 & 3) at Level 1, which represents the leaf node sources.

2.5.2. Do Single-level or Multi-level systems separate better?

Next, we turn toward understanding whether Single-level or Multi-level systems are more effective at separating hierarchical sources. The results shown in Table 2.4 compare Single-level and Multi-level hierarchical models for both Source-Specific Separation (SSS) (top two rows) and Query-by-Example (QBE) (bottom two rows) separation systems. Again, the numbers represent mean SI-SDRi values over the entire test set, so small improvements (i.e., < 0.5 dB) should be taken with a grain of salt. Nonetheless, in both cases the Multi-level hierarchical networks (rows 2 and 4) improve over the Single-level models (rows 1 and 3) at every level. Multi-level systems demonstrate the largest gains over Single-level systems at the lowest levels of the hierarchy (i.e., Level 1). This implies that Multi-level networks can leverage their shared knowledge of the hierarchy to aid themselves at the lower levels, where individual instruments are more difficult to discern in the mix.

From the Level 1 results in Table 2.4, we see that separating sources at this fine level of detail (e.g., clean electric guitars vs. distorted electric guitars) is extremely difficult, especially with a MIDI-synthesized dataset such as Slakh2100, where several different instrument types may sound similar. As a reminder, Level 3 sources can be considered similar to the "Other" source class in MUSDB18 [224]. Therefore, separating sources at the more fine-grained Levels (1 and 2) is more difficult than what is typically attempted in musical source separation.

The results from the QBE networks in Table 2.4 are expanded upon in Figure 2.14, which compares both QBE setups, showing boxplots of the SI-SDRi distributions for the Child level (Level 1). Recall that

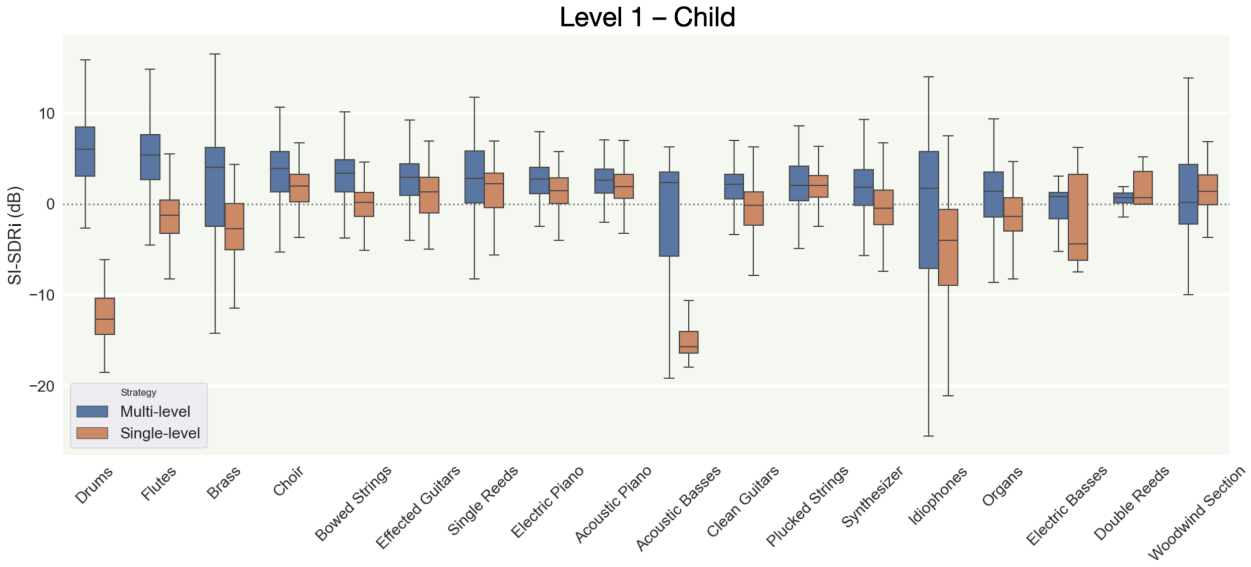


Figure 2.14. Multi-instrument separation SI-SDRi (dB) results for the bottom level ("Level 1 – Child") of the hierarchy comparing on of the **Multi-instrument Single-level** QBE systems (orange, labeled "Single-level") to the **Multi-instrument Multi-level** QBE system (blue, labeled "Multi-level"). The boxes extend from the first quartile to the third quartile of the distribution and the lines in the boxes represent medians. All paired Single/Multi distributions are statistically significant ($p < 0.012$) except for Plucked Strings, Electric Basses, and Double Reeds according to a Mann-Whitney U rank test. The **Multi-instrument Multi-level** QBE system outperforms the **Multi-instrument Single-level** QBE system in 16 of the 18 total source types.

the Single-level QBE setup has one QBE model for each level (i.e., each chart is one model), whereas the Multi-level QBE setup is only one model for all levels (i.e., there is one model shown across all three charts). This figure shows stark differences in the performance between Single-level and Multi-level models, with the Multi-level models winning in the majority of settings. In fact, of the 18 total source types at the Child level, in 16 of them the median is higher; in 16, the top quartile is higher; and in 9 of them, the bottom quartile is higher for Multi-level systems. Furthermore, for each instrument in the Child level, all pairs of Single-level and Multi-level distributions are statistically significant ($p < 0.012$) according to a Mann-Whitney U rank test, with the exception of the Plucked Strings, Electric Basses, and Double Reeds source types. These facts indicate that the Multi-level distributions as a whole have distinctly higher SI-SDRi values over more source types.

Additionally, Figure 2.15 shows an annotated t-SNE [282] visualization of the embedding space made by the query subnetwork of the Multi-level QBE model running on the test set. We can qualitatively see that similar sources (e.g., different types of pianos, or various wind instruments) are put into similar locations of the embedding space. While this is not definitive proof that the embedding space has any hierarchical

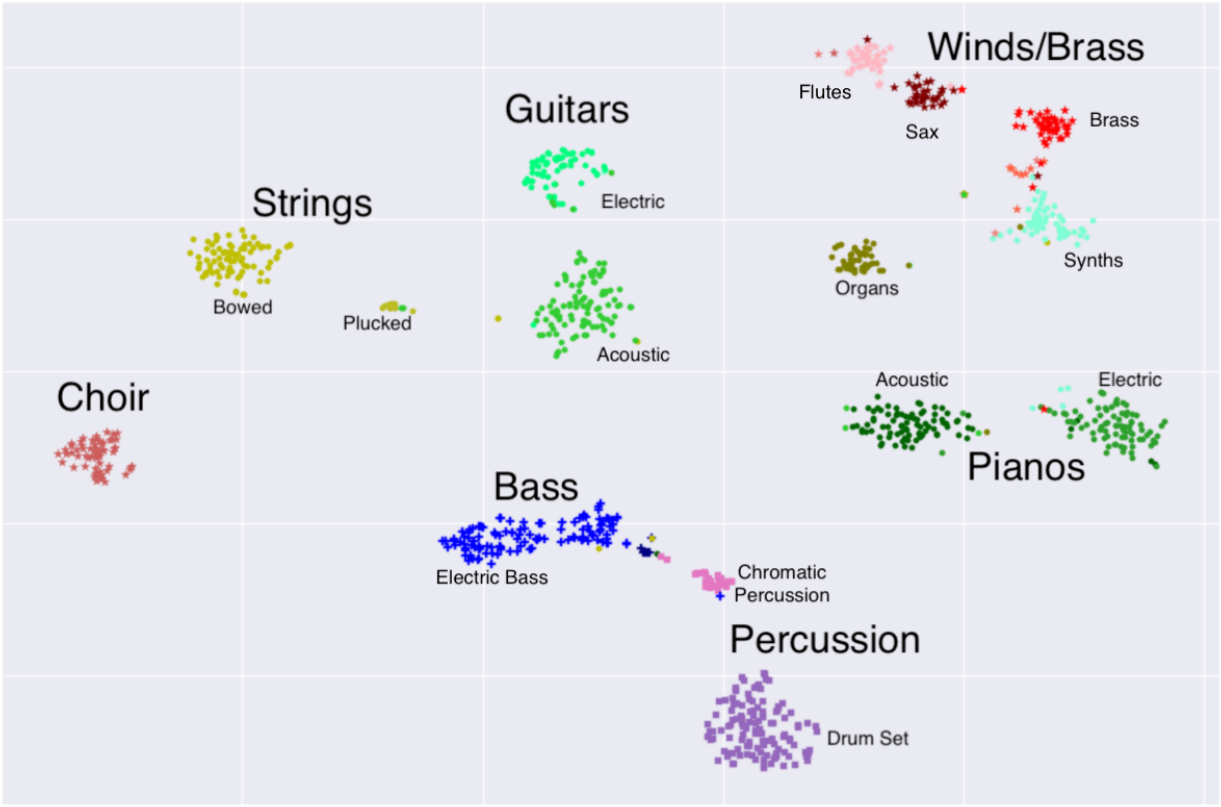


Figure 2.15. Annotated t-SNE [282] plot of the embedding space produced by the Query input of the Multi-level, Multi-instrument QBE separation architecture shown in Figure 2.12. Similar sources are clustered together and dissimilar sources are far apart.

structure (recall that there was no restrictions put on the embedding space during training), this is a nice indication that the proposed hierarchical setup does seem to result in an embedding space that clusters similar sounding sources. Compare this to the pre-defined hierarchy shown in Figure 2.13.

All of this indicates that providing a source separation system with a set of inductive biases akin to our knowledge of auditory hierarchies enables the Multi-level SSS and QBE systems to get substantially better separation performance as compared to their Single-level counterparts. This is especially true for the finest-grained source types. The Multi-level setup enables a QBE system to be more successful when separating more types of sources. In total, the Multi-level QBE system is able to separate 28 different types of sources over all levels and 18 source types at the most granular level. Either number is much more than typical deep learning-based music source separation systems are able to separate. Another striking fact about these results is that the Multi-level systems work better in spite of the fact that they contain just *one* model for all three levels instead of three Single-level models, with each specialized to one of the three levels. The

	Data Reduction		Levels			
	%	type	All	Level 3	Level 2	Level 1
SSS (Guitar)	0	–	3.8	3.5	4.0	4.0
	50	all	3.3	3.1	3.4	3.4
		leaf	3.5	3.3	3.6	3.6
	90	all	0.1	1.5	−0.7	−0.5
		leaf	3.6	3.4	3.7	3.7
QBE	0	–	2.3	3.3	2.1	1.6
	50	all	−1.5	−2.1	−1.4	−1.1
		leaf	2.2	3.4	2.1	1.1
	90	all	−1.8	−2.1	−1.8	−1.5
		leaf	1.9	3.1	1.7	0.8

Table 2.5. Mean SI-SDRi (dB) over the unprocessed mix for hierarchical Source-Specific Separation (SSS) and Query-by-Example (QBE) models (separated by the thick broken line). Each model is trained while removing either just the leaf ("leaf", just Level 1) or the whole example ("all", all levels) for a specified percentage of the data. Reducing just leaf nodes up to 90% shows only a 0.3 dB drop for SSS and 0.8 dB drop for QBE compared to using all of the leaves.

Multi-level models have approximately $\frac{1}{3}$ the number of trainable parameters as the Single-level systems, which seems surprising in the world of deep learning where "more parameters $\implies$ better performance" [15, 148]. This again, shows the power of having a hierarchy available during training: it can overcome a huge parameter deficit to ultimately produce better separation estimates.

In short, inductive biases that reflect auditory hierarchies can enable Multi-level separation systems to better separate more source types with a fraction of the computing cost.

2.5.3. Can Hierarchical separation help with data sparsity at the most-specific source types?

Recall that in this experiment I investigated specifically how well a hierarchical system can leverage information about related nodes (i.e., parent nodes) when separating leaf nodes. By definition, the leaf nodes are the most specific nodes in the hierarchy, and therefore leaf node data is most sparse. Here, the leaf data is data from Level 1 of the hierarchy (also referred to as the "Child" level, earlier). Towards this goal, I tested how well the two Multi-level hierarchical systems performed when trained with all data (as an upper bound), by eliminating full examples from in the training set (as a lower bound), or by only eliminating only the leaf data from the training set (but keeping the parents as training targets).

The full results from this experiment are shown in Table 2.5. The top section shows the Multi-level Source-Specific Separation (SSS) networks and the bottom section shows the Multi-level Query-by-Example

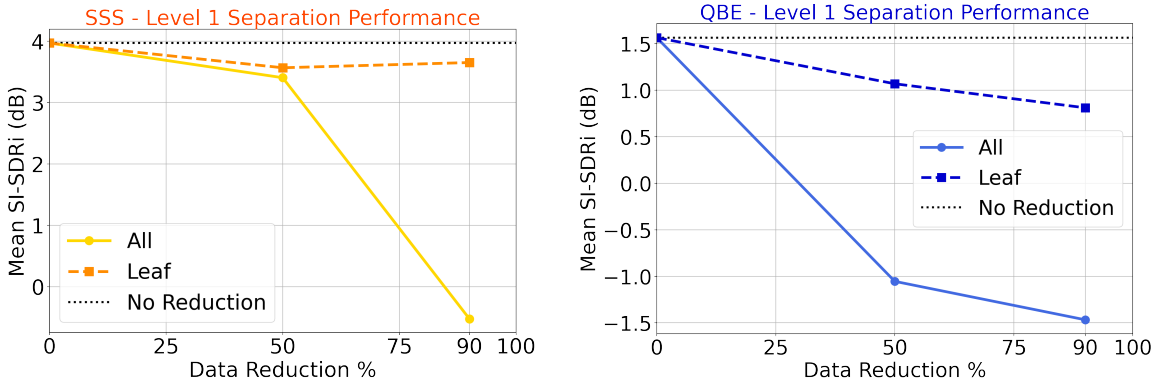


Figure 2.16. Separation performance, in terms of mean SI-SDRi (dB), of the Level 1 sources (i.e., the leaf nodes) for the Source-Specific Separation (SSS) network, left, and the Query-by-Example (QBE), right, as the number of training examples is reduced. Here, I reduce the training data in one of two ways: either by eliminating examples at all levels (labeled "All") or by just eliminating the leaf data (labeled "Leaf"). This is the same data shown in Table 2.5. Reducing the leaf data by up to 90% shows a minimal degradation in performance, indicating that the network is able to leverage information about the hierarchy (i.e., parent sources) when separating leaf sources with limited data.

(QBE) networks. The "Data Reduction" columns show how the data was reduced–"all" means the full example was eliminated, "leaf" means only the leaf data was eliminated–and what percentage of data was eliminated. The top row, marked "0", is the upper bound where none of the data was eliminated from the training set. Compared to reducing all of the data by 50% or 90%, the performance of reducing only the leaves degrades very minimally. In fact, when removing 90% of leaf data (i.e, training on only 10% of the available leaves), the SSS model only loses about 0.3 dB in SI-SDRi performance at the leaf nodes (Level 1) of a system that is trained with all of the data. Similarly, the QBE model only loses roughly 0.8 dB compared to the peak leaf node performance when training on 90% less leaf data. Compare these numbers to the situation in which data is removed at all levels: these values are all below 0.0 for Levels 2 & 1, indicating that the separation systems did worse than doing nothing! Specifically, the Level 1 results from Table 2.5 are also shown in Figure 2.16, to make this trend clearer. For both SSS and QBE cases, the models that were able to leverage information from other nodes in the hierarchy (the dashed lines), had a minimal performance degradation. In cases where we have rich data at higher levels but sparse data at lower levels, hierarchical multi-level networks can do a respectable job at separating lower levels. We see the same story for both SSS and QBE networks: even a small amount of leaf data can help ward off a large drop in performance.

2.6. Envisioned Interactions with Hierarchical Source Separation Systems

One of the major selling points of hierarchical source separation is that resulting systems are able to separate a much wider array of source types compared to typical deep learning-based source separation systems. Because of the existing landscape of deep learning-based source separation, there have been very few opportunities to create interactive separation systems that support *controllable* source separation. Currently, many websites exist where a user can upload a mixture and get back a set of pre-defined sources. These types of non-interactive separation interfaces make sense when the set of pre-defined source types is small (e.g., $N = 4$). Yet, as I have argued, having a small number of supported source types also limits the usefulness of source separation in general. The limitations of current deep learning systems limit the need to think deeply about how a user might interact with them. In contrast, in this chapter I have demonstrated that a hierarchical Query-by-Example source separation can support source types than is typically considered. This provides a chance for us to rethink how a user might interact with a hierarchical separation system.

Specifically, the Multi-level Query-by-Example (QBE) system has a lot in common with steerable separation systems developed before the deep learning era [162]. One interesting direction is so-called Separation by Humming [254], which used an audio query of a user singing or humming their desired source to guide a factorization-based separation system, e.g., Non-negative Matrix Factorization (NMF) [252] or Probabilistic Latent Component Analysis (PLCA) [255]. Such systems are limited by the less performant backend separation algorithms of the day, but they are also limited by a user's ability to vocally imitate their desired source [22, 137]. In a similar thread, the so-called On-the-Fly separation interface [61] enables users to upload a mixture and inform the system what source they want to separate using a set of keywords. The user-selected keywords would then be used to retrieve a set of matching audio clips, where the user would select the clip that most closely matched their desired source. These audio clips would be used to guide a factorization-based separation process and produce separated sources.

Both of these systems embody novel methods for providing users with flexible ways to guide a separation process built around the affordances of backend separation technology that is flexible enough to separate many source types (i.e., NMF). Modern deep learning separation systems offer much better separation performance in comparison to NMF, but, as mentioned, they have historically been limited in the types of sources they can estimate. With more flexible hierarchical Multi-level QBE systems proposed in this chapter, we now have access to backend separation technology that can provide a similar set of afforances

as the Separation by Humming and On-the-Fly separation interfaces. I will draw inspiration from both of these systems when I imagine how a hierarchical separation interface would look like.

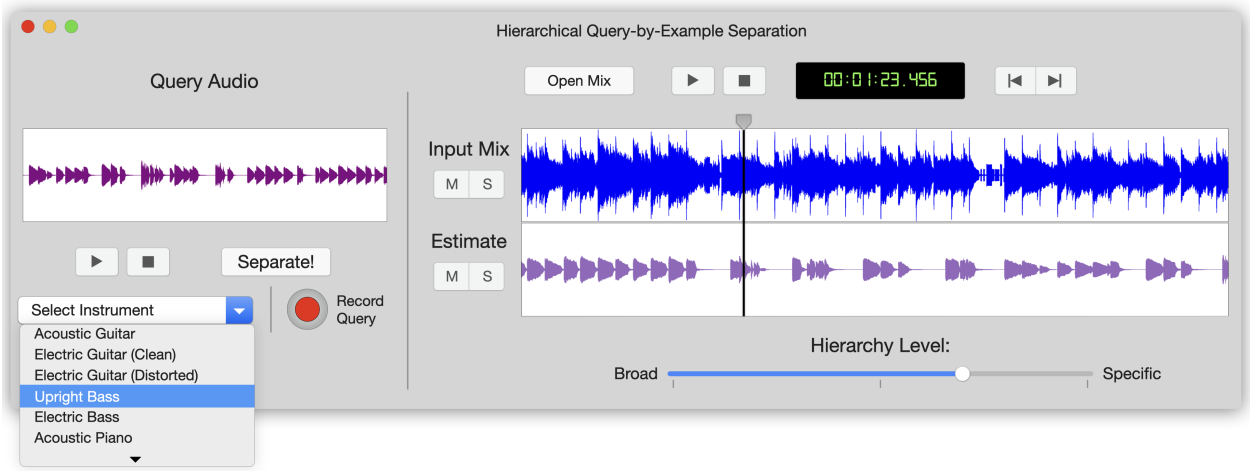


Figure 2.17. An imagined interface for hierarchical source separation. The backend system is a Multi-level Query-by-Example system as outlined in Section 2.3.6. Recall that the query guides the behavior of the separation system. For the query input, a user can either upload a sound of their choosing or select one of the predetermined options from a drop down menu. The predetermined options will load a corresponding audio clip in the background. The output is then displayed to the user, where they can hear their results. A "Hierarchy Level" slider lets the user hear the outputs of each level, and will interpolate between different levels by proportionally adjusting the audio.

The intended user for this system is one that has prior experience with digital audio software, like a Digital Audio Workstation (DAW). Specifically, some DAWs are designed around non-real time audio processing, like iZotope RX [125] or Adobe Audition [1]. Whereas with real time processing where the system processes audio in milliseconds, non real time systems have no such time constraint, so we can expect that the user will tolerate slower turnaround times (e.g., on the order of seconds to minutes). Users accept these slower turnaround times in exchange for enabling an advanced set of functionality that might not be possible or feasible in real time settings, such as spectral editing. As a result of these more advanced features, non real time systems tend to have a user base with more experience with these tools. Typically, the users are audio professionals. Deep learning inference times on commodity hardware (i.e., laptops with no GPUs) can take minutes to finish [83], requiring non real time processing. Because the proposed Multi-level QBE interface is built on deep learning methods and because this is a new technology (compared to older DSP techniques like reverb), the intended audience for this interface is expected to have familiarity with the advanced, non real time DAWs. For example, a post-processing engineer on a movie set or a recording engineer would both be potential users for this interface.

An envisioned interface is shown in Figure 2.17. After providing an input mixture ("Open Mix" button, top center) a user can hear it by pressing the global play/pause/stop controls and the mix is shown in the blue waveform. To perform separation, a user must provide an audio query shown on the left side of the interface. A user could either record their own query audio to upload to the system (red button labeled "Record Query"), or they can select from a drop down menu (bottom left) to pick a predefined query audio clip from a set of sources known by the system. The selections in the drop down menu fetch a corresponding audio clip to use as a query. One the query audio is selected, it is displayed as a waveform (top left) and the user can listen to it. Once the user is satisfied with the query audio, they press "Separate!" and the system provides a separation estimate. When the user presses "Separate!", the system will run inference on a Multi-level QBE system and get a separation estimate. This separation estimate is displayed in a waveform below the mix (labeled "Estimate") and a user can hear it by the pressing "Play" on the global controls. The source estimate track is soloed by default when the separation is first displayed, i.e., all other tracks are muted except the separation estimate. Soloing and muting tracks can be controlled with the "S" and "M" buttons, respectively, to the left of each track (these icons will be familiar to users who have experience with DAWs). Below the tracks there is a slider that lets the user hear the different hierarchical levels of the estimate. In this mockup, I imagined letting the user interpolate between different levels, which can be implemented under the hood as proportionally adjusting the audio volumes between the levels (i.e., if the slider is halfway between Level 1 and Level 2, then each is played at 50% volume).

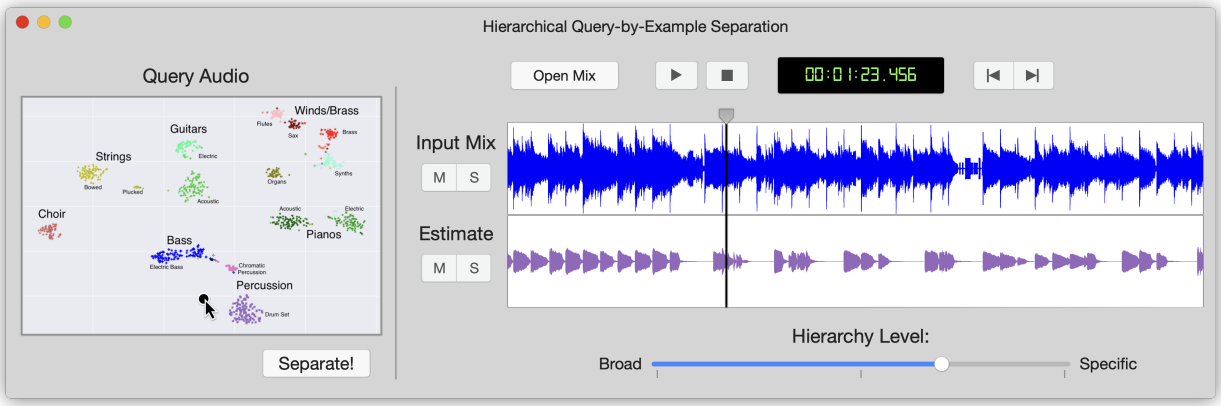


Figure 2.18. An alternate imagined interface for hierarchical source separation. The backend system is a Multi-level Query-by-Example (QBE) system as outlined in Section 2.3.6. Here, instead of selecting an audio clip, a user can move around in an annotated plot of the t-SNE [282] embedding space produced by the QBE model (left).

Finally, I also envisioned a version of this interface where, instead of uploading an audio clip, a user can see and explore the QBE model's latent space directly. This is shown in Figure 2.18. This latent space is annotated, so by moving around the latent space the user can guide the model to separate different sources without having to upload a clip. Under the hood, we can precompute the actual latent matrices that correspond to different locations in the plot shown to the user. Then, once the user makes a selection that latent matrix is put into the model. If a user selects a part of the space that is between precomputed latents, we can interpolate. Instead of being a completely different interface, this iteration could be integrated with the previous one, where a user could choose between uploading audio or interacting with the latent space.

To see either interface to fruition, a set of user studies will need to be designed and implemented such that the interface can be iteratively updated and optimized for user needs and minimize their confusion. Such user studies could focus on two aspects of this system: the intuitiveness of the interface itself and the effectiveness of the backend separation system. Both of these aspects are intertwined; the backend system might provide a perfect separation, but if the user is frustrated by the layout then the interface would not be useful. The opposite is true as well: it is problematic if the interface is intuitive but the backend separation system cannot effectively separate a source they care about. I leave the final design and user studies for future work.

2.7. Conclusion

In this chapter, I have introduced hierarchical source separation, a conceptualization of source separation based on the premise that auditory scenes can be analyzes hierarchically. In doing so, I provided four general strategies for creating hierarchical source separation systems, each of which I instantiated by modifying a standard source separation architecture. I conducted a set of experiments aimed at determining which of these strategies provided the best separation performance and flexibility with respect to number of source types. I found that Multi-level systems always outperform Single-level systems despite the fact that Multi-level have a fraction of the allotted computing resources. I demonstrated that the hierarchical Multi-level Query-by-Example is able to separate many more fine-grained sources than typical separation systems usually consider. I have also demonstrated that hierarchical systems provide a pathway to alleviating the strenuous data requirements in situations where we require very specific, yet sparse, source data. Finally, I have sketched out a design for an interface for a Multi-level Query-by-Example separation system. Fruitful directions for future research might include adapting a more state-of-the-art source separation architecture

to fit within the proposed hierarchical strategies, evaluating hierarchical source separation on recordings of live musicians instead of synthesized music as in Slakh2100, and conducting user-centered design studies to determine the best means for developing interactive hierarchical separation interfaces.

CHAPTER 3

Simultaneous Separation and Transcription with Cerberus

Source separation is an important way to analyze the contents of a musical scene because it promises users or downstream tasks an unobstructed audition into the constituent elements of a musical scene. However, while the resultant audio clips from an ideal separator could perfectly represent every detail of the instrument source audio, audio clips might contain *too* much detail to quickly assess the relevant content within that audio clip when a user wants to make certain kinds of edits. A key piece of musical content that a user might want to assess at a glance is the set of notes contained in the audio. Notes are a high-level and ubiquitous way to represent musical content for human consumption (for more discussion, see Section 1.3.1). The other side of this is that because note representations are typically designed with human readability in mind, these representations are extremely lossy; they strip the data of everything but the essential components, forgoing fidelity in favor of concision.

This brings us to two paradigms of analyzing a musical scene: on one side are scenarios in which we want a low-level analysis that perfectly describes our desired content with high-dimensional data, and on the other are scenarios where we want a high-level analysis that privileges some attributes over others, producing low-dimensional data that sacrifices the ability to perfectly describe every detail of the content. As you may have guessed, Source Separation falls under the former paradigm and Automatic Music Transcription (AMT) falls under the second.

So, which of these paradigms is best for analyzing a musical scene? As with many things in life, the true answer to this question is "it depends." If your task is to learn a song so that you can play it in rehearsal, analyzing the notes in a musical scene at a high-level is sufficient most of the time. On the other hand, if you are tasked with remixing a song, then you might require a low-level audio representation that preserves the exact timbre of each instrument in the song, like a waveform. Given that both of the high-level and low-level analyses are useful in different contexts, it is natural to ask if doing both types of analysis is useful when training machine learning systems. Certainly, a system that can do both can support both high- and low-level analyses would be useful in both of the aforementioned scenarios.

As discussed in Section 1.4.2, there is a barrier to making systems that support both of these high- and low-level analysis paradigms. This barrier is the simple fact that most Automatic Music Transcription (AMT) systems only work for recordings of isolated instruments (e.g., a recording of *only* a piano [11, 53, 63, 64, 103, 105, 106, 135, 136, 139, 149, 204, 219, 247, 307], or *only* a drumset [19, 213, 261, 301]). These AMT systems provide these high-level analysis (i.e., extracting notes from raw audio), but because existing systems do not work for multi-instrument audio (i.e., recordings with multiple instruments heard simultaneously), there is a sizeable portion of musical audio content where it is impossible to produce a high-level analysis. Therefore, it is important to make AMT systems that are able to analyze musical scenes that contain multiple instruments heard simultaneously.

When the discussion moves towards multi-instrument AMT, source separation seems natural to bring back into the conversation. The relationship between source separation and multi-instrument AMT has been noted by early music information retrieval scholars [8, 144, 145, 218], who saw both tasks as having much in common; they both try to analyze multiple overlapping instruments in a scene. In fact, some early techniques for source separation, such as Non-Negative Matrix Factorization (NMF) [75, 156, 252, 285], were initially proposed as AMT systems. However, the discussion about the benefits of doing both separation and AMT simultaneously has evaporated in the past decade or so, just as deep learning efforts for both tasks has ramped up. Given the obvious link between these tasks, it seems like a logical step to see if modern deep learning approaches can benefit from doing both separation and transcription simultaneously.

In this chapter, I will describe the first combined source separation and automatic music transcription system, called Cerberus. This system is built by extending a standard source separation architecture to also do transcription, providing a first step in solving the single-instrument problem with AMT systems by transcribing multiple instruments in a mixture. I will demonstrate that training a deep learning system to both simultaneously can enable it to get better performance on both tasks according to standard separation and transcription metrics. Cerberus uses musical score information (i.e., piano roll data) during training, and therefore there is an argument to made that it is Score-Informed, however has the potential to play a significant part as an upstream, pre-processing step for a true Score-Informed system described in the next chapter.

3.1. Contributions of this Chapter

This chapter makes the following contributions:

- I introduce the first deep learning-based system that can simultaneously separate and transcribe multiple harmonic and percussive sources in a mixture. This system is called Cerberus, a modification to a standard source separation architecture that supports both separation and transcription outputs. Cerberus combines architectural insights from state-of-the-art separation systems to include transcription estimates.
- I evaluate this system against separation-only and transcription-only baselines, and I show that doing both separation and transcription simultaneously leads to better performance when compared to single-task systems according to standard metrics for both tasks.
- I demonstrate that Cerberus is able to separate and transcribe mixtures with up to five instruments, making it able to support many more instruments than the vast majority of previous Automatic Music Transcription systems.
- As I will discuss at the end of this chapter, Cerberus provides an important piece of the puzzle for downstream systems that need multi-instrument transcription data, like the Score-Informed source separation system described in Chapter 4, which can edit an output source estimate based on note input data.

Unlike the other two projects in this dissertation, this chapter will not present an envisioned user interface. This is because the next chapter on score-informed separation builds on the work presented here in this chapter. I will present an interface next chapter that uses Cerberus under the hood.

3.2. Cerberus Architecture: Overview

As its name suggests, the Cerberus architecture has three "heads" that are connected to a single "body." These three heads are trained jointly and they each have specific outputs and loss functions. The input to Cerberus is a magnitude spectrogram that represents a mixture. The rest of this section will go into more detail about each of Cerberus' heads and how they work, but briefly the output heads are as follows:

(1) **The Deep Clustering Head (separation)** produces a high-dimensional embedding space for each time-frequency bin ((t, f) bins) in a spectrogram, whereby (t, f) bins that come from the same source are close together and (t, f) bins from different sources are far apart [114].

(2) **The Mask Inference Head (separation)** outputs a real-valued mask for each source, that is multiplied by the input mixture spectrogram to produce a source estimate [69].

(3) **The Transcription Head (transcription)** produces a transcription estimate for each source in the mixture in the form of a piano roll.

The "body" of Cerberus is a stack of Bidirectional Long Short-Term Memory (BLSTM) layers [94, 118] (see Appendix Section 6.1 for more details about BLSTMs), which inputs a mixture spectrogram and outputs a common representation that is given to each head, each of which will do a minimal amount of processing to convert this common representation into the format the head expects. This BLSTM stack architecture is very similar to the Backbone Separation System described in Section 2.3.2, however, here, I make modifications specific to accomplishing the goals of this chapter, namely separation *and* transcription. Cerberus is trained jointly on the loss functions computed with all three heads (to be described below), which means that the body must produce a representation that is suitable for all three. Further training details will be provided in Sections 3.3.4 and 3.3.5.

Cerberus draws heavily from the source separation literature, specifically it is inspired by Chimera [165], which is a two-headed separation-only system. The rest of this section will be dedicated to examining each of Cerberus' three heads and understanding the role each plays in creating a system that can simultaneously separation and transcribe multiple instruments in a mixture.

3.2.1. Deep Clustering

Deep Clustering [114] is a technique originally designed for speech separation, whereby a neural network creates a high-dimensional, discriminative embedding space populated by the time-frequency bins of the mixture spectrogram. The embedding space is trained such that time-frequency bins that are dominated by the same source (i.e., where the same source is loudest) are close together and time-frequency bins that are dominated by different sources are far apart.

To train this embedding space, a set of C ground truth source spectrograms are flattened from shape $\mathbb{R}^{F\times T\times C}$ to shape $\mathbb{R}^{FT\times C}$ and are then converted to a binary label indicator matrix $\mathbf{Y}\in\mathbb{R}^{FT\times C}$ where the value at $Y_{i,j}=1$ if the source j is loudest at time-frequency bin $i=(t,f)$ and 0 otherwise. This binary label indicator matrix, $\mathbf{Y}$, is then used to make a binary affinity matrix $\mathbf{A}=\mathbf{Y}\mathbf{Y}^T$, which represents the 'assignment' of each of the sources: $A_{i,j}=1$ if i and j are dominated by the same source, $A_{i,j}=0$ otherwise. The network is then designed to produce a D dimensional embedding space $\mathbf{V}\in\mathbb{R}^{FT\times D}$ that

has a corresponding estimated affinity matrix $\tilde{\mathbf{A}} = \mathbf{V}\mathbf{V}^T$. The network is trained to minimize the distance between these two affinity matrices with the Deep Clustering loss [114]

$$\mathcal{L}_{\text{DC}} = \|\tilde{\mathbf{A}} - \mathbf{A}\|_F^2 = \|\mathbf{V}\mathbf{V}^T - \mathbf{Y}\mathbf{Y}^T\|_F^2, \tag{3.1}$$

where $\|\cdot\|_F^2$ denotes the squared Frobenius norm. At inference time, source estimates are created by clustering the embedding space that the network produces; typically, k-means clustering is used. Equation 3.1 is the original loss function that was introduced for Deep Clustering, but alternative objective functions have been proposed in the literature that correct for differences in number of time-frequency bins per source, eliminating quiet time-frequency bins, and regularizing and normalizing the embedding space during training so that can more effectively separated by k-means [291].

3.2.2. Mask Inference

As discussed numerous times in this document, Mask Inference [69] is a technique where the network produces a matrix that is the same shape as the input mixture spectrogram. This matrix is called a mask, and has values in the range $[0.0, 1.0]$. The mask is element-wise multiplied with the mixture spectrogram to produce a source spectrogram estimate. For more details on masks, see Section 1.2. Typically Mask Inference systems are trained to minimize the distance between the ground truth source spectrogram and the estimated spectrogram, either using a simple L1 or L2 distance or something more involved like the truncated Phase Spectrogram Approximation (tPSA) [69]. Here, I use the tPSA loss for the Mask Inference head,

$$\mathcal{L}_{\text{MI}} = \mathcal{L}_{\text{tPSA}} = \left\| \hat{M}_c \odot |X| - \text{clamp}\left(|S_c| \odot \cos(\angle S_c - \angle X)\right) \right\|_1, \tag{3.2}$$

where $|X|$ and $\angle X$ denote the magnitude and phase of the mix spectrogram, respectively, and $\text{clamp}(x) = \min(\max(x, 0), |X|)$ is a truncation function ensuring the tPSA target can be reached with a sigmoid activation function. Here, I use the tPSA loss over a simpler L1 loss on spectrograms because deep networks trained with tPSA losses have been shown to have better performance than those with just an L1 loss [69]. However, there are no conceptual issues with using a simpler L1 or MSE loss instead of tPSA. See Section 2.3.2 for more information on the tPSA loss function. The Mask Inference network creates one mask for each source it estimates, e.g., to separation 4 sources the net makes 4 masks.

The types of systems that use Mask Inference outputs are varied; researchers have proposed many types of U-Nets [34, 35, 111, 127, 133, 148] and other types network architectures that have a Mask Inference output layer [268, 269]. A common architecture is a stack BLSTM layers, connected to a fully connected layer the output of which represents the mask [69, 265]. Here, I use the BLSTM stack (see Appendix Section 6.1 for more details about BLSTMs).

3.2.3. Chimera: Deep Clustering and Mask Inference

In 2017, Luo et. al. proposed combining Deep Clustering and Mask Inference in one network architecture [164]. Here, a shared set of neural network layers form a "body" that outputs a representation that is sent to two heads–a Deep Clustering head and a Mask Inference head. Both heads do separation, albeit in different ways; the Deep Clustering head makes a discriminative embedding space, and the Mask Inference head makes a set of masks. The Chimera network is jointly trained with the losses from both heads, in a multi-task setup. The Chimera loss is thus,

$$\mathcal{L}_{\text{CHI}} = \alpha\mathcal{L}_{\text{DC}} + (1-\alpha)\mathcal{L}_{\text{MI}}, \tag{3.3}$$

where α is a scalar that controls the weight between the two losses.

At inference time either of the heads can used for the final separation estimate. By doing both Deep Clustering and Mask Inference, Chimera networks have been shown to outperform networks that only do one or the other [164].

3.2.4. Three Heads are Better than One: Separation and Transcription with Cerberus

Taking inspiration from Chimera [164], in this chapter I propose Cerberus. Shown in Figure 3.1, Cerberus adds a third head to Chimera–a transcription head. Whereas Chimera is a multi-task network that does separation in two different ways, Cerberus extends that multi-task framework to include a completely different output modality: transcription. Now the shared layers of the "body" must transform the input mixture spectrogram into a shared representation that is suitable for both separation and transcription tasks. The separation estimates are output as I described for Chimera, unaltered, but what is new here is the transcription estimate. To produce transcription estimates, Cerberus must train using transcription data, which could make it viewed as a Score-Informed system (in the next chapter, I will describe a true Score-Informed system).

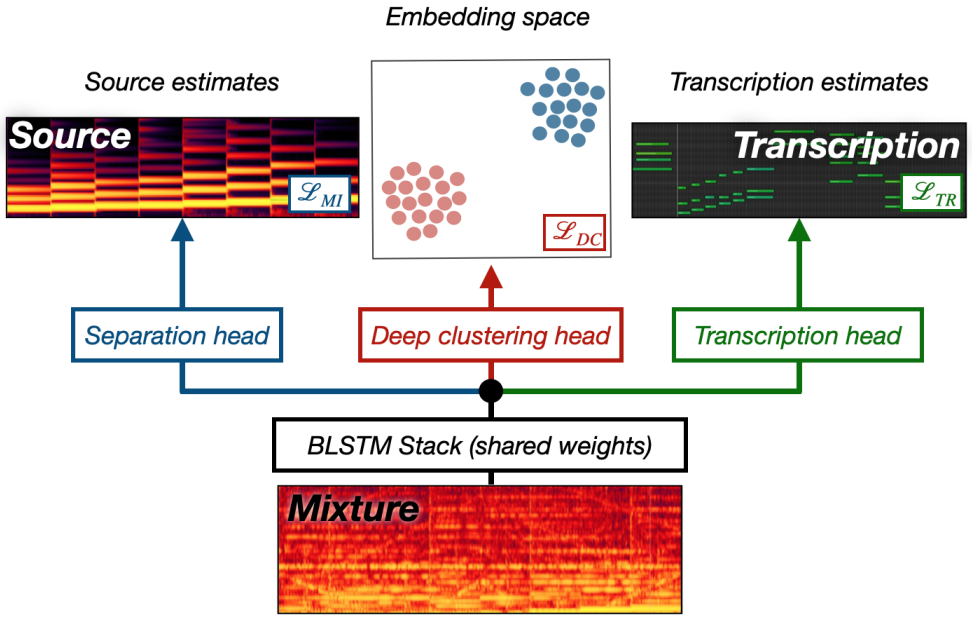


Figure 3.1. The Cerberus architecture for simultaneously separating and transcribing a musical mixture. The input is the magnitude spectrogram of a musical mixture. There are three outputs (heads): an embedding space (trained via $\mathcal{L}_{DC}$ - deep clustering loss), estimated sources (trained via $\mathcal{L}_{MI}$ - mask inference loss) and the piano roll transcription of each source (trained via $\mathcal{L}_{TR}$ - transcription loss).

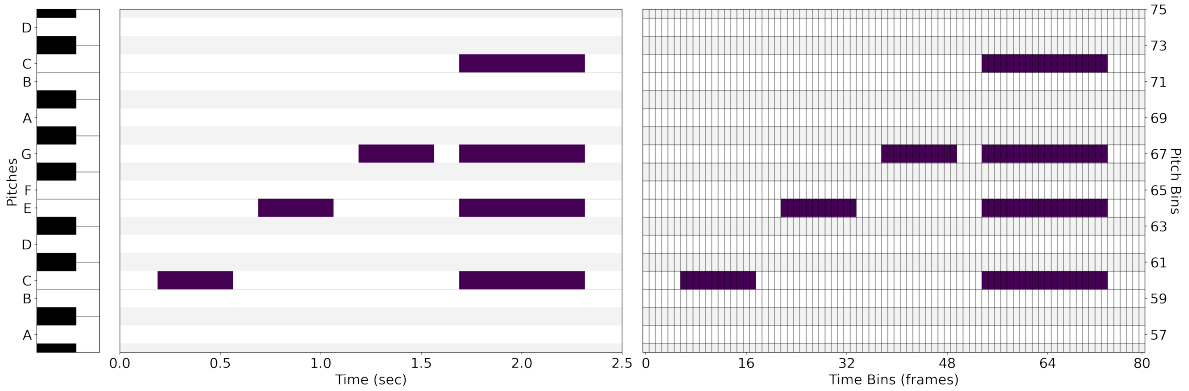


Figure 3.2. Cerberus' transcription output head produces a piano roll as a matrix. A piano roll (left) is easily converted into a matrix representation of the same data (right). The piano roll determines when any particular note is active in a performance, denoting when the note begins, which pitch the note is, the duration of the note, and when it ends. The piano roll is readily converted to and from a matrix representation: the pitch values of the equal temperament scale are used directly as the rows of the matrix and time is discretized to represent the columns of the matrix (in units called "time frames"). In the matrix, a value of 1.0 (dark values) indicates that a note is active in that time-pitch bin, and a value of 0.0 (not shown) denotes silence for that particular time-pitch bin. Piano rolls can also be used to represent the volume of a note (called its "velocity"), however estimating volume is outside the scope of this work.

The transcription estimates are output in the piano roll representation, as shown in Figure 3.2. A piano roll represents a transcription as a matrix where one axis represents time (in units of time frames of e.g., 32ms), the other represents pitch, and values in the matrix represent the presence or absence of a note at that particular pitch at that time frame. Sometimes, the values of the matrix represent the volume of that note (called its "velocity"), but here I binarize the matrix to only estimate whether a note is present or not. I leave the estimating the velocity for future work.

Cerberus outputs a *set* of piano rolls, one for each source that the system estimates. For example, a Cerberus model trained on mixtures of Piano and Guitar will output two separation estimates and two piano roll estimates–one of each for Piano and Guitar. In this example, Cerberus outputs a piano roll for representing Piano transcription and another piano roll representing the Guitar transcription.

Formally, Cerberus' transcription estimates are output as a piano roll matrix with shape $\mathbf{P} \in \mathbb{R}^{K \times T \times I}$, where K is the number of pitches, T is the number of time steps (aligned with the spectrogram input), and I is the number of instruments to transcribe (aligned with number of sources to separate). During training, the transcription head is designed to minimize the MSE between the ground truth transcription and the network's output, $\mathcal{L}_{\text{TR}}$. Combining the Chimera losses with the transcription loss produces the Cerberus loss,

$$\mathcal{L}_{\text{Cerberus}} = \alpha \mathcal{L}_{\text{DC}} + \beta \mathcal{L}_{\text{MI}} + \gamma \mathcal{L}_{\text{TR}}, \tag{3.4}$$

where α, β, and γ are scalars used to control the importance of each loss term. At inference time, any of the three heads can be used, depending on the task. In this work, for separation I use the Mask Inference head for simplicity (skipping the extra clustering step required to get source estimates from the Deep Clustering head). The raw transcription output of Cerberus is a probability estimate that a note is active for each time-pitch entry in the piano roll for each instrument; at inference time, these probabilities are thresholded to make the final transcription output.

Cerberus outputs source separation estimates as well as transcription estimates. The source separation estimate provides a low-level, high-fidelity description of each source in the mixture, and the transcription estimate provides a high-level summarization of the musical content of the mixture. Cerberus provides both, giving a more complete description of a musical scene than a system that only provides one or the other. Furthermore, because it transcribes multiple instruments in a mixture, it breaks free of the limitations posed by existing Automatic Music Transcription systems that only work for recordings of individual instruments.

3.3. Experimental Design

I designed a set of experiments to answer three questions about the Cerberus system. Each of these questions probes a certain aspect of how well the Cerberus system is able simultaneously separate and transcribe. Specifically, I wanted to find answers to the following questions:

- Does learning to simultaneously separate and transcribe using a single network help or hurt performance on either task, versus learning the tasks independently? In practical terms, do we need all three heads to make an effective separation and transcription system, or can we get away with less?
- Given that we need synthesized data (i.e., Slakh) to train Cerberus networks, how well do they generalize to recordings of live musicians?
- How does separation/transcription performance change as more instruments are added?

Each of these questions has a corresponding experiment and results that will elucidate answers to these specific questions, and provide an overall picture of the capabilities of Cerberus as it separates and transcribes in different scenarios. The rest of this section is devoted to expounding on the design of these experiments. First I will discuss these three questions in detail before describing the datasets, evaluation, and model details. I want to make special note that, despite being self-consistent, the results presented in this chapter are not apples-to-apples comparable to the results in the next chapter due to many factors (different ways of producing testing sets, different codebases, different hyperparameters, etc).

Note that while each of these two tasks–source separation and automatic music transcription–have been tackled individually, I am aware of no prior work containing a system that can jointly separate *and* transcribe multiple sources in a mixture. As such, the experiments in this chapter are designed as a proof of existence for the proposed work. Where I can, I will provide baseline systems (i.e., comparing single-task networks to multi-task networks).

3.3.1. Are Separation and Transcription Mutually Beneficial?

The first question I ask aims to understand if learning to separate is symbiotic with learning to transcribe. As I mentioned above, it has been long thought that separation and multi-instrument transcription are tasks that are reciprocally advantageous, however I am unaware of any work that has rigorously tested this hypothesis. The purpose of this first experiment is to answer this question with the Cerberus system.

From a practical standpoint, evaluating this question is simply a matter of training a system with all possible subsets of the three heads (including using all three and excluding using none). Of course, we only need two heads–one for transcription and one for separation–however, given that the Deep Clustering head is known to be valuable when doing another type of separation (i.e., Mask Inference), a natural question to ask is if all three heads can be similarly advantageous for separation *and* transcription. To train the system without a certain head, I set the loss weight for that output head to be exactly 0.0; this eliminates any gradients that will flow through that head and therefore the weights in the layers that make up that head will never change throughout training (i.e., they will have the same weights they did at initialization, namely random). That head is then ignored during all experiments at inference time. Setting the loss weight to 0.0 effectively removes the head.

Specifically, I construct the one Cerberus network (to be discussed shortly in Section 3.3.5) multiple times and train each network using a different combination of losses such that all possibilities of loss functions are used–all possibilities of the three loss functions, $\mathcal{L}_{\text{DC}}$, $\mathcal{L}_{\text{MI}}$, and, $\mathcal{L}_{\text{TR}}$. I initialize each network with the exact same weights each time, and use the same hyperparameters for each training run. The only difference is the weights applied to each loss function.

For this experiment, I only look at mixtures that contain two sources: piano and guitar. I examine the performance of each network on both tasks (where applicable) using standard metrics for source separation and automatic music transcription.

3.3.2. How well do these networks generalize to live recordings?

The next question is about how well Cerberus networks are able to generalize to recordings of live musicians. Given that Cerberus' training data is all Slakh [184] data–which is synthesized–it is logical to wonder how well a trained Cerberus model can perform on more realistic sounding input data.

This question poses an interesting challenge. One of the reasons that the Slakh data is such a valuable resource, especially for Cerberus, is because it is hard to find recordings of live musicians with highly-aligned note data. That is why Slakh is useful: because the alignment of the audio and note data comes for free via the synthesis process. This alignment is a rarity in the field, and while it is great that Slakh has aligned data, the effect of synthesizing data is that it does not sound like musicians when they play their instruments. For example, many of the instrument sounds in Slakh sound stiff and unnatural; they would never be mistaken

for a set of musicians playing the same song.[1] The reality is that there are no large datasets that I am aware of that contain real-world recordings of mixtures, isolated source data, and ground truth transcription data.

Given that this is the state of the world, it will be necessary to make a few concessions in order to answer this experimental question. While these compromises might reduce our ability to trust the resulting experiments carte blanche, they are currently the only way to understanding how well a multi-instrument separation and transcription system will perform on recordings of live musicians. That being said, I propose two ways to leverage existing single-instrument transcription datasets (with live musicians) for use in my experiments. For these experiments, I limit the data to only include mixtures of piano and guitar, as above.

The first way involves looking at transcription performance on these existing single-instrument datasets, ignoring the separation task and assuming the system is a single-instrument transcription system. For example, a common single-instrument piano transcription dataset is the MAPS dataset [64]. MAPS has audio recordings of piano performances along with aligned transcription data. In this case, to evaluate a Cerberus model trained on Piano and Guitar sources, I would run the MAPS piano data through the model and ignore the separation output and look only at the transcription performance of the network's piano output. Similarly, with the same Cerberus trained on Piano and Guitar, I could use a single-instrument guitar transcription dataset, GutiarSet [303], to evaluate the network's guitar transcription performance. Despite the slight mismatch between the training setup–which expects mixes–and the evaluation setup, this should be a decent, albeit noisy, proxy for transcription performance when evaluated on recordings of live musicians.

The second way to leverage single-instrument transcription datasets is to mix audio examples from each dataset together and use those mixtures as input data to the system. For example, in this case I would randomly grab on example from MAPS and another random example from GuitarSet, and mix the audio from both examples together to get input for a Piano-Guitar Cerberus network. I could then evaluate the separation performance comparing the network's output against the solo MAPS example as the source audio for the Piano source, and similarly with the GuitarSet audio being the Guitar source. I could also evaluate the transcription performance by comparing against the transcription data in each of these datasets.

There is a key caveat to this second approach, though: that is that the pianists in MAPS and the guitarists in GuitarSet are recorded separately, with no thought given to ever combining these recordings in the way that I propose. Because of this, when I make mixtures from these two datasets, the results are

[1]Hear some examples for yourself at: `http://www.slakh.com/#examples`

cacophonous and discordant. For instance, the MAPS dataset contains piano recordings of European classical music; GuitarSet contains recordings of guitarists playing in 5 different styles (Rock, Singer-Songwriter, Bossa Nova, Jazz, and Funk). By randomly selecting one example song from each of these datasets, we know that they will not have the same style and we are nearly guaranteed that the examples will be in different keys and have different tempos. I refer to these mixtures as *incoherent* mixtures, as opposed to *coherent* mixtures which all come from the same song. While incoherent mixing is frequently used as an augmentation strategy to train separation-only networks [148, 184, 220, 257, 280], it is rare that it is used for evaluation. For this test, the training data is coherently mixed and the evaluation data is incoherently mixed and as such the evaluation data highly dissimilar from the network's training data.

In the single-instrument evaluation and the incoherent mixing approaches, this unfortunately means that there are two experimental variables that could contribute to the results. The first variable for is the distribution shift from synthesized audio recordings to recordings of live musicians. This is the independent variable we wish to measure the effect of. The second variable is the distribution shift from the coherently mixed training data to either the single instrument audio (in the first evaluation approach) or to the incoherently mixed evaluation data (in the second evaluation approach).

However imperfect these two approaches are, they still provide hints as to a system's performance when evaluated on real world data. There is still plenty of work to be done in providing high-quality, multi-instrument transcription data containing live musicians such that we can create more trustworthy evaluation methods.

3.3.3. How does separation/transcription performance change as more instruments are added?

Given that Cerberus is the first system of its kind to simultaneously separate and transcribe, I want to test the limit of how many instrument sources it can work for. It is generally understood that more instrument sources in the mixture make it harder to separate each individual instrument; this make sense intuitively because when there are more instruments in the mix, each one becomes more difficult for a human listener to hear. The question this experiment asks is if the same thing happens with Cerberus networks: how are separation and transcription performance affected by the number of instruments in the mixture?

To answer this, I train a set of Cerberus networks that are able to support more source types than the networks in the earlier two experiments. Here I train three additional Cerberus networks using the Slakh data such that they can support 3, 4, and 5 sources, respectively. The 3-source Cerberus supports

Piano, Guitar, and Bass; the 4-source Cerberus supports Piano, Guitar, Bass, and Drums; and the 5-source Cerberus supports Piano, Guitar, Bass, Drums, and Strings. I examine how the separation and transcription performance changes using the Slakh test set as I vary the number of sources that each network can support.

3.3.4. Dataset and Evaluation

To train a Cerberus network, a dataset is required that contains mixtures, isolated sources, and ground truth transcriptions for those sources. Slakh2100 [184] is one such dataset. It is comprised of 2100 mixtures made with sample-based professional synthesizers along with isolated sources and accompanying MIDI data for each source. For more information on Slakh, see Section 1.2.2. I downsampled the audio to 16 kHz. To make an example, I randomly chose a mix, selected a random subset of the sources (e.g. piano, guitar, bass) in the mixture that match the desired sources the network will support, and then chose a random 5 second segment where all the desired sources have at least 10 note onsets with MIDI velocity above 30. The source audio for the desired sources was combined to make a mixture of just those sources. STFTs with 1024-point window size and 256 sample hop were calculated from mixture segments as inputs to the network. The musical score was the accompanying MIDI data binarized with a velocity threshold of 30 (i.e., any note with a velocity value over 30 is considered "on", else it is considered "off").

Using this procedure, I made four subsets of the original Slakh dataset, each with 20,000 audio segments (28 hours) for training, 3,000 (4 hours) for validation, and 1,000 (1.4 hours) for testing. These segments were chosen randomly using the procedure described in the previous paragraph, and they were selected according to the train, validation, and test set defined by Slakh (i.e., a training segment is randomly selected from Slakh's training set). These four subsets had different instrumentation. For the first two experiments, I used mixes that had just two sources: Piano and Guitar. For the final experiment, there were three additional datasets with the following instrumentation: the 3-source dataset had Piano, Guitar, and Bass; the 4-source dataset had Piano, Guitar, Bass, and Drums; and the 5-source dataset had Piano, Guitar, Bass, Drums, and Strings.

In addition to the synthesized audio data, I evaluated on recordings of live musicians to answer the second experimental question (Section 3.3.2). As I mentioned, I am aware of no large dataset that contains recordings of live musicians with mixture data, isolated source data, *and* ground truth transcription data. Thus, I made mixtures from two transcription datasets of real solo instrument recordings. The first is

the MAPS dataset[2] [64], which contains 30 live piano performances of classical music recorded with two microphones (60 clips total), and a Disklavier MIDI recorder. The second dataset is GuitarSet [303], which contains 360 30-second guitar excerpts in 5 styles. I downsampled all audio to 16 kHz, selected 5 second segments with at least 10 note onsets, and used the same STFT parameters. I randomly selected segments from each dataset to make 1000 instantaneous mixtures with accompanying sources and score data. These mixes are incoherent, meaning the mixed segments come from different songs with different tempos and key signatures (MAPS is classical piano, GuitarSet is rock guitar). This data is highly dissimilar to the network's training data.

For source separation, I used the scale-*dependent* source-to-distortion ratio (SDR) [154] for evaluation (see Section 1.2.3). As I have discussed earlier in this document, SDR's correlation with human perception is rocky at best [20, 23, 24, 77]. However, SDR is intended to measure how close a system's source estimate is to the ground truth source, which does make it a somewhat useful indicator of how well a system is able to separate. Furthermore, it is much easier to run on the thousands of separation examples in the test set (versus doing listener studies). For evaluating transcription performance I used precision, recall, and F1-score of note onsets and offsets using the `mir_eval` toolbox [222] (see Section 1.3.2). Both SDR and F1-score are both commonly used measures in the literature for their respective tasks.

Note that even though the experiments in this chapter are self-consistent (i.e., same data, same implementation, same evaluation used throughout the chapter), they are not apples-to-apples comparable with the experiments coming up in Chapter 4.

3.3.5. Model Configuration Details

All the networks I trained use a stack of 4 bidirectional long short-term memory (BLSTM) layers. Each BLSTM has 300 hidden units. I trained each network for 100 epochs using an Adam optimizer with an initial learning rate of 2e-4, a batch size of 40, and a sequence length of 400 frames. Each network had three heads. The first head maps each time-frequency point to a 20-dimensional embedding space, with sigmoid activation and unit-normalization. The second head outputs masks for each of the sources I trained the network to separate (between 2 and 5 masks), with a softmax activation across the masks. The third head outputs transcriptions for each source and has a sigmoid activation. Each transcription contains 88 pitches and when each pitch is active. For evaluation, I binarized the network's transcriptions using an experimentally

[2]The MUS partition of both ENSTDkAm and ENSTDkCl.

determined threshold of 0.8, except for drums, which was set to 0.1. Each network was initialized with the same set of weights. The only difference between the networks is the training data (for which instrument combination to separate) and the weights on the the loss functions: deep clustering (DC), mask inference (MI), and transcription (TR).

3.4. Results

In this section I will present and analyze the results from the experiments outlined in the previous section. These experiments are designed to answer questions about the performance of Cerberus. This section is organized around each of the questions posed above.

3.4.1. Are Separation and Transcription Mutually Beneficial?

For this experiment, I was focused on understanding whether the tasks of separation and transcription actually do benefit each other if learned jointly. I trained a Cerberus network to separate and transcribe mixtures of one piano and one guitar from Slakh2100. I compared this to training network with a subset of Cerberus' heads by turning off the losses corresponding to the heads not included in the subset. For example, turning off the transcription loss (i.e., $\gamma = 0.0$ in Equation 3.4) results in a standard Chimera network. The loss weights for the non-zero losses were determined experimentally in such a way that the loss values for each head were all within one order of magnitude difference. For example, the transcription loss weight was set to 0.8 to counteract the larger magnitudes of the separation losses. Transcription F1-scores for an untrained network are on the order of 1e-3.

The results for this experiment are shown in Table 3.1. These results suggest that transcription and separation *can* be learned jointly, given the correct training regime. First, I find that the best performing model for both transcription and separation was the Cerberus model, which surpassed or tied the highest SDR and precision, recall, and F1 scores of the remaining models. Combining the Mask Inference and TRanscription (MI+TR) objectives resulted in higher transcription performance but lowered separation performance very slightly. The difference between the Cerberus and the MI+TR results is small, especially given that the results reported are means. However, the difference between Cerberus and the other single-task systems is much larger; e.g., 1.5 dB difference in SDR between Cerberus and Deep Clustering and 0.03 difference in F1 score. Additionally, combining the deep clustering and transcription (DC+TR) objectives resulted in a large jump in SDR over just deep clustering, suggesting natural synergy between the two tasks.

Network	Loss Weights			Separation	Transcription		
	$\mathcal{L}_{DC}$	$\mathcal{L}_{MI}$	$\mathcal{L}_{TR}$	SDR (dB)	P	R	F1
Deep Clustering	1.0			8.5	–	–	–
Mask Inference		1.0		**10.0**	–	–	–
Transcription			1.0	–	0.48	0.43	0.44
Chimera	0.5	0.5		9.8	–	–	–
DC+TR	0.2		0.8	9.3	0.48	0.41	0.43
MI+TR		0.2	0.8	9.8	**0.51**	**0.46**	**0.47**
Cerberus	0.1	0.1	0.8	**10.0**	**0.51**	**0.45**	**0.47**

Table 3.1. Cerberus networks trained and tested on piano & guitar mixtures. Each row is a distinct network, trained with a distinct combination of three loss functions: Deep Clustering (DC), Mask Inference (MI) and Transcription (TR). The weight applied to a loss function is used as shown where empty cells denote 0.0 weight applied to that loss. Evaluation measures for the transcription task are the mean precision (P), recall (R) and F1 of note onsets & offsets over all examples in the test set. For separation I report the mean of (scale-dependent) source to distortion ratio (SDR) over the test set. Higher values are better. The value in each cell is on the testing data, averaged across both instruments. Empty cells (i.e., cells with '–') indicate the network was not trained for that task.

I will underscore that the best performance for both tasks occurred when using all three of Cerberus' heads, further suggesting that separation and transcription are mutually beneficial.

3.4.2. How well do these networks generalize to live recordings?

Next, I took networks trained on the synthesized Slakh dataset and evaluated them on the dataset I generated from MAPS and GuitarSet. Recall that these evaluation mixtures are *incoherent*, meaning that the sources are mixed randomly and sound cacophonous. The evaluation mixes are extremely unlike the training mixtures.

The results from this experiment are shown in Table 3.2. The first thing to notice is that there is a significant drop in separation and transcription performance when compared to the evaluation numbers in Table 3.1. The separation performance is reduced by half according to SDR, and the transcription F1 performance is roughly 25% of the performance on Slakh data. As I mentioned above in Section 3.3.2, this could be due in large part to the major differences between the training and test data, namely that the training data contains all coherent mixes (instruments all playing the same song) from Slakh (synthesized) and the test data is all incoherent mixes (instruments mixed from different songs) from two disparate datasets.

Test Dataset	Mixes?	Network	Separation SDR (dB)	Transcription P	R	F1
GS		Transcription	–	0.08	0.08	0.08
GS		**Cerberus**	–	**0.13**	**0.11**	**0.12**
M		Transcription	–	0.19	0.08	0.11
M		**Cerberus**	–	**0.19**	**0.10**	**0.12**
M + GS	✓	Deep Clustering	4.3	–	–	–
M + GS	✓	Mask Inference	4.1	–	–	–
M + GS	✓	Chimera	4.5	–	–	–
M + GS	✓	Transcription	–	0.14	0.08	0.09
M + GS	✓	**Cerberus**	**5.0**	**0.16**	**0.10**	**0.12**

Table 3.2. Piano/Guitar performance on MAPS and GuitarSet data using networks from Table 3.1 trained on Slakh2100. *M* means MAPS recordings in isolation, *GS* means GuitarSet recordings in isolation, and *M+GS* means incoherent mixtures of recordings from MAPS and GuitarSet. A check mark under the "Mixes?" column also denotes that the evaluation datasets (i.e., MAPS & GuitarSet) were mixed together. Empty cells (i.e., cells with "–") indicate the network was not trained for that task. Evaluation measures for the transcription task are the mean precision (P), recall (R) and F1 of note onsets & offsets over all examples in the test set. For separation I report the mean of scale-dependent source to distortion ratio (SDR) over the test set. Higher values are better.

In spite of this, I note that the Cerberus model that was trained with all three loss functions out-performs all of the single-task networks for both separation and transcription, suggesting that its multi-task separation and transcription approach leads to better generalization.

3.4.3. How does separation/transcription performance change as more instruments are added?

Finally, I trained and tested a Cerberus model on Slakh datasets of increasing numbers of simultaneous polyphonic instruments to see how the system scales up to more complex and busy mixtures. The results, shown in Table 3.3, show that as more sources are added to the mixture, performance across both separation and transcription predictably degrades. This same data is shown in Figure 3.3, where the Bass, Piano, and Guitar separation and transcription performance are plotted in an effort to more clearly show this trend. In these plots, I only include Bass, Piano, and Guitar because these are the three instruments that are consistent through all of experiments in Table 3.3. These three instruments are also all harmonic sources (i.e., they have a well-defined pitch. Contrast this with a percussive source, like Drums.), which means that they are a good basis for comparing similar sounding sources.

I want to tease out some subtleties about the interactions that occur between sources, and how those interactions are reflected in the results in Table 3.3 (and Figure 3.3). Notice how drastic the separation

Cerberus trained/tested on data that contains:		Separation SDR (dB)	Transcription P	R	F1
3 Sources	Piano	7.6	0.44	0.42	0.42
	& Guitar	6.9	0.46	0.35	0.38
	& Bass	10.1	0.85	0.80	0.82
4 Sources	Piano	6.1	0.38	0.36	0.36
	& Guitar	5.8	0.42	0.32	0.34
	& Bass	7.7	0.82	0.78	0.79
	& Drums*	11.3	0.61	0.76	0.63
5 Sources	Piano	3.4	0.31	0.28	0.28
	& Guitar	3.1	0.29	0.20	0.22
	& Bass	6.4	0.77	0.72	0.74
	& Drums*	10.6	0.62	0.75	0.64
	& Strings	4.1	0.39	0.35	0.35

Table 3.3. Results for individual instruments from three Cerberus networks trained on different sets of instrument combinations, separated by horizontal lines. Each model has its own training, validation, and test set which depend on the instruments it is trained to separate and transcribe. Evaluation measures for the transcription task are the mean precision (P), recall (R) and F1 over all examples in the test set. Drum (*) transcription evaluation measures are note onset only, all other instruments are note onset/offset precision/recall/f-score. For separation I report the mean of scale-dependent source to distortion ratio (SDR) over the test set.

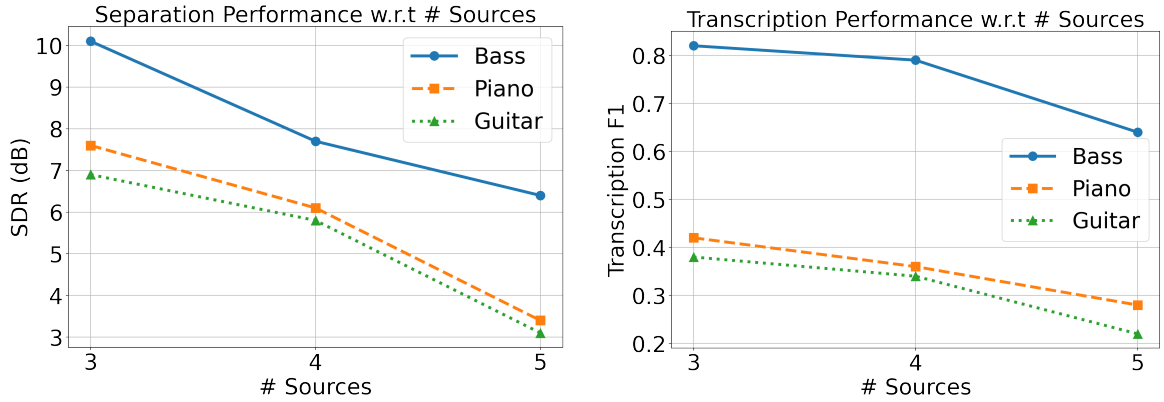


Figure 3.3. Results for Bass, Piano, and Guitar for separation (left) and transcription (right) as a function of number of sources in the mix. This is the same data shown in Table 3.3 (Drums and Strings are omitted for clarity). As more instruments are added in a mixture, separation performance (SDR) and transcription performance (note on/off F1) for all three instruments falls.

performance falls between the 4-source ensemble and the 5-source ensemble for the Piano and Guitar sources in Table 3.3; their SDR values in the 5-source ensemble are roughly *half* of their SDR values in the 4-source ensemble. The only new source between these two ensembles is the Strings source (e.g., violins, violas, cellos).

Let us focus on the Piano and Guitar sources. Why do Piano and Guitar sources see a sharper decline in separation performance when Strings were added when compared to when Drums were added? Well, to understand what I think might be happening, I would like to take a brief detour to talk about musical scenes and how different types of instruments typically interact in those scenes. To grossly oversimplify Western European music theory, we can think of instruments as occupying a few different roles in a song. Some of the roles we might expect are: a percussive instrument, a melodic instrument, a bass instrument, and a chordal instrument (i.e., responsible for harmonies). Typically, these roles are associated with certain pitch ranges–e.g., we expect the bass instrument to play low notes–or timbral qualities–e.g., a drumset is a percussive instrument. This is important to know because some instrument sources are associated with some of these roles more than others. For example, the Piano source and the Guitar source are *both* chordal instruments, meaning that they can oftentimes play the same role in a song. As such, these instruments also have a similar range of pitches that they typically play, which means they might have more overlap in frequency content. All in all, these two sources are very difficult to separate from each other.[3] So, when the Strings source is added to these mixtures, there is now a third instrument that occupies the same role as the Piano and Guitar sources: in the Slakh mixes the Strings oftentimes act as a chordal instrument, so adding them makes it harder for the network to do well separating them.

Thinking about each instrument's role also explains why the Bass and Drum sources have higher separation and transcription performance than the other instruments across Table 3.3. Bass has between 2-3dB higher SDR than Piano and Guitar and twice the F1 score across the 3-, 4-, and 5-Source mixes. The Bass plays a different role than others in a musical scene; it typically has a much lower pitch than the Piano, Guitar, or String sources, and usually there are no other instrument sources that occupy the same role at the same time (unlike chordal instruments). A similar story occurs with the Drums source: it is the only instrument playing the percussive role in these mixes, which explains why its separation and transcription performance is much higher than the other sources.

All in all, Cerberus is able to separate and transcribe multi-instrument mixtures that contain up to 5 sources. Even though performance degrades as more instruments are added to the mix, the 5-Source mixes still have consummate results on both separation and transcription. Some work that came after Cerberus examined multi-instrument transcription performance by scaling to more instruments [84, 160, 270], and one

[3] I have done work that provides direct evidence for this intuition; giving a network a hint about the content of one of these sources drastically helps the network separate the other [175].

of the emergent issues as these systems scale is that, while the onset and pitch of the note might be correct, it is difficult to make sure that it is assigned to the correct instrument. This error is a fruitful direction for future work. In the next chapter, I will examine how changing the instrument assignment of a note is useful for score-informed separation.

3.5. Conclusion

In this Chapter I introduced Cerberus, the first deep learning system to simultaneously separate and transcribe multiple instruments in a mixture. Cerberus is an important step for Automatic Music Transcription systems because it breaks through the field's existing paradigm centered around single-instrument transcription systems. Doing both separation and transcription is useful in many respects. First, it provides an end-user with different types of analysis of a musical scene; a low-level, high-detail analysis describing the source audio (i.e., the source estimate) and high-level, sparse-detail analysis describing the note events (i.e., the transcription estimate). Both types of analysis are important for different use cases, so giving a user both enables a more complete understanding of the content of a musical scene. Additionally–and more practically–training a network to separate and transcribe is better for both tasks; the highest performance came when using all three heads of Cerberus (Tables 3.1 and 3.2). Finally, I showed how Cerberus enables the simultaneous separation and transcription of up to five sources in a mixture.

While Cerberus could be viewed as a Score-Informed separation system because it uses a musical score (i.e., the piano roll) during training, Cerberus is also a crucial stepping stone for the next project in this dissertation: Deep Score-informed Separation, which is bona fide Score-Informed system. I will go into more details in the next chapter, but having a multi-instrument transcription system (like Cerberus) as a pre-processing system has the potential to make Score-informed Separation much easier to use for end-users.

CHAPTER 4

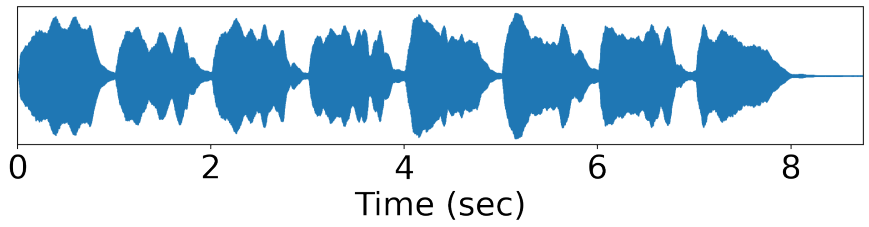
Deep Score-Informed Separation

In Chapter 2, I described a hierarchical Query-by-Example (QBE) separation system that enabled a user to input a query audio clip to steer the separation system to output a different source type. In Chapter 3, I examined how doing both separation and multi-instrument Automatic Music Transcription (AMT) can provide multiple ways to examine a musical scene. In this chapter, I combine these ideas: here I aim to make a controllable separation system conditioned on note data, such as to enable finer-grained, note-level edits of the output source estimates. For example, this system should let a user move the audio content of a *single note* from one source estimate to another. Put succinctly, the goal is to make a system that can enable an end-user to steer the output of a separation system after it is trained by editing a piano roll transcription.

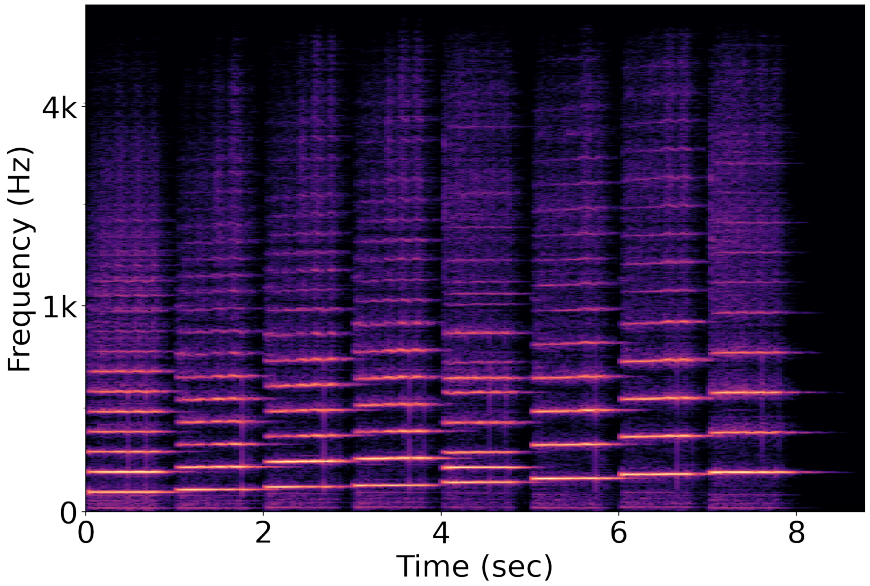
Why is this an important goal? One of the themes woven into the work of this thesis is that the musical scene analysis systems we build will never be flawless, and thus it is critical to make these systems adjustable post-hoc so that end-users have some recourse when they inevitably make mistakes. If the research community aims to put these tools in the hands of musicians and creators, then making the underlying machine learning systems interactive is crucial.

The question then shifts to what the most effective way of providing a way to make adjustments to audio. Without a doubt there are many ways to answer this question depending on the situation, but in this chapter I focus on enabling note-level edits of source estimates. To facilitate this interaction I chose to design a system that uses a piano roll to represent note data.

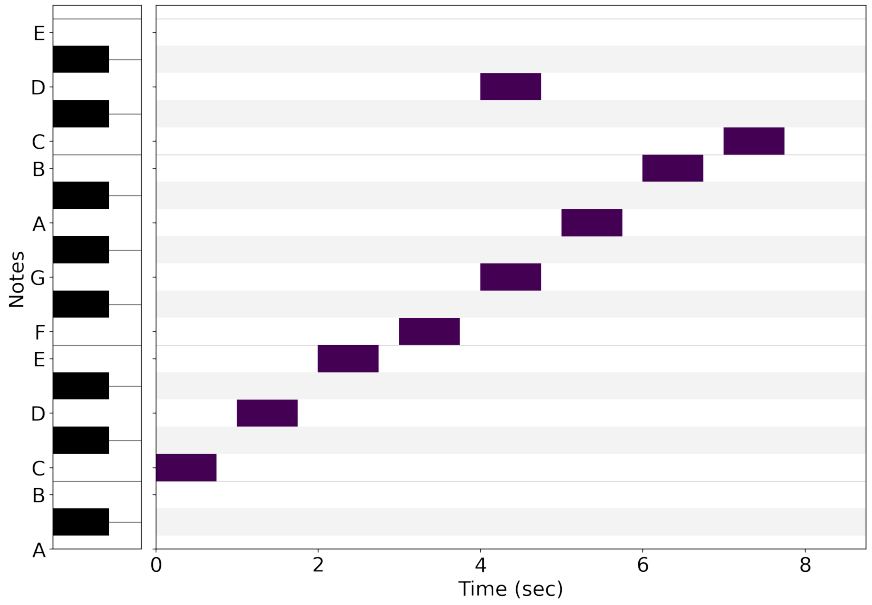
I want to justify why the piano roll is the right way to represent the musical content for the desired interaction in this chapter. Simply put, my argument is that manually editing audio signals is hard, but, on the other hand, a piano roll is a distilled representation of *note* data, the information we care about in this situation. To illustrate this, imagine that we are tasked with editing the output a source separation system and, when listening to the output we hear the audio of an extra note from an interfering source. Let me visualize the data to demonstrate different ways of representing this data. The same musical content is shown as a waveform in Figure 4.1a, as a spectrogram in Figure 4.1b, and as a piano roll in Figure 4.1c.



(a) A Waveform.



(b) A Spectrogram.



(c) A Piano Roll.

Figure 4.1. Three representations of the same musical content, where somewhere there is an extraneous note that we want to remove. The waveform (4.1a) obscures all information except loudness over time; it is impossible to make out the errant note, let alone fix it. The spectrogram (4.1b) shows frequency content, so it is possible to spot the extra note if you know what to look for, but it requires editing tens of thousands of points to fix. The extra note is obvious in the piano roll (4.1c), because the notes are explicitly represented.

Even finding this extraneous note is impossible in the waveform (Figure 4.1a), let alone editing it. The offending note is inextricably enmeshed with the other notes of the desired source. If we were somehow able to determine which of the audio samples needed editing, we would need to change over 64,000 samples[1]– saying nothing about *how* they need to be edited, just where the edits need to occur. In short, editing a waveform by hand is borderline masochistic. In the spectrogram (Figure 4.1b), finding where an extra note occurs is at least possible, but editing a spectrogram manually is a laborious process. While there are user-facing tools available that make spectrogram editing a little easier [1, 16, 125], in many cases there is significant overlap between the content we want to keep and the content we want to scrap, making the edits possible but challenging. Practically speaking, the most granular of control over the output audio comes from editing the spectrogram (because the waveform is too detailed to meaningfully edit by hand). Even in easy cases, knowing identifying which of the $\geq$750,000 time-frequency bins (in this example) need to be edited and figuring out how modify them to correct a stray note is difficult and requires a considerable learning curve.

On the other hand, looking at the piano roll in Figure 4.1c, there are only 9 dark rectangles in the image, which are the 9 active notes in the audio. Determining which of these notes is out of place[2] is much simpler because there is much less information to contend with. The note data in this piano roll is a distilled version of musical content that preserves note pitches, onsets, and offsets. Compared to the spectrogram, which has a lot of detail, and the waveform, which has perhaps *too* much detail, the piano roll is much less information dense, and therefore easier to parse for determining how to edit a note in the output separation. Furthermore, to actually do the edit, we only change one of these rectangles that represents the notes, rather than the 10s or 100s of thousands of data points needed to edit the spectrogram. For this interaction the piano roll presents the relevant information to us at the right level of granularity to edit a note in the output of a source separation system.

Zooming out from this example, state-of-the-art deep learning-based source separation systems offer no recourse to edit their audio output. Lots of effort has gone into making these models work better and better [33, 35, 43, 44, 111, 141, 148, 161, 198, 233, 265, 268, 269], however relatively little work has been focused on adding making the output of these systems changeable or steerable. The work that does exist that enables changing a separation system's output centers around changing the entire source that the system

[1]Assuming a sample rate of 44.1 kHz
[2]The extra note is the D that starts at 4 seconds in.

outputs (as done in with the QBE system in Chapter 2 and by many other similar systems [30, 34, 87, 157, 160, 182, 192, 232, 241, 251]). To make more granular edits the output of any of these systems, a user is stuck back at editing a spectrogram. Existing systems do not support making note-level changes to their output audio, which is the goal of this chapter.

In this chapter, I revive the now-dormant idea of Score-Informed Separation and approach it with modern deep learning methods. Score-Informed Separation is a type of source separation method where the system is conditioned on a musical score as a means to guide the output source estimates. Score-Informed Separation systems were very popular prior to the deep learning era, but fell out of fashion in recent years. Here, I propose Deep Score-Informed Separation systems that use a piano roll score as auxiliary conditioning data for separation. Importantly, I will demonstrate that these systems allow an end-user to make note-level edits to the audio content in the output source estimates.

4.1. Contributions of this Chapter

This chapter makes the following contributions:

- I propose Deep Score-Informed Separation systems that are able to use a musical score as additional conditioning data when doing source separation. In doing so, I revive a long-dormant idea that predates modern techniques and update it for use with deep neural networks. I specifically propose to modify an existing source separation network in three ways such that it can accommodate input score data in the piano roll format.
- I also propose a complementary task, Deep Source-Informed Transcription, which does multi-instrument Automatic Music Transcription (AMT) from a mixture conditioned on isolated source data. I experimentally verify that this system performs better than a baseline system according to standard transcription metrics. While this task setup has some practical limitations, it does lead to interesting questions about how multi-instrument AMT should be designed in future work.
- I provide a set of experiments that verify that the Deep Score-Informed Separation systems do in fact lead to better separation performance than a baseline system that does not use score data as input.
- I show off another set of experiments demonstrating that a Deep Score-Informed Separation system responds to changes in its input score data. I this in two ways: first by using a systematic method

for changing the input score and examining how this affects the output source estimates, and the second way is to explore a set of qualitative examples simulating a few common usage scenarios.

- I envision an interface that could be built on the Deep Score-Informed Separation system described in this chapter and I outline how such an interface can enable an end user to make edits to the source estimates in a way that is easier than manually editing a spectrogram.

4.2. Score-Informed Separation

Score-Informed Separation systems use a musical score, or transcription, as additional conditioning information when separating an input mixture. This score gives the source separation system a distilled representation of the musical content in the musical scene. The goal of a Score-Informed Separation system is that it should be able to use this score data to both aide separation performance and, more importantly, the score input can act as a control mechanism for an end user to change the output separation after the system is trained (see Figure 4.2).

Historically, Score-Informed Separation received a lot of attention prior to the rise of deep learning. These early systems were built on Non-Negative Matrix Factorization (NMF) [75, 156, 252, 285]. NMF works by decomposing a mixture spectrogram into two matrices, one where the columns represent a basis set of spectral templates (i.e., a single column in a spectrogram) and the other whose rows represent when those templates are active. The problem is that most realistic signals contain contain sources with many spectra, and thus NMF would *over*separate a musical scene if left to its own devices. Therefore additional score input information was used as means to constrain the system and produce better separation estimates [26, 27, 49–51, 58, 59, 70–72, 78, 79, 81, 82, 109, 131, 162, 195–197, 201, 202, 228, 248, 249, 254, 259, 300]. As deep learning-based systems superseded older NMF systems because of their improved separation performance, the need for additional score information fell by the wayside, and–with few minor exceptions [28, 91, 176, 197]–Score-Informed Separation was largely ignored.

Unlike typical deep learning-based source separation systems, which offer no intrinsic means of making alterations to the source estimates (outside of editing the output with a spectrogram editor), Score-Informed Separation promises that a user could make changes to the output separation estimate at the granularity of the individual notes from the source. The score provides a means for the user to input their desired changes to the system when it produces source estimates. For example, if a separation system accidentally puts the

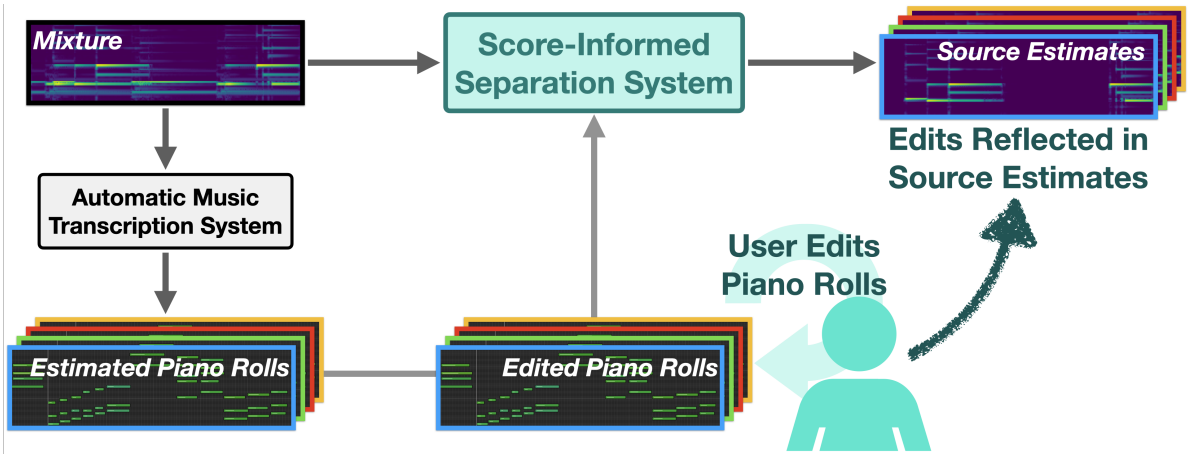
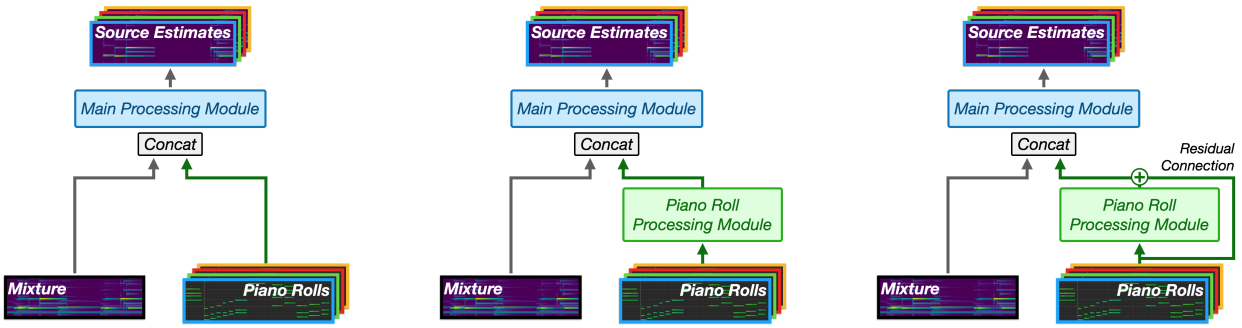


Figure 4.2. Score-Informed Separation systems use score information (shown here as a set of piano rolls) as an auxiliary conditioning when doing source separation. Given an input mixture (top left), a multi-instrument Automatic Music Transcription (AMT) system could be used as a pre-processing step to make an initial set of piano roll estimates (bottom left), that a user could then edit and refine (bottom right). The piano rolls are used as input to the Score-Informed Separation system (top center) and they guide the source estimates that the system produces (top right), in such a way that the edits made in the piano roll are reflected by the source estimates that the system outputs.

sound of a few piano notes into a the Guitar source estimate, a user could use the score data to move that piano note audio data from the Guitar source estimate to the Piano source estimate.

I want to outline one of the explicit design limitations of Score-Informed Separation systems, which is that no new audio data is created. That is, as I have formulated the separation problem in the Introduction Section 1.2.1, the separation system is designed to only *remove* audio from a mixture to produce a source estimate. This is important to mention because it might be tempting to imagine that a user can move notes arbitrarily and expect that the system will produce sound for any possible input score. In other words, if a user inputs a note that does not correspond to anything in the input mixture, that note will not be generated into audio by this system. On the other hand, if there is audio in the input mixture that *does* correspond to a desired note, it should be able to be moved around to different sources. Generating new audio data based on notes is outside the scope of this work, but is a fruitful direction for future work.

Of course, one of the prerequisites for training and running a score-informed system is actually having a score to use as input. This score data could come from a number of places: a user could source a score from the internet, they could create one from scratch, or perhaps the score could come from a multi-instrument Automatic Music Transcription (AMT) system. Creating the score from scratch might be difficult; in this case the user must transcribe their desired instrument(s) that they hear in the scene. As discussed earlier in



(a) Vanilla DSIS (b) Piano Roll Processing DSIS (c) Residual DSIS

Figure 4.3. All of the variations of Deep Score-Informed Separation (DSIS) architectures that I use in this dissertation. For the Main Processing Module, I use a stack of BLSTM layers. For the Piano Roll Processing Module, I use a smaller stack of BLSTM layers.

this document (Introduction Section 1.3.1), transcribing even one instrument by hand is a tedious problem even for trained musicians. Therefore, using a multi-instrument AMT system (like Cerberus from Chapter 3, for example) as a pre-processing step for a Score-Informed Separation system is a good idea. Even if the estimated score from the AMT system is imperfect, it gives the user a starting point to further refine. This might make it easier for a user to localize their edits around a region of interest, focusing their energy on making tweaks to the output of the AMT system rather than starting from a blank page.

Score-informed Separation systems promise to provide a powerful way for end-users to manipulate the output of a separation system that is finer than a QBE system and at a more appropriate granularity (the note) than a spectrogram. Pairing them with a multi-instrument AMT pre-processor can make editing the output of source separation systems easier. Outside of those benefits, a Score-informed Separation system could also produce better separation estimates based on the fact that the score input data gives the system a hint of the musical content of each source needed to separate. The rest of this section will be devoted to detailing my proposed Score-informed Separation neural network architecture.

4.2.1. Deep Score-Informed Separation: System Architecture

To accomplish the goals of the Score-Informed separation system, I designed a neural network architecture such that it had an additional way to input score data alongside the mixture data. As with the previous two chapters in this dissertation, I modified a standard source separation system such that it could accept the score data as conditioning input. In fact, it is simple to modify any source separation architecture to be score-informed: just let the system use the score (Figure 4.2).

While there are many ways to represent the input score data, here I again turn to using a piano roll.[3] Recall that a piano roll represents score data as a matrix, where one axis represents pitches, the other represents time frames, and values indicate whether a note is active at a given time-pitch bin. For more information on piano rolls, see the Introduction Section 1.3.1 and Cerberus Section 3.2.4. In this chapter, the system is input a binary piano roll matrix with shape $\mathbf{P} \in \{0,1\}^{K \times T \times I}$, where K is the number of pitches, T is the number of time steps (aligned with the spectrogram input), and I is the number of instrument sources the system can separate. Zero (0) values denote that there is no active pitch for that instrument at a given time-pitch bin, and one (1) values denote an active pitch.

In this chapter, I will explore three different variations on a standard source separation that I modified to ingest piano roll data. I call these systems Deep Score-Informed Separation (DSIS) systems. The variation in these three DSIS systems stems from how they process the piano roll data. Before I go into details about how the different parts of these three systems work, I will give an overview of how the parts in these systems fit together. The three different variations on the DSIS system are as follows:

Vanilla DSIS: which only has Main Processing Module, shown in Figure 4.3a. This system has one set of layers that jointly process the mixture and piano roll input data.

Piano Roll Processing DSIS: which has the same core Main Processing Module as the Vanilla system, as well as a Piano Roll Processing Module dedicated to processing the piano roll input before being combined with the mixture input and given to the Main Processing Module. The Piano Roll Processing Module is designed to learn a representation of the piano roll score that can help aid the Main Processing Module do separation.

Residual DSIS: which has the same building blocks as the Piano Roll Processing architecture: the Main Processing Module and the Piano Roll Processing Module. In this case, the dedicated Piano Roll Processing Module has a skip connection around it (i.e., a residual connection). Residual connections can often produce well-behaved gradients during learning [4] and have been proven useful for many tasks in the image [107] and audio domains [116]. Just as before the Piano Roll Processing Module is designed to learn a representation of the piano roll score that can help aid the Main Processing Module do separation.

[3]In this chapter, I will frequently discuss the piano roll representation–recall that a piano roll can represent note data from many instruments (e.g., Guitars, Bass, Drums, etc). I will also discuss the Piano source which corresponds to audio from a Piano instrument. So as not to confuse these terms, I will write 'piano roll' uncapitalized, and I will refer to the Piano source with capital letters. Therefore the Piano piano roll refers to representing note data from a Piano instrument with a piano roll. Same for a Guitar piano roll: Guitar instrument, piano roll representation.

In this chapter, the Main Processing Module consists of a stack of Bidirectional Long Short-Term Memory (BLSTM) layers [94, 118] (see Appendix Section 6.1 for more details about BLSTMs, and see the experiments Section 4.3.5 for specifics on the model configuration). In a similar manner to the work in Chapters 2 and 3, these Main Processing Layers output a set of masks that are applied to the mixture spectrogram to produce a set of separation estimate. For more on masking see Section 1.2.1 in the Introduction. Just as before, I use the truncated Phase Spectrogram Approximation (tPSA) [69], as the objective function for this system:

$$\mathcal{L}_{\text{MI}} = \mathcal{L}_{\text{tPSA}} = \left\| \hat{M}_c \odot |X| - \text{clamp}\left(|S_c| \odot \cos(\angle S_c - \angle X)\right) \right\|_1, \tag{4.1}$$

where $|X|$ and $\angle X$ denote the magnitude and phase of the mix spectrogram, respectively, and $\text{clamp}(x) = \min(\max(x, 0), |X|)$ is a truncation function ensuring the tPSA target can be reached with a sigmoid activation function. tPSA is a variant of L1 loss that accounts for phase differences between the ground truth source and the mixture. This loss has been shown to perform better than just plain L1 [69].

The Piano Roll Processing Module used in the latter two Deep Score-Informed Separation (DSIS) variations is also made of BLSTM layers in this work, albeit fewer layers than used in the Main Processing Module. I will provide specifics about the module's size when I discuss the experiments in Section 4.3.5. The Piano Roll Processing Module is tasked with learning a representation of the piano roll that is helpful when doing separation. For example, one could speculate that these layers might learn to map between the pitch bins in the piano roll and the frequency bins in a spectrogram, freeing up capacity in the Main Processing Module to focus on separation directly. These layers have no specific loss term applied directly to them; their only loss is the source separation loss, which is applied to the whole system. Therefore, whatever these representations are, they must be in service of producing better source estimates.

In all systems, normalization layers are applied separately to the mixture inputs and piano roll inputs. Here, I use Batch Normalization [123]. Specifics of the model's configuration–like hyperparameters, number of layers, etc–will be provided when I discuss the experiments in more detail in Section 4.3.5.

The design of these Deep Score-Informed Separation systems originates from the goals of this chapter. I want to make a separation system that can enable a user to make edits to the separation by editing note data. Because I am designing a source separator, this system consumes a mix and outputs source estimates. The note editing piece is serviced by the piano roll input. The expectation is that, just by making the piano roll act as an additional conditional input, these systems will enable a note editing interaction at inference time. I will test and verify this assumption with a set of experiments over the rest of this chapter.

4.2.2. Source-Informed Transcription

Similar to Score-Informed Separation, we can set up the complementary task, *Source-Informed Transcription*. Here, a multi-instrument Automatic Music Transcription (AMT) network is given both the mixture and isolated source data. The architecture of the system can be set up nearly identically to the Score-Informed Separation system, other than the fact that the inputs and outputs are changed for the task. The output of this system is a set of piano rolls that estimate a transcription for each instrument in a given mixture. Just as before, this network is trained with an MSE loss against the ground truth piano rolls, and at inference time the network's output is thresholded to get a set of binary piano rolls.

This setup by itself presents a difficult use case to justify; if a user has both the isolated source audio and a mixture, it seems rare that they would use both to produce a transcription estimate rather than just using a transcription system that could transcribe isolated source data. However, this setup bridges a gap between the literature on single-instrument transcription and multi-instrument transcription. Later in the chapter, I will show results for this setup, though I will not place a heavy emphasis on them due to their limited application towards the goals of this chapter.

4.3. Experimental Design

I designed experiments to test how well the proposed Deep Score-Informed Separation (DSIS) system achieves the goals set out in Section 4.2. Namely, I want to understand how the additional score input information affects the output separation performance. For completeness, I will also examine the Source-Informed Transcription system and see how additional source data input affects transcription performance. Perhaps the most pertinent question I want to examine is if the score can also be used as means of controlling or changing the source separation output.

Specifically I want to design experiments to answer the following questions:

- How does additional score input information affect output separation performance? Are certain architectures better than others? I.e., do the additional score processing layers help? Does the residual connection help?
- How does additional source input data affect the output transcription performance?
- Can the input score information be used to alter the output source estimates?

The next three parts of this section will be centered around expanding these research questions into experiments, before I provide details on the data, evaluation, and model configurations that will be used in

the across all the experiments. I want to make special note that, despite being self-consistent, the results presented in this chapter are not apples-to-apples comparable to the results in the previous chapter due to many factors (different testing sets, different codebases, different hyperparameters, etc).

As with the prior two projects in this document, the problem setup in this chapter is entirely novel (to my knowledge) and represents a new task (i.e., editing a source estimate via editing a piano roll) rather than an iterative improvement on an existing task. Just as before, I will compare to appropriate baselines by doing ablation studies (i.e., varying the components of the system) where it makes sense to do so.

4.3.1. How does additional score input information affect output separation performance?

In this first experiment, I want to ask the fundamental question of whether adding score information as an auxiliary input to a source separation network helps or hinders separation performance. To do this, I will compare the separation performance of the Deep Score-Informed Separation (DSIS) network detailed in Section 4.2.1, to a baseline system. Unlike the DSIS network, which requires a mixture and a piano roll score as input, most typical source separation systems only require a mixture. Therefore, for a fair comparison against a stronger baseline, I want to consider a system that also uses piano roll score data. With that in mind, I will use the Mask Inference and TRanscription (MI+TR) network described in Chapter 3 as a baseline for this work. This MI+TR network does not take the transcription as input–it only inputs the mix–but MI+TR does use transcription data during training as it optimizes a transcription output head. So, the MI+TR network has seen the same data that the DSIS systems have (i.e., mixes, sources, and piano roll data), even though it uses it differently. In this experiment, I will only look at the separation estimates of the MI+TR network, ignoring the system's transcription output. The MI+TR system uses the piano roll as a loss target, whereas the proposed DSIS system uses the piano roll as additional conditioning data. This, of course, comes with similar requirements at inference time, where the DSIS system needs the additional score data.

In response to this last point, I will also compare the performance of the three different variants of the DSIS system proposed in Section 4.2.1. In this case, all three variants have the same input requirements during training and testing time (i.e., all take in the mix and a piano roll). Diagrams that outline these three variants are shown in Figure 4.3. During training, all DSIS systems are given ground truth piano rolls as input (i.e., they are teacher forced). For these experiments, I will also provide the system with ground truth piano rolls at inference time; in a later section, I will explore using piano rolls that are estimated from a

multi-instrument Automatic Music Transcription (AMT) system. Using ground truth piano roll performance gives us an upper bound as to the performance of the DSIS systems.

For this experiment, I will use the improvement of the scale-*independent* source-to-distortion ratio (SI-SDRi) [154] for evaluate source separation performance. As a reminder, the SI-SDRi measures the difference in SI-SDR when using the mix as input versus using the source estimate as input (see Section 1.2.3 for more info). Again, all of the caveats about SI-SDRi still apply here: while it has a weak correlation with human auditory perception [20, 23, 24, 77], it is still useful as a measure because it measures the similarity between a source estimate and the ground truth source audio. Also note that, while this is the regular SI-SDRi discussed earlier, in Section 4.3.3 I will introduce a special variation on this to report results from altering a the piano roll input.

4.3.2. How does additional source input data affect the output transcription performance?

Similar to the previous question, here I want to understand how source input data impacts the quality of a multi-instrument Automatic Music Transcription (AMT) system. Here, I set up the experiment analogously to the preceding section: I use the Mask Inference and TRanscription (MI+TR) network from Chapter 3 as a baseline, comparing against its transcription output. I will also look at three variants as before, a residual system, a source processing system, and a vanilla system. The only difference between the network in these experiments and those from the previous experiments is the inputs and outputs of the system. As discussed, this system takes in a mixture and a set of isolated sources and outputs a transcription estimate for each instrument in the mix. As before, I teacher force these systems (i.e., always provide ground truth as input) during training and inference to get an upper bound on the systems performance.

The measure of performance, as used in Chapter 3, is the note onset/offset F1 score. Higher values indicate that the estimated transcription is closer to the ground truth transcription. I use the same threshold values as used in Chapter 3: 0.8 for every instrument except Drums, which are set to 0.1 [179].

As I mention in Section 4.2.2, the problem set up for this system is quite different from situations that a user might encounter in the real world, however this experiment still has some merit in its own right. Namely, this experiment aims to solidify the intuition that it is easier to transcribe an instrument in isolation than if that instrument is in a mixture. This perhaps seems obvious, but I am unaware of anyone experimentally validating this notion. This experiment provides a step towards actualizing the science behind this idea, and is perhaps an important thought to keep in mind as future research into multi-instrument AMT progresses.

For this experiment, I will report F1-score of note onsets and offsets using the `mir_eval` toolbox [222] (see Section 1.3.2), which is the standard way of measuring transcription performance.

4.3.3. Can the input score information be used to alter the output source estimates?

Perhaps the most important question for the goals of this Chapter is this one. Here, I want to know whether the input score data can be used to change the output source estimates of the Deep Score-Informed Separation (DSIS) system. If the answer is in the affirmative, this system will enable edits where a user can make changes to notes in the piano roll and the separation output changes in response. I will examine both quantitative and qualitative methods of answering this question.

4.3.3.1. Quantitative Evaluation. While qualitative examples might give some evidence that the DSIS system is responding to the piano roll input, before I do that I want to provide a more rigorous, quantitative analysis of the system to ensure that any example I might selected is not a fluke. To do this, I have devised an experiment where I randomly eliminate notes in the piano roll input and examine the output separation performance of a trained DSIS model. This experiment has many important details to explain, so I will break this subsection up into a few pieces. First I will describe the process of altering a piano roll input systematically, then I will describe how I measure the scale of these alterations. What I want is to find the relationship between the scale of the alterations and the separation performance of the system. The hope is that when a change happens to reduce the accuracy of the input piano roll, then that change is reflected by a lower separation performance of the system. If altering the input piano roll produces a lower separation performance, that is a strong indicator that the DSIS system is responsive to changes in the input piano roll, which in turn is a crucial step towards the goal of making a source separation that enables editing notes.

Altering the Input Piano Roll To alter the input systematically, I formulate a strategy to randomly interpolate between the ground truth piano roll and the empty piano roll (i.e., a piano roll with no active notes). This interpolation strategy works as follows: given a randomly ordered list of all of the notes in the ground truth piano roll, I divide that list into N equal subsets. The number of subsets is the maximum number of interpolation steps that are possible. Starting with an empty piano roll, for each step I add in the notes for the corresponding subset all at once. This process is shown in Figure 4.4, where the ground truth piano roll is shown greyed out at Step 0, and at each step a certain number of notes are added (shown in pink) as the interpolation proceeds. By the last step, Step 3 in this example, the interpolated piano roll is equivalent to the ground truth piano roll. For a given example, I randomly select a number of interpolation

steps between 1 and the max number of steps, N, to do before inputting the resulting piano roll into the network. Choosing the number of interpolation steps randomly ensures that there is wide coverage over many possible interpolated alterations of the piano roll.

I chose this strategy for two reasons. First, notice that the smallest unit of interpolation is the note, no matter how long or short it is. This is to simulate that in modern audio editing software, it is trivial to make a note longer or shorter in a piano roll. Second, I chose to use a stepped interpolation instead of a randomly selecting a number of notes to add in because I wanted to guarantee that the experiments had adequate coverage of the empty piano roll case (Step 0) and the perfect ground truth case (Step N). For these experiments, I set the number of steps $N = 5$, which includes the empty piano roll case at Step 0 and the perfect ground truth piano roll at Step 5.

The interpolation example shown in Figure 4.4 starts with an *empty* piano roll and notes are slowly added until the interpolated input matches the ground truth. I also conduct a second version of this experiment where, instead of starting with an empty piano roll, I start from an estimated piano roll transcription provided by a multi-instrument Automatic Music Transcription (AMT) system, and run the interpolation process described to move the piano roll towards ground truth. This system is a network with 4 Bidirectional Long Short-Term Memory (BLSTM) layers, that outputs a piano roll for a set of multiple instruments. This network is has the same architecture as the Transcription-only network from the previous chapter; in other words, this network is a Cerberus network without the Deep Clustering or Mask Inference heads. The network used in this chapter is a distinct implementation from the network from last chapter (i.e., uses a different codebase, different hyperparameters, and different testing set), therefore this AMT system is not apples-to-apples comparable to the previous one. Further details about this network are provided in the Appendix, Section 6.2. By taking the output of this BLSTM multi-instrument AMT network and using it as a starting point for the interpolation experiments, I can simulate an end-user using that as a starting point for editing a DSIS separation.

This process of removing notes from a ground truth could easily be viewed in reverse, where notes are *added*, moving the piano roll closer to the ground truth transcription. This reverse view of adding notes simulates how an end-user might actually use this system: they might start from an AMT estimate (or empty piano roll) and refine the piano roll, moving it closer their desired stopping point (here approximated by the ground truth). This process is more similar to how a deployed DSIS system might be used in the wild than a process that might

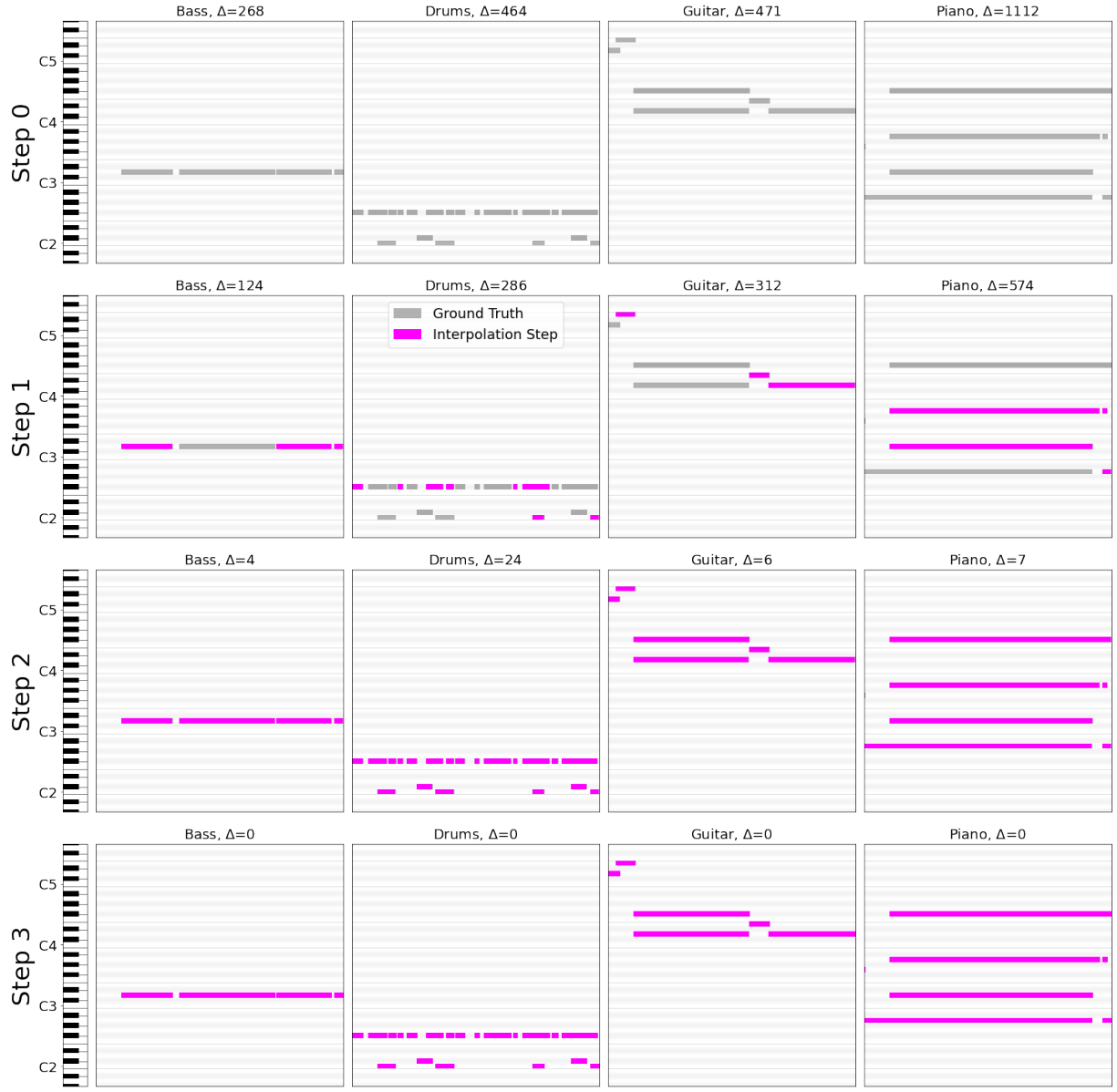


Figure 4.4. An example of the interpolation process for going from a set of empty piano roll inputs (top row, Step 0) to the full ground truth inputs (bottom row, Step 3). The active notes in the ground truth piano roll are shown in grey to illustrate where they are, however grey values are not provided as input to a system. Only the notes in pink are used as input to the system. The Δ's above each piano roll are the (unnormalized) Hamming distance between the ground truth and the interpolated input at that step.

Measuring alterations to the Piano Roll Now that I have established my method for making systematic alterations to a piano roll, I want to measure how different the altered piano roll is from the ground truth. The independent variable in this experiment is the difference between the ground truth and the changed piano roll, as determined by the interpolation described above. I calculate how different the input piano

roll is using the Hamming distance [97] between the ground truth piano roll and the randomly altered one, excluding all time-pitch bins that are not active according to the ground truth piano roll.

Here, as with the rest of this chapter, I assume the input piano roll is binary, and then calculate the Hamming distance by comparing each time-pitch bin of both piano rolls (see Figure 3.2 for an illustration of a piano roll as a matrix). The Hamming distance is defined like,

$$\text{Hamming}(G, A) = \sum_{k \in K, t \in T} |G_{k,t} - A_{k,t}| \tag{4.2}$$

where $G \in \{0,1\}^{K \times T}$ is the binary-valued ground truth piano roll, and $A \in \{0,1\}^{K \times T}$ is the binary-valued piano roll we are altering. Here, K is the number of pitches in the piano roll, T is the number of time steps in the piano roll, and both G and A have the same shape. In this experiment, I will report Hamming distances for each instrument separately, so I omit the instrument dimension of the piano roll in this calculation.

If viewed in reverse, the interpolation process described above starts from the ground truth piano roll and eliminates notes until it gets to an empty piano roll; no new notes are added, notes are only removed. Therefore, the inactive bins of the ground truth will always stay inactive throughout this interpolation process (i.e., the interpolation does not introduce false positives, only false negatives). However, different piano rolls have a wildly variable number of active notes, and therefore wildly different maximum possible Hamming distances (see the different values for Hamming distance in the first row of Figure 4.4). Therefore, I normalize the Hamming distance for each example such that I only consider the active time-pitch bins of the ground truth piano roll, and ignore the inactive bins.The Normalized Hamming distance is thus:

$$\text{Normalized Hamming}(G, A) = \frac{1}{|G_+|}\text{Hamming}(G, A) \tag{4.3}$$

where $|G_+|$ is the normalization factor, which is the number of time-pitch bins in the ground truth piano roll that are active. Now, all distance values are in the interval $[0.0, 1.0]$, where a Normalized Hamming distance of 1.0 means that none of the true notes are input into the system, and a Normalized Hamming distance of 0.0 means that the input to the system is exactly the ground truth piano roll. Ground truth piano rolls with no active notes are set to a Normalized Hamming distance of 0.0. Note that this definition is $1 - \text{Recall}()$, the inverse of the Recall score.

Quantitative Evaluation: Experimental Setup Given this way of randomly altering piano rolls, I ran a trained DSIS network on the test set using these altered piano rolls as input. I ran these networks with

altered input piano rolls that follow the interpolation procedure as described above. I will show the results where the input starts from an empty piano roll and gets interpolated to the ground truth piano roll. I will also show results where input starts from the BLSTM multi-instrument AMT system's piano roll estimate and gets interpolated to to the ground truth piano roll.

To evaluate the performance of these systems, I define a modified SI-SDRi such that I can compare two source estimates. Here, I want to find the change in SI-SDRi between a source estimate produced by a DSIS system using some initial interpolated piano roll step (i.e., at interpolation Step 0) and a source produced by a DSIS system using some other interpolated piano roll step at some Step i. Recall that Step 0 can either be an empty piano roll, or an estimated piano roll transcription from the BLSTM multi-instrument AMT system. Letting m be some mixture, s be the corresponding ground truth isolated source audio, $\hat{s}_0$ be a source estimate from a DSIS system that uses the interpolation Step 0 piano roll as input, and $\hat{s}_i$ is a source estimate from a DSIS system that uses the interpolation Step i piano roll as input, I define the ΔSI-SDRi as

$$\begin{aligned} \Delta\text{SI-SDRi}(\hat{s}_i, \hat{s}_0, s) &:= \text{SI-SDRi}(\hat{s}_i, m, s) - \text{SI-SDRi}(\hat{s}_0, m, s) \\ &= (\text{SI-SDR}(\hat{s}_i, s) - \text{SI-SDR}(m, s)) - (\text{SI-SDR}(\hat{s}_0, s) - \text{SI-SDR}(m, s)) \\ &= \text{SI-SDR}(\hat{s}_i, s) - \text{SI-SDR}(\hat{s}_0, s), \end{aligned} \tag{4.4}$$

where I used the definition of SI-SDRi from Equation 1.5 in the Introduction, to cancel out the SI-SDR terms that use the mix as the source input (i.e., SI-SDR(m, s)). This measure then simplifies to the difference in SI-SDR between $\hat{s}_0$ and $\hat{s}_i$. Despite being conceptually the same as SI-SDRi–although with different source inputs–I refer to this measure as ΔSI-SDRi to avoid confusion; SI-SDRi is an established term of art that implies that the mixture is used as the initial source estimate–which is not what I do here. The reader should apply the same caution to ΔSI-SDRi which they apply to SI-SDRi; SI-SDRi and ΔSI-SDRi are variants of SDR, which itself has a sketchy relationship to human perception [20, 23, 24, 77].

I will report ΔSI-SDRi as a function of the Normalized Hamming distance of the input piano rolls. If these DSIS systems respond well to changes in their input piano rolls, we would hope to see that as the piano rolls get closer to ground truth (i.e., have a Normalized Hamming distance of 0.0), the separation results would get better, and vice versa. This would be evidence that a change in the piano roll input produces a change in the source separation output. Not only that, but by systematically changing the input piano roll from empty to ground truth (or from an AMT estimate to ground truth), we can glean *how* the separation estimates change. For instance, if making the input piano roll better (i.e., closer to ground truth) produces

better source estimates, then this tells us that a) not only does the system respond to changes in the input piano roll, but b) it responds in such a way that would intuitive sense to an end-user. If the system produced source estimates that were *worse* when given a *better* input piano roll, then this system would be unhelpful and frustrating to use. What is most desirable is a system that produces *better* source estimates when given *better* piano rolls. In other words, as Normalized Hamming distance goes to 0.0, ΔSI-SDRi increases. This experiment is constructed to elucidate the relationship between changing the input piano roll and the quality of a DSIS system's resulting source estimates.

Using the experimental procedure outlined in this section, I will investigate the responsiveness of the proposed Score-Informed system using the trained vanilla DSIS and residual DSIS models. This experiment will provide a bird's eye view of how these Score-Informed models respond to the input piano roll.

4.3.3.2. Qualitative Examples. Compared to the quantitative evaluation just described, the qualitative method is simple: given an example mixture and corresponding piano roll transcription, if changes to the input piano roll result in respective changes in the source estimates, then that is evidence that the DSIS system supports this kind of edit. I will show a few examples of altering the input piano roll in this way and resulting source estimates. In this case, I will use the trained DSIS residual network from the experiments in Section 4.3.2 and even though that network is trained to separation four instruments total, here I will focus on the Piano source and the Guitar specifically. I choose to focus on these two instruments in particular, because as discussed in the results section of the Cerberus chapter (Chapter 3, Section 3.4.3) these two instruments are easy for a source separation system to mistake for each other. Therefore, having a note-level granularity of control is most useful because the estimates for one of these sources might have audio from the other source, and an end-user can easily move notes between these sources to correct these issues.

With this set of both qualitative and quantitative experiments, I hope to paint a picture of how well this network is able to change its output based on the input piano roll.

4.3.4. Dataset Preparation

The dataset requirements for training and evaluating are similar to the requirements of the last chapter. As with any source separation system, we need mixtures with paired source data, however to do score-informed separation, we also need aligned score data per source. Because of these conditions, again I turn to the Slakh dataset [184], which checks all the boxes; it contains mixtures with paired source data and aligned score data for each source. For more information on Slakh, see Section 1.2.2.

In this chapter I processed the Slakh data in a similar manner as before, albeit with some slight differences. As before, I downsampled the audio to 16 kHz. I chose relevant segments from the audio in a similar manner as before, I chose every 5 second segment where the mixture contained all desired sources (e.g., Piano, Guitar, Bass, Drums). There was a 2.5 second hop between adjacent segments. Here, instead of enforcing that all sources be active, as I did in Chapter 3, I only required that 2 of the 4 sources were active. I declared that a source was 'active' if it had at least 5 note onsets with MIDI velocity above 30. If a segment did not have enough active sources, it was excluded from the dataset. If a mixture had two (or more) instances of a single source (e.g., [Piano1, Piano2, Guitar1, Bass1, Drums1]), then this example was split into multiple training examples (one for [Piano1, Guitar1, Bass1, Drums1] and another for [Piano2, Guitar1, Bass1, Drums1]). Using this strategy, the full training set contained roughly 206,000 examples, the validation set had roughly 42,000 examples, and the test set had roughly 24,000 examples. The source audio for the desired sources was combined to make a mixture of the selected sources. STFTs with 1024-point window size and 256 sample hop were calculated from mixture segments and used as inputs to the network. The musical score was the accompanying MIDI data binarized with a velocity threshold of 30 (i.e., any note with a velocity value over 30 is considered "on", else it is considered "off").

4.3.5. Model Configuration Details

As I mentioned above, the Main Processing Module for all variants of the Deep Score-Informed Separation (DSIS) networks was a stack of Bidirectional Long Short-Term Memory (BLSTM) layers [94, 118] (see Appendix Section 6.1 for more details about BLSTMs). This module ingested a mixture spectrogram and output a mask, which is applied to the input mix spectrogram to make a source estimate. The first layer in this module is a batch normalization [123] layer, followed by 4 BLSTM layers with 600 units in each layer. The output of the last BLSTM layer is given to a fully connected layer with a leaky ReLU nonlinearity. The output of this final fully connected layer is the mask.

Only two variants used the Piano Roll Processing Module: the Piano Roll Processing DSIS network and the Residual DSIS network. For these two configurations, the Piano Roll Processing module consisted of a smaller stack of BLSTM layers. The piano rolls were fed into a batch normalization layer followed by 2 BLSTMs with 600 units each. That output was processed through a fully connected layer with a leaky ReLU nonlinearity and this final layer produced an embedding with 512 dimensions. This embedding was

Model	Piano Roll Processing Module?	Residual?	Piano	Guitar	Bass	Drums
MI+TR	–	–	14.2	14.6	11.2	14.9
Score-Informed			13.6	**17.2**	11.6	14.9
	✓		14.9	16.3	**11.8**	15.0
	✓	✓	**15.1**	16.2	**11.8**	**15.0**

Table 4.1. Separation performance in terms of mean SI-SDRi (dB) over the test set, comparing the baseline MI+TR network (described in Chapter 3) to three Score-Informed variants. Higher values are better, bolded values indicate the highest score for that instrument. A check mark means that the Module/Residual layer was used, whereas "–" indicates not applicable. The Score-Informed networks are given the ground truth piano rolls as input in this test. The Score-Informed networks outperform the baseline for all four sources, with the version that has some additional piano roll processing layers with a residual connection performing best overall. This indicates that additional score input data is useful when doing source separation.

concatenated to the mixture input along the frequency dimension. This concatenated feature was input into the Main Processing Module, which proceeded as described above.

I trained these networks for 100,000 iterations each, using a batch size of 64 and using the Adam optimizer [142] with a learning rate of 3e-4. The learning rate decayed by a factor of 0.98 after every 10,000 training steps.

4.4. Results

In this section I will provide an overview and analysis of the results of the experiments described in the previous section.

4.4.1. How does additional score input information affect output separation performance?

This question asks whether a Deep Score-Informed Separation (DSIS) network can use its additional score input data productively, to improve its separation estimates. These results are shown in Table 4.1, where I use the Mask Inference and TRanscription (MI+TR) network described in Chapter 3 as a baseline and compare it against three different DSIS systems.

There are no instances where the baseline MI+TR system achieves the highest SI-SDRi score, across all four sources that I test. In fact, the MI+TR only beats one of the DSIS networks in one source: the Piano source in the top row. Other than that, the DSIS networks (bottom three rows) outperform the baseline across all sources. In the case of the Guitar source, the best DSIS system beats the baseline by 2.6 dB, which

Model	Source Processing?	Residual	Piano	Guitar	Bass	Drums
MI+TR	–	–	0.30	0.25	0.55	0.09
Source-Informed			0.47	0.41	0.65	0.18
	✓		**0.49**	**0.44**	**0.69**	**0.21**
	✓	✓	0.47	0.43	0.66	0.19

Table 4.2. Transcription performance in terms of mean note onset/offset F1 score over the test set, comparing the baseline MI+TR network (described in Chapter 3) to three Source-Informed Transcription networks. Higher values are better, bolded values indicate the highest score for that instrument. "3" means that the Module/Residual layer was used, whereas "–" indicates not applicable. The Source-Informed Transcription networks outperform the baseline in every single case across all four sources. The best version had additional source processing layers without a residual connection. This suggests that separating instruments is easier if the source data is available in addition to the mixture data.

is a sizeable improvement. Overall, the Residual DSIS variant (bottom row) achieved the best separation performance in three of the four sources.

These results point to the fact that, yes, these DSIS networks are able to use the piano roll inputs to increase separation performance. Just as before with Cerberus (and MI+TR), these networks seem to be able to learn the relationship between the audio source data and the corresponding source data. Some of the differences between these results are minute (e.g., the difference between MI+TR and DSIS performance is less than 1.0 dB in both the Bass and Drums sources). However, an important thing to note is that the DSIS networks use a piano roll score as input. This indicates that DSIS networks might be able to used for the ultimate goal of this chapter: the ability to change a separation output by editing note data. A further exploration of this goal is upcoming in Section 4.4.3, however before we examine those results, I will first turn to the results for Source-Informed Transcription.

4.4.2. How does additional source input data affect the output transcription performance?

As discussed, the Deep Source-Informed Transcription (DSIT) system is analogous to the Deep Score-Informed Separation (DSIS) system. The DSIT system is a multi-instrument transcription system that transcribes from a mixture but is also conditioned on isolated source data. As discussed, this system does not have a lot of practical value, however these results might provide interesting insight into the relationship between single-instrument transcription and multi-instrument transcription systems.

The results for this experiment are shown in Table 4.2. The story here is clear cut: the Source-Informed systems comprehensively outperform the Mask Inference and TRanscription (MI+TR) baseline system. In

most cases, the F1 score is nearly 0.20 points higher, which is a large improvement. This is perhaps not a surprise given the difference in problem setups between the MI+TR system and the DSIT systems; the MI+TR only sees the mix as input, whereas DSIT sees the mixture and isolated source data. These results may seem obvious but they point to the difference in the difficulty between these two problem setups. I included these results for completeness, but further exploration of this topic would be a terrific subject for future work.

4.4.3. Can the input score information be used to alter the output source estimates?

This final set of results is perhaps the most important towards verifying the goal of this chapter. The goal of this chapter is to create a source separation system that is able to change its separation output based on note data given as input. For practical purposes, it is important that this interaction happen at inference time, and require no additional training or optimization steps. Thus, for this experiment I use the trained Deep Score-Informed Separation (DSIS) networks described in Section 4.3.1 and whose results are shown in Section 4.4.1. Let me first analyze the results of the quantitative evaluation before showcasing some qualitative examples.

4.4.4. Quantitative Results

Here I discuss the results from the experiments outlined in Section 4.3.3. The results from measuring the mean ΔSI-SDRi of a residual Deep Score-Informed Separation (DSIS) network as a function of the Normalized Hamming distance of the input piano roll are shown in Figure 4.5. The top row shows the results from doing the interpolation of the input piano roll starting with an empty piano roll, and the bottom row shows the results where there interpolation started from a transcription estimate.

Starting with the empty piano roll results (top row of Figure 4.5), most sources see a large increase in separation performance as the Normalized Hamming distance gets closer to 0.0 (where indicates the ground truth is used as input to the system). Bass, Guitar, and Piano show a ΔSI-SDRi of between 15 and 20 dB as the Normalized Hamming distance moves to 0.0! Even as the Normalized Hamming distance gets closer to 0.90, the ΔSI-SDRi still shows an improvement of around 7-10 dB, which is considerable. Drums shows a steady ΔSI-SDRi improvement of 7-10 dB for nearly all but the largest Normalized Hamming distances. This is the first indication that the output this system is very influenced by the input piano roll, which is a positive result towards the goal of making a system whose output can be changed at inference time.

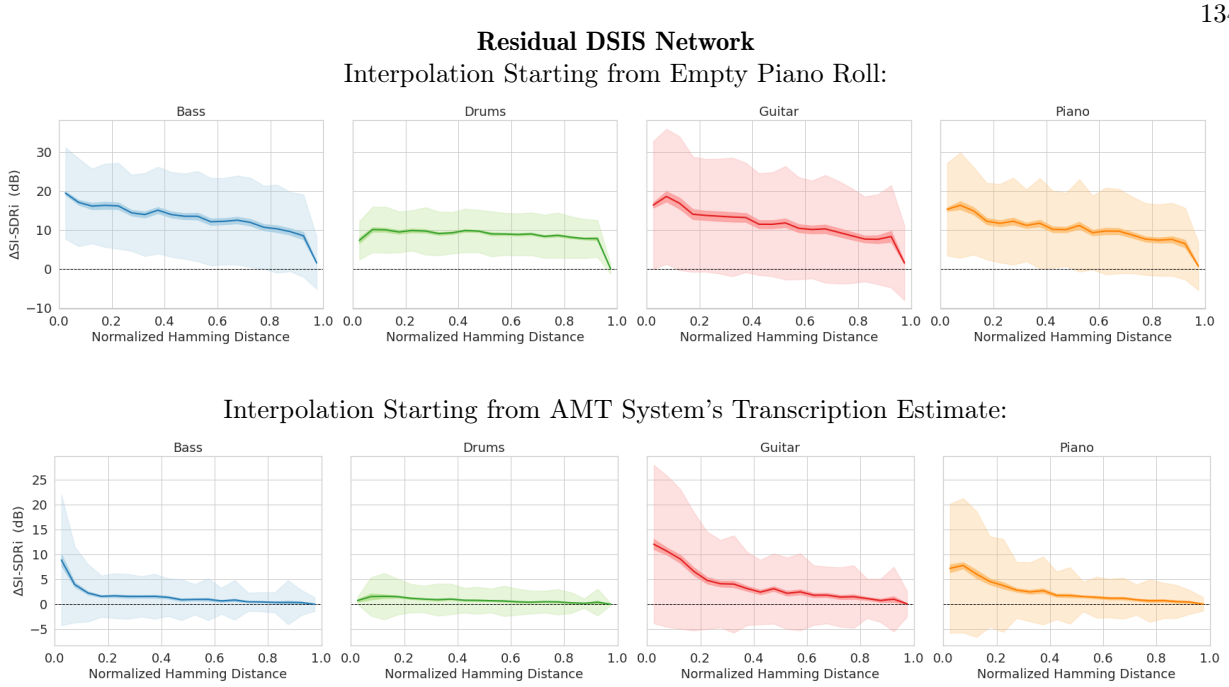


Figure 4.5. Mean ΔSI-SDRi (dB) of a residual Deep Score-Informed Separation (DSIS) network as a function of the Normalized Hamming distance of the input piano roll. A Normalized Hamming distance of 0.0 means the ground truth piano roll is used as input and higher ΔSI-SDRi values mean better separation performance. The top row shows the results from doing the interpolation of the input piano roll starting with an empty piano roll, and the bottom row shows the results where there interpolation started from a transcription estimate. For each line, the outer band represents on standard deviation and the inner band represents the 95% confidence interval of the mean.

There is also another important aspect to discuss that is a little more subtle, which is the fact that the ΔSI-SDRi is still quite elevated at very high values for Normalized Hamming distances (ΔSI-SDRi stays around 10 dB until about a 0.90 Normalized Hamming distance). At those high values for Normalized Hamming distance, the input piano rolls contain very few of the ground truth notes. This means that even if an end user is only able to provide minimal note annotations as input, the separation performance could vastly increase.

The second row in Figure 4.5 shows the results when interpolating between the output of an AMT system and the ground truth. Here, the AMT system makes its initial estimate, so there is less ΔSI-SDRi improvement that is possible. Still, Bass, Guitar, and Piano see an increase in ΔSI-SDRi of over 5 dB, with a lot of that gain happening below a Normalized Hamming distance below 0.2. Unlike before, Drums do not see much of a performance gain at all, perhaps due to the fact that Drums are the only percussive source, which makes them easier to separate (perhaps supported by Table 3.3 in Chapter 3, where Drum separation

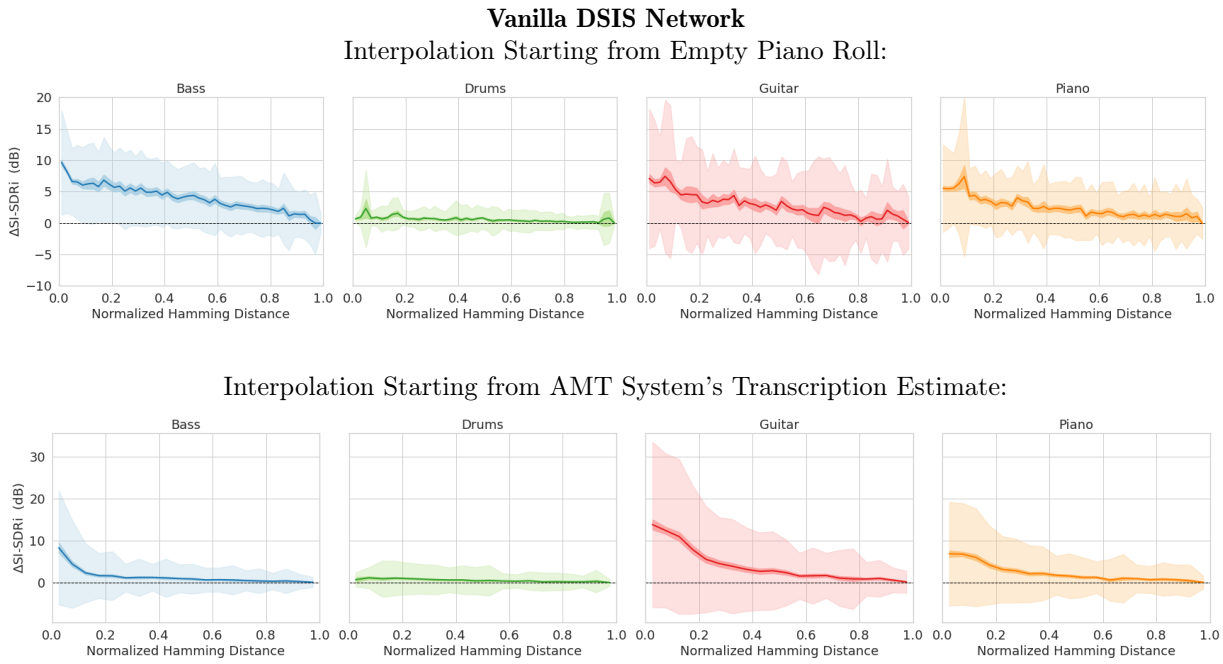


Figure 4.6. Mean ΔSI-SDRi (dB) of a vanilla Deep Score-Informed Separation (DSIS) network as a function of the Normalized Hamming distance of the input piano roll. A Normalized Hamming distance of 0.0 means the ground truth piano roll is used as input and higher ΔSI-SDRi values mean better separation performance. The top row shows the results from doing the interpolation of the input piano roll starting with an empty piano roll, and the bottom row shows the results where there interpolation started from a transcription estimate. For each line, the outer band represents on standard deviation and the inner band represents the 95% confidence interval of the mean.

performance is significantly higher than other sources). This makes incremental changes to the input piano roll for Drums difficult to provide additional separation improvement.

A corresponding set of results using the vanilla DSIS network are shown in Figure 4.6. Here we see vaguely similar trends, even if they are a little more noisy and muted than the residual DSIS network. Interpolating from an empty piano roll (top row) shows an improvement of approximately 5-10 dB in ΔSI-SDRi at a Normalized Hamming distance of 0.0 for the Bass, Guitar, and Piano sources, although, as the interpolations move toward 1.0, they are much noisier and they decrease much quicker than the residual DSIS network. When the interpolation starts from the AMT system (bottom row), the Bass, Guitar, and Piano sources start at a roughly 6-13 dB ΔSI-SDRi but show a sharper decline at a Normalized Hamming distance of 0.2. Again, in both cases, the Drums source is barely affected by changing the piano roll inputs.

To get another sense of how the separation performance of the residual DSIS system changes as the input piano roll interpolation proceeds, Figure 4.7 shows the a single example of the interpolated steps for the Piano input and the resulting SI-SDRi (which can be converted to ΔSI-SDRi by subtracting the SI-SDRi

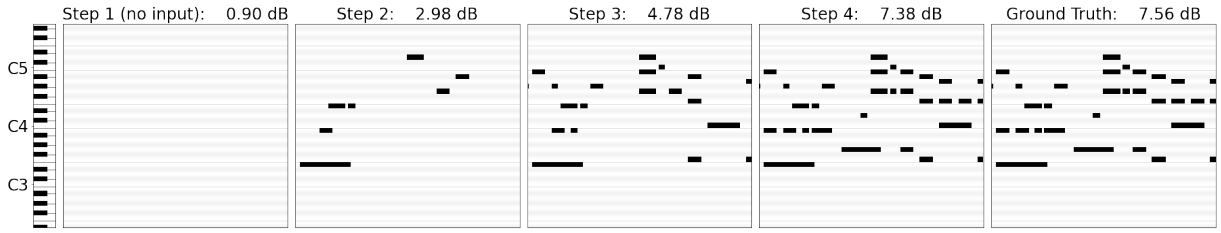


Figure 4.7. On example of the interpolated input piano rolls for the Piano source over 5 steps using the residual DSIS network. From left to right, the input starts completely empty, and then notes are incrementally added until the last step which uses the full ground truth piano roll (labeled "Ground Truth") all the way on the right. The SI-SDRi is shown at the top of each plot; to get the ΔSI-SDRi, subtract the SI-SDRi value at Step 0 (0.90 dB) from all values. As the input piano roll gets closer to the ground truth, the SI-SDRi raises by over 6.5 dB for this example! This suggests that the piano roll input plays a large part in how the network determines its output. Notice, too, that the SI-SDRi still improves between Step 4 and the Ground Truth, despite the input piano roll for the Piano source not changing. This interpolation is happening across all four input sources, so while the Piano source is unchanging between Step 4 and Ground truth, another source's input piano roll is still changing (but is not shown here). This a further indication that the piano roll inputs influence the separation output–the network has refined its Piano source estimate based on piano roll input from a completely different source, perhaps using that other information to refine its Piano source estimate.

values from the value at Step 0, which is 0.90 dB). As the input piano roll gets closer to the ground truth, the ΔSI-SDRi is over 6.5 dB for this example. This is further evidence that the piano roll input is a significant factor in the network's output. Something interesting also happens: the SI-SDRi still improves between Step 4 (the penultimate step) and the last step (labeled Ground Truth) even though the input piano roll for the Piano source is unchanging. The trick here is that the interpolation happens across all four input sources, even those not shown in the figure. So, while the Piano piano roll is static between Step 4 and Ground truth, another source's input piano roll is still changing (but is not shown here). This a neat example of how the piano roll inputs influence the separation output. In this case, the network refined its Piano source estimate based on piano roll input from a completely different source.

Overall, it seems like having the extra layers dedicated to processing the piano roll, like the residual DSIS network has, plays a major part in making a system that is able to respond to changes in the piano roll input. Even though the all-around performance of both systems in Table 4.1 from Section 4.4.1 show that both systems do well when provided ground truth as input, the results from this section indicate that the residual DSIS system is much better suited towards the goals of making an interactive separation system.

4.4.5. Qualitative Results

Now that it has been established that Deep Score-Informed Separation (DSIS) networks can respond to their input piano rolls from a high-level statistical perspective, I want to concretize these results by showing how the source estimates change when given different input piano rolls. Here, I will illustrate this using one example mixture and examining how the separation output changes in response to changes in the input piano roll. I use the residual DSIS network discussed in the previous sections. This example is only slightly cherry-picked; I chose this example because the changes to the output audio separations were easy to visualize in a spectrogram, however when using the system there was no shortage of examples I could have selected from, but were harder to visualize. Additionally I only focus on the Guitar and Piano sources, because, as discussed in the last chapter (Section 3.4.3), these sources can often play the same role in an ensemble and thus are easy for a separation system to mix up. Without further adieu, let me walk through these results.

In Figure 4.8, I show the output of the residual DSIS network when the ground truth piano roll is input into the system. The top row shows the piano roll that is used as input for both instruments. Here, I color the Guitar's notes in green and the Piano's notes in yellow. The bottom row shows spectrograms of the DSIS network's source estimates for the Guitar and Piano sources, respectively.

Before I show you the next set of results, I want to pause here and expound on the content of each source in this figure because understanding it will be important to comprehending the results shown in the next two figures. Notice the Guitar has two notes (G3 and G4) that begin prior to this segment (before 0.0 seconds in the plot) and then at roughly 2.5 seconds three are quick ascending notes (B♭3, C4, E♭4) before the same two notes as before start again (G3 and G4). Looking at the Piano, after 2.5 seconds it alternates between two notes (B♭5 and B♭6) for the rest of the audio segment.

Notice the relationship between how these notes are represented in the piano rolls and in the source estimates. I find it is helpful to look at the timing of events to line them up. One more thing that is important to notice is that this figure is a great illustration of why we would want to build a separation system that enables an end user to control the output by changing the piano roll. I will take this opportunity to reiterate that it is much easier to pick out the relevant notes in the piano rolls (top row) than in the source estimate spectrograms (bottom row).

I want to show a note-level change to the input piano rolls. Here, I move individual notes from the Guitar's piano roll to the Piano's piano roll. The results are shown in Figure 4.9, where the top row shows

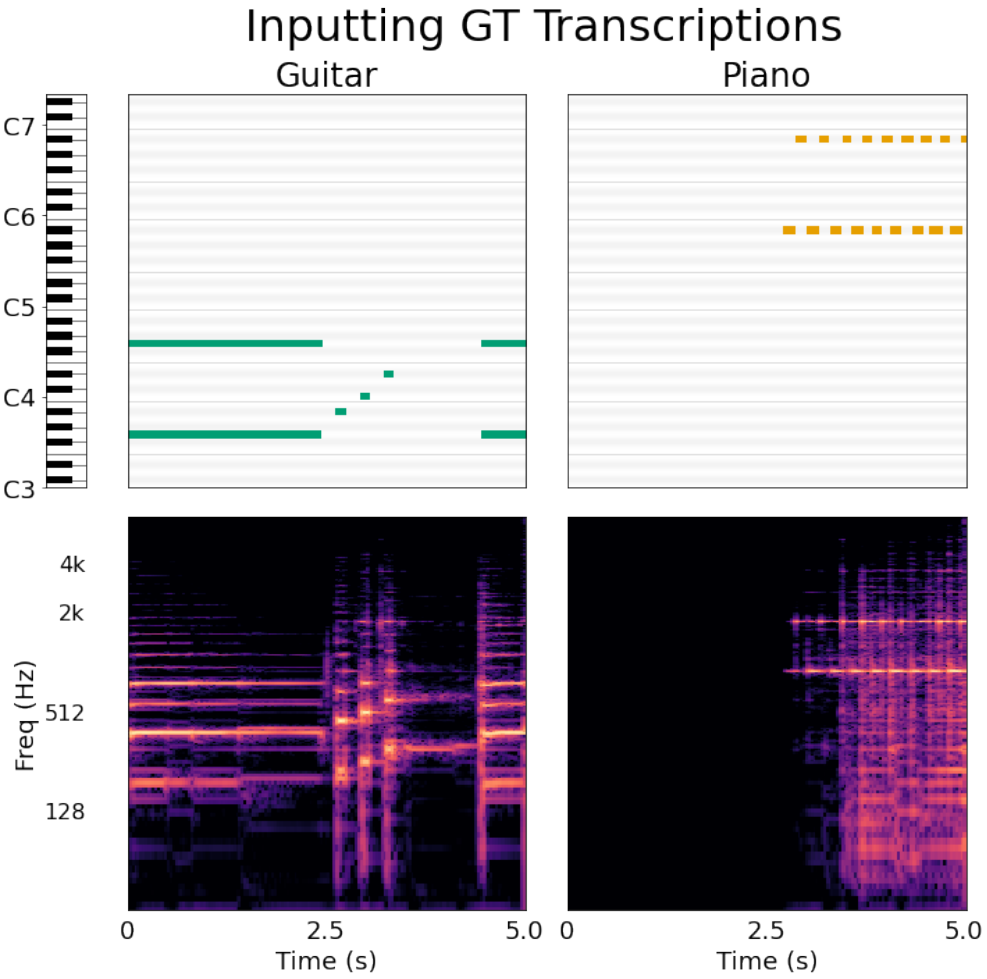


Figure 4.8. Example source estimates (bottom row) when the ground truth piano rolls (top row) are input into a trained residual DSIS model.

how I grabbed the three notes in the middle of the Guitar piano roll (B♭3, C4, E♭4 at roughly 2.5 seconds) and moved them to the Piano piano roll (circled in the top row). The bottom row shows the DSIS network's output, where the audio from those three notes are output in the Piano source, and not the Guitar source. In this case, I have moved individual notes from one piano roll to the other and the DSIS separation system moved the audio corresponding to those same notes in its output. This is direct evidence that this system accomplishes the goals set out for this chapter: this system enables a user to change the notes of the input piano roll and have them reflected in the source separation output.

The main goal of this chapter is to develop a source separation system that is able to alter its source estimates based on note data as input. The results of the quantitative experiments and these qualitative

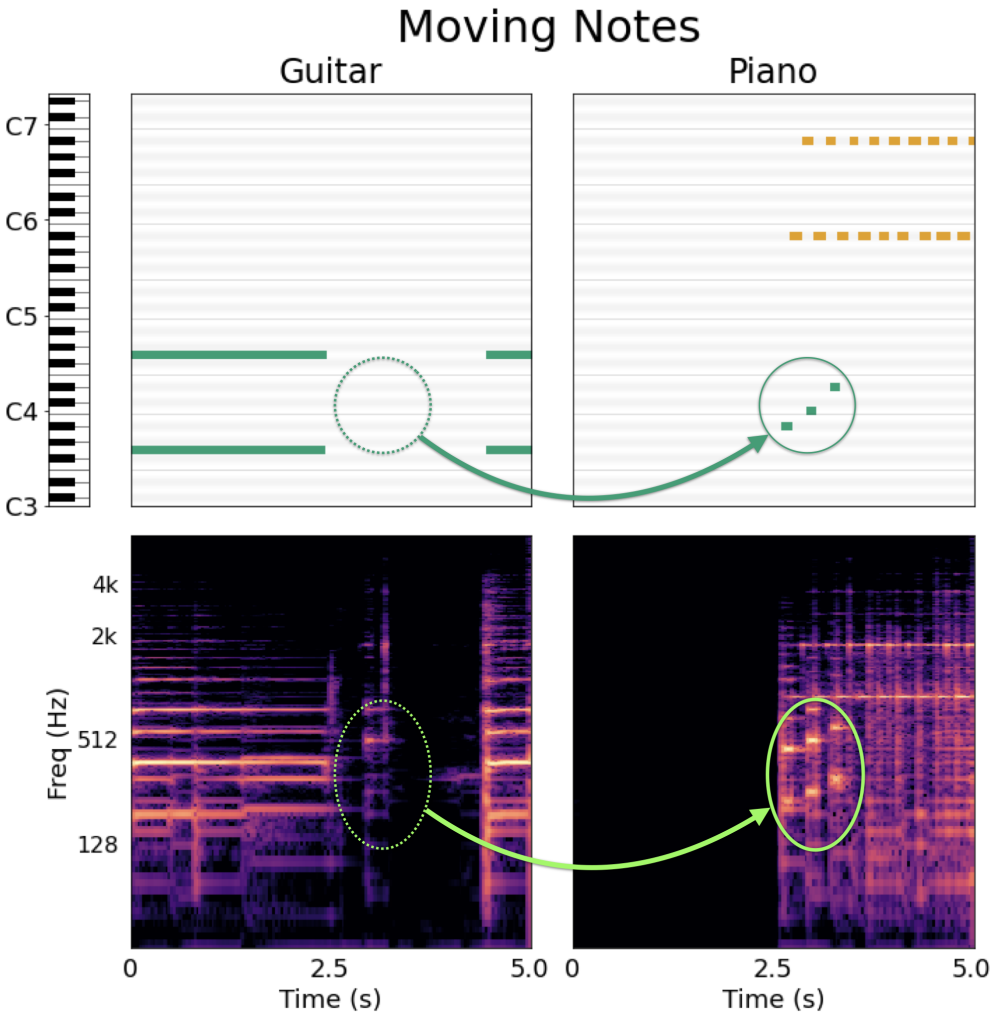


Figure 4.9. The same mixture used in Figure 4.8, where I move three notes from the Guitar piano roll to the Piano piano roll (top row, circled). In its source estimates, the DSIS model puts the audio corresponding to these three notes into the Piano source estimate (bottom row, circled). This shows that the DSIS model can move the audio of individual notes.

examples are all evidence that the strategies for Deep Score-Informed Separation to achieve the goals set out of the beginning of this chapter. The quantitative experiments in Section 4.4.4 show that, over a whole test dataset, the resulting models respond to changes in the input piano rolls, and the qualitative examples explored in Section 4.4.5 enable us to verify on a single example that the model does express this desired behavior. In the next section, I will explore how a user interface built on this system might work.

4.5. Envisioned Interactions with Score-Informed Separation Tools

In this final section, I mock up a user interface to investigate how a Score-Informed Separation system could be baked into an interactive system. Score-Informed Separation systems, like the Deep Score-Informed

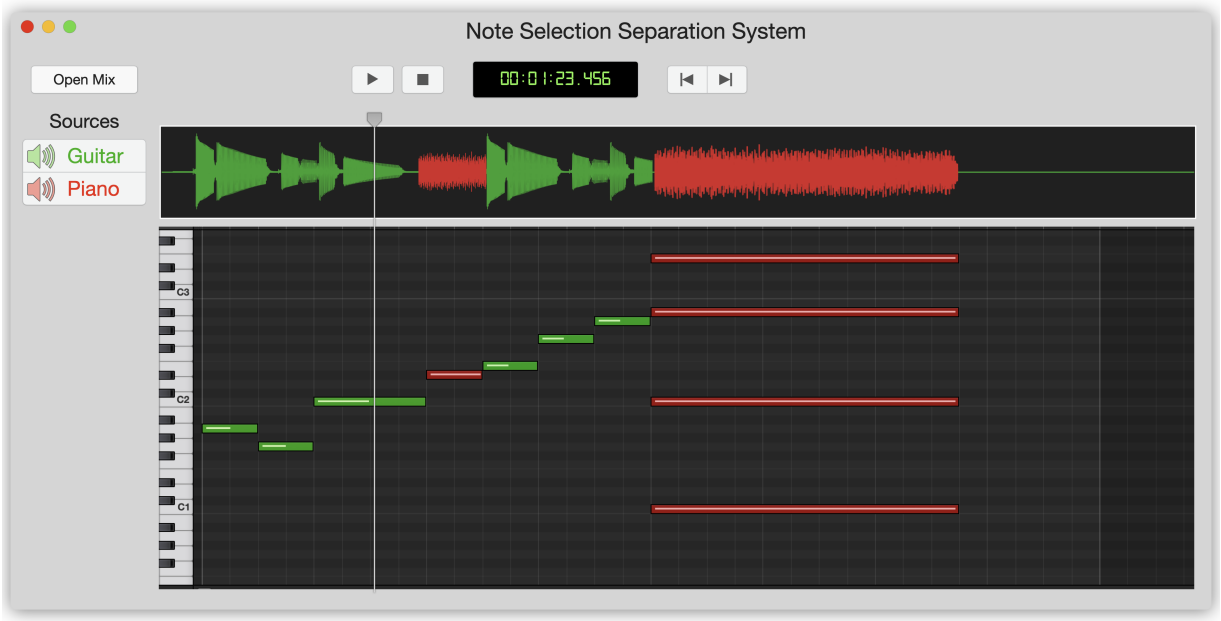


Figure 4.10. My envisioned Score-Informed Separation interface. After a user opens a mixture, the interface populates using DSIS and multi-instrument AMT systems running in the background. Here, Guitar data is in green and Piano data is in red.

Separation (DSIS) networks I presented, enable an end-user to make edits to score data to modify the separated output. So far, I have been using the piano roll to represent score data, and this is on purpose: most potential users of a system like this will be familiar with the piano roll because of its widespread use in Digital Audio Workstations.

My proposed interface is shown in Figure 4.10. Here I only consider a system that separates Piano and Guitar sources, as an illustration of the concept. Once a user imports a mixture audio file, a DSIS system and multi-instrument Automatic Music Transcription (AMT) are both run on the mixture and the interface is populated with the separated sources and estimated transcription data. In this example, the Guitar data is in green and the Piano data is in Green. Below the playback controls, the source data is shown as overlaid waveforms with interspersed red and green, representing each of the sources respectively. To the left of that is a set of toggle switches to mute each of the sources, as shown in Figure 4.11, where the Piano data is muted. Finally, below the waveform is the piano roll. This is where a user will do the majority of their edits. Because this system enables a user to make note-level edits to the source output, like switch which source the audio of a particular note goes in, this interface elevates that interaction; here just clicking a note swaps

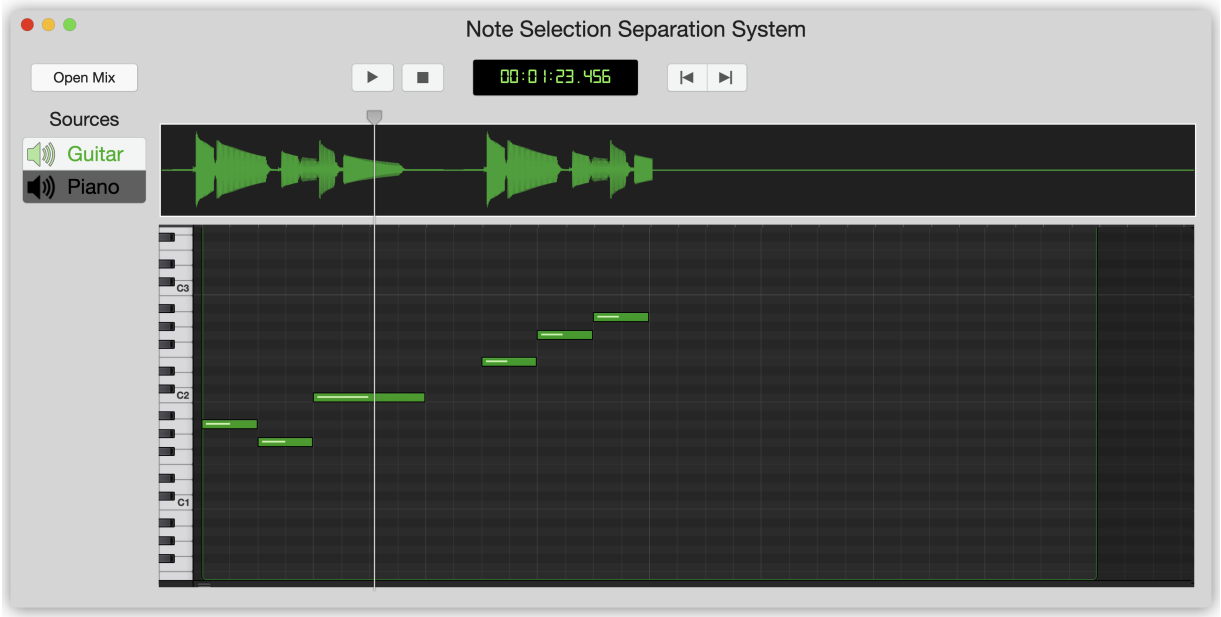


Figure 4.11. A user can mute one source to hear the estimate for a single source. Here, a user mutes the Piano source and realizes that there is one Guitar note missing. It turns out that it was accidentally put into the Piano source.

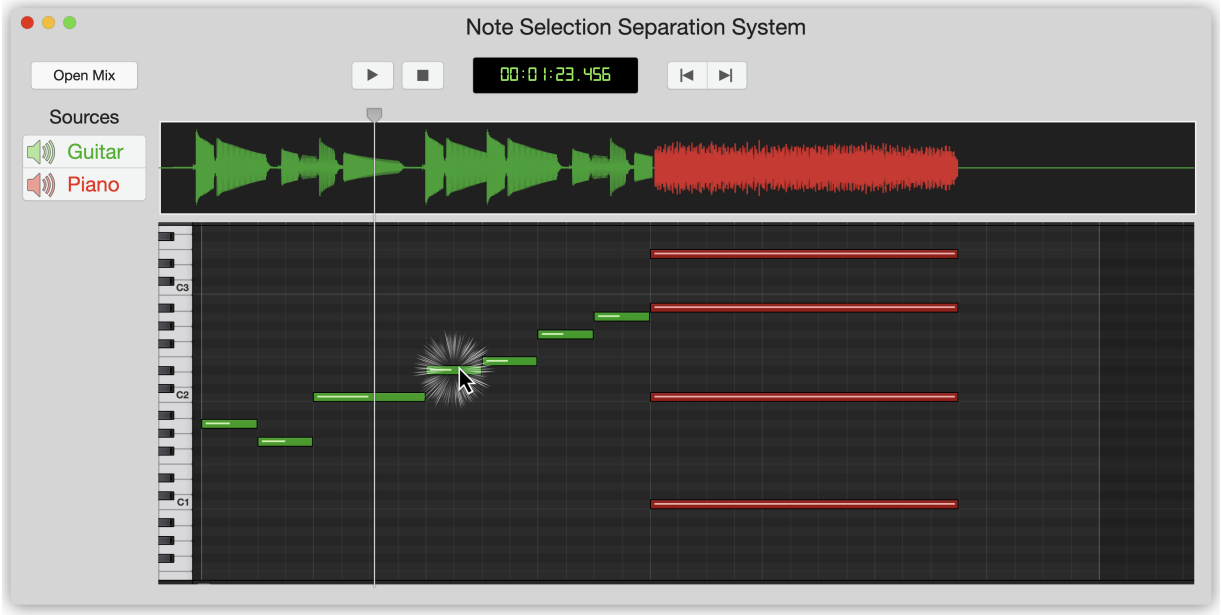


Figure 4.12. The user can click a note to swap which source it is allocated to. In this example, the user corrects the Guitar note that the system initially misallocated to the Piano source in Figure 4.11.

an errant Piano note to a correct Guitar note, as shown in Figure 4.12. As shown, the audio for that missing Guitar note is restored in the waveform.

These figures are just an sketch of what an interface for a Deep Score-Informed Separation system might look like. Refinements to these initial visions should be driven by following iterative user-centered design principles, focusing on user needs and eliminating pain points. The consequential user-studies should focus on two aspects of this system, specifically, making the interface itself more intuitive and ensuring that the backend DSIS and multi-instrument AMT systems work as intended. Future work can also solicit feedback from users about what other types of edits this system should handle, even if their desired edits fall outside the scope of a DSIS system.

4.6. Conclusion

In this chapter, I set out with the goal of making a source separation system that enables modifying individual notes in the system's source estimates. Towards this goal, I turned to Score-Informed separation, a technology that was initial developed nearly a decade ago. Score-Informed separation systems are designed to use a musical score to aid in separation. Here, I revitalize this idea and modify a neural network-based separation system to have additional input conditioning from a musical score. I call the resulting systems Deep Score-Informed Separation (DSIS) systems. I show that DSIS models are able to achieve better separation performance than a baseline system that does not use the score as input. These initial DSIS results are an important hint about whether or not the system is able to respond to the input score. The most pertinent results toward achieving the goals in this chapter are final set of results, where I explore how a trained DSIS model is able to alter its source estimate outputs based on modifications to the input score data. I show that the DSIS models are highly sensitive to changes in the input score data, showcasing examples where I was able successfully to move the audio of individual notes from one source estimate to another by changing the score data. Finally, I showed how I envision how a user would be able to interact with an interactive source separation system built on a DSIS model.

CHAPTER 5

Conclusion

In this dissertation, I have focused on expanding the capabilities of Musical Scene Analysis systems. Through the lens of source separation and Automatic Music Transcription (AMT), I sought to extend the machine learning systems that will power a new generation of audio workstations in an effort to aid musicians in their creative processes. Towards this goal, I set out to make source separation and AMT systems using Score-Informed and hierarchical techniques.

In Chapter 2, I presented a framework for Hierarchical Source Separation that assumes that audio sources in a scene are related according to some hierarchy. Specifically, I looked at musical scenes because musical instruments have a long history of being classified hierarchically [287]. Given existing precedent that sounds can be organized hierarchically, I was able to formulate the structure of auditory hierarchies such that child sources are mixed together to make parent sources, building a tree of submixtures by combining leaf sources all the way up the the root, which is the full mixture. With this formulation in hand, I defined four ways of designing hierarchical source separation systems: **Single-instrument Single-level**, **Single-instrument Multi-level**, **Multi-instrument Single-level**, and **Multi-instrument Multi-level**. By modifying a standard source separation network, I instantiated each of these four systems and experimentally verified them. I found that the Multi-level systems always outperformed the Single-level systems, especially when separating leaf sources. Leveraging the hierarchical structure of a musical scene provides a lot of affordances to the separation system as it learns: it can use the general concept of a broader source type (e.g., "stringed instruments") to help it when separating more specific source types (e.g., guitar or violin). Because of this, I showed that the **Multi-instrument Multi-level** system (a multi-level Query-by-Example system) was able to better separate leaf sources in 16 of the 18 leaf node sources I test. Finally, I presented a mockup of a potential interactive source separation system for end-users, where the proposed QBE system was used under the hood such that a user could choose between many different sources to separate without retraining a model from scratch.

In Chapter 3, I turned my attention toward uniting the tasks of source separation and Automatic Music Transcription (AMT) in one system. In doing so, I described Cerberus, which is deep learning system to simultaneously separate and transcribe multiple percussive and harmonic instruments in a mixture. Cerberus

extends the worldview behind source separation–namely, that a musical scene has multiple overlapping events belonging to different instruments–to the task of AMT, which solves a longstanding issue with prior work in AMT, specifically, that the majority of previous systems could only transcribe the audio of a single instrument at a time. By combining multi-instrument AMT with two types of separation outputs (Deep Clustering [114] and Mask Inference), I showed that Cerberus networks achieve better for both tasks according to standard metrics, with the highest performance coming when using all three output heads. Importantly, doing both separation and transcription is useful because it provides an end-user with different types of analysis of a musical scene; a low-level, high-detail analysis describing the source audio (i.e., the source estimate) and high-level, sparse-detail analysis describing the note events (i.e., the transcription estimate). Both types of analysis are important for different use cases, so giving a user both enables a more complete understanding of the content of a musical scene. Having multi-instrument AMT systems is also an important stepping stone towards making Score-Informed separation systems with modern techniques, as I did in Chapter 4.

On that note, the final chapter of this dissertation had the goal of making a separation system whose output is modifiable by making edits to note data. To that end, I described a system which I called a Deep Score-Informed Separation (DSIS) system, which is a deep learning system that ingests a piano roll score as an auxiliary conditioning input when doing source separation. This system is a fresh take on Score-Informed Separation systems that were built on pre-deep learning technologies (e.g., Non-negative Matrix Factorization [75, 156, 252, 285]) which were less performant than modern neural network techniques but did enable modifying the system's output at a more granular level than existing deep learning systems. As an upper bound on performance, I show that Deep Score-Informed Separation (DSIS) systems using ground truth transcription data as input are able to outperform a baseline system that does not use transcription data. This initial set of results is an important step towards showing that the DSIS systems are able to respond to the input piano roll data. The most significant results towards the goal of editing note data in the source estimates come from the final set of experiments wherein I explored how changing the input piano roll affected the output separation performance. I found that as more correct notes are added, the separation results produce better outcomes according to standard measures of performance. This was a very strong indicator that the output separation estimates are able to be modified by making changes to the input piano roll. As a final qualitative result, I showed how I was able to move a few notes from one source to the other in the input piano roll, and the separation output for both of those sources changed in kind. Confident that the DSIS system had reached the goals set out for the chapter, I turned towards envisioning what an

end-user system might look like with the DSIS system as a backend separation system. In this mockup, I imagined how an end-user could use a piano roll to make changes to the separation estimates.

The three main projects discussed in this dissertation all make contributions to the field of Musical Scene Analysis, with special attention paid to making systems that (a) are able to be controlled at inference time without retraining (Chapters 2 and 4), (b) are able to support many more instrument types that were previously considered (Chapters 2 and 3). I accomplished this via looking at the tasks of source separation and automatic music transcription. I look forward to seeing how future researchers and build on this work in an effort towards making the next generation of tools for artists and musicians.

5.1. Limitations and Future Work

Overall, one of the most major limitations of all the work I described in this dissertation is the training data that these systems are built on. All of the projects described here use the Slakh [184] dataset, which is a dataset of audio mixtures with corresponding source data and aligned note data. The problem with the Slakh data is that it is all synthetic data, because the audio data was synthesized from note data (specifically, MIDI data), the audio does not sound like the music was recorded by live musicians. The distribution of audio in Slakh does not match the distribution of musical audio found in the wild; it is–at best–a minute subset of the larger distribution of musical audio in the world. Modern deep learning systems are extremely data hungry, which necessitates large synthetic datasets like Slakh to be able to reach the scales required to successfully train these systems. However, modern deep learning systems are notoriously brittle when exposed to out-of-distribution data. Therefore, it should not be assumed that the systems presented in this work generalize well to recordings of live musicians (see Chapter 3, Section 3.3.2 for some experimental evidence of this). Further work is needed to create realistic-sounding datasets that can support systems like the ones described in this dissertation: datasets with a large, diverse set of audio and instrumentation that contains mixture data, source data, and aligned notation data.

One exciting future challenge for the hierarchical separation system would be to add the capability to train a hierarchical system without the need to explicitly define a hierarchy a priori. The primary question would be to understand if a system can learn a set of hierarchical relationships between sources, and if so, understanding if the learned hierarchy produces better source estimates than an established hierarchy. Furthermore, can such a system discover new sources not in its training data? In presenting my hierarchical

work in Chapter 2, the system was able to leverage the hierarchy to reduce its dependency on training data at the leaf nodes. Can hierarchical priors further help future reduce the dependency on isolated source data?

For Cerberus, one of the main limitations of the system is that the number of sources it supports must be decided ahead of time. In other words, you train a Cerberus system for mixes that contain a particular set of instruments (e.g., Pianos and Guitars) and then the system cannot be extended to support new instruments (e.g., Piano, Guitar, *and Bass*) without retraining a new system from scratch. In the time since I originally published this work, researchers have proposed a some solutions to this problem for multi-instrument AMT [84, 270] or for source separation, but rarely [160] have these tasks been handled jointly like I do here. More work into general purpose source separation and/or transcription systems is needed; I dream of a system that can automatically identify an arbitrary number of sources in a mix and separate and transcribe the instruments therein.

Finally, the major shortcoming of the Deep Score-Informed Separation (DSIS) system is that the edits are restricted to removing audio data or swapping which source the audio data gets allocated to; there is no way for the system to create new audio data not already present in the mix. Of course, this would require the system to be a generative model–something that DSIS is not. This line of reasoning brings up many interesting ideas for future work. Specifically, the majority of prior work in generative modeling for musical audio has focused on generating audio for a single source at a time.[1] In contrast, doing separation and generation jointly, allows generative audio modeling to be much more flexible with respect to supporting multi-instrument audio (just as happened with AMT). Adding note conditioning is also a great entryway for adding controllability to the system [302] because many musicians are very familiar with editing note data.

5.2. Looking Further Ahead

Taking a wider view, the work in this dissertation contributes toward a broader and more capable Computational Musical Scene Analysis toolkit. The ability to separate sources hierarchically, to transcribe multiple instruments in a mixture, and to use those transcriptions to make edits to musical scenes, we open up a world where these systems have a richer understanding of musical content. In turn, these systems could let us interact with musical content in new and exciting ways.

The work in this dissertation paves the way for making the advanced technology of the recording studio available anywhere. We will no longer need to perfectly record each instrument individually in an expensive

[1] There are few notable exceptions to this [47, 104].

sonically-treated isolation booth in order to change their volume levels or apply audio effects to them; we can use source separation technologies. We do not need perfectly isolated recordings to change a few sour notes; we can edit the notes using transcribed note data. We will no longer need dedicated, specialized hardware to get transcriptions of what we play; Automatic Music Transcription systems will do that for us. Let me concretize these ideas: imagine a set of musicians performing on street corner and recording their performance on their phone. From this ragtag phone recording, these musicians could make it sound like they recorded themselves in a million dollar studio. They could eliminate street noises and isolate each instrument, they could fix some botched notes, and from there, they could apply all of the usual studio effects to make themselves sound great.

Extrapolating further out from the direct applications of the technology developed in this dissertation, this field of study has the potential to make a big impact in more everyday scenarios as well. Outside of just making music, these systems can help people as they interact with general audio scenes. Imagine hearing aids that can identify what music you are listening to, and selectively boost certain frequencies based on your existing surroundings and the musical content, all with the goal of enhancing your experience with the piece. Such a system could identify non-relevant sounds to remove extraneous noise (source separation/enhancement) and could understand that an important violin solo is coming up (Automatic Music Transcription & score following). These systems could use their keen sense of musical scenes (or more general auditory scenes) to aid us in any way that we might want to interact with them.

Personally, I am most excited to see how Computational Musical Scene Analysis can enable new modes of interactive expression for musicians. To hearken back to the discussion at the beginning of this document, listening is a critical part of music making. What new things could we do if our musical tools were able to listen to what we create in the same way that another human musician might? How might the 'musical conversations' we have with listening machines alter how we make music? How do we build systems that *empower* users rather than sideline them? These are very big questions–perhaps each is deserving of its own dissertation–but as these systems start to percolate from the academic literature into the hands of end-users [83], I am enthusiastic about the types of new art that musicians can envision by using (or misusing) future interactive Computational Musical Scene Analysis systems.

CHAPTER 6

Appendix

6.1. (Bidirectional) Long Short-Term Memory Layers

Long Short-Term Memory layers [94, 118] (LSTMs) are a type of recurrent neural network. Recurrent layers are network layers that where the output of the layer is wired back into themselves (i.e., they have feedback connections) as well as onto the next layer. One practical problem with classic recurrent neural network layers is that, when trained with back-propagation, the gradients can be unstable, tending to vanish (go to zero) or explode (go to infinity). LSTM nodes solve this problem by adding a "forget gate" that allows information and gradients to flow through the node unchanged. A single node is shown in Figure 6.1. LSTM layers are composed of many LSTM cells.

Because of the feedback loops, LSTMs are well-suited for learning data that varies over time, like audio. Each individual loop processes the data step by step. For use in source separation in this dissertation, an audio signal is converted to a log-scale magnitude spectrogram and the LSTM layers ingest the audio as it changes along time. So, for example, an LSTM layer will ingest one column of the spectrogram at a time (i.e., all of the frequency components or the entire spectrum at that time step).

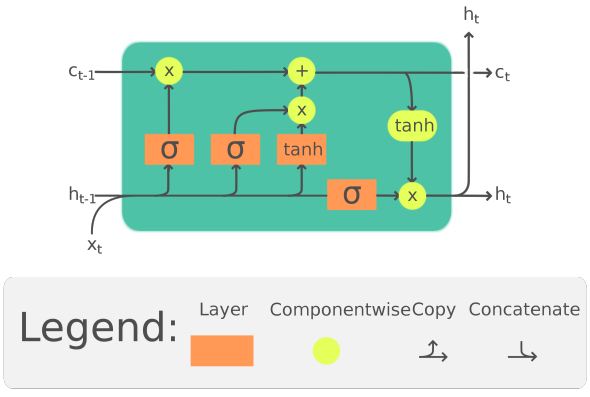


Figure 6.1. A single Long Short-Term Memory (LSTM) cell. LSTMs extend traditional recurrent neural network nodes by adding a "forget gate" (the σ and tanh nodes in the center) which allows gradients to flow through the cell unchanged. This prevents gradients from vanishing or exploding. Image by Guillaume Chevalier, licensed under CC BY-SA 4.0 (`https://creativecommons.org/licenses/by-sa/4.0/deed.en`).

The number of units an LSTM has defines its receptive field or the amount of time steps the layer can see at any given time. So for instance, if a spectrogram has 1024 time steps but the LSTM layer only has 4512 units, only 512 time steps of the spectrogram will be processed by the recurrent layers at a time. It will start with the first time step, and work its way across every one of the 1024 time steps, advancing by one step at a time. Because of this, if you train an LSTM that only has 512 time steps, it can accept spectrograms of any length at inference time.

An important variant of LSTMs that are used in this dissertation is the The Bidirectional LSTM, or BLSTM. BLSTMs are actually two sets of LSTMs, one that processes a signal forwards in time, $t = 0 \dots T$, and the other that processes the signal backwards, $t = T \dots 0$ (for a signal with T time steps). BLSTMs are thus anti-causal, which also means that they are difficult or impossible to use in realtime settings, however none of the work in this dissertation is designed with that in mind, so this extra processing that comes from being able to 'see into the future' is useful for making the resulting systems more performant.

6.2. Details about the AMT System in Chapter 4

In this section, I will describe the multi-instrument Automatic Music Transcription (AMT) system used as a pre-processing step to provide an estimated transcription as input to the Deep Score-Informed Separation (DSIS) experiments in Chapter 4, Section 4.3.3. The architecture of this AMT system is very similar to the Transcription Only system described in the Cerberus Chapter (Chapter 3), however its training data and implementation make it not directly comparable to that system.

This multi-instrument AMT system consists of a stack of Bidirectional Long Short-Term Memory (BLSTM) layers [94, 118] (see Appendix Section 6.1 for more details about BLSTMs), which ingested a mixture spectrogram and output a set of $N = 4$ piano rolls, where the piano roll frames were aligned with the spectrogram frames. The first layer is a batch normalization layer [123], followed by 4 BLSTM layers with 600 units each. The output of the last BLSTM layer was given to a fully connected layer with a leaky ReLU nonlinearity and this output is reshaped to become the piano roll estimates for each of the 4 instruments used in these experiments. This network was trained for 100,000 iterations with a batch size of 64 and using the Adam [142] optimizer with a learning rate of 3e-4. The learning rate was multiplied by a factor of 0.98 every 10,000 training steps.

The results of this multi-instrument AMT system are shown in Table 6.1, where I show both note onset/offset F1 scores and the frame F1 scores (see Introduction Section 1.3.2 for further details). Even

Metric	Piano	Guitar	Bass	Drums
Onset/offset F1	0.03	0.03	0.05	0.00
Frame F1	0.30	0.29	0.49	0.06

Table 6.1. Transcription F1 scores for the 4-layer BLSTM multi-instrument Automatic Music Transcription (AMT) system used as a pre-processing step for the Deep Score-Informed Separation (DSIS) interpolation experiments in Chapter 4, Section 4.3.3. Here, I provide both the note onset/offset score and the frame score (discussed in the Introduction Section 1.3.2) to show how even though this system does not get many onsets & offsets correct, it does do a decent job at getting some of the frames correct. An imperfect AMT system is actually desirable for the DSIS experiments because it lets us better simulate how the DSIS system's separation estimates will change when a user makes edits; if the AMT system were perfect there would be less opportunities to simulate potential changes to the piano roll.

though this system barely gets any of the note onsets and offsets correctly, it does produce a fair amount of correct results according to the frame F1 scores. This system serves as input for the interpolation experiment in Chapter 4, Section 4.3.3, therefore an imperfect AMT system is preferred. In the interpolation experiment, I wanted to simulate a user making edits to correct the piano roll input. Thus, if the upstream AMT system was perfect, then it would have been of no use to simulate the desired edits. With this system, there are lots of edits to simulate.

References

[1] Adobe. *iZotope RX9*. `https://www.adobe.com/products/audition.html`. Accessed 2022-06-14. 2022.

[2] Audionamix. *Audionamix*. `https://audionamix.com/`. 2021.

[3] AudioSourceRE. *AudioSourceRE*. `https://www.audiosourcere.com/products/`. Accessed 2021-04-13. 2021.

[4] David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. "The shattered gradients problem: If resnets are the answer, then what is the question?" In: *International Conference on Machine Learning*. PMLR. 2017, pp. 342–350.

[5] Lucien Barret, Jean-Baptiste Membrado, and Lucie Perrotta. "TRACING HISTORICAL DEVELOPMENTS OF CHORD PATTERNS IN WESTERN POPULAR MUSIC BETWEEN 1955 AND 1995". In: ().

[6] Adam Patrick Bell. *Dawn of the DAW: The Studio as Musical Instrument*. Oxford University Press, 2018.

[7] Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. "Automatic music transcription: An overview". In: *IEEE Signal Processing Magazine* 36.1 (2018), pp. 20–30.

[8] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. "Automatic music transcription: challenges and future directions". In: *Journal of Intelligent Information Systems* 41.3 (2013), pp. 407–434.

[9] Emmanouil Benetos, Anssi Klapuri, and Simon Dixon. "Score-informed transcription for automatic piano tutoring". In: *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*. IEEE. 2012, pp. 2153–2157.

[10] Rachel M Bittner, Brian McFee, Justin Salamon, Peter Li, and Juan Pablo Bello. "Deep Salience Representations for F0 Estimation in Polyphonic Music." In:

[11] Sebastian Böck and Markus Schedl. "Polyphonic piano note transcription with recurrent neural networks". In: *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE. 2012, pp. 121–124.

[12] Albert S Bregman. *Auditory scene analysis: The perceptual organization of sound*. MIT press, 1994.

[13] Guy J Brown and Martin Cooke. "Computational auditory scene analysis". In: *Computer Speech & Language* 8.4 (1994), pp. 297–336.

[14] Guy Jason Brown. "Computational auditory scene analysis: a representational approach." PhD thesis. University of sheffield, 1992.

[15] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

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

[17] Andreas Bugler, Bryan Pardo, and Prem Seetharaman. "A Study of Transfer Learning in Music Source Separation". In: *arXiv preprint arXiv:2010.12650* (2020).

[18] David Byrne. *How music works*. Three Rivers Press, 2017.

[19] Lee Callender, Curtis Hawthorne, and Jesse Engel. *Improving Perceptual Quality of Drum Transcription with the Expanded Groove MIDI Dataset*. 2020. arXiv: 2004.00188 [cs.SD].

[20] Estefanıa Cano, Derry FitzGerald, and Karlheinz Brandenburg. "Evaluation of quality of sound source separation algorithms: Human perception vs quantitative metrics". In: *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE. 2016, pp. 1758–1762.

[21] Mark Cartwright and Juan Pablo Bello. "Increasing drum transcription vocabulary using data synthesis". In:

[22] Mark Cartwright and Bryan Pardo. "Vocalsketch: Vocally imitating audio concepts". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM. 2015, pp. 43–46.

[23] Mark Cartwright, Bryan Pardo, Gautham Mysore, and Matt Hoffman. "Fast and Easy Crowdsourced Perceptual Audio Evaluation". In: *Proceedings of the 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Shanghai, China, March 20-25, 2016.

[24] Mark Cartwright, Bryan Pardo, and Gautham J Mysore. "Crowdsourced Pairwise-Comparison for Source Separation Evaluation". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 606–610.

[25] Rodrigo Castellon, Chris Donahue, and Percy Liang. "Codified audio language modeling learns useful representations for music information retrieval". In: *Proceedings of the 22nd International Society for Music Information Retrieval Conference*. ISMIR, Nov. 2021, pp. 88–96. DOI: 10.5281/zenodo.5624605.

[26] Estefanıa Cano Ceron. "Pitch-informed solo and accompaniment separation." PhD thesis. Universität Ilmenau, 2014.

[27] Tak-Shing T Chan and Yi-Hsuan Yang. "Informed group-sparse representation for singing voice separation". In: *IEEE Signal Processing Letters* 24.2 (2017), pp. 156–160.

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

[29] Jingjing Chen, Qirong Mao, and Dong Liu. "Dual-path transformer network: Direct context-aware modeling for end-to-end monaural speech separation". In: *arXiv preprint arXiv:2007.13975* (2020).

[30] Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. "Zero-shot audio source separation through query-based learning from weakly-labeled data". In: *AAAI 2022*. (2021).

[31] Zhuo Chen, Yi Luo, and Nima Mesgarani. "Deep attractor network for single-microphone speaker separation". In: *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2017, pp. 246–250.

[32] Hyeong-Seok Choi, Jang-Hyun Kim, Jaesung Huh, Adrian Kim, Jung-Woo Ha, and Kyogu Lee. "Phase-aware speech enhancement with deep complex u-net". In: *International Conference on Learning Representations*. 2018.

[33] Woosung Choi, Minseok Kim, Jaehwa Chung, and Soonyoung Jung. "LaSAFT: Latent Source Attentive Frequency Transformation for Conditioned Source Separation". In: *arXiv preprint arXiv:2010.11631* (2020).

[34] Woosung Choi, Minseok Kim, Jaehwa Chung, and Soonyoung Jung. "LaSAFT: Latent Source Attentive Frequency Transformation for Conditioned Source Separation". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 171–175.

[35] Woosung Choi, Minseok Kim, Jaehwa Chung, Daewon Lee, and Soonyoung Jung. "Investigating u-nets with various intermediate blocks for spectrogram-based singing voice separation". In: *arXiv preprint arXiv:1912.02591* (2019).

[36] Woosung Choi, Minseok Kim, Marco A Martınez Ramırez, Jaehwa Chung, and Soonyoung Jung. "Amss-net: Audio manipulation on user-specified sources with textual queries". In: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 1775–1783.

[37] Boaz Cogan, Noah Schaffer, Ethan Manilow, and Bryan Pardo. "Cooking With MSG: Enhancing the Output of Source Separation Using Generative Adversarial Networks". In: *International Society for Music Information Retrieval – Music Demixing Workshop (MDX)* (2021).

[38] Nicholas J Conard, Maria Malina, and Susanne C Münzel. "New flutes document the earliest musical tradition in southwestern Germany". In: *Nature* 460.7256 (2009), pp. 737–740.

[39] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[40] Ian Cross. "Music and cognitive evolution". In: *Oxford handbook of evolutionary psychology* (2007), pp. 649–667.

[41] Roger B Dannenberg, William P Birmingham, Bryan Pardo, Ning Hu, Colin Meek, and George Tzanetakis. "A comparative evaluation of search techniques for query-by-humming using the MUSART testbed". In: *Journal of the American Society for Information Science and Technology* 58.5 (2007), pp. 687–701.

[42] Matthew EP Davies and Mark D Plumbley. "Context-dependent beat tracking of musical audio". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.3 (2007), pp. 1009–1020.

[43] Alexandre Défossez. "Hybrid spectrogram and waveform source separation". In: *arXiv preprint arXiv:2111.03600* (2021).

[44] Alexandre Défossez, Nicolas Usunier, Léon Bottou, and Francis Bach. "Demucs: Deep extractor for music sources with extra unlabeled data remixed". In: *arXiv preprint arXiv:1909.01174* (2019).

[45] Joanna Demers. "Sampling the 1970s in hip-hop". In: *Popular Music* 22.1 (2003), pp. 41–56.

[46] Melvil Dewey. *A classification and subject index, for cataloguing and arranging the books and pamphlets of a library*. Brick row book shop, Incorporated, 1876.

[47] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. "Jukebox: A generative model for music". In: *arXiv preprint arXiv:2005.00341* (2020).

[48] David Ditter and Timo Gerkmann. "A multi-phase gammatone filterbank for speech separation via tasnet". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 36–40.

[49] Christian Dittmar, Jonathan Driedger, and Meinard Muller. "A separate and restore approach to score-informed music decomposition". In: *2015 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE. 2015, pp. 1–5.

[50] Christian Dittmar and Meinard Müller. "Reverse engineering the amen break—score-informed separation and restoration applied to drum recordings". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.9 (2016), pp. 1535–1547.

[51] Christian Dittmar and Meinard Müller. "Towards transient restoration in score-informed audio decomposition". In: *Proc. Int. Conf. Digital Audio Effects*. 2015, pp. 145–152.

[52] Daniel Dixon. *Mix bus 101: Why, when, and how to group tracks into a bus*. Aug. 2019. URL: `https://www.izotope.com/en/learn/mix-buses-101.html`.

[53] Matthias Dorfer, Andreas Arzt, and Gerhard Widmer. "Learning audio-sheet music correspondences for score identification and offline alignment". In: *arXiv preprint arXiv:1707.09887* (2017).

[54] J Stephen Downie. "Music information retrieval". In: *Annual review of information science and technology* 37.1 (2003), pp. 295–340.

[55] J Stephen Downie, Andreas F Ehmann, Mert Bay, and M Cameron Jones. "The music information retrieval evaluation exchange: Some observations and insights". In: *Advances in music information retrieval*. Springer, 2010, pp. 93–115.

[56] J. Stephen Downie. "The music information retrieval evaluation exchange (2005ndash;2007): A window into music information retrieval research". In: *Acoustical Science and Technology* 29.4 (2008), pp. 247–255. DOI: 10.1250/ast.29.247.

[57] Andis Draguns, Emıls Ozoliņš, Agris Šostaks, Matıss Apinis, and Kārlis Freivalds. "Residual shuffle-exchange networks for fast processing of long sequences". In: *arXiv preprint arXiv:2004.04662* (2020).

[58] Jonathan Driedger, Harald Grohganz, Thomas Prätzlich, Sebastian Ewert, and Meinard Müller. "Score-informed audio decomposition and applications". In: *Proceedings of the 21st ACM international conference on Multimedia*. 2013, pp. 541–544.

[59] Zhiyao Duan and Bryan Pardo. "Soundprism: An online system for score-informed source separation of music audio". In: *Selected Topics in Signal Processing, IEEE Journal of* 5.6 (2011), pp. 1205–1215.

[60] Edward Douglas. *The Beatles: Get Back Sound Editing Team on Revisiting and Cleaning Up Audio From 50 Years Ago for Peter Jackson's Hit Docuseries*. https://www.btlnews.com/interviews/beatles-get-back-sound-team-interview/. Accessed 2022-06-14. 2022.

[61] Dalia El Badawy, Ngoc QK Duong, and Alexey Ozerov. "On-the-fly audio source separation—A novel user-friendly framework". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.2 (2016), pp. 261–272.

[62] Daniel PW Ellis. "Beat tracking by dynamic programming". In: *Journal of New Music Research* 36.1 (2007), pp. 51–60.

[63] Anders Elowsson. "Polyphonic pitch tracking with deep layered learning". In: *The Journal of the Acoustical Society of America* 148.1 (2020), pp. 446–468.

[64] Valentin Emiya, Roland Badeau, and Bertrand David. "Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle". In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.6 (2009), pp. 1643–1654.

[65] Valentin Emiya, Emmanuel Vincent, Niklas Harlander, and Volker Hohmann. "Subjective and objective quality assessment of audio source separation". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 19.7 (2011), pp. 2046–2057.

[66] Jesse Engel, Lamtharn Hantrakul, Chenjie Gu, and Adam Roberts. "Ddsp: Differentiable digital signal processing". In: *arXiv preprint arXiv:2001.04643* (2020).

[67] Jesse Engel, Rigel Swavely, Lamtharn Hanoi Hantrakul, Adam Roberts, and Curtis Hawthorne. "Self-supervised Pitch Detection by Inverse Audio Synthesis". In: (2020).

[68] Brian Eno. "The studio as compositional tool". In: *Audio culture: Readings in modern music* (2004), pp. 127–130.

[69] Hakan Erdogan, John R. Hershey, Shinji Watanabe, and Jonathan Le Roux. "Phase-Sensitive and Recognition-Boosted Speech Separation using Deep Recurrent Neural Networks". In: *ICASSP*. Apr. 2015, pp. 708–712.

[70] Sebastian Ewert and Meinard Müller. "Using score-informed constraints for NMF-based source separation". In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE. 2012, pp. 129–132.

[71] Sebastian Ewert, Bryan Pardo, Mathias Muller, and Mark D Plumbley. "Score-informed source separation for musical audio recordings: An overview". In: *Signal Processing Magazine, IEEE* 31.3 (2014), pp. 116–124.

[72] Sebastian Ewert and Mark B Sandler. "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)*. IEEE. 2017, pp. 2277–2281.

[73] Sebastian Ewert, Siying Wang, Meinard Müller, and Mark Sandler. "Score-informed identification of missing and extra notes in piano recordings". In: (2016).

[74] Morwaread M Farbood, David J Heeger, Gary Marcus, Uri Hasson, and Yulia Lerner. "The neural processing of hierarchical structure in music and speech at different timescales". In: *Frontiers in neuroscience* 9 (2015), p. 157.

[75] Cédric Févotte, Nancy Bertin, and Jean-Louis Durrieu. "Nonnegative matrix factorization with the Itakura-Saito divergence: With application to music analysis". In: *Neural computation* 21.3 (2009), pp. 793–830.

[76] Hugo Flores Garcia, Aldo Aguilar, Ethan Manilow, and Bryan Pardo. "Leveraging Hierarchical Structures for Few-Shot Musical Instrument Recognition". In: *International Society for Music Information Retrieval*. 2021.

[77] Brendan Fox, Andrew Sabin, Bryan Pardo, and Alec Zopf. "Modeling perceptual similarity of audio signals for blind source separation evaluation". In: *Independent Component Analysis and Signal Separation*. Springer, 2007, pp. 454–461.

[78] Joachim Fritsch, Joachim Ganseman, and Mark D Plumbley. "A comparison of two different methods for score-informed source separation". In: *Proc. Int. Workshop Machine Learning Music (MML)*. 2012, p. 2.

[79] Joachim Fritsch and Mark D Plumbley. "Score informed audio source separation using constrained nonnegative matrix factorization and score synthesis". In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 888–891.

[80] Hiromasa Fujihara, Masataka Goto, Jun Ogata, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. "Automatic synchronization between lyrics and music CD recordings based on Viterbi alignment of segregated vocal signals". In: *Eighth IEEE International Symposium on Multimedia (ISM'06)*. IEEE. 2006, pp. 257–264.

[81] Joachim Ganseman, Paul Scheunders, Gautham J Mysore, and Jonathan S Abel. "Evaluation of a score-informed source separation system." In: *ISMIR*. 2010, pp. 219–224.

[82] Joachim Ganseman, Paul Scheunders, Gautham J Mysore, and Jonathan S Abel. "Source separation by score synthesis". In: *ICMC*. 2010.

[83] Hugo Flores Garcia, Aldo Aguilar, Ethan Manilow, Dmitry Vedenko, and Bryan Pardo. "Deep Learning Tools for Audacity: Helping Researchers Expand the Artist's Toolkit". In: *NeurIPS 2021 Workshop on Machine Learning for Creativity and Design*. 2021.

[84] Josh Gardner, Ian Simon, Ethan Manilow, Curtis Hawthorne, and Jesse Engel. "MT3: Multi-Task Multitrack Music Transcription". In: *arXiv preprint arXiv:2111.03017* (2021).

[85] Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. "Audio Set: An ontology and human-labeled dataset for audio events". In: *IEEE ICASSP*. 2017.

[86] Timo Gerkmann, Martin Krawczyk-Becker, and Jonathan Le Roux. "Phase processing for single-channel speech enhancement: History and recent advances". In: *IEEE Signal Processing Magazine* 32.2 (2015), pp. 55–66.

[87] Beat Gfeller, Dominik Roblek, and Marco Tagliasacchi. "One-shot conditional audio filtering of arbitrary sounds". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 501–505.

[88] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C Smith. "Query by humming: Musical information retrieval in an audio database". In: *Proceedings of the third ACM international conference on Multimedia*. 1995, pp. 231–236.

[89] Jon Gillick, Adam Roberts, Jesse Engel, Douglas Eck, and David Bamman. "Learning to Groove with Inverse Sequence Transformations". In: *International Conference on Machine Learning (ICML)*. 2019.

[90] Stan Godlovitch. *Musical performance: A philosophical study*. Routledge, 2002.

[91] Matan Gover. "Score-informed source separation of choral music". In: (2020).

[92] Theodore C Grame. "Bamboo and music: a new approach to organology". In: *Ethnomusicology* 6.1 (1962), pp. 8–14.

[93] Charles L Granata and Tony Asher. *Wouldn't it be Nice: Brian Wilson and the Making of the Beach Boys' Pet Sounds*. Chicago Review Press, 2016.

[94] Alex Graves and Jürgen Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures". In: *Neural networks* 18.5-6 (2005), pp. 602–610.

[95] Daniel Griffin and Jae Lim. "Signal estimation from modified short-time Fourier transform". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 32.2 (1984), pp. 236–243.

[96] D. Gunawan and D. Sen. "Iterative Phase Estimation for the Synthesis of Separated Sources From Single-Channel Mixtures". In: *IEEE Signal Processing Letters* 17.5 (May 2010), pp. 421–424. ISSN: 1070-9908. DOI: 10.1109/LSP.2010.2042530.

[97] Richard W Hamming. "Error detecting and error correcting codes". In: *The Bell system technical journal* 29.2 (1950), pp. 147–160.

[98] Stephen Handel. *Listening: An introduction to the perception of auditory events*. The MIT Press, 1993.

[99] Dorothea E Hast, James R Cowdery, and Stanley Arnold Scott. *Exploring the world of music: an introduction to music from a world music perspective*. Kendall Hunt, 1999.

[100] Verena Haunschmid, Ethan Manilow, and Gerhard Widmer. "audiolime: Listenable explanations using source separation". In: *arXiv preprint arXiv:2008.00582* (2020).

[101] Verena Haunschmid, Ethan Manilow, and Gerhard Widmer. "Towards Musically Meaningful Explanations Using Source Separation". In: *arXiv preprint arXiv:2009.02051* (2020).

[102] Marc D Hauser and Josh McDermott. "The evolution of the music faculty: A comparative perspective". In: *Nature neuroscience* 6.7 (2003), pp. 663–668.

[103] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. "Onsets and frames: Dual-objective piano transcription". In: *arXiv preprint arXiv:1710.11153* (2017).

[104] Curtis Hawthorne, Ian Simon, Adam Roberts, Neil Zeghidour, Josh Gardner, Ethan Manilow, and Jesse Engel. *Multi-instrument Music Synthesis with Spectrogram Diffusion*. 2022. DOI: 10.48550/ARXIV.2206.05408. URL: https://arxiv.org/abs/2206.05408.

[105] Curtis Hawthorne, Ian Simon, Rigel Swavely, Ethan Manilow, and Jesse Engel. "Sequence-to-Sequence Piano Transcription with Transformers". In: *International Society for Music Information Retrieval*. 2021.

[106] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. "Enabling factorized piano music modeling and generation with the MAESTRO dataset". In: *arXiv preprint arXiv:1810.12247* (2018).

[107] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[108] Toni Heittola, Anssi Klapuri, and Tuomas Virtanen. "Musical instrument recognition in polyphonic audio using source-filter model for sound separation." In: *ISMIR*. 2009, pp. 327–332.

[109] Romain Hennequin, Bertrand David, and Roland Badeau. "Score informed audio source separation using a parametric model of non-negative spectrogram". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE. 2011, pp. 45–48.

[110] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. "Spleeter: a fast and efficient music source separation tool with pre-trained models". In: *Journal of Open Source Software* 5.50 (2020), p. 2154.

[111] Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. "Spleeter: A fast and state-of-the art music source separation tool with pre-trained models". In: *Late-Breaking/Demo ISMIR 2019* (2019).

[112] Carlos Hernandez-Olivan, Ignacio Zay Pinilla, Carlos Hernandez-Lopez, and Jose R Beltran. "A comparison of deep learning methods for timbre analysis in polyphonic automatic music transcription". In: *Electronics* 10.7 (2021), p. 810.

[113] J.R. Hershey, Z. Chen, J. Le Roux, and S. Watanabe. "Deep Clustering: Discriminative Embeddings for Segmentation and Separation". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Mar. 2016, pp. 31–35. DOI: 10.1109/ICASSP.2016.7471631. URL: http://www.merl.com/publications/TR2016-003.

[114] John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. "Deep clustering: Discriminative embeddings for segmentation and separation". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 31–35.

[115] John R. Hershey, Zhuo Chen, and Jonathan Le Roux. "Deep Clustering: Discriminative Embeddings for Segmentation and Separation". In: *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2016, pp. 31–35.

[116] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, Malcolm Slaney, Ron Weiss J., and Kevin Wilson. "CNN architectures for large-scale audio classification". In: *2017 ieee international conference on acoustics, speech and signal processing (icassp)*. IEEE. 2017, pp. 131–135.

[117] Andrew Hines, Jan Skoglund, Anil C Kokaram, and Naomi Harte. "ViSQOL: an objective speech quality model". In: *EURASIP Journal on Audio, Speech, and Music Processing* 2015.1 (2015), pp. 1–18.

[118] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[119] Matthew Homer. "Beyond the studio: the impact of home recording technologies on music creation and consumption". In: *Nebula* 6.3 (2009), pp. 85–99.

[120] Henkjan Honing, Carel ten Cate, Isabelle Peretz, and Sandra E Trehub. *Without it no music: cognition, biology and evolution of musicality*. 2015.

[121] Ying Hu and Guizhong Liu. "Separation of singing voice using nonnegative matrix partial cofactorization for singer identification". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23.4 (2015), pp. 643–653.

[122] Don Ihde. *Technics and praxis: A philosophy of technology*. Vol. 24. Springer Science & Business Media, 2012.

[123] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[124] ITU. *Recommendation ITU-R BS. 1534-2: Method for the subjective assessment of intermediate quality level of audio systems*. 2014.

[125] iZotope. *iZotope RX9*. `https://www.izotope.com/en/products/rx.html`. Accessed 2021-06-14. 2022.

[126] Andreas Jansson, Rachel M Bittner, Sebastian Ewert, and Tillman Weyde. "Joint Singing Voice Separation and F0 Estimation with Deep U-Net Architectures". In: ().

[127] Andreas Jansson, Eric Humphrey, Nicola Montecchio, Rachel Bittner, Aparna Kumar, and Tillman Weyde. "Singing voice separation with deep U-Net convolutional networks". In: (2017).

[128] Vivek Jayaram and John Thickstun. "Parallel and Flexible Sampling from Autoregressive Models via Langevin Dynamics". In: Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021. PMLR, 2021, pp. 4807–4818. URL: `http://proceedings.mlr.press/v139/jayaram21b.html`.

[129] Kristoffer Jensen and David G Hebert. "Evaluation and Prediction of Harmonic Complexity Across 76 Years of Billboard 100 Hits". In: *International Symposium on Computer Music Multidisciplinary Research*. Springer. 2015, pp. 283–296.

[130] Nicholas Jillings, David Moffat, Brecht De Man, Joshua D Reiss, and Ryan Stables. "Web audio evaluation tool: A framework for subjective assessment of audio". In: (2016).

[131] Cyril Joder and Björn Schuller. "Score-informed leading voice separation from monaural audio". In: *Proceedings 13th International Society for Music Information Retrieval Conference, ISMIR 2012*. 2012.

[132] Andrew Johnston, Linda Candy, and Ernest Edmonds. "Designing and evaluating virtual musical instruments: facilitating conversational user interaction". In: *Design Studies* 29.6 (2008), pp. 556–571.

[133] Venkatesh S Kadandale, Juan F Montesinos, Gloria Haro, and Emilia Gómez. "Multi-task U-Net for Music Source Separation". In: *arXiv preprint arXiv:2003.10414* (2020).

[134] Margaret J Kartomi. *On concepts and classifications of musical instruments*. University of Chicago Press Chicago, 1990.

[135] Rainer Kelz, Sebastian Böck, and Gerhard Widmer. "Deep polyphonic adsr piano note transcription". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 246–250.

[136] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. "On the potential of simple framewise approaches to piano transcription". In: *arXiv preprint arXiv:1612.05153* (2016).

[137] Bongjun Kim, Madhav Ghei, Bryan Pardo, and Zhiyao Duan. "Vocal imitation set: a dataset of vocally imitated sound events using the audioset ontology". In: *Proceedings of the 2018 Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE2018)*. 2018.

[138] Chungeun Kim, Emad M Grais, Russell Mason, and Mark D Plumbley. "Perception of phase changes in the context of musical audio source separation". In: *Audio Engineering Society Convention 145*. Audio Engineering Society. 2018.

[139] Jong Wook Kim and Juan Pablo Bello. "Adversarial learning for improved onsets and frames music transcription". In: *arXiv preprint arXiv:1906.08512* (2019).

[140] Jong Wook Kim, Justin Salamon, Peter Li, and Juan Pablo Bello. "Crepe: A convolutional representation for pitch estimation". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 161–165.

[141] Minseok Kim, Woosung Choi, Jaehwa Chung, Daewon Lee, and Soonyoung Jung. *KUIELab-MDX-Net: A Two-Stream Neural Network for Music Demixing*. 2021. DOI: 10.48550/ARXIV.2111.12203. URL: https://arxiv.org/abs/2111.12203.

[142] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[143] Anssi Klapuri and Manuel Davy. "Signal processing methods for music transcription". In: (2007).

[144] Anssi Klapuri, Tuomas Virtanen, Antti Eronen, and Jarno Seppänen. "Automatic transcription of musical recordings". In: *Consistent & Reliable Acoustic Cues Workshop, CRAC-01, Aalborg, Denmark*. 2001.

[145] Anssi P Klapuri. "Automatic music transcription as we know it today". In: *Journal of New Music Research* 33.3 (2004), pp. 269–282.

[146] Donald Ervin Knuth. *The art of computer programming*. Vol. 1. Pearson Education, 1968.

[147] Morten Kolbæk, Dong Yu, Zheng-Hua Tan, and Jesper Jensen. "Multi-Talker Speech Separation with Utterance-Level Permutation Invariant Training of Deep Recurrent Neural Networks". In: *IEEE/ACM Transactions on Audio, Speech and Language Processing* (2017), pp. 1901–1913.

[148] Qiuqiang Kong, Yin Cao, Haohe Liu, Keunwoo Choi, and Yuxuan Wang. "Decoupling Magnitude and Phase Estimation with Deep ResUNet for Music Source Separation". In: *ISMIR*. 2021.

[149] Qiuqiang Kong, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. "High-resolution Piano Transcription with Pedals by Regressing Onset and Offset Times". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 3707–3717.

[150] Naoko Kosugi, Yuichi Nishihara, Tetsuo Sakata, Masashi Yamamuro, and Kazuhiko Kushima. "A practical query-by-humming system for a large music database". In: *Proceedings of the eighth ACM international conference on Multimedia*. 2000, pp. 333–342.

[151] Sebastian Kraft and Udo Zölzer. "BeaqleJS: HTML5 and JavaScript based framework for the subjective evaluation of audio quality". In: *Linux Audio Conference, Karlsruhe, DE*. 2014.

[152] Cyril Laurier, Jens Grivolla, and Perfecto Herrera. "Multimodal music mood classification using audio and lyrics". In: *2008 Seventh International Conference on Machine Learning and Applications*. IEEE. 2008, pp. 688–693.

[153] Jonathan Le Roux, Gordon Wichern, Shinji Watanabe, Andy Sarroff, and John R Hershey. "Phasebook and friends: Leveraging discrete representations for source separation". In: *IEEE Journal of Selected Topics in Signal Processing* 13.2 (2019), pp. 370–382.

[154] Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R Hershey. "SDR–half-baked or well done?" In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 626–630.

[155] Jonathan Le Roux, Scott T. Wisdom, Hakan Erdogan, and John R. Hershey. "SDR – half-baked or well done?" In: *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2019.

[156] Daniel Lee and Hyunjune Seung. "Algorithms for Non-negative Matrix Factorization". In: *Adv. Neural Inform. Process. Syst.* 13 (Feb. 2001).

[157] Jie Hwan Lee, Hyeong-Seok Choi, and Kyogu Lee. "Audio query-based music source separation". In: *arXiv preprint arXiv:1908.06593* (2019).

[158] Marc Leman. *Embodied music cognition and mediation technology*. MIT press, 2007.

[159] Kjell Lemstrom. "String matching techniques for music retrieval." In: (2002).

[160] Liwei Lin, Qiuqiang Kong, Junyan Jiang, and Gus Xia. *A Unified Model for Zero-shot Music Source Separation, Transcription and Synthesis*. 2021. arXiv: 2108.03456 [cs.SD].

[161] Haohe Liu, Qiuqiang Kong, and Jiafeng Liu. *CWS-PResUNet: Music Source Separation with Channel-wise Subband Phase-aware ResUNet*. 2021. DOI: 10.48550/ARXIV.2112.04685. URL: https://arxiv.org/abs/2112.04685.

[162] Antoine Liutkus, Jean-Louis Durrieu, Laurent Daudet, and Gaël Richard. "An overview of informed audio source separation". In: *2013 14th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*. IEEE. 2013, pp. 1–4.

[163] Antoine Liutkus, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave. "The 2016 Signal Separation Evaluation Campaign". In: *International Conference on Latent Variable Analysis and Signal Separation*. Springer. 2017, pp. 323–332.

[164] Yi Luo, Zhuo Chen, John R Hershey, Jonathan Le Roux, and Nima Mesgarani. "Deep clustering and conventional networks for music separation: Stronger together". In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE. 2017, pp. 61–65.

[165] Yi Luo, Zhuo Chen, John R Hershey, Jonathan Le Roux, and Nima Mesgarani. "Deep Clustering and Conventional Networks for Music Separation: Stronger Together". In: *arXiv preprint arXiv:1611.06265* (2016).

[166] Yi Luo, Zhuo Chen, John R. Hershey, Jonathan Le Roux, and Nima Mesgarani. "Deep Clustering and Conventional Networks for Music Separation: Stronger Together". In: *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Mar. 2017, pp. 61–65.

[167] Yi Luo, Zhuo Chen, and Takuya Yoshioka. "Dual-path rnn: efficient long sequence modeling for time-domain single-channel speech separation". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 46–50.

[168] Yi Luo and Nima Mesgarani. "TasNet: Surpassing Ideal Time-Frequency Masking for Speech Separation". In: *arXiv preprint arXiv:1809.07454* (Sept. 2018).

[169] Yi Luo and Nima Mesgarani. "TasNet: Surpassing ideal time-frequency masking for speech separation". In: *arXiv preprint arXiv:1809.07454* (2018).

[170] Yi Luo and Nima Mesgarani. "Tasnet: time-domain audio separation network for real-time, single-channel speech separation". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 696–700.

[171] Thor Magnusson. "Of epistemic tools: Musical instruments as cognitive extensions". In: *Organised Sound* 14.2 (2009), pp. 168–176.

[172] Thor Magnusson. *Sonic writing: technologies of material, symbolic, and signal inscriptions*. Bloomsbury Academic, 2019.

[173] Veli Mäkinen, Gonzalo Navarro, and Esko Ukkonen. "Algorithms for transposition invariant string matching". In: *Annual Symposium on Theoretical Aspects of Computer Science*. Springer. 2003, pp. 191–202.

[174] Shoji Makino. *Audio source separation*. Vol. 433. Springer, 2018.

[175] Ethan Manilow, Patrick O'Reilly, Prem Seetharaman, and Bryan Pardo. "Source Separation By Steering Pretrained Music Models". In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 126–130.

[176] Ethan Manilow and Bryan Pardo. "Bespoke Neural Networks for Score-Informed Source Separation". In: *International Society for Music Information Retrieval - Late Breaking Demo* (2020).

[177] Ethan Manilow and Bryan Pardo. "Leveraging Repetition to Do Audio Imputation". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2017)*. New Paltz, NY, USA, October 15-18, 2017.

[178] Ethan Manilow, Prem Seetharaman, and Bryan Pardo. "Simultaneous Separation and Transcription of Mixtures with Multiple Polyphonic and Percussive Instruments". In: *arXiv preprint arXiv:1910.12621* (2019).

[179] Ethan Manilow, Prem Seetharaman, and Bryan Pardo. "Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 771–775.

[180] Ethan Manilow, Prem Seetharaman, Fatemeh Pishdadian, and Bryan Pardo. "Predicting Algorithm Efficacy for Adaptive Multi-Cue Source Separation". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2017)*. New Paltz, NY, USA, October 15-18, 2017.

[181] Ethan Manilow, Prem Seetharman, and Justin Salamon. *Open Source Tools & Data for Music Source Separation*. https://source-separation.github.io/tutorial, 2020. URL: `https://source-separation.github.io/tutorial`.

[182] Ethan Manilow, Gordon Wichern, and Jonathan Le Roux. "Hierarchical Musical Instrument Separation". In: *International Society for Music Information Retrieval*. 2020.

[183] Ethan Manilow, Gordon Wichern, and Jonathan Le Roux. "Hierarchical Musical Source Separation". In: *Proc. International Society for Music Information Retrieval (ISMIR) Conference*. Oct. 2020.

[184] Ethan Manilow, Gordon Wichern, Prem Seetharaman, and Jonathan Le Roux. "Cutting Music Source Separation Some Slakh: A Dataset to Study the Impact of Training Data Quality and Quantity". In: *Proc. IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE. 2019.

[185] Pranay Manocha, Adam Finkelstein, Richard Zhang, Nicholas J Bryan, Gautham J Mysore, and Zeyu Jin. "A differentiable perceptual audio metric learned from just noticeable differences". In: *arXiv preprint arXiv:2001.04460* (2020).

[186] Pranay Manocha, Zeyu Jin, Richard Zhang, and Adam Finkelstein. "CDPAM: Contrastive learning for perceptual audio similarity". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 196–200.

[187] Josh McDermott and Marc Hauser. "The origins of music: Innateness, uniqueness, and evolution". In: *Music perception* 23.1 (2005), pp. 29–59.

[188] Samuel A Mehr, Manvir Singh, Dean Knox, Daniel M Ketter, Daniel Pickens-Jones, Stephanie Atwood, Christopher Lucas, Nori Jacoby, Alena A Egner, Erin J Hopkins, et al. "Universality and diversity in human song". In: *Science* 366.6468 (2019).

[189] Francisco José Cuadrado Méndez. "The use of sequencer tools during the composition process: A field study". In: *Journal of Music, Technology & Education* 8.1 (2015), pp. 55–70.

[190] Meredith Woerner. *Peter Jackson Details How 'Get Back' Used Machine Learning to Restore the Beatles' Sound and Footage*. `https://variety.com/2021/music/news/peter-jackson-get-back-restored-beatles-f` Accessed 2022-06-14. 2022.

[191] Annamaria Mesaros and Tuomas Virtanen. "Automatic recognition of lyrics in singing". In: *EURASIP Journal on Audio, Speech, and Music Processing* 2010.1 (2010), p. 546047.

[192] Gabriel Meseguer-Brocal and Geoffroy Peeters. "Conditioned-U-Net: Introducing a Control Mechanism in the U-Net for Multiple Source Separations". In: *arXiv preprint arXiv:1907.01277* (2019).

[193] Paul D Miller. *Sound unbound: sampling digital music and culture*. Mit Press, 2008.

[194] Stylianos Ioannis Mimilakis, Konstantinos Drossos, Joao F Santos, Gerald Schuller, Tuomas Virtanen, and Yoshua Bengio. "Monaural singing voice separation with skip-filtering connections and recurrent inference of time-frequency mask". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 721–725.

[195] Marius Miron et al. "Source separation methods for orchestral music: timbre-informed and score-informed strategies". PhD thesis. Universitat Pompeu Fabra, 2018.

[196] Marius Miron, Julio J Carabias Orti, and Jordi Janer Mestres. "Improving score-informed source separation for classical music through note refinement". In: *Müller M, Wiering F, editors. Proceedings of the 16th International Society for Music Information Retrieval (ISMIR) Conference; 2015 Oct 26-30; Málaga, Spain. Canada: International Society for Music Information Retrieval; 2015*. International Society for Music Information Retrieval (ISMIR). 2015.

[197] Marius Miron, Julio J Carabias-Orti, Juan J Bosch, Emilia Gómez, and Jordi Janer. "Score-informed source separation for multichannel orchestral recordings". In: *Journal of Electrical and Computer Engineering* 2016 (2016).

[198] Yuki Mitsufuji, Giorgio Fabbro, Stefan Uhlich, and Fabian-Robert Stöter. "Music demixing challenge at ISMIR 2021". In: *arXiv e-prints* (2021), arXiv–2108.

[199] Yuki Mitsufuji, Tatsuya Haraguchi, and Yoshikazu Takashima. *AI Sound Separation x Entertainment*. `https://www.sony.net/SonyInfo/sony_ai/audio_2.html`. Accessed 2021-04-26. 2021.

[200] Bhavya Mor, Sunita Garhwal, and Ajay Kumar. "A systematic literature review on computational musicology". In: *Archives of Computational Methods in Engineering* (2019), pp. 1–15.

[201] Meinard Müller. "Musically Informed Audio Decomposition". In: *Fundamentals of Music Processing*. Springer, 2015, pp. 415–480.

[202] Antonio Jesús Muñoz-Montoro, Pedro Vera-Candeas, Raquel Cortina, Elıas F Combarro, and Pedro Alonso-Jordá. "Online score-informed source separation in polyphonic mixtures using instrument spectral patterns". In: *Computational and Mathematical Methods* 1.4 (2019), e1040.

[203] *MUSDB18 Benchmark (Music Source Separation) | Papers With Code*. 2021. URL: `https://paperswithcode.com/sota/music-source-separation-on-musdb18` (visited on 04/26/2021).

[204] Eita Nakamura, Emmanouil Benetos, Kazuyoshi Yoshii, and Simon Dixon. "Towards complete polyphonic music transcription: Integrating multi-pitch detection and rhythm quantization". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 101–105.

[205] Luc Nijs, Micheline Lesaffre, and Marc Leman. "The musical instrument as a natural extension of the musician". In: *the 5th Conference of Interdisciplinary Musicology*. LAM-Institut jean Le Rond d'Alembert. 2009, pp. 132–133.

[206] Sergio Oramas, Oriol Nieto, Mohamed Sordo, and Xavier Serra. "A deep multimodal approach for cold-start music recommendation". In: *Proceedings of the 2nd workshop on deep learning for recommender systems*. 2017, pp. 32–37.

[207] Bobby Owsinski. *The mixing engineer's handbook*. Nelson Education, 2013.

[208] Bryan Pardo. *Lecture slides in Machine Perception of Music*. 2008.

[209] Manuel Pariente, Samuele Cornell, Antoine Deleforge, and Emmanuel Vincent. "Filterbank design for end-to-end speech separation". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 6364–6368.

[210] Doheum Park, Juhan Nam, and Juyong Park. "Novelty and influence of creative works, and quantifying patterns of advances based on probabilistic references networks". In: *EPJ Data Science* 9.1 (2020), p. 2.

[211] Gloria G Parras, Javier Nieto-Diego, Guillermo V Carbajal, Catalina Valdés-Baizabal, Carles Escera, and Manuel S Malmierca. "Neurons along the auditory pathway exhibit a hierarchical organization of prediction error". In: *Nature communications* 8.1 (2017), pp. 1–17.

[212] Brian Patton, Yannis Agiomyrgiannakis, Michael Terry, Kevin Wilson, Rif A Saurous, and D Sculley. "AutoMOS: Learning a non-intrusive assessor of naturalness-of-speech". In: *arXiv preprint arXiv:1611.09207* (2016).

[213] Jouni Paulus and Tuomas Virtanen. "Drum transcription with non-negative spectrogram factorisation". In: *2005 13th European Signal Processing Conference*. IEEE. 2005, pp. 1–4.

[214] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[215] Jonathan E Peelle, Ingrid Johnsrude, and Matthew H Davis. "Hierarchical processing for speech in human auditory cortex and beyond". In: *Frontiers in human neuroscience* 4 (2010), p. 51.

[216] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. "Film: Visual reasoning with a general conditioning layer". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[217] Juan I. Perotti and Orlando V. Billoni. "On the emergence of Zipf 's law in music". In: *Physica A: Statistical Mechanics and its Applications* 549 (July 2020), p. 124309. ISSN: 0378-4371. DOI: 10.1016/j.physa.2020.124309. URL: http://dx.doi.org/10.1016/j.physa.2020.124309.

[218] Mark D Plumbley, Samer A Abdallah, Juan Pablo Bello, Mike E Davies, Giuliano Monti, and Mark B Sandler. "Automatic music transcription and audio source separation". In: *Cybernetics &Systems* 33.6 (2002), pp. 603–627.

[219] Graham E Poliner and Daniel PW Ellis. "A discriminative model for polyphonic piano transcription". In: *EURASIP Journal on Advances in Signal Processing* 2007 (2006), pp. 1–9.

[220] Laure Prétet, Romain Hennequin, Jimena Royo-Letelier, and Andrea Vaglio. "Singing voice separation: A study on training data". In: *ICASSP*. 2019.

[221] Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching*. Columbia University, 2016.

[222] Colin Raffel, Brian McFee, Eric J Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, Daniel PW Ellis, and C Colin Raffel. "mir_eval: A transparent implementation of common MIR metrics". In: *Proc. of the 15th International Society for Music Information Retrieval Conference* (2014).

[223] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stter, Stylianos Ioannis Mimilakis, and Rachel Bittner. *The Musdb18 Corpus for Music Separation*. Dec. 2017. DOI: 10.5281/zenodo.1117372. URL: https://zenodo.org/record/1117372.

[224] Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. *The MUSDB18 corpus for music separation*. Dec. 2017. DOI: 10.5281/zenodo.1117372. URL: https://doi.org/10.5281/zenodo.1117372.

[225] Antony W Rix, John G Beerends, Michael P Hollier, and Andries P Hekstra. "Perceptual evaluation of speech quality (PESQ)-a new method for speech quality assessment of telephone networks and codecs". In: *2001 IEEE international conference on acoustics, speech, and signal processing. Proceedings (Cat. No. 01CH37221)*. Vol. 2. IEEE. 2001, pp. 749–752.

[226] Matthias Robine, Pierre Hanna, Pascal Ferraro, and Julien Allali. "Adaptation of string matching algorithms for identification of near-duplicate music documents". In: *Workshop on Plagiarism Analysis, Authorship Identification, and Near-Duplicate Detection (PAN07)*. 2007, pp. 37–43.

[227] Tara Rodgers. "On the process and aesthetics of sampling in electronic music production". In: *Organised Sound* 8.3 (2003), pp. 313–320.

[228] Francisco J Rodriguez-Serrano, Zhiyao Duan, Pedro Vera-Candeas, Bryan Pardo, and Julio J Carabias-Orti. "Online score-informed source separation with adaptive instrument models". In: *Journal of New Music Research* 44.2 (2015), pp. 83–96.

[229] David F Rosenthal and Hiroshi G Okuno. *Computational auditory scene analysis.* Lawrence Erlbaum Associates Publishers, 1998.

[230] Justin Salamon and Emilia Gómez. "Melody Extraction from Polyphonic Music Signals using Pitch Contour Characteristics". In: *IEEE Transactions on Audio, Speech and Language Processing* (2012).

[231] Justin Salamon, Emilia Gómez, Daniel PW Ellis, and Gaël Richard. "Melody extraction from polyphonic music signals: Approaches, applications, and challenges". In: *IEEE Signal Processing Magazine* 31.2 (2014), pp. 118–134.

[232] David Samuel, Aditya Ganeshan, and Jason Naradowsky. "Meta-learning extractors for music source separation". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 816–820.

[233] Ryosuke Sawata, Stefan Uhlich, Shusuke Takahashi, and Yuki Mitsufuji. "All for One and One for All: Improving Music Separation by Bridging Networks". In: *arXiv preprint arXiv:2010.04228* (2020).

[234] Ryosuke Sawata, Stefan Uhlich, Shusuke Takahashi, and Yuki Mitsufuji. "All for One and One for All: Improving Music Separation by Bridging Networks". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 51–55.

[235] R Keith Sawyer. "Music and conversation". In: *Musical communication* 45 (2005), p. 60.

[236] Nicolas Scaringella, Giorgio Zoia, and Daniel Mlynek. "Automatic genre classification of music content: a survey". In: *IEEE Signal Processing Magazine* 23.2 (2006), pp. 133–141.

[237] Pierre Schaeffer. *In search of a concrete music.* Vol. 15. Univ of California Press, 2012.

[238] Susan Schmidt Horning. "Chasing sound: Technology, culture, and the art of studio recording from Edison to the LP". in: (2013).

[239] Michael Schoeffler, Fabian-Robert Stöter, Bernd Edler, and Jürgen Herre. "Towards the next generation of web-based experiments: A case study assessing basic audio quality following the ITU-R recommendation BS. 1534 (MUSHRA)". in: *1st Web Audio Conference*. 2015, pp. 1–6.

[240] Prem Seetharaman, Gordon Wichern, Bryan Pardo, and Jonathan Le Roux. "AutoClip: Adaptive Gradient Clipping for Source Separation Networks". In: *Proc. International Workshop on Machine Learning for Signal Processing (MLSP)*. Oct. 2020.

[241] Prem Seetharaman, Gordon Wichern, Shrikant Venkataramani, and Jonathan Le Roux. "Class-conditional embeddings for music source separation". In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 301–305.

[242] Joan Serra, Emilia Gómez, and Perfecto Herrera. "Audio cover song identification and similarity: background, approaches, evaluation, and beyond". In: *Advances in Music Information Retrieval*. Springer, 2010, pp. 307–332.

[243] Joan Serrà, Álvaro Corral, Marián Boguñá, Martın Haro, and Josep Ll Arcos. "Measuring the evolution of contemporary western popular music". In: *Scientific reports* 2.1 (2012), pp. 1–6.

[244] Marc Serra-Peralta, Joan Serrà, and Álvaro Corral. *Heaps' Law and Vocabulary Richness in the History of Classical Music Harmony*. 2021. arXiv: `2104.04143 [cs.SD]`.

[245] Amanda Sewell. "A typology of sampling in hip-hop". PhD thesis. Indiana University, 2013.

[246] Bidisha Sharma, Rohan Kumar Das, and Haizhou Li. "On the Importance of Audio-Source Separation for Singer Identification in Polyphonic Music." In: *INTERSPEECH*. 2019, pp. 2020–2024.

[247] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. "An end-to-end neural network for polyphonic piano music transcription". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24.5 (2016), pp. 927–939.

[248] Umut Şimşekli and A Taylan Cemgil. "Score guided musical source separation using generalized coupled tensor factorization". In: *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*. IEEE. 2012, pp. 2639–2643.

[249] Umut Şimşekli, Jonathan Le Roux, and John R Hershey. "Hierarchical and coupled non-negative dynamical systems with application to audio modeling". In: *2013 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. IEEE. 2013, pp. 1–4.

[250] Olga Slizovskaia, Gloria Haro, and Emilia Gómez. "Conditioned Source Separation for Music Instrument Performances". In: *arXiv preprint arXiv:2004.03873* (2020).

[251] Olga Slizovskaia, Gloria Haro, and Emilia Gómez. "Conditioned Source Separation for Musical Instrument Performances". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 29 (2021), pp. 2083–2095. DOI: 10.1109/TASLP.2021.3082331.

[252] Paris Smaragdis. "Non-negative matrix factor deconvolution; extraction of multiple sound sources from monophonic inputs". In: *Independent Component Analysis and Blind Signal Separation*. Springer, 2004, pp. 494–499.

[253] Paris Smaragdis and Judith C Brown. "Non-negative matrix factorization for polyphonic music transcription". In: *Applications of Signal Processing to Audio and Acoustics, 2003 IEEE Workshop on*. IEEE. 2003, pp. 177–180.

[254] Paris Smaragdis and Gautham J Mysore. "Separation by "humming": User-guided sound extraction from monophonic mixtures". In: *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA'09. IEEE Workshop on*. IEEE. 2009, pp. 69–72.

[255] Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka. "A probabilistic latent variable model for acoustic modeling". In: *Advances in models for acoustic processing, NIPS* 148 (2006), pp. 8–1.

[256] Juliu O Smith III. *Spectral audio signal processing*. W3K publishing, 2011.

[257] Xuchen Song, Qiuqiang Kong, Xingjian Du, and Yuxuan Wang. "CatNet: Music source separation system with mix-audio augmentation". In: *arXiv preprint arXiv:2102.09966* (2021).

[258] Soundslide. *Soundslice*. https://www.soundslice.com/. Accessed 2021-04-13. 2021.

[259] Pablo Sprechmann, Pablo Cancela, and Guillermo Sapiro. "Gaussian mixture models for score-informed instrument separation". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2012, pp. 49–52.

[260] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[261] Ryan Stables, Jason Hockman, and Carl Southall. "Automatic Drum Transcription using Bi-directional Recurrent Neural Networks." In: dblp. 2016.

[262] Daniel Stoller, Sebastian Ewert, and Simon Dixon. "Jointly detecting and separating singing voice: A multi-task approach". In: *International Conference on Latent Variable Analysis and Signal Separation*. Springer. 2018, pp. 329–339.

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

[264] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. "The 2018 Signal Separation Evaluation Campaign". In: *Latent Variable Analysis and Signal Separation: 14th International Conference, LVA/ICA 2018, Surrey, UK*. 2018, pp. 293–305.

[265] Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. "Open-unmix-a reference implementation for music source separation". In: *Journal of Open Source Software* 4.41 (2019), p. 1667.

[266] Cem Subakan, Mirco Ravanelli, Samuele Cornell, Mirko Bronzi, and Jianyuan Zhong. "Attention is all you need in speech separation". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2021, pp. 21–25.

[267] Cees H Taal, Richard C Hendriks, Richard Heusdens, and Jesper Jensen. "An algorithm for intelligibility prediction of time–frequency weighted noisy speech". In: *IEEE Transactions on Audio, Speech, and Language Processing* 19.7 (2011), pp. 2125–2136.

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

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

[270] Keitaro Tanaka, Takayuki Nakatsuka, Ryo Nishikimi, Kazuyoshi Yoshii, and Shigeo Morishima. "Multi-instrument music transcription based on deep spherical clustering of spectrograms and pitchgrams". In: *International Society for Music Information Retrieval Conference (ISMIR), Montreal, Canada*. 2020.

[271] Jonathan David Tankel. "The Practice of Recording Music: Remixing as Recoding." In: *Journal of Communication* 40.3 (1990), pp. 34–46.

[272] John Thickstun, Zaid Harchaoui, and Sham M. Kakade. "Learning Features of Music from Scratch". In: *International Conference on Learning Representations (ICLR)*. 2017.

[273] Thilo Thiede, William C Treurniet, Roland Bitto, Christian Schmidmer, Thomas Sporer, John G Beerends, and Catherine Colomes. "PEAQ-The ITU standard for objective measurement of perceived audio quality". In: *Journal of the Audio Engineering Society* 48.1/2 (2000), pp. 3–29.

[274] Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. *Deep Complex Networks*. 2017. DOI: 10.48550/ARXIV.1705.09792. URL: https://arxiv.org/abs/1705.09792.

[275] John Tresch and Emily I Dolan. "Toward a new organology: instruments of music and science". In: *Osiris* 28.1 (2013), pp. 278–298.

[276] Rainer Typke. "Music retrieval based on melodic similarity". In: ASCI. 2007.

[277] George Tzanetakis and Perry Cook. "Musical genre classification of audio signals". In: *IEEE Transactions on speech and audio processing* 10.5 (2002), pp. 293–302.

[278] Efthymios Tzinis, Shrikant Venkataramani, Zhepei Wang, Cem Subakan, and Paris Smaragdis. "Two-step sound source separation: Training on learned latent targets". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 31–35.

[279] Efthymios Tzinis, Zhepei Wang, and Paris Smaragdis. "Sudo rm-rf: Efficient networks for universal audio source separation". In: *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE. 2020, pp. 1–6.

[280] Stefan Uhlich, Marcello Porcu, Franck Giron, Michael Enenkl, Thomas Kemp, Naoya Takahashi, and Yuki Mitsufuji. "Improving music source separation based on deep neural networks through data augmentation and network blending". In: *ICASSP*. 2017.

[281] Aäron Van Den Oord, Sander Dieleman, and Benjamin Schrauwen. "Deep content-based music recommendation". In: *Neural Information Processing Systems Conference (NIPS 2013)*. Vol. 26. Neural Information Processing Systems Foundation (NIPS). 2013.

[282] Laurens Van der Maaten and Geoffrey Hinton. "Visualizing data using t-SNE.". In: *Journal of machine learning research* 9.11 (2008).

[283] Emmanuel Vincent, Rémi Gribonval, and Cédric Févotte. "Performance measurement in blind audio source separation". In: *Audio, Speech, and Language Processing, IEEE Transactions on* 14.4 (2006), pp. 1462–1469.

[284] Emmanuel Vincent, Tuomas Virtanen, and Sharon Gannot. *Audio source separation and speech enhancement*. John Wiley & Sons, 2018.

[285] Tuomas Virtanen. "Monaural sound source separation by nonnegative matrix factorization with temporal continuity and sparseness criteria". In: *IEEE transactions on audio, speech, and language processing* 15.3 (2007), pp. 1066–1074.

[286] Anja Volk, Frans Wiering, and P Kranenburg. "Unfolding the potential of computational musicology". In: *Proceedings of the 13th International Conference on Informatics and Semiotics in Organisations*. 2011.

[287] Erich M Von Hornbostel and Curt Sachs. "Classification of musical instruments: Translated from the original German by Anthony Baines and Klaus P. Wachsmann". In: *The Galpin Society Journal* (1961), pp. 3–29.

[288] DeLiang Wang. "On ideal binary mask as the computational goal of auditory scene analysis". In: *Speech separation by humans and machines* (2005), pp. 181–197.

[289] DeLiang Wang and Jitong Chen. "Supervised speech separation based on deep learning: An overview". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.10 (2018), pp. 1702–1726.

[290] Siying Wang, Sebastian Ewert, and Simon Dixon. "Identifying missing and extra notes in piano recordings using score-informed dictionary learning". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.10 (2017), pp. 1877–1889.

[291] Zhong-Qiu Wang, Jonathan Le Roux, and John R Hershey. "Alternative Objective Functions for Deep Clustering". In: *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018.

[292] Zhong-Qiu Wang, Jonathan Le Roux, and John R. Hershey. "Alternative Objective Functions for Deep Clustering". In: *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Apr. 2018.

[293] Zhong-Qiu Wang, Jonathan Le Roux, DeLiang Wang, and John R Hershey. "End-to-end speech separation with unfolded iterative phase reconstruction". In: *arXiv preprint arXiv:1804.10204* (2018).

[294] Felix Weninger, Martin Wöllmer, and Björn Schuller. "Automatic assessment of singer traits in popular music: Gender, age, height and race". In: *Proc. 12th Intern. Society for Music Information Retrieval Conference, ISMIR 2011, Miami, FL, USA*. 2011.

[295] Felix J. Weninger, John R. Hershey, Jonathan Le Roux, and Björn Schuller. "Discriminatively Trained Recurrent Neural Networks for Single-Channel speech Separation". In: *GlobalSIP Machine Learning Applications in Speech Processing Symposium*. Dec. 2014.

[296] Catherine M Wessinger, John W VanMeter, Biao Tian, Jennifer Van Lare, Juraj Pekár, and Josef P Rauschecker. "Hierarchical organization of the human auditory cortex revealed by functional magnetic resonance imaging". In: *Journal of cognitive neuroscience* 13.1 (2001), pp. 1–7.

[297] Gordon Wichern, Joe Antognini, Michael Flynn, Licheng Richard Zhu, Emmett McQuinn, Dwight Crow, Ethan Manilow, and Jonathan Le Roux. "WHAM!: Extending Speech Separation to Noisy Environments". In: *INTERSPEECH 2019*. 2019.

[298] Gordon Wichern and Jonathan Le Roux. "Phase reconstruction with learned time-frequency representations for single-channel speech separation". In: *2018 16th International Workshop on Acoustic Signal Enhancement (IWAENC)*. IEEE. 2018, pp. 396–400.

[299] Thomas Wilmering, David Moffat, Alessia Milo, and Mark B. Sandler. "A History of Audio Effects". In: *Applied Sciences* 10.3 (2020). ISSN: 2076-3417. DOI: 10.3390/app10030791. URL: https://www.mdpi.com/2076-3417/10/3/791.

[300] John F Woodruff, Bryan Pardo, and Roger B Dannenberg. "Remixing Stereo Music with Score-Informed Source Separation." In: *ISMIR*. 2006, pp. 314–319.

[301] Chih-Wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, Meinard Müller, and Alexander Lerch. "A review of automatic drum transcription". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.9 (2018), pp. 1457–1483.

[302] Yusong Wu, Ethan Manilow, Yi Deng, Rigel Swavely, Kyle Kastner, Tim Cooijmans, Aaron Courville, Cheng-Zhi Anna Huang, and Jesse Engel. "MIDI-DDSP: Detailed Control of Musical Performance via Hierarchical Modeling". In: *arXiv preprint arXiv:2112.09312* (2021).

[303] Qingyang Xi, Rachel M Bittner, Johan Pauwels, Xuzhou Ye, and Juan Pablo Bello. "GuitarSet: A Dataset for Guitar Transcription." In: *ISMIR*. 2018, pp. 453–460.

[304] Yong Xu, Jun Du, Li-Rong Dai, and Chin-Hui Lee. "An Experimental Study on Speech Enhancement Based on Deep Neural Networks". In: *IEEE Signal Processing Letters* 21.1 (2014), pp. 65–68.

[305] Haici Yang, Shivani Firodiya, Nicholas J Bryan, and Minje Kim. "Don't Separate, Learn To Remix: End-To-End Neural Remixing With Joint Optimization". In: *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2022, pp. 116–120.

[306] Muqiao Yang, Martin Q Ma, Dongyu Li, Yao-Hung Hubert Tsai, and Ruslan Salakhutdinov. "Complex transformer: A framework for modeling complex-valued sequence". In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 4232–4236.

[307] Adrien Ycart and Emmanouil Benetos. "Polyphonic music sequence transduction with meter-constrained LSTM networks". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 386–390.

[308] Yousician. *Yousician*. `https://yousician.com/`. Accessed 2021-04-13. 2021.

[309] Simon Zagorski-Thomas. *The musicology of record production*. Cambridge University Press, 2014.

[310] Jerry Zolten. "The Beatles as recording artists". In: *The Cambridge Companion to the Beatles*. Cambridge University Press, 2009, pp. 33–62.