NORTHWESTERN UNIVERSITY

Bootstrapping the Learning Process for Computer Audition

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Prem Seetharaman

EVANSTON, ILLINOIS

September 2019

ABSTRACT

Bootstrapping the Learning Process for Computer Audition

Prem Seetharaman

The human auditory system is remarkable, easily parsing the complex mixtures of speech, music, and environmental sounds that we encounter every day. To appreciate the complexity of the task set before our ears, consider this thought experiment: imagine a crowded lake during summer. Boats are traveling across it in different directions and people are swimming and splashing around. All of this activity manifests as waves rippling across the surface of the lake. Two small channels of water extend out from the lake. The activity on the lake causes waves that reach the channels, causing them to swell and recede. Your task is to describe the activity taking place on the lake, such as how many boats there are, their direction of travel, what kind of boats, how many people there are, how far the boats are, etc. The catch is that you can only observe the two small channels extending from the lake. This seems impossible. Yet, the human auditory system deals with precisely this task all the time and does it with astounding efficacy. The lake of activity (the sounds around us) causes waves (sound pressure waves in air) that reach the two channels (our ears). We process these waves in real-time into an understanding of the auditory world around us.

Computer audition is the study of how machines can organize, parse, and understand the auditory "lake" of activity around us. A fundamental problem in computer audition is *audio source separation.* Source separation is the isolation of a specific sound (e.g. a single speaker) in a complex audio scene, like a cocktail party. Humans, as evidenced by our daily experience with sounds as well as empirical studies [8, 35, 114, 116, 156], manage the source separation task very effectively, attending to sources of interest in complex scenes. In this work, I create computational methods for audio source separation that are inspired by the abilities of humans. It is important to note that algorithms that are inspired by human abilities do not necessarily have to use the same underlying mechanics that humans use to perform these tasks.

Deep learning approaches are currently state-of-the-art for source separation tasks. They are typically trained on many examples (e.g. tens of thousands) where each source (e.g. a voice) was recorded in isolation and then the recordings are artificially mixed together. The mixture and the ground truth decomposition of the mixture are presented to the learner as the desired input/output examples. Although we have large collections of isolated speech, we don't have large collections of isolated recordings for every arbitrary sound. Artificial mixing also cannot capture the full range of sound source motions, reverberations, and sound radiation patterns that exist in real-world mixtures. This fundamentally limits the range of mixtures and sounds that existing models can learn to separate.

In contrast, humans are never given a decomposition of the complex auditory scenes they encounter, yet they develop broadly robust abilities to parse audio scenes. There is experimental evidence [8, 114, 116] that the brain uses fundamental grouping mechanisms, called primitives, to segment audio scenes even when they are composed of never-before-heard sounds. This, in turn allows learning about sound sources without the requirement to hear them in isolation.

The problems I am concerned with are the following: How can we learn audio source separation models directly from complex auditory scenes instead of from artificial scenes created to give us perfect ground truth? How can we leverage a combination of learned and primitive source separation models to produce robust source separation that performs well even when mixing conditions vary?

My approach for training a deep computer audition model without ground truth is as follows. I first apply primitive audio source separation algorithms to an auditory scene. These algorithms are inspired by human audition, such as our tendencies to group sources by spatial location, repeating vs not repeating, common fate, pitch and time proximity, and so on [8, 114, 116, 156, 160, 161]. Then, I estimate the confidence I have in the labels produced by the primitives. The goal is to focus the learning process for the deep network on time-frequency points whose labels are more likely to be correct. Finally, we train a network using the labels produced by the primitive audio source separation algorithms in conjunction with a modified deep clustering loss function that incorporates the concept of confidence weights. This training methodology makes the network focus on points that we are more confident in. The trained network can then be applied in situations where primitives fail. The result is a source separation model that can learn to segment the auditory scene without ground truth by bootstrapping its understanding of the auditory world using primitive unsupervised audio source separation algorithms.

Acknowledgments

First and foremost, I'd like to thank Bryan Pardo for being an incredible advisor. His tireless efforts to improve my scholarship and his remarkable ability to break literally any interface or algorithm I bring to him (seriously, it's a superpower) has resulted in years of fruitful collaboration. Through our interactions, I learned so much and found fresh perspectives on every problem we encountered together. I first learned of Bryan's lab as a freshman at Northwestern and was immediately hooked. To graduate all these years later (10, to be exact) with a PhD advised by Bryan is surreal. Thank you for getting me through this PhD, giving me the tools I needed, guiding me in how to do science, and finally, giving me so many lunches. Looking forward to our first lunch on me! And more papers, of course.

Thank you to Ken Forbus and Oliver Coissart, who make up the rest of my thesis committee. Thank you for taking the time out of your busy schedules to give me valuable feedback and your unique perspectives on my work.

Thanks to Zafar Rafii, Shashank Merchant, Bob Coover and Markus Cremer of Gracenote. Thanks to Gautham Mysore and Paris Smaragdis of Adobe. Finally, thanks to Gordon Wichern and Jonathan Le Roux of MERL. I asked them all so many dumb questions while I was interning with them and they responded admirably, resulting in very productive summers with each of them that I look back on with fondness. I look forward to future collaborations with each of you!

My parents - Balachandran and Jayanthi Seetharaman - are the only reason getting this PhD was even possible. For years, they sacrificed to put me through school, instill in me the value of knowledge and education, and encourage all of my extra-curricular interests, from childhood. Thanks for getting me my first instrument, my first computer, my first everything. You guys have been unbelievably encouraging and are amazing parents. I love you both.

I'd like to thank my sister, Deepa. You are the best sister anyone could ask for. My first editor and my first (intense) critic. Thank you so much. I'd like to thank my brother-in-law, Rolfe Winkler, for all the support in finishing this thesis. Late night phone calls with the two of you were invaluable to maintaining my sanity down the stretch. Finally, I have to thank little Yash Winkler, who has been a source of great joy in the past year for all of us! I'd like to thank my cousins Hayawardh Vijayakumar and Prammodh Vijayakumar. You guys have been so supportive. Hayawardh, as the other PhD in the family, thanks for the support (and all of the warnings!). Thanks to all of my extended family as well.

I'd like to thank my grandparents - K.N. and Sankari Venkataraman and N. and Sharada Balachandran. My grandfather on my Mom's side was a professor at University of Madras many years ago and passed before I was born. His memory is no small part of my finishing this thesis. My grandmother on my Mom's side is a pillar of my family, raising us kids, feeding us, and giving us everything we needed to succeed in life. My grandmother on my Dad's side raised me from a young age, and showered me with love and kindness. My grandfather on my Dad's side gave me my first mridangam, taught me all the stories of old, and showed me how to be kind and curious. Thank you all so much.

I'd like to thank my current and former lab-mates - Fatemeh Pishdadian, Ethan Manilow, Bongjun Kim, Max Morrison, Mark Cartwright, Zhiyao Duan, and Zafar Rafii. Our discussions have been invaluable and many of them led to work that I am really proud of. Thanks even more for the discussions that had nothing to do with work. Thanks for all the valuable mutual procrastination over the years. Thanks to all the undergraduates that I had the privilege of working with over the years: Julia Wilkins, Nathan Shelly, Taylor Zheng, Olivia Morales, Brian Margolis, Michael Donovan.

Thanks to all of the people in computer science that I've had the lovely experience of working next to for all these years. Maciej Swiech, Kyle Hale, Marcel Flores, John Rula for welcoming me into grad school (for better or worse) and being steadfast and amazing friends. Thanks to Irina Rabkina, Scott Cambo, Joe Blass, German Espinosa, Abhratanu Dutta, Nick Vincent, and Jamie Gorson for helping to transform the student experience in CS at Northwestern together. Thanks to Emily Harburg, Sarah D'Angelo, Michalis Mamakos, Simone Bianconi, Marc Warrior, Sam Taggart, Chandrasekhar Bhagavatula, Yongsung Kim, Robin Brewer, Daniel Rees Lewis, Bryan Head McMahan, Alison Wahl, Enrico Deiana, Ettore Trainiti, Vincent St-Amour, Jason Wilson, and Johes Bater for the friendship (and commiseration) over the years.

Thanks to Derren Gergle and Haoqi Zhang for serving on my qualifier committee and giving me great advice. Thanks to Stephen Tarzia and Peter Dinda for giving me my start in research as an undergrad. Thanks to my college friends, especially Madhav Suresh, Andrew Gomez, Zach Puller, Sudi Srivatsan, Anvesh Tanuku, Andy Zheng, Jack Hudson, Vijay Raghunathan, Emily Kesler, Katie Warnecke, for years of fun, idiocy, and support.

My work was made possible by funding support from the National Science Foundation (NSF) via grants 1420971, 1617497, 1116384, the Northwestern Data Science program, and the Center for Interdisciplinary Research in the Arts.

There are so many others that I've likely left out. So many people have provided support and kindness over the years and I am grateful for every one of you. Thank you!

Thanks to Suri Venkataraman and Uma Suri for providing a great refuge to work from and for all the support during the home stretch of this thesis. Finally, thanks to Priya Suri, my partner. Your support, love, and understanding is important to me beyond words and gave me the energy I needed to finish this thesis. Throughout all the ups and downs, you have been by my side, helping me to see the bright side of every problem I encountered. Your companionship is invaluable to me. Thank you.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

“ *The innate influences on [audition] should not be seen as being in opposition to principles of learning. The two must collaborate, the innate influences acting to ‘bootstrap’ the learning process.* ”

Albert Bregman, *Auditory Scene Analysis*, p. 40, 1994

The human auditory system is remarkable, easily parsing the complex mixtures of speech, music, and environmental sounds that we encounter every day[1]. To appreciate the complexity of the task set before our ears, consider this thought experiment, depicted in Figure 1.1: imagine a crowded lake during summer. Boats are traveling across it in different directions and people are swimming and splashing around. All of this activity manifests as waves rippling across the surface of the lake. Two small channels of water extend out from the lake. The activity on the lake causes waves that reach the channels, causing them to swell and recede. Your task is to describe the activity taking place on the lake, such as how many boats there are, their direction of travel, what kind of boats, how many people there are, how far the boats are, etc. The catch is that you can only observe the two small channels extending from the lake. Your task seems impossible. Yet, the human auditory system deals with precisely this task all the time and does it with astounding efficacy. The lake of activity (corresponding to the sounds around us) causes waves (sound pressure waves in air) that reach the two channels (our ears). We process these waves in real-time into an understanding of the auditory world around us.

Computer audition is the study of how machines can organize, parse, and understand the auditory “lake” of activity around us. A fundamental problem in computer audition is *audio source separation*, the act of isolating sound producing sources (or groups of sources) in an audio scene. Recordings of audio scenes with multiple overlapping sounds are often referred to as *mixtures*. Example source separation tasks include separating the lead vocals from a music mixture (vocal extraction - [141]), or separating a target speaker

[1]This chapter has supplementary audio examples: `https://pseeth.github.io/public/thesis/ch1.html`

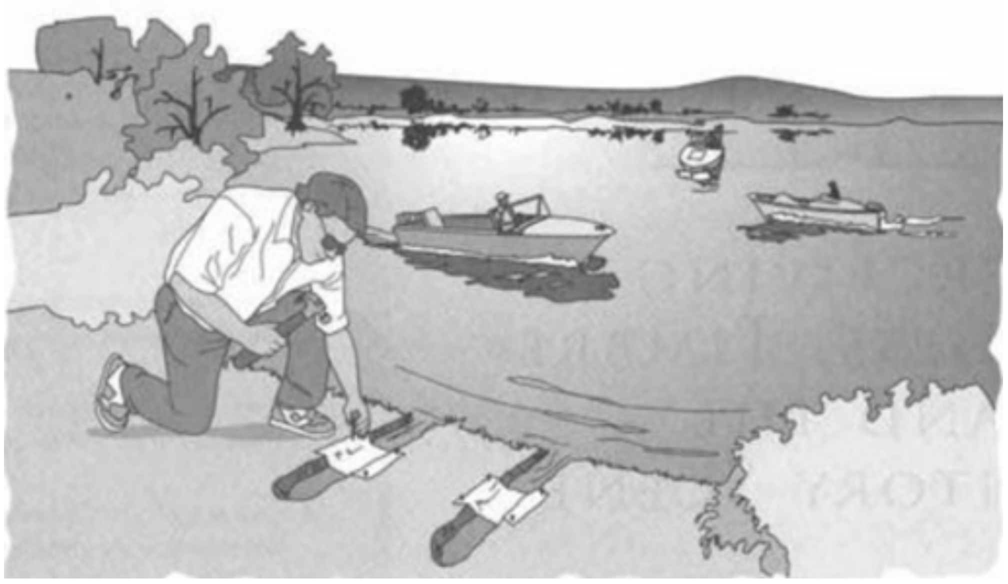

Figure 1.1. A thought experiment: by observing the activity of the two channels of water, can you describe the activity on the lake? Figure from Bregman [8]
.

from a mixture of people talking at a cocktail party [35]. In this work, the primary problem we will be dealing with is that of audio source separation.

1.1. Broader impact

Robust, reliable machine audio source separation would be transformative for many applications. Like visual segmentation for visual tasks (e.g. object classification - [89]), audio source separation is an enabling technology. It is useful for any auditory recognition or editing task where there are multiple concurrent sounds recorded on the same microphone that one might want to attend to or edit individually.

For example, we could create better hearing aids that increase the volume of only the sounds that matter in a mixture (e.g. the voice you are listening to) rather than superfluous sounds (the background chatter at a crowded party). Today's hearing-assistive devices cannot amplify only the desired sound, so they instead amplify all sounds in a frequency range or from a general direction. As a result, many hearing aid users simply avoid crowded public places, or tune out of a conversation they cannot follow. This technical limitation can translate to serious societal costs. Previous estimates place the number of individuals suffering from hearing loss at 30 million in the US alone [100]. When these individuals' needs are unmet by hearing-assistive technology, it can lead to loss of economic output and overall quality of life, and in extreme cases isolation and social alienation, both of which have serious health effects. A hearing aid that used state-of-the-art computer audition methods to amplify only the desired sound would be transformative for many people.

Improved audio source separation would improve speech recognition applications by eliminating unimportant sound sources from the audio (anything that isn't speech from the person of interest). We could also leverage knowledge of the constituent sources in an auditory scene to create more efficient algorithms for remixing (e.g. remixing stereo audio accompanying a movie into surround sound) ([127]). Algorithms to separate audio sources can be used to replace speech from vintage film soundtracks, where the speech and music are all on one channel (e.g. Casablanca), with dialog in another language (e.g. making a Tamil version of Casablanca). Source separation could also allow actors on a stage to be individually amplified without the need for individual microphones clipped to each person.

The Sounds of New York City (SONYC) [5] project and ShotSpotter [162] are examples of a wide range of initiatives to deploy automated listening stations to identify sounds (e.g. dog barks, gunshots) in our natural and built environments. Sound event identification technology [67, 206] is also transforming search through databases of audio recordings. These, too would benefit from source separation, as overlapped sounds are much more likely to be mislabeled.

More broadly, source separation algorithms can be applied anywhere where signals are comprised of multiple sources, such as biomedical imaging ([79]), seismology ([1]) and more recently, audio data sensed by personal assistants such as the Google Home and the Amazon Echo.

1.2. Problem statement

Source separation is the isolation of a specific sound (e.g. a single speaker) in a complex audio scene (hereafter called a mixture), like a cocktail party. Humans, as evidenced by our daily experience with sounds as well as empirical studies [8, 35, 114, 116, 156], manage the source separation task very effectively, attending to sources of interest in complex scenes. I want to create computational methods for audio source separation that are inspired by the abilities of humans. It is important to note that algorithms that are inspired by human abilities do not necessarily have to use the same underlying mechanics that humans use to perform these tasks. The goal is to create computational methods that have abilities comparable to humans rather than create computational methods that use the same underlying mechanics that humans use. An analogy would be airplanes, which do not flap their wings (hopefully) to fly (un)like birds do.

Setting up the source separation problem more formally, assume N audio sources, where a source is something like a person's voice, a car horn, or a cello. Each source is recorded in M channels (using M microphones). Each source $s_{n,m}(t)$ is represented as a sequence of amplitudes in time, indexed by which source it is n and which channel recorded the signal m. We do not observe the individual sources but rather the mixture x:

$$x_m(t) = \sum_{n}^{N} s_{n,m}(t)$$

$x(t)$ is an M channel mixture that contains the sum of the amplitudes of all of the sources at time step t. The audio source separation problem is this: Given an M-channel mixture of sounds $x_m(t)$, return the N isolated sound sources $s_{i,m}(t)$.

When M is greater than N, and the sources are statistically independent, the source separation problem is trivially solved by using independent component analysis (ICA) [95] for most sources of interest (e.g. speech, music, etc.). This is the *over-determined* source separation case. In this work, we are concerned with the far more difficult *under-determined* case - where there are fewer channels M than sources N. In these cases, ICA is not applicable. When $M < N$, the source separation problem becomes very difficult, almost like trying to take sugar out of a baked cake.

When audio is mixed together, the constituent sources can cancel each other out (either entirely or partly), making information about the sources unrecoverable given the mixture. For example, consider two single-channel sources $s_{0,0}$ and $s_{1,0}$, where $s_{1,0} = -s_{0,0}$. In this case, the mixture of these sources $x(t) = s_{0,0} + (-s_{0,0}) = 0$. Noise-canceling headphones leverage this phenomenon to cancel out sounds that are coming from outside the headphones (e.g. airplane noise), leaving only the audio the user wants to hear (e.g. music). The sounds coming from outside are recorded, multiplied by -1 and "added" to the mixture that reaches the listeners ears, canceling those sounds out. Effectively, this is source separation where the desired source signal (the source to be separated out, the content playing through the headphones) is known down to the individual samples taken $44,100$ times per second (for compact disc quality audio).

In general source separation problems, the exact sample-by-sample characteristics of neither the desired nor the undesired source are known. The audio mixture contains both the desired and undesired source with no additional information as to which is which. Further, when audio is mixed, the sources are overlaid on top of each other, resulting in both constructive and destructive interference. This is contrast to the

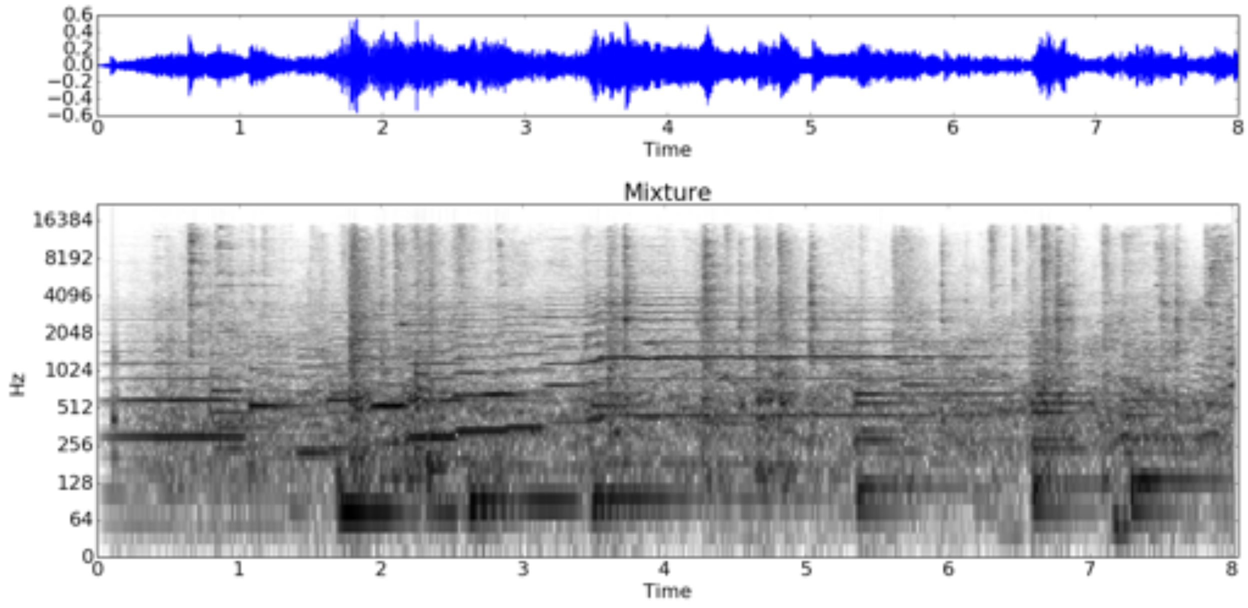

Figure 1.2. An example of audio representations. The top plot depicts the time-series representation, while the bottom plot depicts the spectrogram (time-frequency) representation. The auditory scene consists of a small jazz band (consisting of saxophone, piano, drums, and bass), chatter produced by a crowd of people, and dishes clattering as they are being washed. It is difficult to visually separate the three sources from each other in either the time-frequency representation or the original time-series.

analogous problem in vision of scene segmentation - identifying which pixels belong to which object in a visual scene. The source separation problem is like doing visual scene segmentation, if the visual world consisted of transparent/translucent objects that could possibly cancel each other out. The complexity of this task is apparent in Figure 1.4, where the top three panels that display isolated sources become difficult to visually separate in the mixture of the sources in the bottom panel.

For audio source separation, the audio must be encoded into a machine-readable representation. The first level of representation of audio is the time-series, which encodes audio as a sequence of real numbers (seen in Figure 1.2. These sequences often come in one channel (mono) or two channels (stereo). Each number in the time-series represents the amplitude of the signal at that moment in time (as measured by the displacement of the diaphragm of a microphone in response to the pressure waves in the air that make up sound). From the time series audio, more complex representations can be derived. The most commonly used representation is the *time-frequency representation*. This represents audio as a two dimensional matrix where the energy of a particular frequency at a specific moment in time is visible. The human inner ear performs a similar time-frequency decomposition using the cochlea [139]. Sound waves travel into the ear and along a vibrating membrane that varies in thickness. Along the membrane, hair cells send electrical signals to the brain depending on their displacement due to the sound wave. The membrane is sensitive in different locations to different frequencies because it varies in thickness. The brain uses this information to perform a frequency decomposition of the incoming sounds.

In computer audition, it is often useful to obtain a similar representation of the signal. In time-frequency representations, patterns that are not apparent in the time-series representation become salient. One such time-frequency representation and its associated time series is shown in Figure 1.2. In the top time-series representation, it is unclear what the content of the signal is. In the time-frequency representation, one can see (with practice) that there is a harmonic source (the rising harmonic stack visible from 2 seconds to 5 seconds), some broadband bursts periodically (the dishes clattering), and some noise in the background (the people chattering). The exact classes of these sounds may not be immediately apparent but clearly more information is visible than in the time-series representation. We can have more intuition about the behavior of time-frequency representations and more easily design audio source separation algorithms around them. The time-frequency representation is also known as the spectrogram. I use these two terms interchangeably.

An individual value in the spectrogram is called a *time-frequency point*. The time-frequency representation of an audio signal $x(t)$ is $X(t, f)$, where t and f indicate time and frequency. The value of $|X(t, f)|$, where $|.|$ indicates the absolute value, is the energy of the signal at the point in time and frequency. In complex transforms, which map a real number to the complex plane (like the Fourier transform), there is also a component of phase at each time-frequency point. One time-frequency representation of interest is the short-time Fourier transform (STFT) which, like its name suggests, is calculated using the Fourier transform [2]. The STFT is calculated by first splitting the time-series representation into overlapping windows that progress forward in time. Each window is then converted to the frequency domain using the Fourier transform. The magnitude and phase at each frequency for each window in time are then stacked vertically and become a column of the time-frequency representation. The time-frequency representations shown in Figure 1.4 and 1.2 are calculated using this approach. The images in these figures show magnitude spectrograms and do not display the phase information. This is the most common visual representation, as phase is difficult to interpret visually.

The STFT can be inverted back to the time-series representation by reversing the operations used to construct it (e.g. the Fourier transform is swapped with the inverse Fourier transform). It is important to have an inverse procedure associated with the time-frequency representation, as it is what allows one to recover audio to play back through speakers to listen to. The general procedure for source separation is to first map the time-series representation to a time-frequency representation. Then, one manipulates the time-frequency representation to achieve some effect (e.g. separating out vocals from accompaniment). After manipulation the representation, we can invert the manipulated representation back to a time-series that can be played back on computer speakers to hear the results.

Some approaches exist that manipulate learned representations of audio rather than time-frequency representations [107, 182, 183]. However, these models often learn representations that are highly domain-specific (e.g. trying to separate music using a representation learned on speech does not work). Therefore, the vast majority of source separation research has restricted itself to the analysis and manipulation of time-frequency representations, as a broad range of tasks is possible using these representations. However, even in learned representations, the idea is the same - one still manipulates the representation to separate the sources.

The most common way to perform audio source separation on underdetermined mixtures is to assign time-frequency points as belonging either in whole or in part to one of the sources. The former, where each time-frequency point is assigned in whole to one source, is referred to as a *hard mask*, while the latter, where a time-frequency point can be shared among multiple sources, is a *soft mask*. This mask is a two dimensional matrix M with the same dimensions as the time-frequency representation X. Elements of the mask are between 0 and 1. 0 indicates that the time-frequency point does not belong to the source being separated by the mask, while 1 indicates the opposite. Values between 0 and 1 indicate that the energy of the time-frequency point is shared between multiple sources. This is common in musical auditory scenes (e.g. an oboe and clarinet playing the same melody simultaneously), for example. To separate audio, we element-wise multiply the mask M with the time-frequency representation X: $S = M \odot X$. Then S, the separated source, can be mapped back to the audio domain using the appropriate inverse transform.

Much of the audio source separation literature is dedicated to how to assign time-frequency points in the spectrogram to each sound source (see Section 1.4). An example of such an assignment is shown in Figure 1.3. Throughout this thesis, we will be dealing with this problem - assigning time-frequency points in a spectrogram to meaningful sources. The complex audio mixture shown in Figure 1.2 can be separated into three sources - the jazz band, the chatter of the crowd, and the clattering of the dishes - using the time-frequency assignments shown in Figure 1.3.

I stated previously that I concentrate on solving the underdetermined (more sources than mixture channels) source separation problem. Most publicly available prerecorded audio (e.g. YouTube) and most

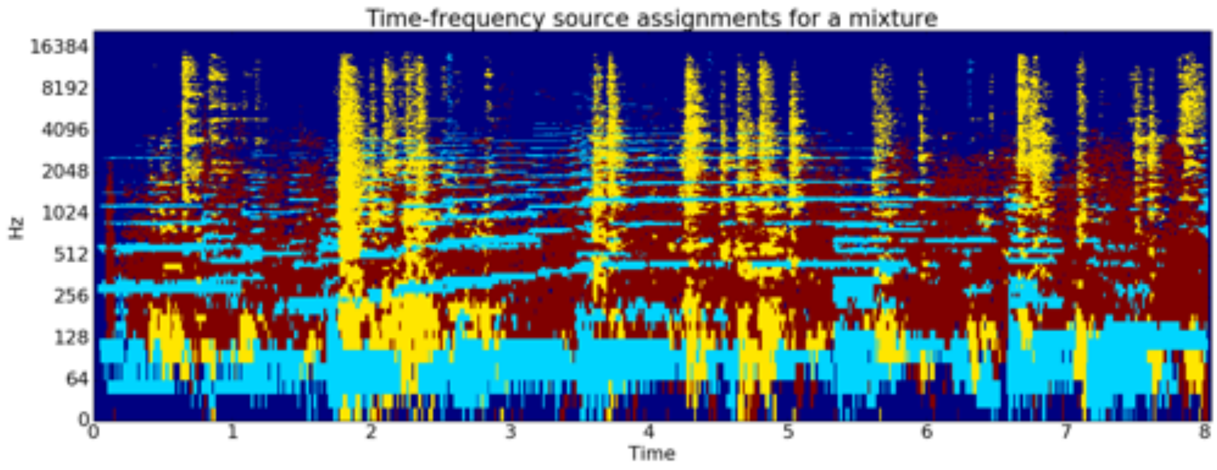

Figure 1.3. An example of time-frequency assignments in the spectrogram from 1.2. The auditory scene consists of a small jazz band consisting of saxophone, piano, drums, and bass, chatter produced by a crowd of people, and dishes clattering as they are being washed. Dark blue represents silence, light blue represents the jazz band, red the chatter, and yellow the dishes.

live recordings in real-world environments are under-determined, making it an important case to solve. Due to its wide applicability, a large body of research has developed on this topic [3, 4, 8, 9, 11, 13, 23, 26, 27, 30, 33, 36, 37, 40, 41, 43, 45–48, 51, 56–59, 62, 63, 65, 69–71, 73, 76, 81, 82, 88, 95, 101, 102, 107, 113, 116, 126, 128–131, 140–142, 144, 147, 148, 156, 164–166, 168, 170, 177, 178, 181, 184–188, 190–192, 200–202, 205]. Approaches can be broadly grouped into techniques that are not specific to audio (e.g independent component analysis [95]), techniques leveraging generic cues common in auditory scenes (e.g. repeating structure [116, 141]), and data-driven techniques that learn source models from training data (e.g. deep learning [65, 90, 104, 107, 123, 172, 179, 194]).

Data-driven techniques, specifically deep learning [53] models, are the most successful approaches to underdetermined source separation [103, 173] of one or two channels. This corresponds to the case that humans deal with - two ears means two channels. This is also the most common case encountered in real world mixtures (e.g. stereo music mixtures). In this work, we are explicitly interested in this case - separating auditory scenes given at most 2 channels of information. Source separation approaches that exploit more than 2 channels of information, such as beamforming with a microphone array[82, 136, 181] are not applicable to this problem, as they require far more microphones than are usually available for many audio recordings.

Deep learning approaches (e.g. the state-of-the-art deep clustering [65]) approach are typically trained on many examples (e.g. tens of thousands, or even millions) where each source (e.g. a voice) was recorded in

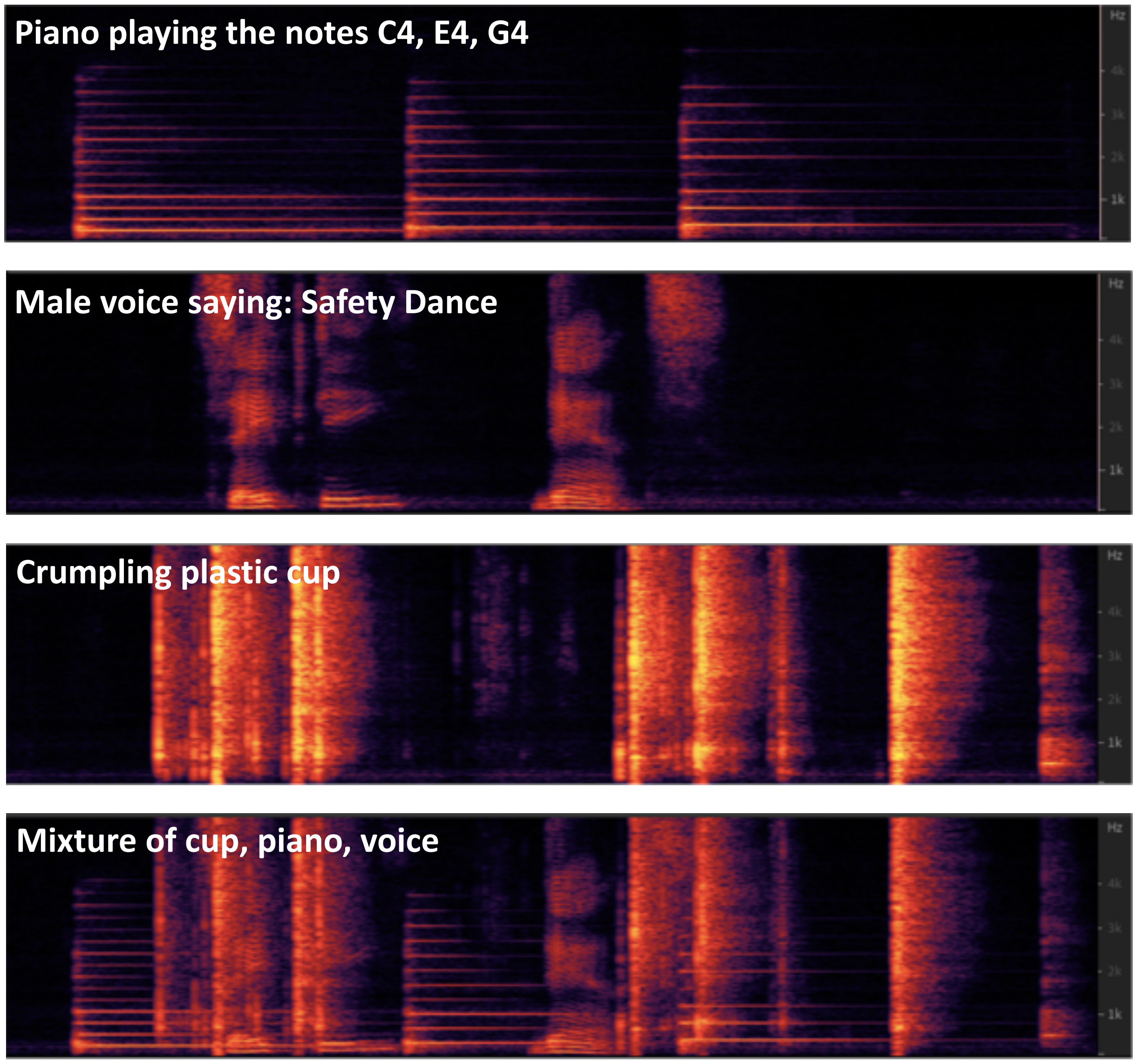

Figure 1.4. An example of audio representations and mixing sources. In the top image, a spectrogram for a single audio source consisting of a piano playing 3 notes is shown. In the second, an example of speech. In the third image, a crumpling plastic cup. Finally, the last image has the time-frequency representation of the three sources mixed together. It is very hard to disambiguate the three sources from one another in the image, especially in the times where the sources overlap in activity.

isolation and then the recordings are artificially mixed together. The mixture (e.g. Figure 1.2) and the ground truth decomposition of the mixture (e.g. Figure 1.3) are presented to the learner as the desired input/output examples. Although we have large collections of isolated speech, we don't have large collections of isolated recordings for every arbitrary sound. Artificial mixing also cannot capture the full range of sound source motions, reverberations, and sound radiation patterns that exist in real-world mixtures. This fundamentally limits the range of mixtures and sounds that existing models can learn to separate.

In contrast, humans are never given a decomposition of the complex auditory scenes they encounter, yet they develop broadly robust abilities to parse audio scenes. There is experimental evidence [8, 114, 116] that the brain uses fundamental grouping mechanisms, called primitives, to segment audio scenes even when they are composed of never-before-heard sounds. Examples include direction of arrival and harmonicity (see Section 1.4.1). This, in turn allows learning about sound sources without the requirement to hear them in isolation.

Both primitives and learning based approaches have many failure modes. Any of the primitives mentioned in Section 1.4.1 produce good results when the assumption they make about the auditory scene is valid. Any of the learning-based approaches can produce good results if the content or structure of the auditory scene is sufficiently similar to the conditions of the training mixtures used to build the model. When these assumptions are invalid, both primitives and learned models can fail.

The problems I am concerned with are the following: How can we learn audio source separation models directly from complex auditory scenes instead of from artificial scenes created to give us perfect ground truth? How can we leverage a combination of learned and primitive source separation models to produce robust source separation that performs well even when mixing conditions vary?

1.3. Contributions

This dissertation bridges the gap between learning-based and primitive-based source separation approaches such that the combination of the two mitigates the failures of either approach in isolation.

In this work I develop methods to bootstrap a learning-based source separation algorithm using the output of primitive-based source separation algorithms, eliminating the need for ground truth sources during training. In doing so, I significantly advance the state-of-the-art in computational auditory scene analysis. Specific advances include:

(C1) Algorithms that apply primitive cues known to be used in human audition that had not previously been used in algorithmic source separation. (Chapter 2). Specifically, I contribute a source separation algorithm based on a modulation-based representation that can be used for separation by frequency micromodulation or repetition.

(C2) A method to combine multiple primitive source separation algorithms in a way that out-performs any single primitive algorithm in isolation (Chapter 2).

(C3) Methods to estimate the efficacy of source separation algorithms without access to ground truth separated sources (Chapter 3).

(C4) Methods to bootstrap a learned source separation model from primitive cues without access to ground truth (Chapter 4).

(C5) A source separation system that embodies these advances to learn directly from audio mixtures without access to ground truth. This work is foundational to building a lifelong learner that will not require ground truth and can constantly train itself off and improve directly from new audio mixtures (Chapter 4).

Methods in this thesis are applicable to machine learning in general. It is well established that deep learning generalizes well when trained on large data sets consisting of thousands to millions of training examples that are cleanly annotated with ground truth labels [53]. These annotations are often costly to acquire. In this thesis, I propose methods for training deep learning models without ground truth. These methods could be applied to any perceptual domain where deep learning is useful, such as video, images, or language.

1.4. Related work

Source separation is broadly studied when many microphones are available (e.g beamforming [82, 136, 181]), or when mathematical assumptions about the signals or mixture can be made (e.g. low-rank, sparse, or statistical properties like non-Gaussian signals) [69, 95]. However, here, we restrict our overview to two specific types of approaches that are relevant to this dissertation: approaches that are loosely inspired by observations of human audition and those that are based on machine learning methods. The first category of approaches are modeled after what famed psychologist Albert Bregman [8] calls primitives - unlearned responses to auditory scenes that humans have. Again, it is important to note that the algorithms covered are not cognitive models. They are inspired by phenomena rather than mechanics of human audition or the exact specifics of human behavior. That is to say, the mechanics of these algorithms are not modeled after mechanics of human auditory systems but try to capture the same general principles that humans and other mammals are known to leverage in response to auditory stimuli.

1.4.1. Primitive-based source separation

Many source separation algorithms are based on primitives - innate grouping principles the human auditory system uses to parse auditory scenes. Evidence for these primitives playing a role in human perception of audio is strong [8]. Primitives are unlearned and exist from birth, and have been identified in multiple species [8]. In this section, we present auditory primitives along with the source separation algorithms that leverage the same cues inherent in the auditory scene that these primitives use.

1.4.1.1. Amplitude and frequency modulation. One way the human auditory system groups parts of the auditory scene is via correlations in amplitude and frequency change over time [8, p. 248]. Parts of the spectrum that exhibit correlated changes in loudness over time are grouped together as a single source. Similarly, parts of the spectrum are grouped into a source when they have correlated frequency changes. These two facts together make up the common fate primitive. Common fate refers to the fact that we group sounds that start, stop, and move together (this principle is at play in the visual system as well [8]). Since it is highly unlikely that unrelated parts of an auditory scene will behave in parallel with one another by accident, common fate is a good way to group the auditory scene. The common fate primitive is the foundation for several source separation algorithms.

Stöter et al. [172] considers the problem of separating musical sources that are overlapped in frequency (playing the same pitch, but with different modulation patterns). When presented with these mixtures, the human auditory system is capable of attending to one of the sources despite this frequency overlap by using common fate - leveraging the fact that the two instruments have different frequency modulation characteristics. Stöter et al. [172] uses a representation called the common fate model which groups parts of the spectrum according to frequency modulation characteristics. Separation is then performed using this representation to group sources via a parts-based decomposition. The multi-resolution common fate transform [137, 138] takes this further, creating a four-dimensional representation consisting of time, frequency, scale, and rate. The last two of these, scale and rate, correspond to features of common fate - modulation speeds along frequency and time, respectively. Neurons in the auditory cortex of mammals have been discovered empirically to be highly responsive to distinct scale-rate filters [49, 161], indicating that the auditory cortex actually derives these higher-order representations.

Wolf et al. [199] also implements common fate source separation using a technique adapted from the visual domain. In the visual domain, object segmentation is easier in video data than in static image data.

This is because an assumption can be made about the rigidity of the objects and how they move throughout the video. By applying an optical flow model, objects in the video can be segregated. Wolf et al. [199] create an analogous algorithm in audio by considering time-frequency *velocity*, or common fate.

The perception of singing voice and speech is heavily dependent on amplitude and frequency modulation. Chowning [20] discovered that when a complex tone - meaning a sine wave at a fundamental frequency combined additional sine waves with frequency at integer multiples of that fundamental frequency (e.g. 440 Hz, 880 Hz, 1320 Hz, etc.) - is coherently modulated in frequency, the human auditory system perceives it as a singing voice. This phenomenon is known as frequency micromodulation. It is used to make singing voice and speech synthesizers more realistic [18, 19]. Modulation-based representations have proven useful for speech and music processing and have been shown to be used explicitly by the human auditory system [34]

In Chapter 2, I present an algorithm that leverages the phenomenon of micromodulation for the purposes of singing voice separation from musical mixtures [158] by leveraging a modulation-based representation for source separation.

1.4.1.2. Repetition. Repetition has been shown to be an effective cue that humans use to parse the auditory scene. McDermott et al. [116] present the results of experiments that test the ability of humans to segregate previously unheard sources using only repetition as a primitive. In the experiment, mixtures are presented to a human participant in which a target source is presented repeatedly over many different distracting sources. Because the target source remains unchanged and is presented over a variety of distractors, the participants were able to reliably identify the target source as being in the mixtures. Kaernbach [80] and Warren et al. [197] show the limits of detection of repeating patterns by presenting listeners with periodic target sources juxtaposed with uncorrelated noise. The periodic target sources are noisy in nature. The target source is made longer and longer and the performance of listeners is recorded. Listener performance significantly degraded once the period exceeded 10-20 seconds, indicating that there is a buffer in the human auditory system of limited size.

Rafii et al. [141] use the repetition primitive to great effect for the task of machine audio source separation. In the REPET system, the repeating background model is formed by leveraging the statistical regularities of the periodic repetition of the target source. First, the periodicity of the target source is calculated by using the beat spectrum. Then, every iteration of the target source according to the period is layered on top of each other. Finally, the iterations are fused into a repeating background model by using an element-wise median of each time-frequency point in the iterations. The REPET algorithm responds to mixtures such as the ones used in the study by McDermott et al. [116] in the same way humans do. Rafii et al. [144], Liutkus et al. [101] and Huang et al. [69] all extend this idea of using statistical regularities of repeating patterns for source separation.

In Chapter 2, I show how the same modulation-based representation that encoded the frequency micro-modulation primitive also encodes the repetition primitive. The representation is processed in a different way such that it leverages repetition for singing voice separation [158].

1.4.1.3. Old plus new heuristic. "Old plus new" is a primitive coined by Bregman [8, p. 222] and refers to the idea that the order of presentation of auditory events affects how the human auditory system groups them. In "old plus new", a sound source is first presented to a listener. The listener acclimates to the first sound source, grouping corresponding parts of the spectrum to it using other primitives. Later in the mixture, it is joined by a different sound source. "Old plus new" states that the model that the listener builds for the first source also attempts to model the combination of the first source and the second source. Whatever the model cannot explain is then considered to be a new source. Alternatively, "look for a continuation of what went before and then pay attention to what has been added to it" [8, p. 224].

This idea is used extensively by composers, as it will reliably guide a listener's ears to areas of interest in the audio (e.g. presentation of a fugue subject in the exposition of a fugue or layering in instruments in a string quartet). Darwin [24] puts this idea to the test by presenting listeners with mixtures where a distractor source is presented first and then is later joined by a target source. The distractor consisted of a single tone at 500 Hz. The target consisted of a phoneme utterance with energy focused around 500 Hz. Because the tone was presented before the utterance, listeners were able to use the "old plus new" primitive to perceptually subtract the tone and attend to the phoneme. Winkler et al. [198] cast this primitive in a predictive coding light by suggesting that auditory objects are represented somehow in memory. These representations are used to model current auditory events. If the model fails due to the entrance of a new source that is not included in the model, the "old plus new" heuristic is used to start building a model for the new source. Though out of scope in this dissertation, I did create an algorithm that implements the "old plus new" primitive [157].

1.4.1.4. Harmonicity. Harmonicity is an important primitive for grouping parts of the spectrum. Many natural sounds that are harmonic in nature (e.g. a clarinet playing a note, a piano tone, a singing voice, a bird call) consist of a fundamental frequency as well as integer multiples of the fundamental frequency, called harmonics. These are visible in the top panel of Figure 1.4, which depicts the spectrogram of a piano chord. The human auditory system groups the fundamental frequency and its harmonics into a single stream [8, p .83]. This primitive forms the basis for many source separation algorithms. Duan et al. [31] uses the fact that sources will group together in integer multiples of a fundamental frequency to identify parts of the spectrum that belong to a single source. Similarly, Salamon et al. [154] estimates the predominant melody of a mixture by first looking for large peaks corresponding to a harmonic structure in the spectrogram. Bertin et al. [7] constrain a parts-based decomposition of the spectrogram by optimizing a loss function that corresponds to the discovery of harmonic sources in the spectrogram. However, harmonicity only explains how the sounds at a single instant in time are grouped. The grouping of multiple harmonic frames organized sequentially in time falls to the next primitive.

1.4.1.5. Frequency and temporal proximity. Parts of the spectra that are near each other in both frequency and time are grouped into a single source. This primitive is "peak streaming", in which time-frequency points are grouped by minimizing jumps in time and frequency [8, p. 137]. Salamon et al. [154] and Duan et al. [31] use this in conjunction with harmonicity to separate out sources in a mixture. Virtanen [188] and Li et al. [98] both implement singing voice separation systems by streaming together vocal contours, identified through predominant harmonic peaks, according to frequency and temporal proximity.

1.4.1.6. Timbral similarity. Timbral similarity is one primitive that humans use to group sounds emitted from the same instrument or voice as one source [8, p. 478]. Both Jaiswal et al. [78] and Spiertz et al. [169] group sources by timbre after performing a parts-based decomposition of the spectrogram. First, a basis set of recurring spectral templates is found by factorizing the spectrogram into the product of a template matrix and an activation matrix. The basis set is then clustered, using a machine learning approach (K-Nearest Neighbor, K-Means, or non-negative matrix factorization), according to timbral characteristics. These timbral characteristics are encoded by either using low mel-frequency cepstral coefficients[2], as in speech, or some other approach. The clusters are then used to recover the associated sources from the mixture.

One way to organize the human perception of timbre is along two dimensions - how harmonic a sound is versus how percussive a sound is [91]. Examples of harmonic sounds are a violin playing a *legato* tone (no attack), a police siren, or a singing voice. Examples of percussive sounds are a snare drum being hit,

[2]Mel-frequency ceptral coefficients, also known as MFCCs, are acoustic features that capture the timbre of a sound. They work by analyzing the rate of change across the frequencies, or spectrum. This is a “spectrum-of-a-spectrum” approach. The name “cepstral” comes from reversing the “spec” part of “spectrum”.

a gunshot, or a clap. Most sounds possess characteristics of both harmonic and percussive sounds. For example, a cellist that begins their bow stroke with a hard attack and then transitions to a smooth tone starts off as a percussive sound and transitions to a harmonic sound. Similarly, a singer singing the word "cash" will start with a percussive sound (the fricative of the "c"), transition to a harmonic sound ("a") and finish with a noisy percussive exist [26, 27, 42]. Harmonic/percussive source separation often does not result in distinct sources, but rather breaks apart a mixture into two sub-mixtures that can be passed along to downstream tasks. For example, the percussive component can be used for beat tracking in a music mixture [117], while the harmonic component could be used for melody extraction [154].

1.4.1.7. Direction of arrival. Parts of the spectra that are emitted from the same spatial location are grouped into a single source. While direction of arrival by itself has been shown to not be enough for reliable sound segregation by humans [156], it is a helpful primitive for parsing the auditory scene. It can also be used to ensure that streams are segregated by placing enough spatial distance between the sounds. Direction of arrival has formed the foundation for many source separation algorithms and is often used in conjunction with other primitives [128].

Rickard et al. [148] presented DUET, a system for the blind separation of speech by exploiting spatial information. In DUET, a 2D histogram is first obtained from the stereo mixture. One dimension of the histogram corresponds to differences in amplitude between the two channels (gain). The other dimension corresponds to differences in phase between the two channels (delay). Sounds emitted from that the same spatial location will have a consistent gain and delay. Sounds are then grouped together using the gain-delay histogram. Time-frequency points that correspond to a peak in the gain-delay histogram are grouped as a single source and separated from the mixture. Extensions of this method that use Gaussian Mixture Models on extracted spatial features have been developed for speech separation and music separation [84].

Fitzgerald et al. [44] presents PROJET, another method for exploiting spatial information in auditory scenes. The key idea in PROJET is to first calculate projections of the mixture along different angles. When these projections are subtracted from the original mixture, they will cancel out sources that exist in the original mixtures if the spatial angle of the source in the original mixture is orthogonal to the angle of the source in the projection. Sources not at the angle of the projection are left intact.

1.4.2. Learning-based source separation

Learning-based source separation approaches learn constraints about source behavior via a learning algorithm. There are two kinds of training regimes for this approach - those that learn directly from mixtures, and those that learn from isolated sources - recordings of sounds entirely by themselves with no interfering sounds.

State-of-the-art in source separation is dominated by learning-based approaches that are trained using isolated sources. For example, an isolated recording of a voice would have nothing in the background (e.g. no cars driving by outside, air conditioner units, background music, and so on). The isolated source data is used to train a model, which can consist of neural network trained via back-propagation [68], non-negative matrix factorization [21, 22, 60, 94, 96, 109, 125, 155, 163, 188, 189] or probabilistic latent component analysis [6, 10, 12, 29, 86, 121, 145, 167] trained using expectation-maximization update rules. The trained model can then be used to separate out the target source from mixtures later on.

All of these systems mentioned in the previous paragraph must have isolated source data for training. These isolated sources must then exist in the mixtures that the model is applied to. For example, one can train a model using non-negative matrix factorization on a set of recordings of whistling. The model would learn the various properties of a whistle - namely, the spectral characteristics (timbre, pitch range, overtone characteristics). Then given the trained whistle separator model, it can be applied to a mixture that contains a whistling sound. The model will then extract the whistling sound. Models trained in this way are very specific to the type of source they were trained on. A whistle separator will only be able to separate whistles, and even not all whistles, at that. A whistle separator trained on one whistle may not be able to separate a higher or lower pitched whistle. These models fail when the training data and testing data differ, sometimes even slightly. An example of this is shown in Figure 1.9 and discussed in Section 1.5.

In a parts-based decomposition approach, generally two sets of latent variables are learned: a basis set and an activation set. The basis set consists of recurring frequency patterns that are in the scene. These could be things like the timbre of repeating notes, drum hits, etc. The activation set indicates when different patterns in the basis set are active in the mixture. Techniques like non-negative matrix factorization, probabilistic latent component analysis, and deep auto-encoders [55, 85, 122, 176] find basis sets of "parts" directly from the audio mixture. By enforcing certain constraints on the two sets (e.g. sparsity), these approaches can discover distinct components of sources in the mixture without direct supervision. However,

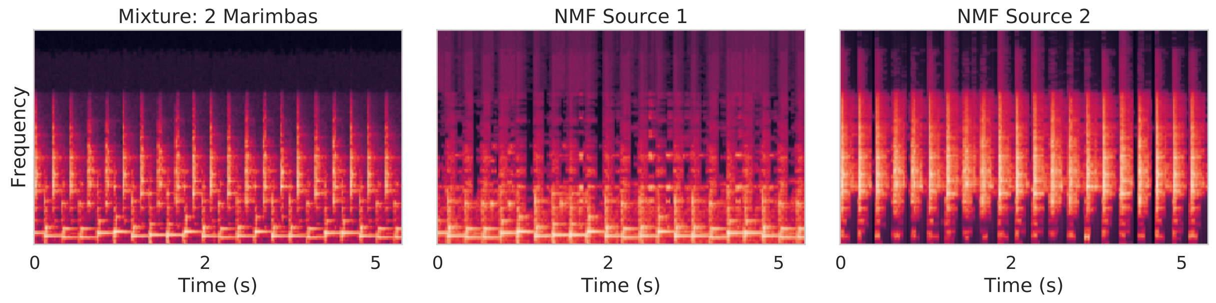

Figure 1.5. Non-negative matrix factorization (NMF) applied to a mixture of two marimbas playing simultaneously. The two marimbas have different timbres. One is played with a soft mallet, resulting in a more harmonic tone, while the other uses a hard mallet, resulting in a more percussive tone. Applying NMF to the mixture and clustering the basis set by timbre recovers the two sources successfully.

the grouping of components in the basis set is left to external methods (e.g. clustering by timbre [60, 78], user-guidance [9, 10, 165], or external information [30, 41]).

This approach works well when the elements in an auditory scene have different characteristics and are repetitive in nature. The inherent assumption that factorization methods make is that there is something in the scene that can be represented by a single pattern that is activated multiple times over the course of a mixture. For example, a snare drum is played in a similar way multiple times in a musical mixture. A parts-based decomposition would represent the snare hit with a single pattern, and track its activation over the course of the mixture.

In Figure 1.5, I show a mixture of two marimbas playing simultaneously, but with different timbres. One marimba is played with a hard mallet while the other is played with a soft mallet. The hard mallet results in sounds that are very percussive, represented as vertical lines in the mixture spectrogram. The soft mallet results in harmonic sounds, represented as the horizontal lines in the mixture spectrogram. A parts-based decomposition, here using NMF, is applied to the mixture. Then the basis set is clustered using timbre [78], recovering the two sources.

The assumption made here can be easily violated, causing the parts-based decomposition to fail. In Figure 1.6, I show a similar marimba mixture. However, in this mixture, the two sources have the same characteristics. Because of this, a parts-based decomposition fails to separate the sources. It instead discovers the “attack” and the “sustain” of each note. As can be seen clearly in the right-most spectrogram (NMF Source 2), the two sources are mixed still, resulting in a failed separation. One reason for this is that the optimization criterion of a parts-based decomposition approach is simply to reconstruct the mixture, not

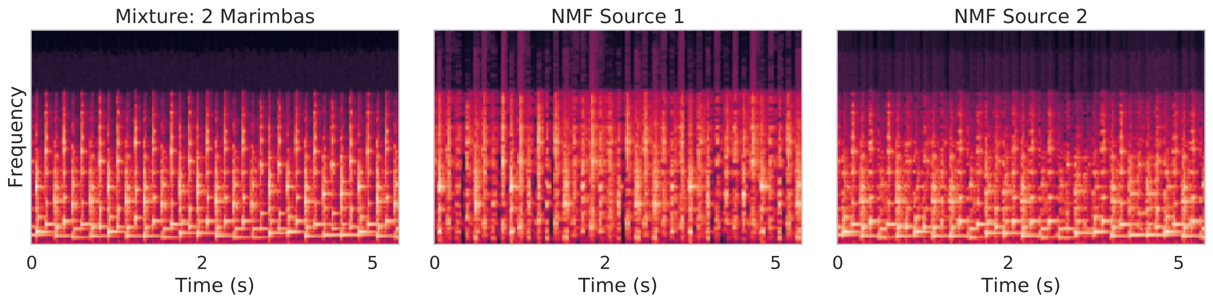

Figure 1.6. Non-negative matrix factorization (NMF) applied to a different mixture of two marimbas playing simultaneously. The two marimbas have the same timbre, but are separated in pitch. NMF fails to separate the two marimbas. The first source it discovers is merely the “attack” (the percussive start of each note), while the other source is the “sustain” (the harmonic horizontal component of each note). The two sources have similar characteristics, causing the parts-based decomposition approach to fail.

separate it. The under-determined nature of the source separation problem makes this approach unreliable in the general case.

Recent state-of-the-art learning-based source separation methods use supervised learning of deep neural networks. The way to train these networks is to construct them such that they takes the mixture as input and output a decomposition of that mixture into sources directly. These models require a mixture and a ground truth decomposition of that mixture for training. Because the ground truth decomposition is explicitly defined in the training process (unlike the parts-based decomposition), this approach can learn to separate sounds more effectively, even if they have similar timbre (e.g. separating two speakers talking simultaneously in a mixture [65]).

Supervised models are usually neural networks that are trained via the back-propagation algorithm [68]. Huang et al. [71] and Huang et al. [70] present such a system for separating singing voice from accompaniment. Uhlich et al. [179] presents a system in which many sources are separated from mixture using a multichannel deep neural network. Due to its success in other domains, deep learning has been broadly applied to source separation in recent years [16, 38, 39, 50, 70, 92, 106, 107, 123, 150, 159, 171, 182, 193, 207].

Hershey et al. [66] presented deep clustering, in which the ideal grouping of time-frequency points in the mixture spectrogram is used to train a neural network that maps time-frequency points in the mixture spectrogram to an embedding space that is learned by the network and used to solve the source separation problem. In this embedding space, every time-frequency point is represented by a D-dimensional point. Points in the embedding space that are near each other should belong to the same source. The loss function used to train the system enforces this. As training progresses, the system increasingly represents time-frequency points in an embedding space where points from one source occupy a different region than points from another source. Sources can be recovered from a good embedding space by applying a simple clustering technique (e.g. Gaussian Mixture Models). Points belonging to the same cluster are presumed to belong to the same source. Once a deep clustering network is trained from ground truth, sources can be recovered from a mixture by feeding the mixture spectrogram to the deep clustering model and using the clusters it outputs to build time-frequency masks. Luo et al. [105] expands on this work by simultaneously predicting the clusters as well as the target sources. Deep clustering is covered further in Section 1.6.3.1.

1.4.3. Combining primitives and learning

There has been some work in connecting in primitive and learning-based source separation approaches. In this section, we present past approaches that leverage aspects of both. McVicar et al. [118] presents a system in which conditional random fields are used as an agreement mechanism to combine the output of multiple singing voice separation algorithms. Some of these algorithms are primitive (such as REPET) and some are learning-based (such as the system by Huang et al. [70]). The output of each is fed into a conditional random field that is trained using the ideal binary mask. At testing, each algorithm is run and the output of the conditional random field is used to separate the audio. This system requires ground truth for training the conditional random field, unlike the work in this thesis. Driedger et al. [26] presents a system which uses multiple primitives by cascading primitive-based source separation algorithms on the mixture. The output of the last layer of separated sources are then combined to form singing voice and accompaniment estimates. However, this system failed to significantly out-perform any of the primitive source separation algorithms that went into the system. In Chapter 2, I present a separation algorithm that combines multiple primitive-based algorithms in such a way that it out-performs any of the primitive algorithms in isolation.

Kim [83] creates a system for speech enhancement in which a high-level learning-based model chooses from multiple simpler models for speech enhancement. Each of the simpler models corresponds to a straight-forward speech enhancement method (e.g. removing a high frequency tone, removing repetitive street noise). The learning-based model is a deep autoencoder that is trained on many samples of clean speech. In the final system, the output of the simple models are input to the autoencoder. The system then selects the model that results in the least reconstruction error according to the autoencoder. The autoencoder part of this system is trained using ground truth clean speech. The proposed work for this thesis will not require ground truth at any stage to learn models and perform source separation.

Both Ozerov et al. [128] and Liutkus et al. [101] present general frameworks for source separation where multiple primitive-based constraints can be placed on a parts-based decomposition of an audio scene. Ozerov et al. [128] allows for constraints to be encoded in a parts-based decomposition of the spectrogram via the loss function. These constraints can include spatial correspondence, harmonicity, temporal smoothness, etc. The specific constraints used are hand-crafted by the user of the system. The constraints must be specified for each source. For example, if the goal is vocal separation, one can specify temporal smoothness and harmonicity constraints. The parts-based decomposition (often non-negative matrix factorization) is then applied to the mixture. There can also be knowledge-based constraints, such as musical score information or source models derived through a learning algorithm. Liutkus et al. [101] allows for constraints to be encoded in the form of kernels, which group time-frequency points in the spectrogram according to a similarity function. These kernels can encode repetition, spectro-temporal characteristics [42, 204]. The general approach of specifying which constraints will apply for which source is not ideal as hand specifying constraints for every possible source is not feasible and must be determined by an educated end-user of the system. My system leverages general principles drawn from study of the human auditory system that do not need to be hand-balanced.

1.5. Limitations of existing work

Both primitives and learning based approaches have many failure modes. Any of the primitives mentioned in Section 1.4.1 produce good results when the assumption they make about the auditory scene is valid. Any of the learning-based approaches can produce good results if the content or structure of the auditory scene is sufficiently similar to the conditions of the training mixtures used to build the model. When these assumptions are invalid, both primitives and learned models can fail. These approaches are brittle, susceptible to characteristics of the auditory scene that they are applied to. An example of this is shown in Figure 1.7, where different primitive-based separation algorithms are run on different types of mixtures. When the assumptions made by the algorithm are valid, they work well, as shown in the first column. However, when the assumptions are invalid, the algorithm fails to recover the target sound, as shown in the second column. In the third column of Figure 1.7, we see that an algorithm based on direction-of-arrival worked where a repetition-based algorithm failed.

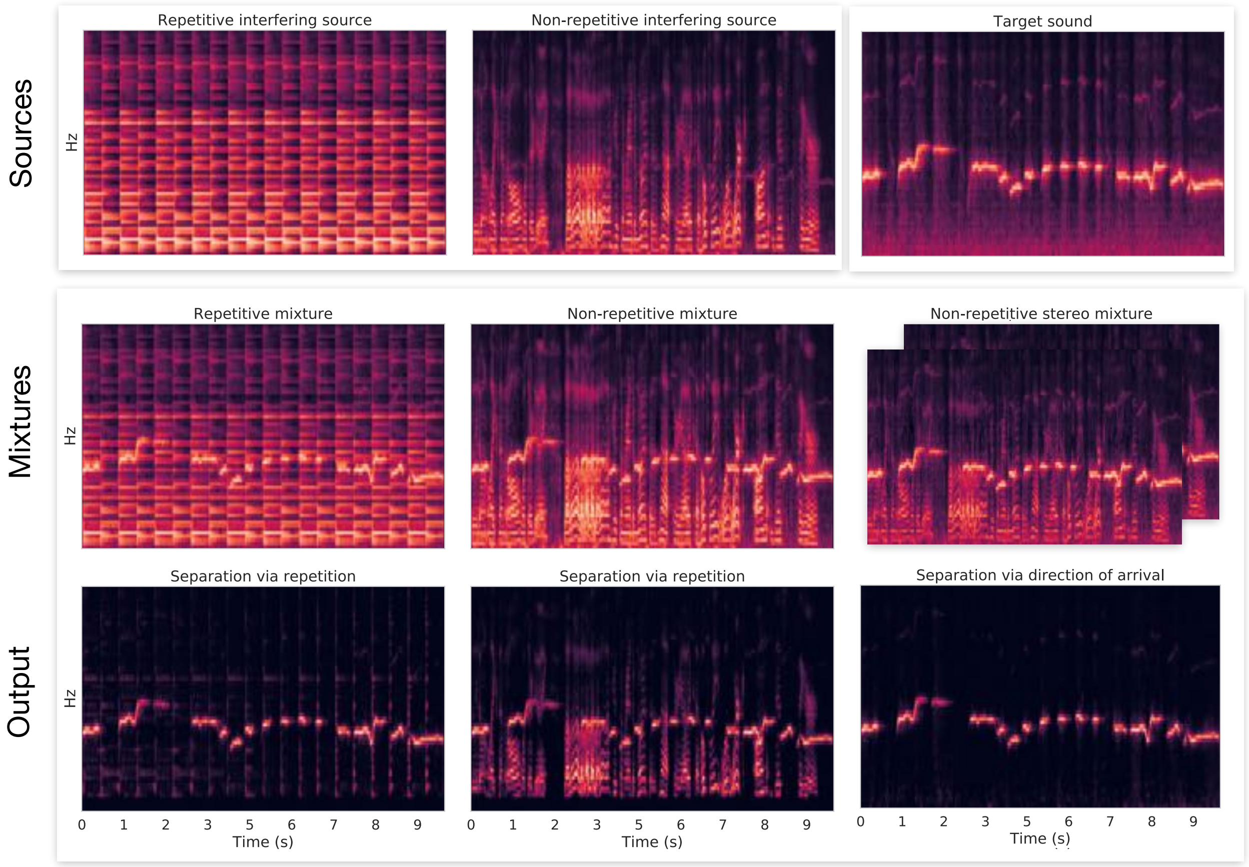

Figure 1.7. Output of primitive separation algorithms on different kinds of mixtures. In the top row, the constituent sources are shown. The goal is to extract the target sound, shown in the top right. When the target is mixed with repeating interference (first column), a repetition-based algorithm [141] works. When the target is mixed with non-repetitive interference (second column), the algorithm fails. Finally, a direction-of-arrival-based algorithm [147] (third column) works on the same non-repetitive mixture, as the two sources are spatially separated in the auditory scene.

Learning-based approaches are also highly dependent on characteristics of the mixtures that they were trained on. In Figure 1.9, I show the output of a deep learning model on different kinds of mixtures. The deep learning model is one trained using deep clustering [195] and achieves a state-of-the-art score for deep clustering on speech mixtures. This model is highly effective at separating mixtures of unknown speakers (the methodology behind deep clustering models is described in more detail in Section 1.6.3.1). The model is trained using mixtures of speech where the ground truth sources that went into the mixtures is known. Because it is trained on speech mixtures, it is highly effective at separating speaker mixtures, as can be seen in the first column of Figure 1.9 (the speech + speech column). Compare the recovered sources in the lower two rows in the first column to the ground truth sources in the top row of the figure. However, when

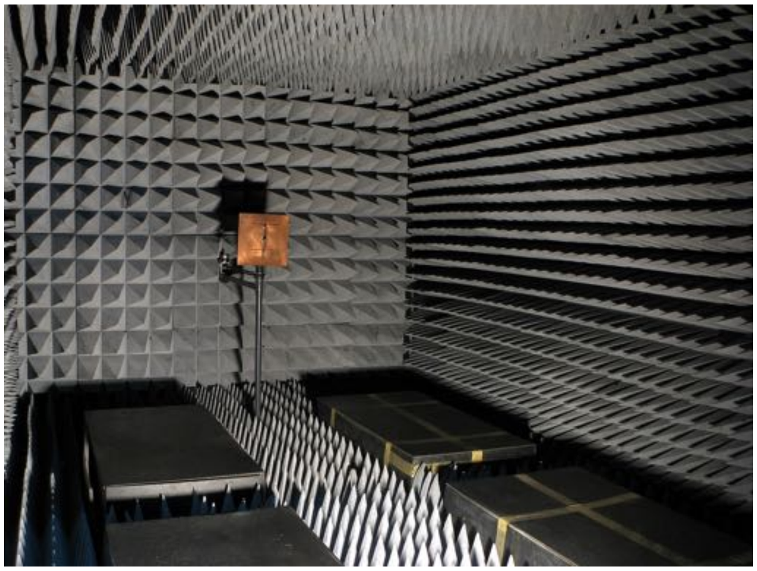

Figure 1.8. An anechoic chamber - the only place in the world where sounds have no echo and are closest to being heard in complete isolation.

presented with mixtures of a sort that it was not trained on, the deep clustering model fails completely. On the speech/whistle mixture and the speech/clap mixtures, the associated recovered sources (bottom two rows, right two columns) do not match the associated ground truth sources in the top row. The model was not taught using mixtures of whistling and clapping or whistling and speech. The nature of the learning algorithm is such that it cannot generalize to mixtures that contain different kinds of sources.

All state-of-the-art learning-based approaches leverage ground truth isolated sources to build training examples that are artificial mixtures made from those isolated sources to learn source separation models. Isolated sources are recordings of sounds entirely by themselves with no interfering sounds (cars driving by outside, air conditioner units, and so on). While we have isolated recordings for certain sound classes, like speech (made using resources like the chamber shown in Figure 1.8), we do not have a large set of isolated recordings for every possible sound class in the real world. Learning models then becomes very difficult for these under-represented sound classes. However, we can easily obtain recordings of many sounds “in the wild”, but not always in isolation. For example, learning music separation models requires “stems” - isolated recordings of each instrument in a piece of music (e.g. the drums, bass, guitar, vocals, etc). Stems are often only available to a limited number of people - usually the recording engineer, the artists, the label, etc. Very rarely are stems made available to source separation researchers.

Stem datasets for the purposes of training music source separation models are usually at most 50 hours in length. For comparison, the amount of data required to train a state-of-the-art speech recognition system is in the thousands of hours (at least for the only large-scale freely available dataset - Mozilla Voice[3]). Industry leaders likely have access to far more data for training. No such dataset exists for audio source separation. Further, the stem datasets that do exist are genre-limited (e.g. mostly rock and pop) and are designed for the separation of certain instruments (e.g. bass, drums, and vocals in a band, not violin I and II, viola and cello in a string quartet). However, the amount of publicly available (and purchasable) music recordings that exist in the world is orders of magnitude larger (over 50 million unique tracks on Spotify[4]). It would be desirable to have methods that learn directly from mixtures, as we could leverage a resource that is easy to access (recordings available on YouTube, Spotify, and so on). Further, these recordings capture a much wider range of recording environments, mixing conditions, source behaviors and source types that we could not account for in an artificial mixing environment as is currently used by learning-based source separation models.

This dissertation bridges the gap between learning-based and primitive-based source separation approaches such that the combination of the two mitigates the failures of either approach in isolation.

1.6. Approach

My approach to audio source separation bootstraps learning-based computer audition directly from mixtures using primitive source separation algorithms. My approach has four major steps:

(1) **Primitive source separation algorithms are applied to a mixture to obtain estimates of how each time-frequency point is assigned.** The primitives I consider in this work are: repetition [116, 141], direction of arrival [84, 156], time-frequency proximity [8, 154], harmonic/percussive timbre [8, 42] and frequency micro-modulation [8, 158]. The primitives can be either used by themselves (e.g. direction of arrival only) or ensembled via a method I introduce in Chapter 2 called primitive clustering.

(2) **The system estimates confidence in each primitive's assignment of time-frequency points to each source, and confidence in the the combination of the primitives.** This is done using unsupervised quality assessment that does not depend on knowing ground-truth separated sources.

(3) **A deep learning model is trained using the confidence-weighted output of the primitive separation algorithm(s).** The deep learning approach we select is deep clustering [65].

[3]https://voice.mozilla.org/en
[4]https://newsroom.spotify.com/company-info/

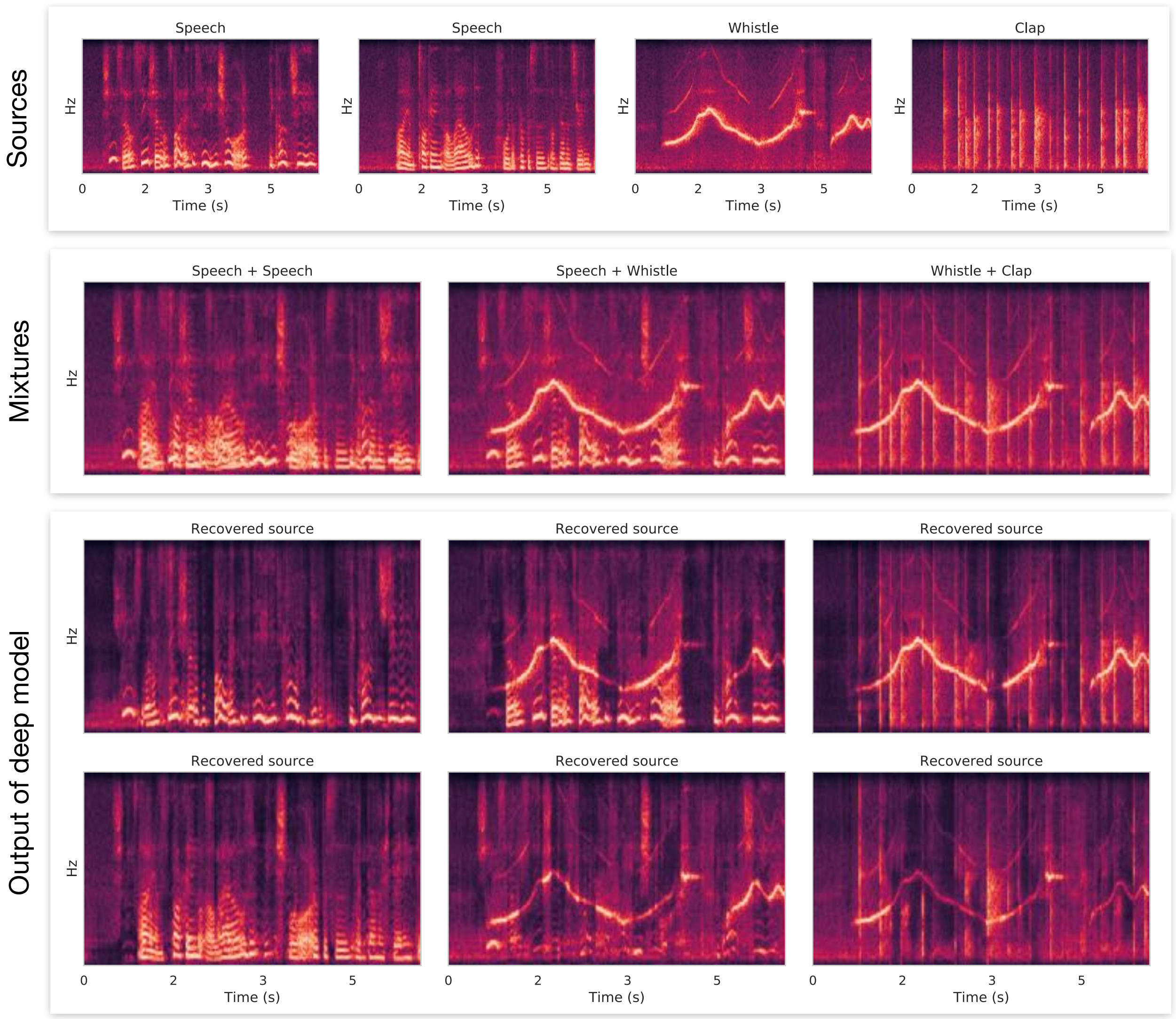

Figure 1.9. Output of a deep model on different mixtures. The model here is state-of-the-art and is trained to separate speech mixtures [195]. In the top row are the ground truth sources used to create the mixtures in the second row. They are recordings of speech, whistling, and clapping. On the speech + speech recording, the deep model performs well (first column), as it was trained for this task. However, on the other two mixtures, it fails to separate the scene into coherent sources, mixing up the whistle and speech (second column) and the whistle and clapping (third column).

(4) **Finally, the deep learning model is applied to separate new mixtures, either independently of the primitives, or in an ensemble with them.** The resulting learned model is often able to separate sources in situations where the primitives it learned from fail (Chapter 4).

To illustrate the approach, I will use a simple example throughout the remainder of this section. The mixture for the example is depicted in Figure 1.10. The setup is as follows: in the first part of the mixture (Situation 1), there is a drum and a whistle sounding simultaneously. Both sound sources are coming from

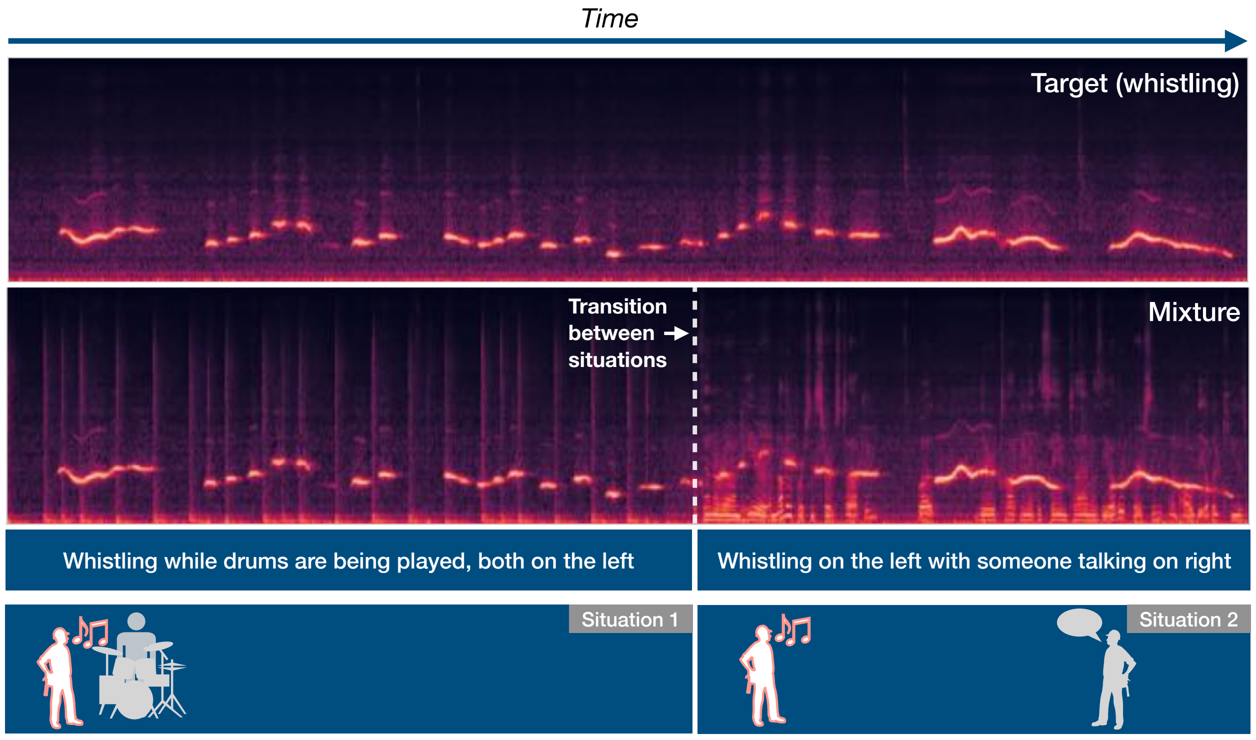
Time
Target (whistling)
Transition between situations
Mixture
Whistling while drums are being played, both on the left
Whistling on the left with someone talking on right
Situation 1
Situation 2

Figure 1.10. An auditory scene (second panel from top) consisting of whistling, drums and speech sounds. In the first part of the mixture, there is whistling simultaneous from drums, both spatially located to the left. In the second part, the whistling continues on the left while someone starts talking on the right. The top panel is a spectrogram of the whistling in isolation. The goal is to separate the whistle from the rest of the mixture.

the left. Halfway through the mixture, the drum ceases but someone starts talking (Situation 2). The speech is coming from the right, while the whistling continues on the left.

1.6.1. Separating via primitives

The first step is to separate the auditory scene using primitive source separation methods. The primitive source separation algorithms used in this work (which I have made available for use through the open-source library "nussl" [111]) are:

(1) Repeating background extraction - non-repeating elements are separated from repeating elements automatically [116]. Most often used to extract vocals from accompaniment in music. The REPET family of algorithms (including the original REPET, REPET-SIM, and Adaptive REPET) [141, 143] encode this primitive, as does an approach I present in Chapter 2 which leverages a modulation-based representation [158].

(2) Direction of arrival - elements of an auditory scene coming from different directions are considered separate sources by the humans [156]. This primitive is encoded in spatial clustering algorithms, which extract spatial features from every time-frequency point in a stereo spectrogram (e.g. inter-aural time of arrival and level differences). A clustering algorithm is then applied to spatial features to recover sources coming from different directions [84, 147]. In this work, we extract inter-aural phase and amplitude difference and apply a clustering algorithm to this feature space [120].

(3) Frequency micromodulation and common fate. Common fate states that elements of an auditory scene that are moving together (e.g. rising chirps versus falling chirps) are automatically grouped together by the auditory system [8, p. 248]. Frequency micromodulation states that elements of an auditory scene that are modulating in frequency are automatically grouped apart from elements that are not modulating. I implemented the 2DFT source separation algorithm, which encodes these primitives for the purposes of separating vocals from accompaniment using a 2D spectro-temporal modulation analysis of the spectrogram.

(4) Proximity in time and frequency - elements of an auditory scene that are close in pitch and time are grouped together by the human auditory system [8, p. 143]. Melodia [154] is one algorithm that is based on this and is used in this dissertation to separate singing vocals from background accompaniment.

(5) Harmonic/percussive separation - harmonic sounds are grouped apart from percussive sounds due to their difference in timbre by the human auditory system. Fitzgerald [42] implements this primitive.

To get intuition about the approach, let's pick one of these primitives and apply it to the mixture presented in Figure 1.10. A separation algorithm based on direction of arrival works as follows: given a stereo mixture (two channels), calculate two measures for every time-frequency bin in the left channel[5]: the *inter-phase difference* (IPD), and the *inter-level difference* (ILD). If a sound is coming from a specific location, it will result in a peak in the IPD-ILD feature space, as it will be louder in one channel than the other (ILD) and will arrive earlier in one channel than the other (IPD). If sounds are coming from multiple locations, you will observe multiple peaks in the IPD-ILD feature space. Separation of sounds then becomes a clustering problem in this feature space, which here has two dimensions. We can apply a clustering algorithm like K-Means to the IPD/ILD feature space. The cluster assignments are for each time-frequency bin and

[5]The features could be calculated for the right channel instead, but they would just be symmetric with respect to the left channel, so the left channel is chosen by convention.

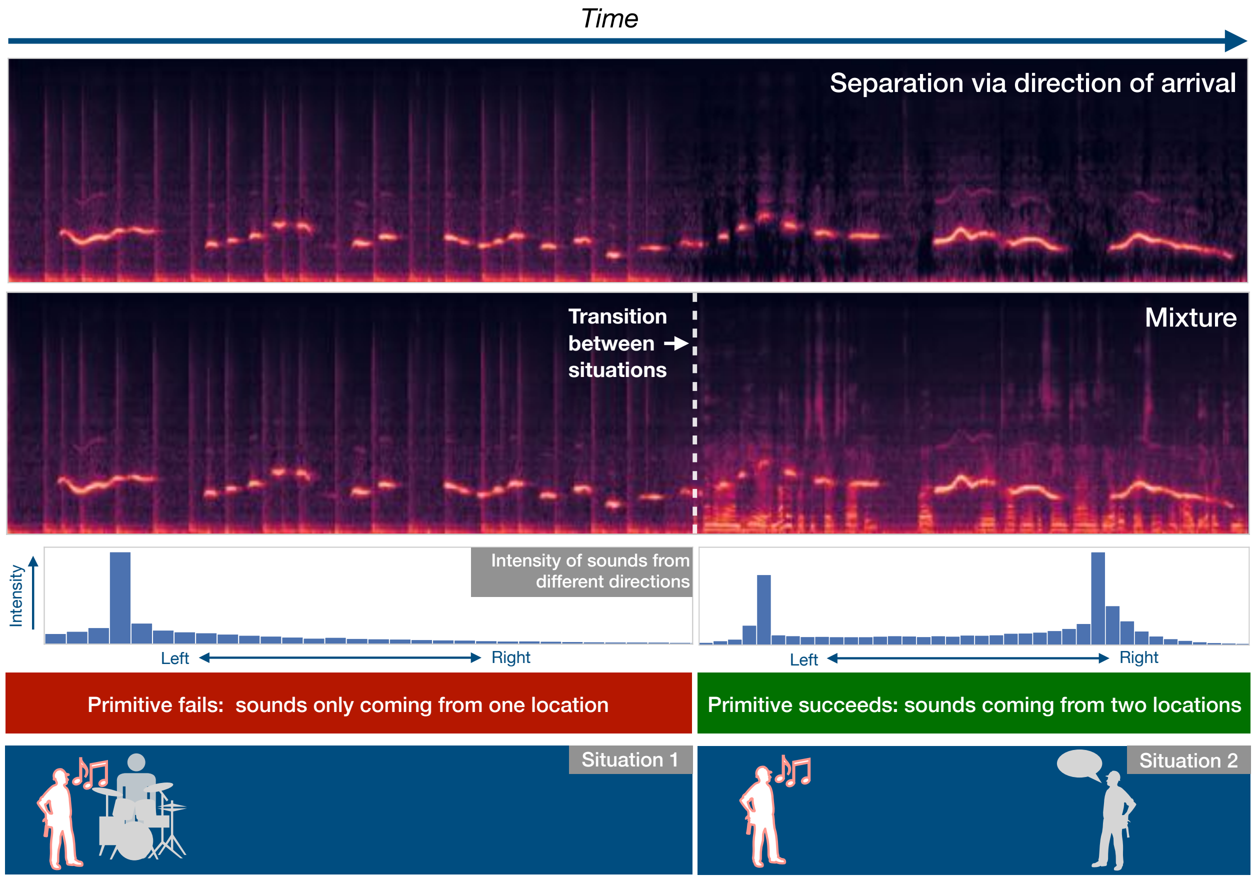

Figure 1.11. Separating the audio mixture using direction of arrival. The top panel shows the output of the algorithm. The second panel shows the mixture. The third row contains a visualization of the spatial features. In the first part of the mixture, the algorithm fails as sounds are coming from only one location. In the second part, the algorithm succeeds.

can then be used as time-frequency masks on the stereo mixture. The number of sources extracted from the mixture is the same as the number of clusters used in K-Means.

In Figure 1.11, we apply this approach to attempt to separate out the whistle from the interfering sounds (drums and speech) in the mixture. In the first part of the mixture, where the sounds are coming from the same location, direction of arrival fails to separate the mixture. The top panel of the figure shows that the drums and the whistle are mixed in the separation, as they are coming from the same place in the scene. A visualization in Figure 1.11 of the IPD/ILD features that are being used to separate the scene shows a single peak. However, in the second part of the mixture, direction of arrival has great success in separating the mixture, as the speech and whistle sounds are coming from multiple directions. In the visualization in Figure 1.11 of the IPD/ILD features, we see two peaks. One peak corresponds to the sounds coming from the whistle while the other is from the speech. By clustering in the space, the direction of arrival algorithm is able to separate the auditory scene.

While the direction of arrival algorithm is able to isolate the whistle in the second situation, it has failed in the first. However, other primitive separation algorithms may work in the first situation. In Figure 1.12, I show the output of other primitives on the mixture. Where direction of arrival confused the drums and the whistle, the other primitives - frequency micro-modulation [158] and time and pitch proximity [154] - have succeeded. Both of the spectrograms depicted are clear. However, in the second situation, the common fate primitive fails to isolate the speech from the whistle as both are sources that exhibit frequency modulation. The time and pitch proximity fails during the transition between the first situation and the second situation due to the time-frequency overlap between the speech and the whistle. Each primitive in isolation has made a mistake on some part of the mixture. The combination of the primitives, shown in the fourth row of Figure 1.12, succeeds in isolating the whistle from the entire recording. The combination is done here via a "winner-take-all" method where each time-frequency point is labeled using a majority vote between the three primitives. Where primitives fail in isolation, ensembles of primitives can succeed. I show three primitives in this example but more can be incorporated into the approach. I describe this ensemble approach in detail in Chapter 2.

1.6.2. Establishing confidence

The second step of my approach is to establish confidence in the time-frequency assignments produced by the primitives. In Chapter 3, I introduce a confidence measure that is based on cluster analysis. Specifically, the confidence measure works for any separation algorithm that is based on clustering, of which there are many. In clustering-based separation algorithms, each time-frequency point in a spectrogram is represented by a feature vector. The features might be direction-of-arrival, harmonicity, or learned features. A clustering algorithm such as K-Means is then applied to the feature space resulting in cluster assignments for each time-frequency point. The assignments are then used as a time-frequency mask on the mixture spectrogram that will be used to recover the individual sound sources. The direction of arrival primitive is one such clustering-based algorithm.

The confidence measure works by analyzing the feature space that is used for clustering and separating sources. In Figure 1.11, I applied the direction of arrival primitive to the mixture. The failure in the first half is because sounds are coming from the same direction rather than multiple directions. The visualization

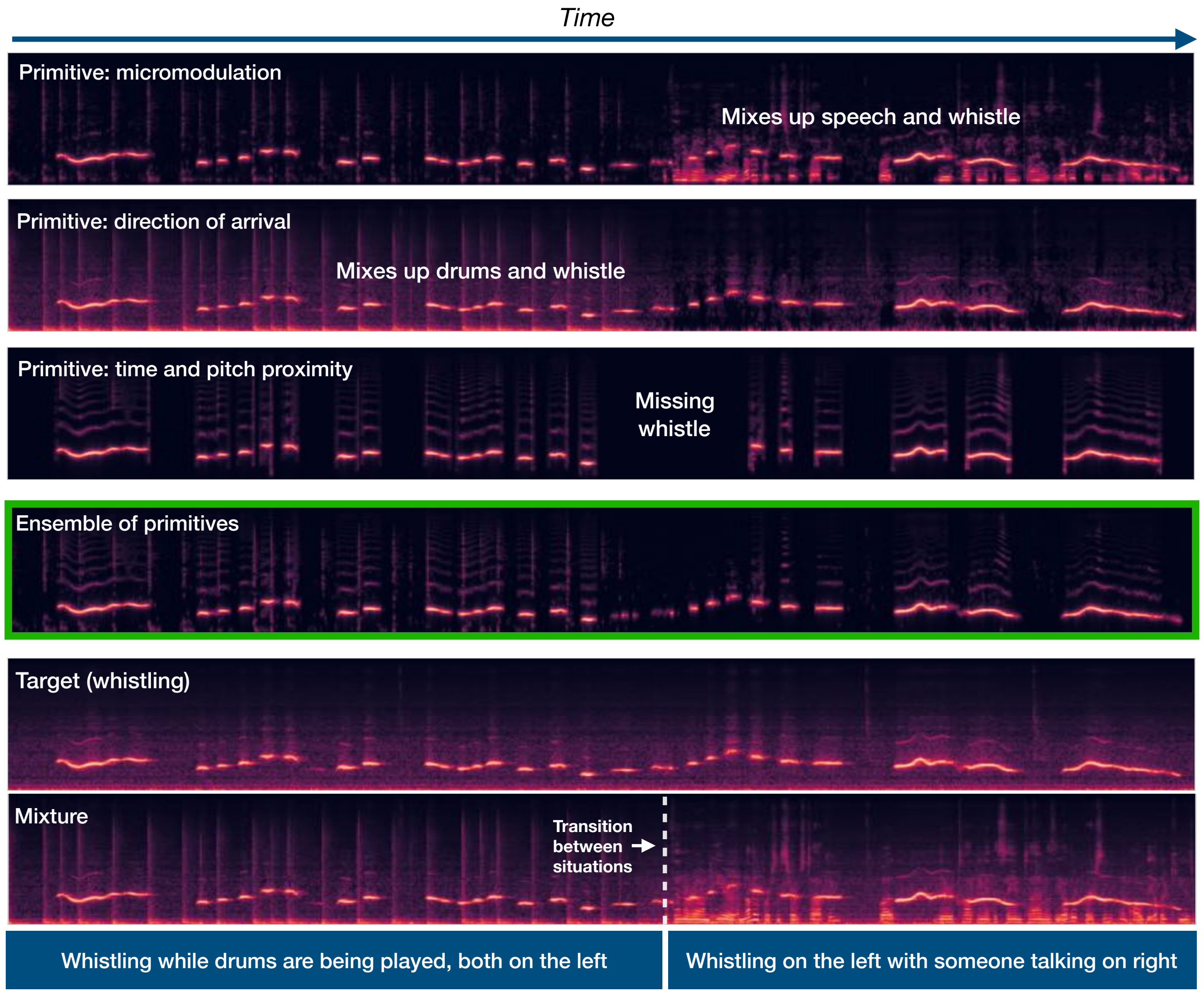

Figure 1.12. Output of primitive separation algorithms on the mixture. Each primitive has made a mistake on some part of the mixture, but the combination of the primitives (4th row) has succeeded in isolating the whistle over the course of the entire mixture, despite the conditions of the mixture changing.

of the features used for clustering show exactly one peak in the first situation and two peaks in the second situation. However, there are two sources in the scene (whistling and speech). This gives rise to our first confidence measure: the "clusterability" of the feature space used in the algorithm. In the first situation, there is only one cluster. Thus, the direction of arrival algorithm fails to separate the auditory scene at all. However, in the second situation, there are two natural well-separated clusters in the feature space, leading to separation into coherent sources (whistle vs speech). In Chapter 3, I describe a method that can capture the difference between these feature spaces automatically.

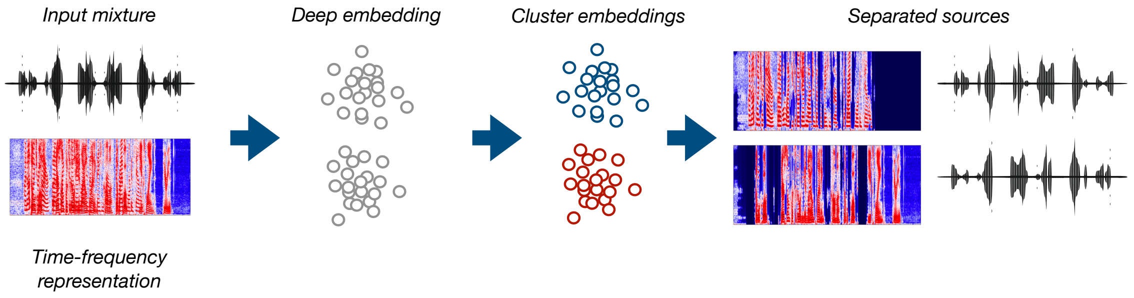

Figure 1.13. Deep clustering is a supervised deep source separation method that takes in an audio representation, maps every point in the representation to an embedding space, and then clusters the points in that embedding space. The cluster assignments are then used to mask the representation and produce separated sources.

1.6.3. Bootstrapping a deep model

The third step of my approach is to bootstrap a deep learning model from the output of primitives, rather than ground truth. Before I show how to apply my bootstrapping technique to the mixture in Figure 1.10, I first describe the learning algorithm used throughout this work - deep clustering.

1.6.3.1. Deep clustering. Deep clustering [66] is a deep neural network approach to source separation where the input to the neural network is a representation of the mixture audio (a time-frequency representation). It has achieved state-of-the-art results on many source separation tasks, such as singing voice separation [105] and speech separation [66, 75, 104]. The output of the deep clustering network, unlike other learning-based source separation methods [70, 71, 179], is not the source itself but rather an embedding space. It maps every point in the representation of the audio signal (e.g. every time-frequency point in the spectrogram) to a D-dimensional space. The key idea is to map each point into the embedding such that the points that belong to the same source are near each other in the embedding and points that belong to different sources are far away from each other in the embedding. The model is then trained with a loss function that encourages such an embedding. The details of deep clustering, most of which involve the construction of the loss function, now follow.

First, consider a matrix X which has dimensions T by F. In our case X contains the values of a magnitude spectrogram, like the one in Figure 1.4. F is the number of frequencies in the spectrogram, and T is the number of time frames. In all, X contains $T * F$ points. Each of these $T * F$ points is assigned to a single source (hard masking - like in Figure 1.3). We can encode all of these assignments by using a matrix Y where every row is a “one-hot” vector. Y has dimensions $(T * F)$ by N, where N is the number of sources in the mixture. For example, if the first point in X belongs to source i, then the first row of Y would be a vector with a 1 at the ith index and 0s everywhere else.

Now, the idea is to learn whether two points in our spectrogram X belong to the same source or different sources, rather than learn directly which source a point in X belongs to. This can be encoded by using the ideal affinity matrix A , which is constructed via $A = YY^T$, which has dimensions $(T * F)$ by $(T * F)$. If two points a and b in X belong to the same source, then $A(a, b) = 1$, and if they don't then $A(a, b) = 0$.

Deep clustering aims to learn an embedding function which, if handed a spectrogram of dimensions $T * F$, will return a representation of size $T * F$ by D, where D is the dimension of the embedding space. If X is the spectrogram, let V the D dimensional embedding of X. D can be set arbitrarily but is often set to some number between 10 and 60, depending on how complicated the system builder anticipates it will be to separate two sources. The aim is that the if two time-frequency points a and b in X belong to the same source, then their representations in the embedding space should put them close to each other. The similarity between two time-frequency points can be measured by taking an inner product between their embeddings.

Intuitively, we take every pair of time-frequency points and turn them into D dimensional embeddings. If two points belong to the same source, the dot product of the embedding of the two points should be high (they are near each other in the embedding space). If they belong to different sources, the dot product should be low (they are in different parts of the embedding space). The dot product between all pairs can be encoded by building an estimated affinity matrix $\hat{A}$, which is constructed via $\hat{A} = VV^T$. The value at each point in the estimated affinity matrix $\hat{A}$ contains the dot product of a pair of time-frequency points. We want the estimated affinity matrix $\hat{A}$ to be close to the ideal affinity matrix A, since this would mean that points belonging to the same source are very close in the embedding space and points belonging to different sources are in different parts of the embedding space.

Initially, the deep network produces random embeddings. To learn an embedding useful for source separation, one must train it using the cost function $C(\theta)$ for deep clustering. $C(\theta)$ is, with F indicating the Frobenius norm and θ indicating the parameters of model (e.g. weights of a neural network):

$$C(\theta) = |\hat{A} - A|_F^2 \tag{1.1}$$

$$= |VV^T - YY^T|_F^2 \tag{1.2}$$

Once a deep clustering network is trained, the sources are recovered by clustering the embedding space, using a clustering algorithm (e.g. K-Means). A depiction of this process can be seen in Figure 1.13.

A time-frequency representation is very large, usually considering hundreds of frequencies across hundreds of time frames. This can easily be on the order of a million points, making both VV^T and YY^T on the order of one million by one million which is much too large for optimization. By using the *trace trick* and some other handy tricks [135], we can reformulate the deep clustering loss as

$$L_{DC} = ||V^TV||_2^F - 2||V^TY||_2^F + ||Y^TY||_2^F \tag{1.3}$$

This formulation is far more tractable with matrix sizes of DxD, DxN and NxN, respectively for each term in the reformulated loss function.

The architecture of the neural network that produces the embedding can vary. Recent implementations of deep clustering [66] [105] have used an architecture with layers consisting of bidirectional long short-term memory units (LSTMs). This architecture will use the temporal context (both forward and backward in time) of each time-frequency point to produce the embedding. Other architectures, such as convolutional architectures, and other recurrent architectures, have also been investigated [119] for deep clustering. The focus of this work is not to investigate new architectures, but rather to find new ways to train existing architectures without ground truth.

Existing deep clustering work uses ground truth sources to construct the ideal affinity matrix required for training. This means the system builder must acquire a collection of sound sources recorded in isolation and then artificially mixed together. Having artificial mixtures allows one to know exactly which time-frequency points in the mixture correspond to source A and which correspond to source B. This allows the construction of the true affinity matrix, A, upon which this entire learning procedure depends. This also greatly limits the kind and amount of training data that can be used, since it requires training from artificial mixtures built in the studio from recordings of isolated sound sources.

1.6.3.2. Learning without ground truth. My approach does not require learning from artificial mixtures created by the system developer. Instead, an audio mixture time-series x, with an associated magnitude spectrogram representation X, is first decomposed using a primitive-based algorithm $\mathbb{P}$, as depicted in Figure 1.12. $\mathbb{P}(x)$ is a decomposition of x into constituent sources $[x_0, x_1, ..., x_{N_\mathbb{P}}]$ with associated magnitude spectrogram representations $[X_0, X_1, ..., X_{N_{\mathbb{P}}}]$ where $N_{\mathbb{P}}$ is the number of constituent sources produced by the primitive $\mathbb{P}$.

Recall that a time-frequency point is the energy of the sound mixture at a particular time and frequency. This maps onto a luminance value of a pixel in the image of the magnitude spectrogram of the sound. Given a time-frequency point $X(t, f)$, and the magnitude spectrogram representations of constituent sources in the audio mixture according to a primitive $\mathbb{P}$ $[X_0, X_1, ..., X_{N_p}]$, we decide which sound source the point belongs to with the following formula:

$$argmax_{t,f}([X_0(t, f), X_1(t, f), ..., X_{N_{\mathbb{P}}}(t, f)])$$

It is important to note that this is according to a source separation algorithm based off a primitive cue known to be used in human audition and not according to ground truth that results from building an artificial mixture from isolated sound sources. Each time-frequency point is exclusively assigned to a single source in a mixture according to each primitive. This produces an assignment for each time-frequency point in the representation according to that primitive. Using this information, we can construct the *primitive affinity matrix* $A_{\mathbb{P}}$. Whereas before, the deep clustering loss function was computed between the estimated affinity matrix $\hat{A}$ and the ideal affinity matrix A, we now train with the $\hat{A}$ and the primitive affinity matrix $A_{\mathbb{P}}$. We thus optimize:

$$C(\theta) = |\hat{A} - A_{\mathbb{P}}|_F^2 \tag{1.4}$$

$$= |VV^T - Y_{\mathbb{P}} Y_{\mathbb{P}}^T|_F^2 \tag{1.5}$$

where $Y_{\mathbb{P}}$ corresponds to the one-hot encoding of which source the primitive estimated a point belongs to for every point in the representation. I refer to this method of learning as *bootstrapping*, where a deep clustering is learned without ground truth but rather bootstrapped off the output of primitives. The training labels are a function of the primitives used to separate the mixture and both single primitive and multiple primitive cases are considered. The training methodology is depicted in Figure 1.14.

The key idea to bootstrapping is to incorporate the notion of *confidence* in the labels given by the primitives. The more confident we are in a set of time-frequency labels provided by a primitive, the more we want to train the deep clustering network off of those labels. We adapt a version of the original deep

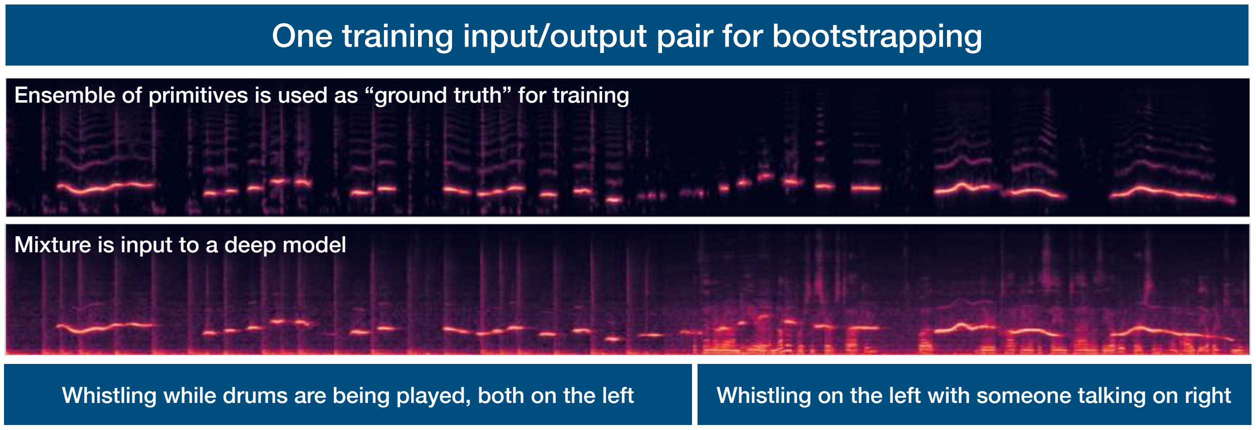

Figure 1.14. Bootstrapping a deep model from primitives. Instead of being trained with ground truth, the model is learned from the output of primitive source separation algorithms.

clustering objective [65] that was modified to include a weight w_i for each time-frequency point $i = (t, f)$ [195]:

$$\mathcal{L}_{\mathrm{DC},W}(V, Y) = \|W^{1/2}(VV^T - YY^T)W^{1/2}\|_F^2 \tag{1.6}$$

$$= \sum_{i,j} w_i w_j [\langle v_i, v_j\rangle - \langle y_i, y_j\rangle]^2, \tag{1.7}$$

where W is a diagonal matrix with the weights w on the diagonal. w_i is the weight for a time-frequency point. In prior work [195], the weights corresponds to loudness of the ith time-frequency point, forcing the network to focus on learning embeddings for louder time-frequency points during training. In this work, I introduce *confidence weighting* into the training process for deep clustering, where each point is labeled by our confidence in the label provided by the primitives. I use the confidence weights to guide the training process, making the network focus on points whose labels are more likely to be correct while down-weighting the importance of labels that are likely to be incorrect.

1.6.3.3. Example. In Figure 1.14, I show how the bootstrapping process works for the mixture shown in Figure 1.10. An ensemble of primitives is applied to the mixture generating “ground truth”. In this example, I use the clustering-based confidence measure to mediate the training process. For this example, this confidence measure reduces to tracking how much agreement there is for the label of each time-frequency point between the three primitives. Since there are only 3 primitives, and each primitive makes a binary decision, the confidence measure can take two values: 1.0 (everyone agrees) or $\frac{2}{3}$ (one primitive disagrees with the other two). This means that in the loss function in Equation 1.6, time-frequency points where all

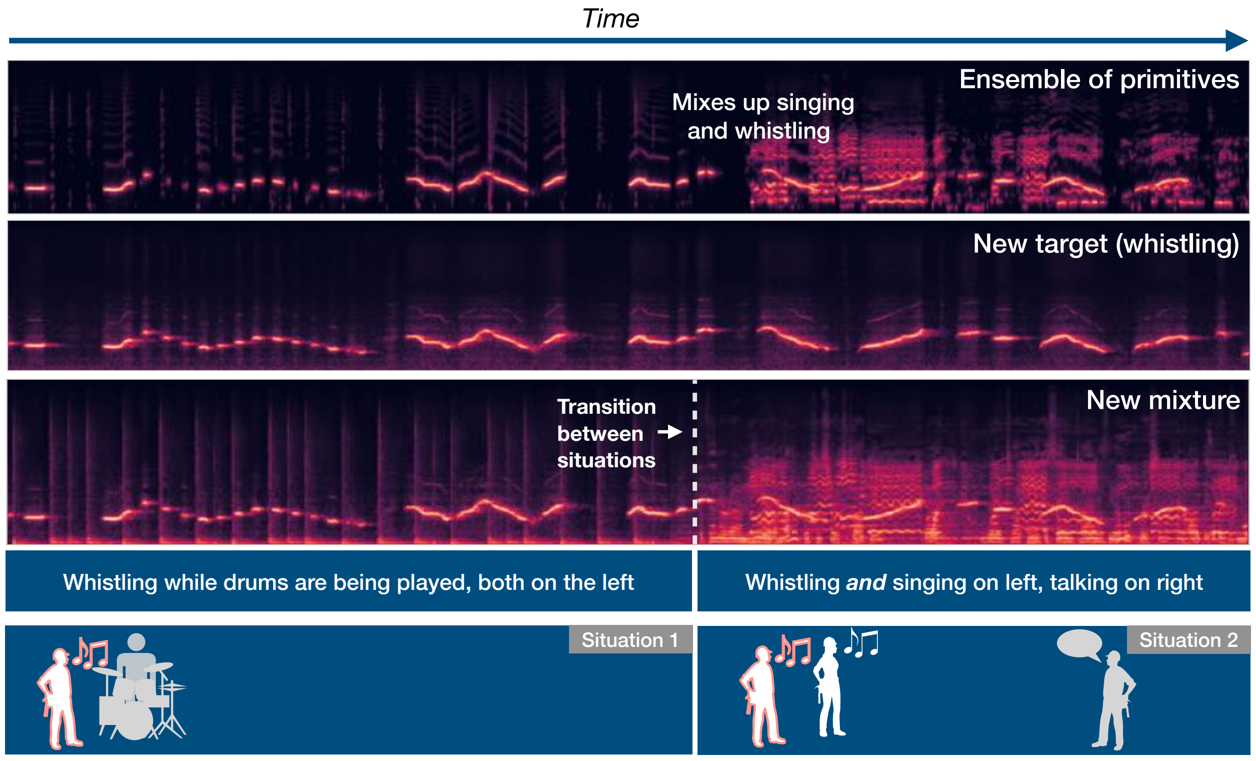
Time
Ensemble of primitives
Mixes up singing and whistling
New target (whistling)
Transition between situations
New mixture
Whistling while drums are being played, both on the left
Whistling *and* singing on left, talking on right
Situation 1
Situation 2

Figure 1.15. An auditory scene consisting of whistling, drums, speech, and singing sounds. The first part of the mixture is similar to the setup in the first part of the mixture in Figure 1.10. However, in the second part of the mixture, someone starts singing from the same location as the whistling, causing all of the primitives to fail. As a result, the ensemble of primitives (the top panel) fails to isolate the whistling sound.

primitives agree are worth 1.5x that of points where one primitive disagrees with the other two. This is a tunable ratio that we explore in this work. It can be tuned by raising the confidence measure to some power α prior to incorporating it into the loss. At an extreme setting of $\alpha = 10$, points where all the primitives agree are worth 57x that of points with disagreement in the loss function. I train the deep learning model using the confidence-mediated loss function so that it focuses its learning on time-frequency points that are likely to be correct (the primitives tend to agree on their assignment to a particular source).

1.6.4. Applying the model to new mixtures

Once the deep model is learned it can be applied on mixtures where the primitives it was trained from may fail. Figure 1.15 shows one such mixture. This mixture is similar to the one in Figure 1.10, with a few notable differences - the target whistle has changed - it is a different recording of whistling. The second situation is now different - simultaneously with the speech from the right and the whistling on the left is someone singing. The singing is coming from the same location as the whistling. This causes the direction

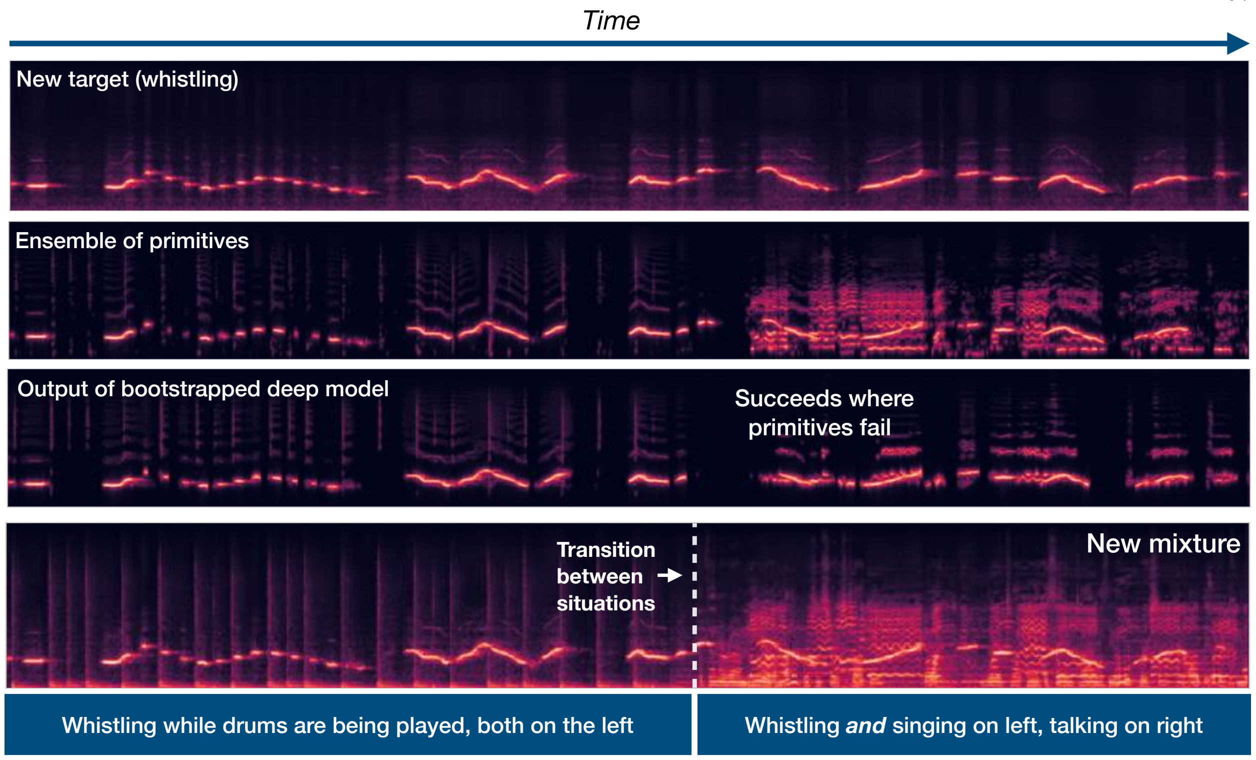

Figure 1.16. The output of the bootstrapped deep learning model on the new mixture. Unlike the ensemble of primitives that it was trained from, it is able to isolate the whistle in the second situation where the primitives have failed.

of arrival primitive to fail (as the locations of two sources are the same). The time and pitch proximity cue and the frequency micro-modulation cue also fail, as they both group the singing and the whistle together. Therefore, the ensemble of primitives, depicted in the top row of Figure 1.15, fails to isolate the whistle in the second situation.

The deep model can be applied to this mixture instead of the primitives, as shown in Figure 1.16. The model exhibits different behavior than the primitives it was trained from. In this case, it has learned to extract new whistle sounds from new mixtures, even in situations where the primitives cannot do this. The model depicted here was bootstrapped from a single training pair - the one depicted in Figure 1.14. This is unlike traditional deep learning approaches which require tens of thousands of mixtures with associated ground truth sources. In this specific case, it'd be arduous to obtain thousands of recordings of isolated whistles just to create a "WhistleNet". Instead, one could just download a set of recordings of whistling in any context (e.g. search "whistling" on YouTube and download a bunch of videos). Unfortunately, most of these recordings will not be people whistling in anechoic rooms for the purposes of training a separation algorithm. They will be people whistling over songs, in places with interfering sounds, and so on. The bootstrapping process I've described could be applied to train a WhistleNet. The same process could be applied for any sound for which recordings of any kind are available, circumventing the need for ground truth in the training process.

1.6.5. Summary of approach

To summarize: my approach for training a deep computer audition model without ground truth is as follows. I first apply primitive audio source separation algorithms to an auditory scene. These algorithms are inspired by human audition, such as our tendencies to group sources by spatial location, repeating vs not repeating, common fate, pitch and time proximity, and so on [8, 114, 116, 156, 160, 161]. Then, I establish the confidence I have in the labels produced by the primitives. The goal is to focus the learning process for the deep network on time-frequency points whose labels are more likely to be correct. Finally, we train a network using the labels produced by the primitive audio source separation algorithms in conjunction with a modified deep clustering loss function that incorporates the concept of confidence weights. This training methodology makes the network focus on points that we are more confident in. The trained network can then be applied in situations where primitives fail. The result is a source separation model that can learn to segment the auditory scene without ground truth by bootstrapping its understanding of the auditory world using primitive unsupervised audio source separation algorithms.

CHAPTER 2

Primitive source separation

In this chapter, I present my work on primitive source separation algorithms[1]. These algorithms are designed to separate sources from auditory scenes in an unsupervised fashion by leveraging assumptions about the auditory scene. For example, sources may be spatially separated, or exhibiting certain types of behavior (e.g. repeating vs not repeating) in an auditory scene. An algorithm can exploit this information to separate the sources. These assumptions are often based on observations of human audition. It has been found empirically, for example, that separation based on repetition is inherent to human audition [116]. This same assumption has been used as the basis for several source separation algorithms [101, 140, 141, 144]. In this chapter, I describe two contributions I have made to this branch of source separation:

(1) A source separation algorithm based on a modulation-based representation that can be used for separation by frequency micromodulation or repetition.
(2) A method to combine multiple primitive source separation algorithms in a way that out-performs any single primitive algorithm in isolation.

Before describing my original contributions, I now give an overview of another already existing primitive source separation algorithm that is used throughout this dissertation - separation via direction of arrival.

2.1. Separation via direction of arrival

First, it is helpful to ground the concepts of implement primitive audio source separation in a clear example. In typical human audition, the direction of arrival of sound sources is an important cue for source separation [156]. Sounds manifest as pressure waves through the air around us which reach our ears. Since we have two ears, if a sound is positioned to the left of us, it will reach our left ear before our right ear. By leveraging this observation, we can create a primitive source separation algorithm that imitates this.

To separate via direction of arrival, I use a simple blind source separation method that clusters time-frequency bins based on low-level features present in stereo mixtures. This method belongs to a well studied

[1]This chapter has supplementary audio examples: `https://pseeth.github.io/public/thesis/ch2.html`

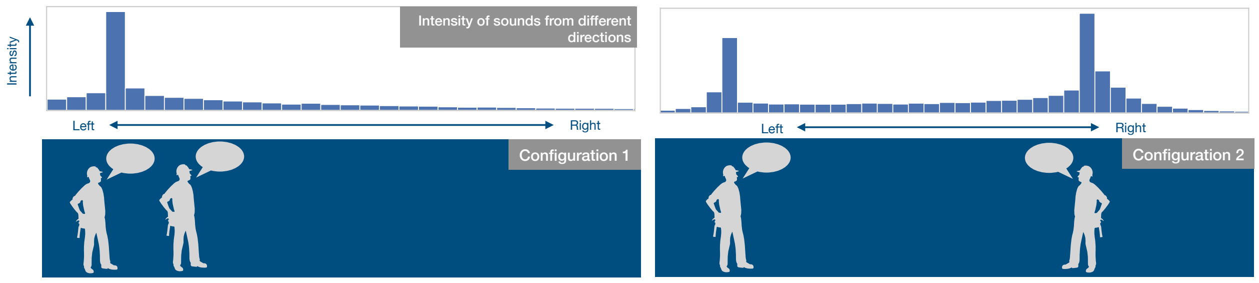

Figure 2.1. Two mixture configurations. *Configuration 1* is a mixture in which the two speakers are both coming from the left. *Configuration 2* is a mixture in which the two speakers are coming from different directions. The difference between the configurations can be seen in the direction of arrival features (top row) that are used to separate the mixture.

family of spatial source separation algorithms [187] such as DUET [84, 147]. The assumption is that time-frequency bins with similar spatial features likely come from the same direction, and that sounds coming from the same direction belong to the same source. If the sources are coming from distinct spatial locations, one will observe significant inter-channel difference, giving a good clustering and separation result. The key idea is to exploit differences between the two channels to decide which time-frequency bins go with which source. First, transform the input stereo audio to a stereo complex spectrogram $X_{t,f}^{(c)}$ where c is the channel, t the time index, and f the frequency index. Then, extract the inter-channel phase difference (IPD) θ and the inter-level difference (ILD) $X^{\log}$:

$$\theta_{t,f} = \angle\left(X_{t,f}^{(0)}\overline{X_{t,f}^{(1)}}\right), \tag{2.1}$$

$$\phi_{t,f} = \frac{20\log_{10}\left(|X_{t,f}^{(0)}|\right)}{20\log_{10}\left(|X_{t,f}^{(1)}|\right)}. \tag{2.2}$$

These two features form an embedding space that is clustered using K-Means to obtain separated sources. To bias the clustering towards bins with significant energy, weight the computation of the means using the magnitude of each time-frequency bin. The posterior assignments are used as masks on the complex spectrogram, using soft K-Means:

$$M_k(t,f) = \frac{e^{-\beta\mathcal{D}([\theta_{t,f},\phi_{t,f}],\mu_k)}}{\sum_j e^{-\beta\mathcal{D}([\theta_{t,f},\phi_{t,f}],\mu_j)}} \tag{2.3}$$

Consider the two mixture configurations shown in Figure 3.4. Both mixtures are of two people talking simultaneously. In the first mixture, the two speakers are coming from noticeably different direction - the angle between the speakers is very large. In the second mixture, the two speakers are coming from similar directions - the angle between them is small. We will apply a primitive separation algorithm to both of these mixtures that leverages direction of arrival.

This procedure is applied to the mixtures in Figure 3.4. A 1D visualization of the features is shown in Figure 3.4. In the mixture where the two speakers were located far from each other, there are two clear peaks in the corresponding visualization. In the other mixture, where the speakers were close, there is a single peak, meaning that separation via direction of arrival would not work well in this case. This algorithm shows how a simple observation about the human auditory system - sounds coming from different directions reach our ears at different times - can be used as inspiration for a primitive-based separation algorithm that uses that same cue.

I will now discuss how other observations about the cues used human audition can be used to develop source separation algorithms.

2.2. Separating via modulation-based representations

In this section, I present a source separation algorithm I have developed that works off of a modulation-based representation. Both repetition and micromodulation cues (see below) are used by humans to group stimuli in the auditory scene and they can be leveraged to separate auditory scenes using this algorithm. I illustrate the approach by showing its response on audio examples from Bregman [8] and [116]. I then apply the approach to a concrete source separation problem - separating vocals from music. Finally, I evaluate the approach using standard source separation metrics, showing that it has superior performance to competing unsupervised algorithms.

Chowning [20] describes an extraordinary phenomenon that occurs in the perception of the human singing voice. When a pure tone (e.g. 220 Hz sine wave) is played, it is perceived as a single sound, having a pitch. Overtones are additional sine waves with frequencies at integer multiples of that fundamental frequency (e.g. 440 Hz, 880 Hz, 1320 Hz, etc.). When overtones are added to the first sine tone, the combined set of sinewaves is perceived as a single sound, a complex tone similar to the sound of an organ, rather than multiple individual sounds. But when the sine tone and all of its overtones are modulated in frequency simultaneously by very small amounts, the tone is suddenly perceived as a singing voice. This auditory scene is depicted in Figure 2.2. This phenomenon is known as frequency micromodulation. It is often used to make voice synthesizers sound more realistic [18, 19].

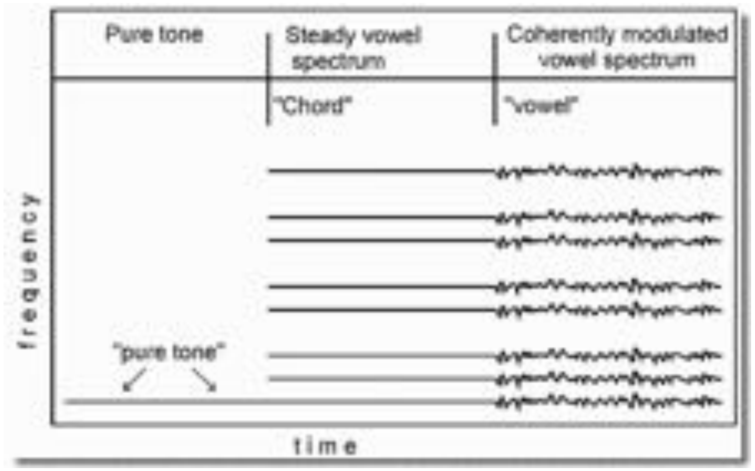

Figure 2.2. The auditory scene starts with a pure sine tone and then a sine tone plus overtones. When the entire stack of sine tones starts to modulate slightly in frequency, the perception is that someone has started singing. Figure from Bregman [8, p. 252].

Bregman [8] describes an auditory phenomenon that suggests that the human auditory system groups sounds by how they are moving. He calls this grouping principle *common fate.* The stimuli starts with a complex tone, which consists of a fundamental pure tone and overtones at integer multiples above that fundamental. Shortly after, half the overtones of the complex tone are modulated in frequency while the other set of overtones continues on without frequency modulation. The perception is that of hearing a voice (from the micromodulated tones) and an organ (from the non-modulated tones).

McDermott et al. [116] describes another auditory phenomenon - that of separating an auditory scene using repetition. They discovered that if in an auditory scene some sound is repetitive while an interfering sound is non-repetitive (like someone talking over the sound of a jackhammer), humans automatically perceive the repeating sound as being different from the interfering sound. To test this, they constructed mixtures of completely synthetic sounds where one sound was repeating and the other was not. After the mixture is played, the sound that was repeating is presented to a human by itself. They are asked to determine whether the repeating sound was present in the mixture. Remarkably, they found that, given enough repetitions, the listener was able to positively identify the sound. This held even when the sound was completely synthetic - never heard before by any human anywhere. We know this because the authors of the study synthesized the sounds themselves for the purposes of the study. The participants had thus never heard these synthesized sounds. Further, the sound was only presented with overlapping sounds that were also synthetic and never heard before by humans. Because of this, the subjects could not rely on past experience with sounds to parse the scene, yet they were still able to do so. This indicates that repetition is a core primitive that is leveraged by humans automatically.

Studies of the auditory system of mammals have revealed higher-order representations are used to parse the auditory world. As I alluded to in Chapter 1, the first stage of auditory processing is a time-frequency decomposition by the cochlea in the human ear [139]. From this time-frequency representation, it has been found [34, 115, 160, 161] that the auditory system extracts higher-order modulation-based representations. Through these representations, the auditory system can represent more complex sounds like a percussive hit, or a rising or falling tone. Where a time-frequency representation only encodes the magnitude and phase of a signal at a specific time, a modulation-based representation encodes how the sound changes and evolves over time.

Using this knowledge as inspiration, I now present an algorithm that is loosely inspired by these facts about the auditory system. It does not attempt to model the underlying mechanics of the auditory cortex (unlike other approaches [17, 108, 133, 152, 160, 161]), but rather attempts to model the phenomenon (e.g. repetition, micromodulation, common fate) itself. My algorithm instead leverages simple techniques from image processing.

2.2.1. The 2D Fourier Transform

The 2D Fourier Transform (2DFT) is an essential tool for image processing, just as the 1D Fourier Transform is essential to audio signal processing. The 2DFT decomposes images into a summation of weighted and phase-shifted 2D sinusoids [153]. The core idea behind this approach is to process the 2DFT of a time-frequency representation. I apply a 2DFT to the magnitude spectrogram of audio mixtures to detect and extract particular patterns such as temporal repetitions. I refer to the vertical and horizontal dimensions of the 2D transform domain as *scale* and *rate*. These terms are borrowed from studies of mammalian auditory systems [17, 134, 152]. In this context, scale corresponds to the spread of spectral energy (e.g. frequency modulation depth) as well as frequency-domain repetitions (e.g. overtones) and rate corresponds to temporal repetitions (e.g. repeating percussive patterns).

In Figure 2.3, I show various time-frequency representations in the top row and their associated 2DFTs in the bottom row. The left two time-frequency representations are simple patterns that can be represented by a single sinusoid. They manifest in their associated 2DFT as two peaks, oriented in the same direction as the pattern. The distance of the peaks from the origin indicates the frequency of the sinusoid (e.g. how

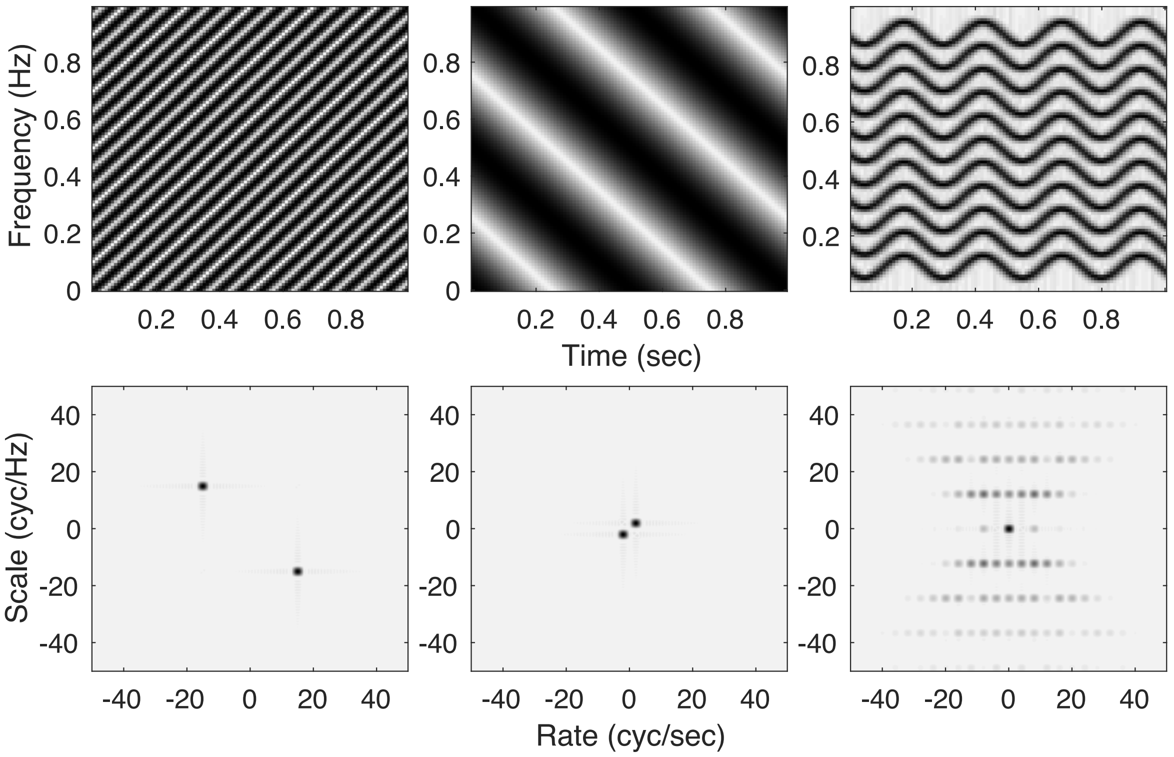

Figure 2.3. Examples of time-frequency-domain signals (top row) and their associated magnitude 2D Fourier transforms (bottom row). The left two show 2D sinusoids and the right-most plot shows a more complex 2D signal. Darker colors show higher values in all plots.

rapidly it goes from black to white). More complex signals, like the one shown in the top right of the figure, are represented by a combination of many sinusoids. This complex signal manifests as more spread out in the 2DFT, with many peaks required to reconstruct the wavy pattern.

To get some intuition, let's look at a simple example of using the 2DFT on a visual image to denoise it. In Figure 2.4, I show in the top left an image of an astronaut on the moon mixed with a noisy grid pattern on top of it. The goal is to get rid of the grid and recover the original image. To do this, you first take the 2DFT of the image. The 2DFT of the noisy image is shown in the upper right of the figure. The grid-like pattern manifests as a set of peaks in the 2DFT. Denoising the image, then, reduces to detecting these peaks in the 2DFT, deleting them, and then inverting the peak-less 2DFT back to the image domain. This procedure is shown in the bottom right (2DFT without peaks) and bottom left (denoised image) of the figure.

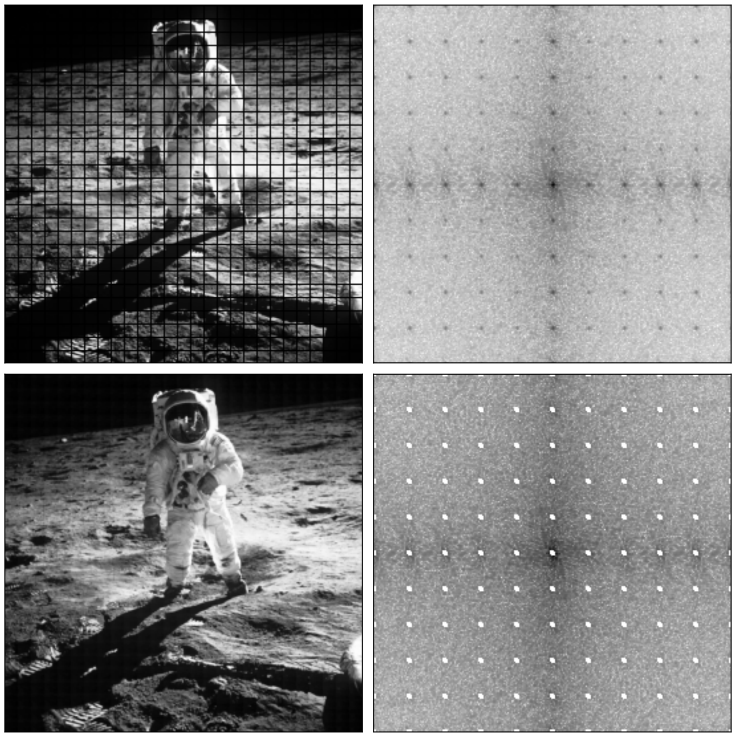

Figure 2.4. An example of periodic noise removal using the 2DFT. The noisy image (upper left) is denoised by taking its 2DFT (upper right), removing local peaks that correspond to the repeating pattern (lower right) and inverting the 2DFT to obtain the denoised image (lower left).

2.2.2. Applying the 2DFT to audio

Let $x(t)$ denote a single-channel time-domain audio signal and $X(t, f)$ its complex Short-time Fourier Transform (STFT), where f is frequency and t is time. I restrict the 2DFT analysis to the magnitudes of the STFT, ignoring the phase. To this end, all the processing will be performed on $|X(t, f)|$, where $|.|$ denotes the magnitude operator. The scale-rate representation of the spectrogram will be denoted by $\tilde{X}(s, r)$, where s and r, stand for scale and rate respectively. The relationship between the spectrogram and its scale-rate transform can then be formulated as

$$\tilde{X}(s,r) = \mathcal{FT}_{2D}\{|X(\omega,\tau)|\}, \tag{2.4}$$

where $\mathcal{FT}_{2D}\{.\}$ denotes the two-dimensional Fourier transform. $\tilde{X}(s, r)$ contains complex values. The magnitude of $\tilde{X}(s, r)$ is what I process for the algorithm.

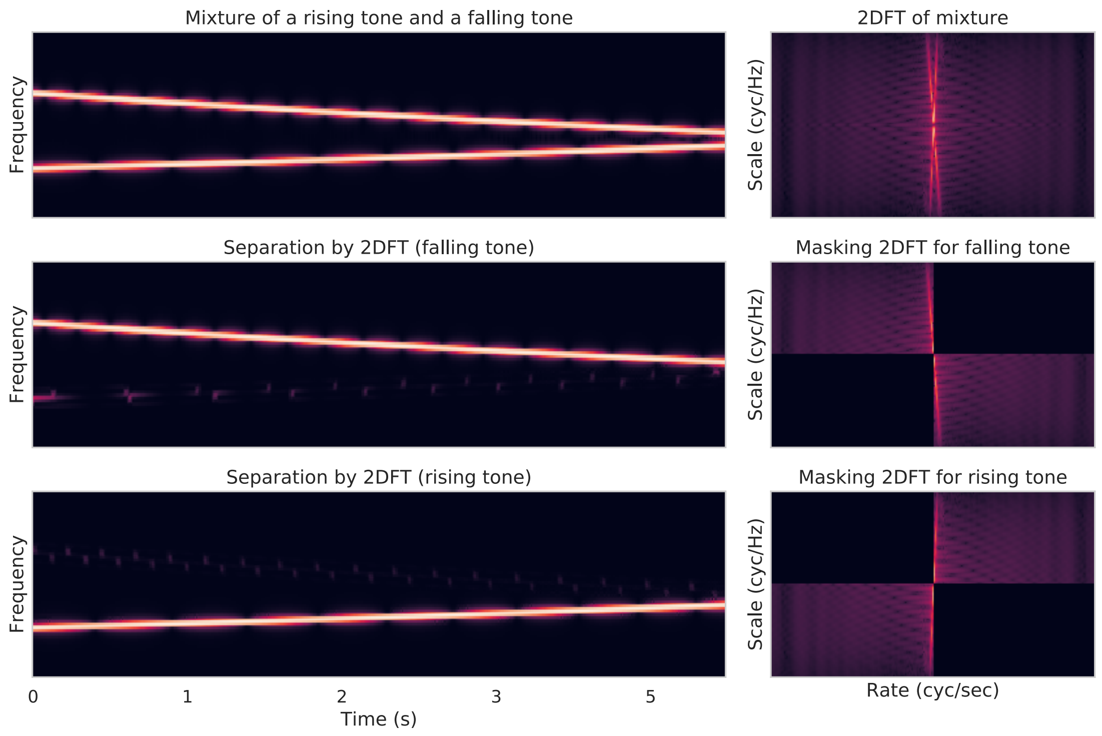

Figure 2.5. Rising tones and falling tones can easily be isolated using the 2DFT. Each row contains the time-frequency representation and its associated 2DFT. When different quadrants are symmetrically masked out in the 2DFT, one can isolate either rising components or falling components in a mixture.

I will now show how to manipulate $|\tilde{X}(s,r)|$ in meaningful ways for audio processing. Recall that, as shown in Figure 2.3, the orientation of the peaks across the origin is the same as the orientation of the pattern in the original image. The peaks are symmetric across the origin. Spectro-temporal patterns that are falling in pitch over time manifest as peaks in the upper left and bottom right quadrants. Rising patterns are in the opposite pair of quadrants. This can be easily exploited to separate rising from falling tones by simply masking out the associated quadrants. Figure 2.5 shows a simple example of this. The auditory scene consists of a rising chirp mixed with a falling chirp. By masking out the associated quadrants, shown in the right-most column of the figure, one can isolate falling tones from rising tones.

2.2.3. Proposed method

A more complex way to manipulate the 2DFT of a time-frequency representation is to find local peaks and mask them out, just as I did to denoise the image in Figure 2.4. It is first important to realize what peaks correspond to in audio mixtures. Peaks in the 2DFT of a time-frequency representation indicates some

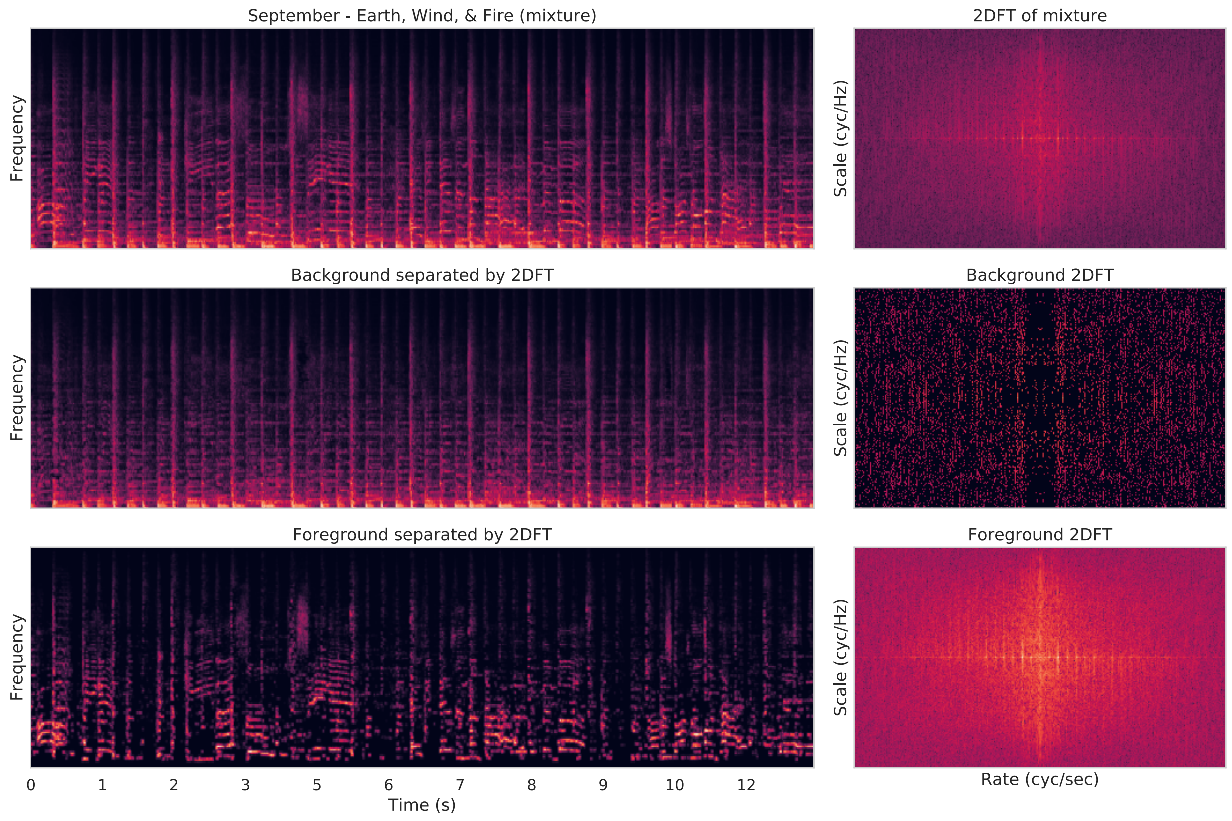

Figure 2.6. Separation using the 2D Fourier Transform (2DFT). In the first row, the left panel shows the mixture spectrogram and the right panel its 2DFT. In the second row, I apply my peak picking technique to obtain a background 2DFT. Then, I invert this 2DFT and apply masking to the mixture spectrogram to get the background spectrogram. In the third row, I show everything from the rest of the 2DFT (i.e. the non-peaks or foreground), which contains the singing voice.

sinusoidal pattern going across time and frequency. Figure 2.6 shows a musical mixture. In that mixture (top row), there is a repetitive snare drum, roughly sounding twice a second. This periodic pattern is essentially a sinusoid oriented along the rate axis of the 2DFT. It manifests as a peak in the 2DFT. By detecting peaks and removing them from the 2DFT, I can separate out repeating patterns from non-repeating patterns. Regular peaks correspond to repeating patterns. Non-peaks correspond to non-repeating audio. One need only find the peaks in the 2DFT, mask them out, then invert this back to a spectrogram (and then back to the original waveform) to remove the repeating audio. I now describe a simple peak-picking procedure.

The goal of this procedure is to locate local peaks in the magnitude of the 2DFT and mask them out to separate a background signal from a foreground signal. Thus, the goal is to produce two masks on the 2DFT - one with the peaks and one without the peaks. these are referred from now as the background mask (the one with peaks) and the foreground mask (the one without peaks). Note that these masks are on the modulation-based representation (the 2DFT) and not on the time-frequency representation (the spectrogram), as seen before.

I pick peaks by comparing the difference between the maximum and minimum magnitude values over a neighborhood surrounding each point in the 2DFT. For each point in the 2DFT, consider it to be the center of an arbitrary rectangular neighborhood. The center is some point $c = (s_c, r_c)$ and its neighborhood is denoted by $N(c)$. The dimensions of the neighborhood along the scale and rate axes are tunable parameters. The neighborhood can be of arbitrary shape. The bigger the neighborhood, the more significant the peak needs to be in the 2DFT to be detected. The problem reduces to deciding for each point in the 2DFT, whether it is a peak relative to its neighborhood.

For each neighborhood surrounding a point, I first calculate the maximum value of the neighborhood $M_{N(c)}$, the mean value of the neighborhood $\mu_{N(c)}$, and the standard deviation of the neighborhood $\sigma_{N(c)}$. Next, I calculate how many standard deviations above the neighborhood mean is the neighborhood max:

$$\frac{M_{N(c)} - \mu_{N(c)}}{\sigma_{N(c)}} \tag{2.5}$$

The idea is that strong peaks will be significantly above the mean of their neighborhoods, which can be detected by using the standard deviation of the neighborhood. The next step is to mask out the peaks. For each point in the 2DFT, I check if it is equal to its neighborhood max. I denote the computed background mask over the entire scale-rate domain representation by $M_{bg}(s, r)$. The scale-rate domain *foreground* mask can then be computed as $M_{fg}(s, r) = 1 - M_{bg}(s, r)$. If it is, then it will be assigned to the background as follows:

$$M_{bg}(s_c, r_c) = \begin{cases} \frac{M_{N(c)} - \mu_{N(c)}}{\sigma_{N(c)}} & |\tilde{X}(s_c, r_c)| = \max\limits_{N(c)} |\tilde{X}(s, r)| \\ 0 & otherwise \end{cases} \tag{2.6}$$

$M_{bg}(s_c, r_c)$ is then normalized so its values are between 0 and 1 via:

$$\frac{M_{bg}(s_c, r_c)}{\max_{M_{bg}(s_c, r_c)}} \tag{2.7}$$

This creates a soft mask on the 2DFT that is used to filter out patterns related to the peaks in the time-frequency representation. The foreground 2DFT mask is then simply $M_{fg} = 1 - M_{bg}(s_c, r_c)$. Intuitively, this is simply a way to discover local maxima in $|\tilde{X}(s,r)|$ and weight their contribution to the background mask by how much of a local maximum it is. It should be noted that neighborhood selection and mask value computation is performed for every single point in the scale-rate domain. This procedure results in the foreground and background 2DFTs shown in Figure 2.6.

Next, I compute the separated magnitude spectrogram of the background source from the masked version of the complex scale-rate domain representation, by taking the inverse 2DFT of the masked signal:

$$|X_{bg}(\omega, \tau)| = \mathcal{IFT}_{2D}\{M_{bg}(s,r) \odot \tilde{X}(s,r)\}, \tag{2.8}$$

with $\mathcal{IFT}_{2D}\{.\}$ denoting the inverse 2D Fourier transform and $\odot$ denoting element-by-element multiplication, respectively. The foreground magnitude spectrogram can be similarly computed using the foreground mask. This results in two time-frequency representations: one for the background $X_{bg}(t,f)$ and one for the foreground $X_{fg}(t,f)$. To recover the time-domain audio, I use the foreground and background time-frequency representations to construct a time-frequency mask on the original mixture $X(t,f)$.

I can use either the $X_{bg}(t,f)$ or $X_{fg}(t,f)$ to construct the mask. Which one I use has consequences in terms of separation, as we will see later on. For now, I will denote the one used as $X_s(t,f)$. The time-frequency mask is constructed as follows:

$$M_s(t,f) = \frac{min(X_s(t,f), X(t,f))}{X(t,f)} \tag{2.9}$$

where *min* indicates the element-wise minimum between the recovered source time-frequency representation and the mixture's time-frequency representation[2]. This is required to ensure that the mask $M_s(t,f)$ is between 0 and 1.

With this procedure, I get time-frequency masks for the background ($M_{bg}(t,f)$) and foreground signals ($M_{fg}(t,f)$). Finally, the time-domain background and foreground audio signals are recovered from the masked STFT. In short, $x_{bg}(t) = ISTFT\{M_{bg}(t,f) \odot X(t,f)\}$, where $ISTFT\{.\}$ is the Inverse Short-Time Fourier Transform, computed through the overlap-and-add method. The foreground audio signal

[2]Also, some small value should be added to the denominator in implementation to avoid a potential numerical error.

can be similarly computed by applying the foreground mask to the complex spectrogram and taking the inverse STFT. The associated 2DFTs and time-frequency representations corresponding to the foreground and background can be seen in Figure 2.6.

2.2.4. Behavior on auditory phenomena

The algorithm I have presented has interesting behavior when confronted with auditory stimuli used in lab experiments to investigate the behavior of the human auditory system. In short, while it uses entirely different mechanics and does not attempt to model the human auditory system directly in any way, it leverages similar cues to those that humans use to group sources in the same auditory stimuli.

The first stimulus I will consider is the one used by Chowning [19], establishing the connection between frequency micromodulation and voice perception. A graphical representation of this stimuli is shown in Figure 2.2. When tones are modulated in frequency slightly (e.g. vibrato), we tend to perceive the modulating tone as a human voice. Figure 2.7 shows the response of the 2DFT separation algorithm to this auditory stimuli. When the tone is pure, with no modulation, the 2DFT algorithm rejects it to the background source. Mechanically, this makes sense as the pure tone consists of straight horizontal lines, which correspond to peaks in the 2DFT. When the peaks are masked out, the pure tone is filtered out, leaving the modulating tone. Similarly, note that the wavy complex signal depicted in the top right of Figure 2.3 has a corresponding 2DFT that does not consist of a single peak, but rather is a dispersed set of peaks. Tones that have a bit of vibrato require more 2D sinusoids at various amplitudes and phases to be reconstructed, resulting in a less "peaky" 2DFT representation. Therefore, a peak picking procedure on the 2DFT tends to filter out tones that are modulating in frequency. This shows that the 2DFT algorithm is sensitive to micromodulation cues in auditory scenes. The 2DFT algorithm is the first algorithm I know of that explicitly makes use of micromodulation to separate an auditory scene.

The next stimulus example is one used in Bregman [8] to investigate the effect of common frequency change on auditory perception. In this example, shown in Figure 2.8, a fundamental tone is played with overtones on top of it. Because the fundamental and all of its overtones are static and not moving, the tone is perceived as a single tone. Then, half of the overtones begin to modulate up and down in frequency. This is similar to the micromodulation stimulus, but this time the frequency modulation is much greater. The human perception of this is that two tones are now playing - one that is static and one that is moving up and down. The response of the 2DFT algorithm is the same. The moving tone is not "peaky" in the 2DFT

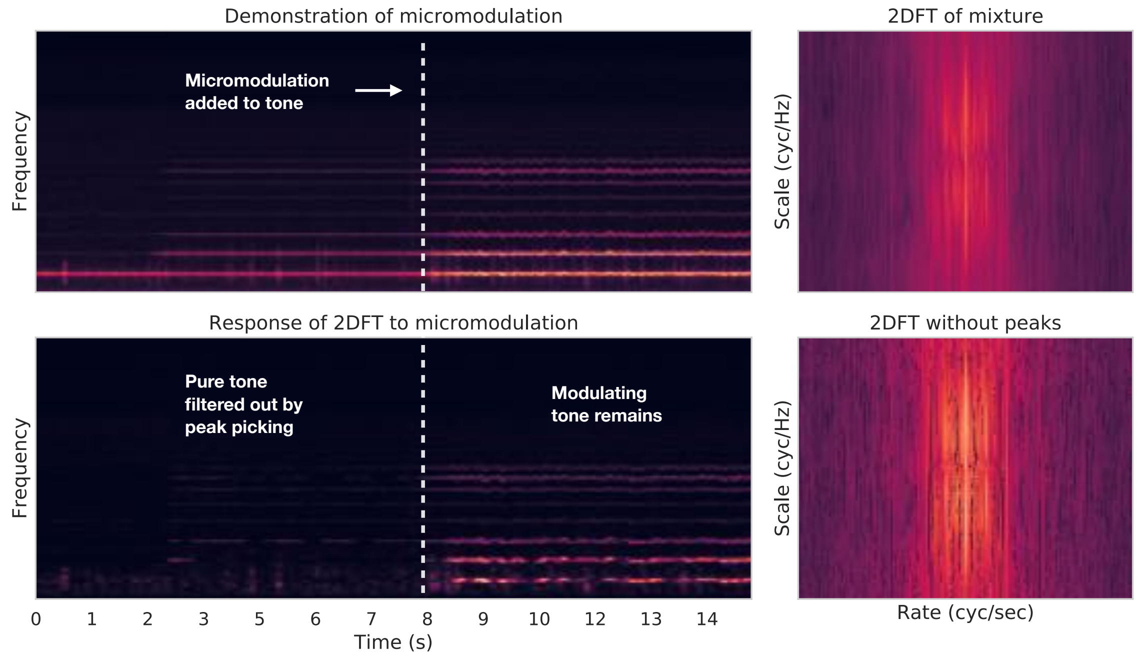

Figure 2.7. Response of the 2DFT separation algorithm to micromodulation. A pure tone is first played, along with overtones. When the tone begins to modulate in frequency, the 2DFT algorithm recognizes it as a foreground source, showing that it is sensitive to micromodulation cues in audio mixtures.

of the mixture, so it is placed in the foreground, as a separate source from the static tone. When the moving tone ceases, the tones fuse again into a single source in the background.

The final stimuli I will consider is the one used by McDermott et al. [116] to establish that repetition is a cue leveraged by humans for parsing auditory scenes. One of the stimuli they used in their experiments is shown in Figure 2.9. It consists of a repeating synthesized sound and a non-repeating interfering sound. The repeating synthesized sound manifests as a set of peaks in the 2DFT. The peak-picking procedure isolates the repeating sound from the interfering sounds successfully. This shows that the 2DFT algorithm is also sensitive to repetition in the auditory scene. It is not the first algorithm to leverage repetition for source separation [101, 141, 144], but as we will now see, it is the most effective algorithm for leveraging repetition.

2.2.5. Experiments

I now apply the 2DFT source separation algorithm to a practical problem - separating vocals from accompaniment in musical mixtures. In musical mixtures, there is often an element of repetition in the accompaniment source while a vocalist sings on top of it. The vocalist will often put vibrato on his or her voice, causing some frequency modulation to occur. Further, regardless of amount of vibrato, singing voice will always exhibit

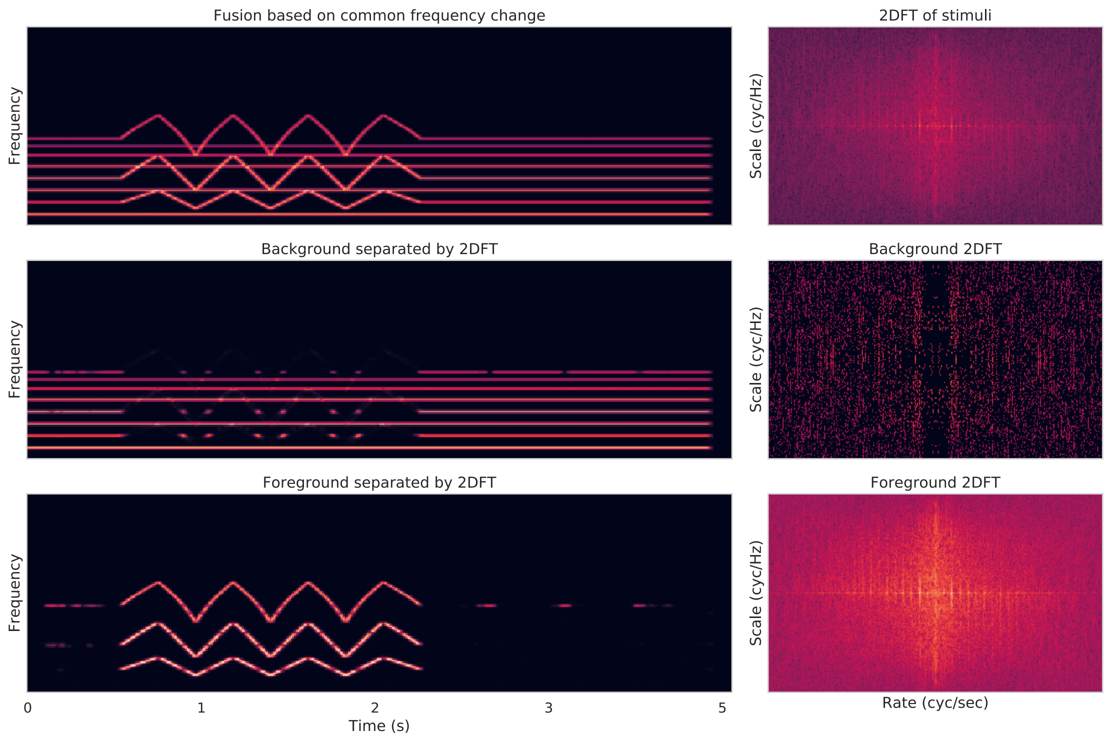

Figure 2.8. Response of the 2DFT separation algorithm to common frequency change. The tone is initially a complex tone with a fundamental and its overtones. It is perceived as a single sound. When half of its overtones begin to move, we perceive the mixture as containing two sounds. When the overtones stop moving, we once again perceive a single sound. The response of the 2DFT algorithm is the same.

some amount of frequency micromodulation [8, 18]. That these two cues are salient in musical mixtures with vocals makes the 2DFT source separation algorithm a natural fit for separating vocals.

Vocals separation is a well studied problem in the source separation literature [33, 69, 84, 101, 140, 141, 144] with proposed methods in both supervised (e.g. deep learning) and unsupervised methods. Here we are concerned with how 2DFT performs relative to competing primitive separation methods. The most relevant methods are REPET and REPET-SIM, which both leverage repetition in the auditory scene. Another relevant method is Melodia [154], which was not designed for source separation but rather melody tracking. Melodia leverages time and pitch proximity in a mixture to track the vocal melody. One can turn the extracted melody into a time-frequency mask by using the tracked melody along with some number of integer multiples of the melody frequency at each time frame. This mask will try to extract the vocals from the mixture.

2.2.5.1. Datasets. I tested my method against these competing methods on the MUSDB dataset [173], which consists of 150 songs with four isolated stems for each song. The isolated stems consist of vocals, drums,

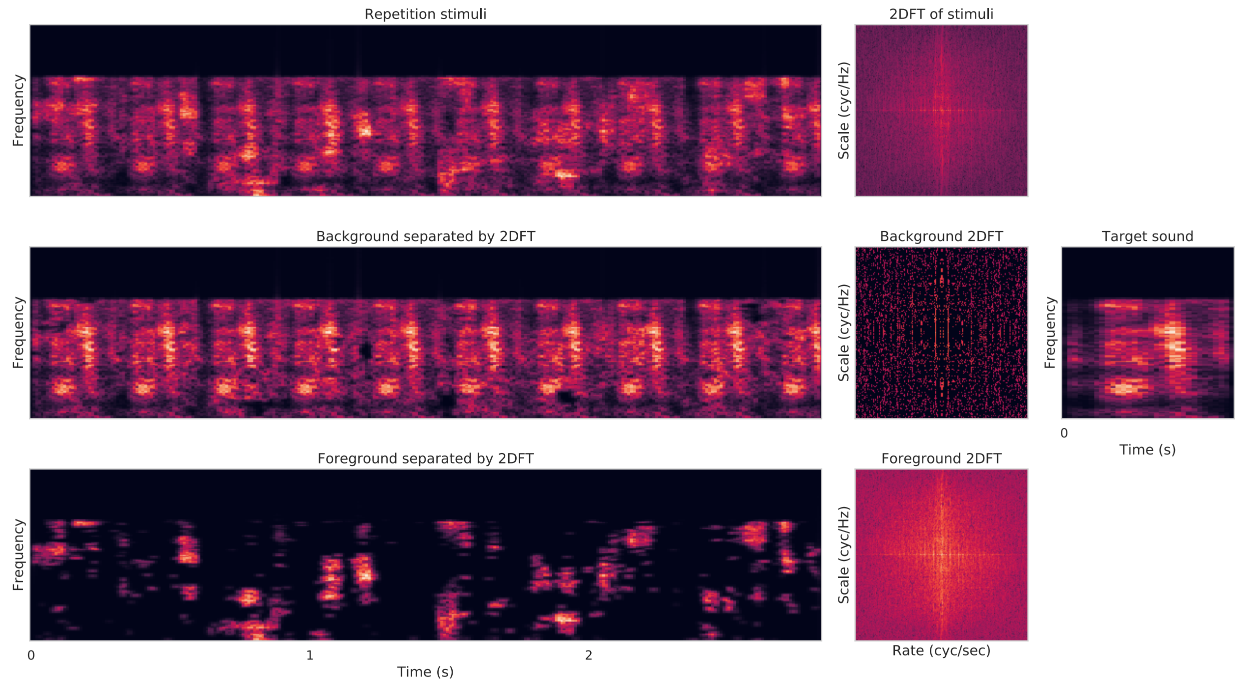

Figure 2.9. Response of the 2DFT separation algorithm to repetition. In the mixture a repeating sound (shown on the right - "target sound") is played while a non-repeating sound is mixed with the target sound. The peak-picking procedure successfully isolates the repeating sound in the mixture, as can be seen by comparing the background spectrogram (middle row) with the target sound spectrogram.

bass, and other. The "other" stem contains sources like keyboards, guitars, etc. In these experiments, the accompaniment consists of the sum of the other, drums and bass stems and the vocals consist of just the vocals stem. The mixture consists of the accompaniment plus the vocals stems. The goal is to separate the vocals from the accompaniment given the mixture.

2.2.5.2. Evaluation. For evaluation, I used the blind source separation evaluation (BSSEval [185]) metrics - source-to-distortion ratio (SDR), source-to-interference ratio (SIR), and source-to-artifact ratio (SAR). This is the most common way to evaluate separation performance [93, 103, 124, 173]. The output of a separation algorithm is the time-series audio waveform for each source (in this case the vocals and the accompaniment). These evaluation measures then compare the estimated time-series audio waveform with the *ground truth* audio waveform - the stem recordings of vocals and accompaniment, prior to mixing, in this case. The three evaluation metrics - SDR, SIR, and SAR - measure the quality of the recovered source in three different ways.

SIR tries to measure how suppressed the other sources in the recovered source. For example, how much accompaniment is left in the estimate of the vocals. SAR tries to measure the amount of artificial noise in the recovered source. Source separation algorithms often result in output that has what is called "musical noise" in them. These are also called artifacts, something like what you would see in the output of an image compression algorithm. Finally, SDR estimates the overall quality of the separated source, which is affected by both how much it suppresses other sources in the mixture as well as how many artifacts are in the source.

It's worth taking a moment to consider the relationship of these metrics to human perception, as well as how these evaluation measures can be "hacked" by an adversary. The first question was considered by Cartwright et al. [14] in which a controlled lab-study where participants judged separated sources along four different criteria: overall quality, target preservation, suppression of other sources, and absence of artificial noise. The authors report the correlation of SDR/SIR/SAR with the lab-study on those same sources. Of the three metrics, they found that SIR had the highest correlation with the results of the lab-study perception (Pearson correlation of .78), followed by SAR (.55) and SDR (.50). This suggests that SIR better correlates with human perception than the SDR or SAR. While these automated evaluation metrics have some flaws, they are still commonly used for comparing algorithms to one another in the literature [93].

The second question was considered by Le Roux et al. [93] in which they show that the original formulation is flawed and can be hacked easily using a deep network. Specifically, they show that they can maximize the SDR with an arbitrary time-frequency mask that clearly does not separate a source. They suggest two new classes of metrics - scale-invariant SDR/SIR/SAR and scale-dependent SDR/SIR/SAR - which are not sensitive to this sort of hack. In this analysis, I use the scale-dependent SDR/SIR/SAR metrics, which are more robust and less prone to adversarial attacks. I will refer to these measures as simply SDR, SIR, and SAR throughout this dissertation.

Finally, when interpreting results, it is important to consider all three metrics and how they interact with each other. For example, using the mixture as the estimated source will have very high SAR - as no separation has taken place, there will be no artifacts. However, it will have very low SIR, as no suppression of the other sources has occurred. The result will be low SDR (very high distortion as the target source is completely mixed with other sources). Similarly, it's possible to get high SIR and SAR by simply outputting a blank source - the other sources have been suppressed, but so has the target source! Since the estimated source is blank, there are also no artifacts. However, the preservation of the target source is poor, resulting in low SDR. Finally, if an estimated source has high SIR, low SAR, and an SDR somewhere in between, it's possible to increase SDR at the cost of SIR. This is done by adding some of the mixture back into the estimated source, reducing the impact of artifacts, increasing SAR. SIR goes down a little because the suppression of other sources is reduced, but SDR is likely to increase. This is commonly done to make the estimated sources more listenable [140, 141, 144]. This suggests there is a trade-off to be made between suppression and artifacts, which makes some intuitive sense - the more you suppress a source, the more likely you are to introduce artifacts into the estimated source.

2.2.5.3. Experimental design. My experiment is designed to investigate the consequences of all the free parameters in the 2DFT algorithm, as well as compare its performance to competing unsupervised vocal separation algorithms. The free parameters in the 2DFT algorithm are:

(1) **The size and shape of the neighborhood used to select peaks.** The bigger the neighborhood, the larger a peak has to be in the 2DFT to be filtered into the background.
(2) **The time-frequency resolution of the spectrogram that the 2DFT is applied to.** The 2DFT detects modulation patterns and repetition patterns in the spectrogram. If the frequency resolution of that spectrogram is too low, the 2DFT will not be able to detect that a sound has frequency modulation. The frequencies of analysis will be too far apart to capture the modulation, resulting in what appears to be a straight line in the spectrogram. If the time resolution is too low, then repeating patterns may not manifest in the spectrogram, causing 2DFT to fail.
(3) **Which time-frequency reconstruction I use to construct the time-frequency masks:** $X_{bg}(t, f)$ or $X_{fg}(t, f)$. $X_{bg}(t, f)$ is constructed by deleting everything but the peaks from the 2DFT and inverting the 2DFT while $X_{fg}(t, f)$ is constructed by keeping everything except the peaks and inverting the resultant 2DFT. This has a marked effect on how the separation sounds and the type of cue that 2DFT is using on the mixture.

To investigate the effect these free parameters have on separation performance, I conducted parameter-sweeps across them for the 2DFT algorithm on the MUSDB *train* set of songs ($N = 100$), using scale-dependent SDR/SIR/SAR for evaluation. Then, I compared the performance of the 2DFT algorithm to REPET [141], REPET-SIM [144], and Melodia [154] on the MUSDB *test* set ($N = 50$).

2.2.5.4. Results and discussion. The effect of neighborhood size on the performance of 2DFT is shown in Figure 2.10. Here, I only consider neighborhood shapes that are rectangles of one row high by some number of columns (1, N). Other neighborhood shapes (diagonal ones, vertical ones, square ones) were found to be unsuccessful in separating vocals from accompaniment in preliminary trials. Overall, 2DFT is fairly robust to changes in the length of the neighborhood, but at extreme settings of (1, 5) and (1, 1000), it has much worse performance. SIR tends to go up as the neighborhood considered gets bigger, as fewer peaks are being

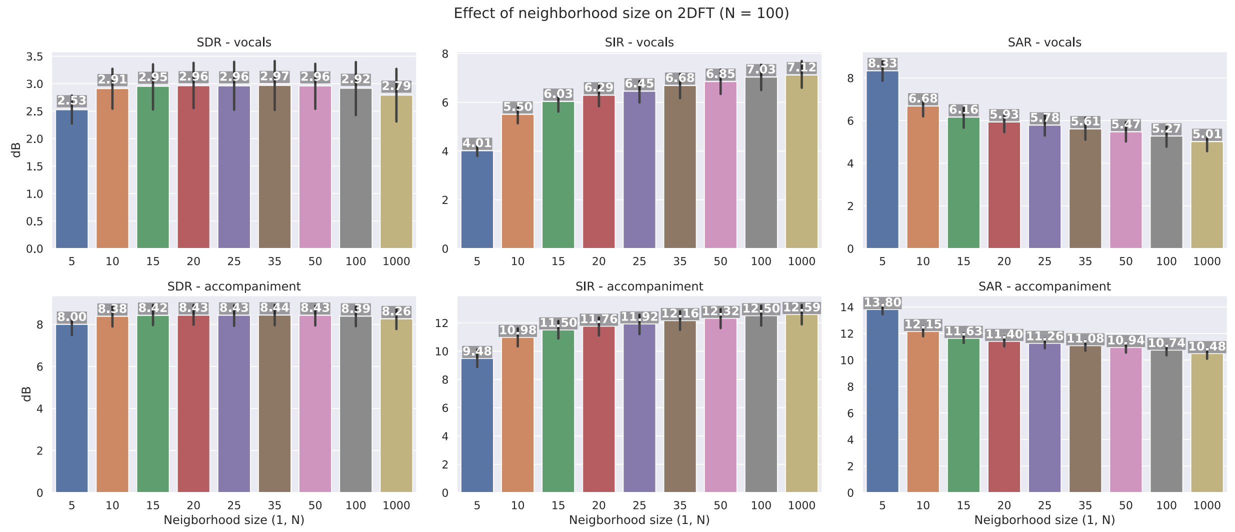

Figure 2.10. Effect of neighborhood size on 2DFT performance for vocals/accompaniment separation. The shape of the neighborhood is (1, N), that is one row by N columns in the 2DFT. The bigger it is, the higher the SIR gets, at the cost of SAR. The mean metric is reported above each bar. A fair balance between the two is seen at (1, 35), where SDR peaks. However, outside of the extreme values at (1, 5) and (1, 1000), there is little effect of neighborhood size on separation performance.

selected, making the accompaniment estimation reject more of the vocals. However, SAR tends to go down as this happens. This causes a peak in SDR at (1, 35). However, for all neighborhoods between (1, 10) and (1, 100), we observe essentially the same SDR.

The effect of time-frequency resolution on 2DFT performance is shown in Figure 2.11. Here, I use a neighborhood size of (1, 35), as it has a good trade-off between SIR and SAR, as seen in Figure 2.10. Recall that an STFT is constructed by taking a window of the time-series audio and computing its Fourier transform. This result then becomes a column of the STFT, containing the magnitudes and phases at each frequency for that window in time. Then the window is moved over by some number of samples, and the next column of the STFT is computed. How much the window is moved over is called the hop length, which dictates the time-resolution of the STFT. The time resolution in seconds is computed by dividing the hop length by the sample rate of the audio. How big the window is dictates the frequency resolution of the STFT. The frequency-resolution in Hz is computed by dividing the sample rate by the window length. In Figure 2.11, I show the performance of 2DFT at various time-frequency resolutions, with frequency resolution in Hz on the x axis and time resolution in milliseconds on the y axis. I note a peak in performance at a frequency resolution of 21.5 Hz.

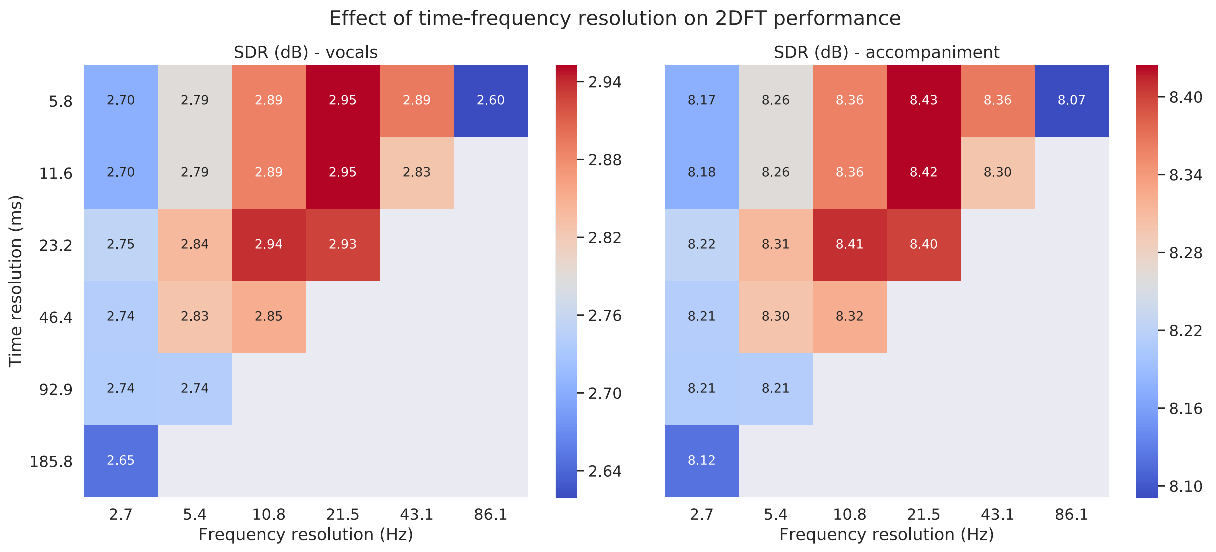

Figure 2.11. Effect of time-frequency resolution on 2DFT performance. Each box displays the mean SDR across the train set of MUSDB. We see a peak in SDR at a frequency resolution 21.5 Hz, which is quite near the minimum amount of frequency modulation observed in singing voice [175], possibly explaining this peak in performance.

Table 2.1. Performance of micromodulation vs repetition on vocals separation on the MUSDB train set. We see that the two approaches have similar performance, but repetition is slightly better.

Approach	SDR	SIR	SAR
2DFT (micromodulation)	2.82	6.56	5.45
2DFT (repetition)	2.96	6.68	5.60

As shown in Section 2.2.4, the 2DFT algorithm leverages frequency micromodulation to parse the auditory scene. In music, the singing voice usually displays some amount of vibrato, which can be picked up by 2DFT for separation. Sundberg [175] studied the vibrato of multiple singers to establish the amount of frequency variation in their singing voice. The authors found that 6% was on average the amount of frequency variation (how much the singer was deviating from the central pitch). If we assume a minimum frequency for singing voice of 300 Hz (a reasonable assumption - most singing voices are somewhere between 300 and 2000 Hz [175]), detecting a vibrato at that frequency would require a resolution of $300 * .06 = 18$ Hz. This may explain why I get the best performance for 2DFT at a close value at 21.5 Hz. Finally, Sundberg [175] establishes a vibrato has a period of around 5 or 6 Hz in time, which may explain why time resolution does not have much effect on the performance of 2DFT.

The final free parameter of the algorithm is whether to use the peaks in the 2DFT to construct the mask or to use the 2DFT without peaks to construct the mask. Recall that the 2DFT balances two primitive

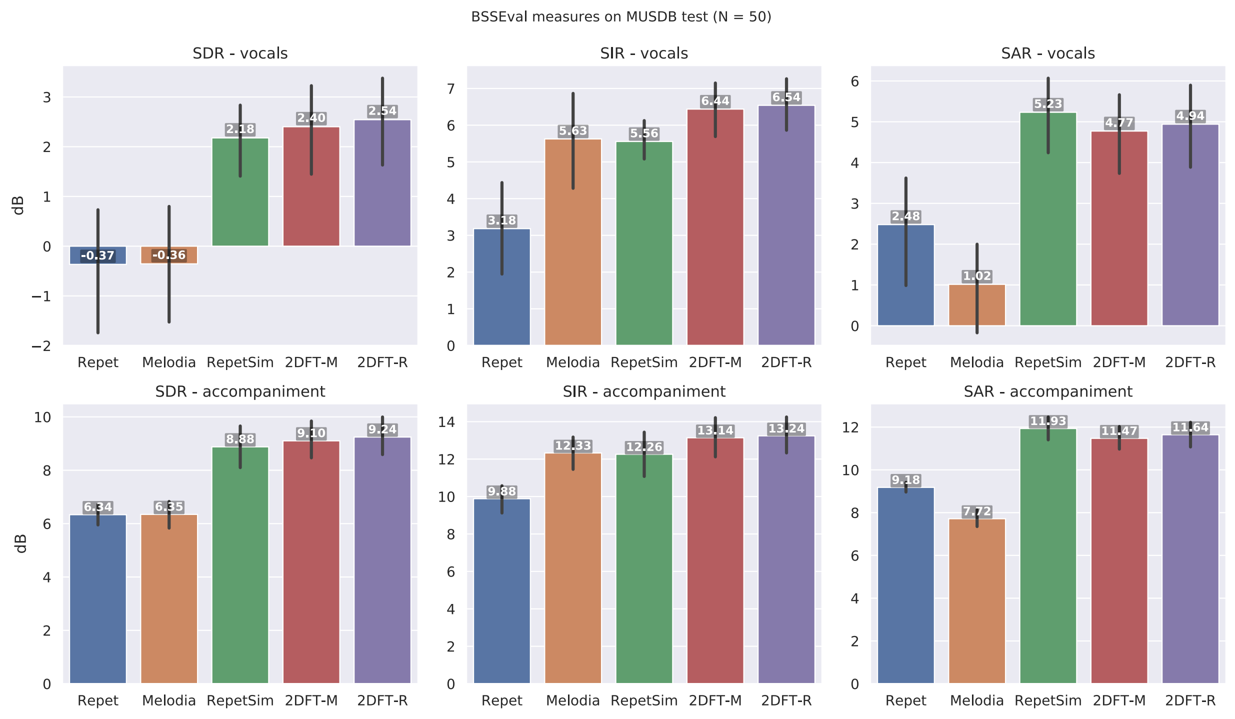

Figure 2.12. Comparison between 2DFT and competing methods. 2DFT out-performs all competing methods in SDR and SIR on the MUSDB test set. It underperforms on SAR compared to REPET-SIM, due to higher SIR. 2DFT-R indicates using the 2DFT background to construct the mask (repetition), while 2DFT-M indicates using the 2DFT foreground to construct the mask (micromodulation).

grouping principles simultaneously - the principle of repetition and the principle of micromodulation. Which one is used more is dictated by whether you use the peaks, which model the repeating background, or the non-peaks, which model the micromodulating patterns that are dispersed in the 2DFT. Table 2.1 shows that the configuration of 2DFT that uses repetition more than micromodulation out-performs the other by .16 dB SDR.

Finally, I compare the 2DFT algorithm to competing unsupervised methods in Figure 2.12. I see that 2DFT out-performs the competing methods on the task of vocals separation. Note that the dataset used here is the *test* set, while previous results in this section were reported on the *train* set. This is explicitly to give a fair comparison of 2DFT to competing algorithms, rather than overfitting 2DFT settings to the training set. I use a window length of 2048 samples and a hop length of 512 samples to construct the STFT (for all algorithms) and a neighborhood size of (1, 35) for 2DFT. I find that 2DFT is the best performing method for the task of vocals separation, achieving the highest SDR and SIR. It achieves slightly lower SAR than REPET-SIM, due to the higher SIR. The closest competing method in terms of performance is

REPET-SIM [144]. Something to note about 2DFT versus REPET-SIM, however, is running time. 2DFT is a highly efficient algorithm, leveraging the fast Fourier transform [146]. REPET-SIM, on the other hand, requires the construction of an TxT matrix, where T is the number of time frames in the spectrogram. This makes the time complexity of REPET-SIM $O(F*(T^2))$, whereas the 2DFT algorithm has a time complexity of $O(T*F*log(T*F))$ - the time complexity of the fast Fourier transform, the bottleneck in the algorithm - where F is the number of frequencies. Considering $T > F$ in general, this works out to $O(T^2)$ for REPET-SIM vs $O(T*log(T))$ for 2DFT. In practice, 2DFT is also very fast to compute as the fast Fourier transform has highly optimized implementations.

2.2.6. Future work

The 2DFT algorithm for source separation can be extended on considerably. One major limitation of it is that it is computed on the magnitudes of the time-frequency representation, rather than the full complex time-frequency representation. This fundamentally limits the quality of the source separation that is possible with 2DFT. Pishdadian et al. [137] proposes a new representation, called the multi-resolution common fate transform or MCFT. MCFT is similar to taking the 2DFT of a time-frequency representation but has the advantage of using the full complex representation. This allows for higher reconstruction quality, as they show in their experiments. A peak-picking strategy similar to the one I have proposed for 2DFT could be applied on MCFT for source separation, but with better quality.

Another parameter to investigate is the input representation to 2DFT. I used an STFT throughout this work, which has linear frequency spacing. This is unlike the human auditory system which has logarithmic frequency spacing [8]. Applying 2DFT to different representations with logarithmic frequencies is unexplored and may yield better results than on an STFT. The MCFT, for example, uses such a representation for its input.

2.3. Combining primitives

I now turn to the problem of combining primitive algorithms. The work here is inspired by the human ability to use multiple strategies at once to parse an auditory scene [156]. In Chapter 1, we saw that when a primitive algorithm is applied to an auditory scene where the assumption that the algorithm relies on is invalid (e.g applying a direction of arrival algorithm to a monaural mixture), that output of the algorithm is not useful. We saw that in situations where one algorithm fails, others may be successful, suggesting that an

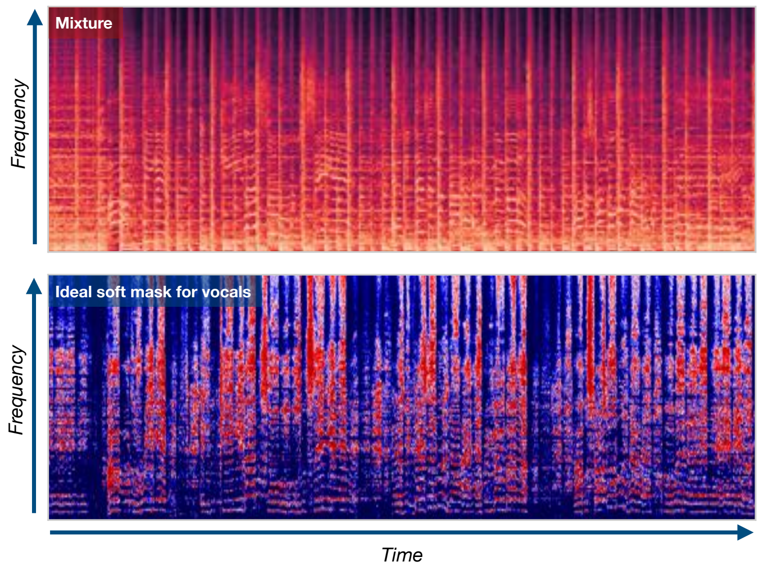

Figure 2.13. Pictured is the spectrogram of a musical mixture (above) and the ideal soft mask to separate out the vocals (below) for the mixture. The goal is to recover the soft mask below as closely as possible. The mask has values between 0 (blue) and 1 (red).

ensemble of algorithms would be helpful for parsing auditory scenes in general. I now describe an approach that combines multiple primitives in a framework I call *primitive clustering*.

2.3.1. Primitive clustering

Recall that the goal in all of the separation algorithms we have seen so far is to construct a time-frequency mask that, when applied to the mixture time-frequency representation, extracts the desired source. In Figure 2.13, I show one such mixture, in the top panel. The ideal soft mask, which has values between 0 and 1, is shown in the lower panel. To extract the source from the mixture, the mixture time-frequency representation is multiplied element-wise with the ideal soft mask.

Primitive source separation algorithms also make soft masks like these, deciding for every time-frequency point whether it belongs to one source or the other. For example, running a repetition-based algorithm will produce a soft mask. Each value in the soft mask is between 0 and 1 and can be interpreted as how repetitive or not repetitive that time-frequency point is. The same applies for an algorithm based on melody streaming - each point is mapped to a value indicating how melodic or non-melodic it is. The basic idea behind primitive clustering is to run several algorithms on a mixture, producing several masks. The mask values are then considered features for each time-frequency point in the spectrogram. Then all of the points are

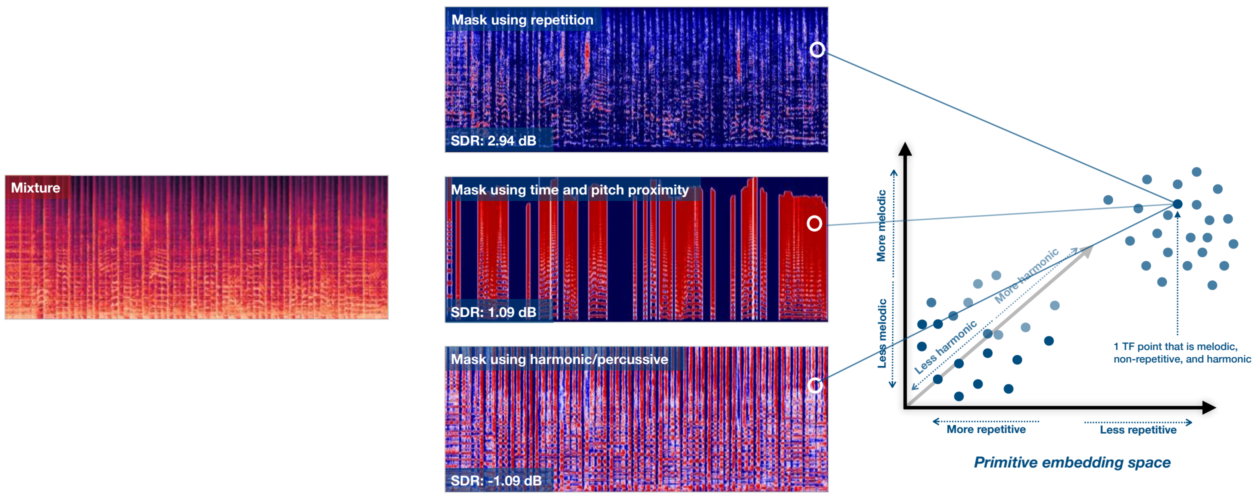
Mixture
Mask using repetition
SDR: 2.94 dB
Mask using time and pitch proximity
SDR: 1.09 dB
Mask using harmonic/percussive
SDR: -1.09 dB
More melodic
Less melodic
Less harmonic
More harmonic
More repetitive
Less repetitive
1 TF point that is melodic, non-repetitive, and harmonic
Primitive embedding space

Figure 2.14. The process for making a primitive embedding space. A set of primitive algorithms is run on the mixture. Each algorithm produces a mask with values between 0 (blue) and 1 (red) that indicates how it is segmenting the auditory scene. The performance (using source-to-distortion ratio - SDR) for each algorithm is shown. Together, the three masks map each time-frequency point to a 3D embedding space, shown on the right. The marked point was classified by the three primitives as melodic, not repetitive, and harmonic.

clustered using these features via a clustering algorithm like K-Means to produce a final mask. This final mask is then applied to the mixture spectrogram to recover the desired source.

In Figure 2.14, I show the process of using primitive separation algorithms to construct the primitive embedding space that is used for clustering. Here, three algorithms are run on the mixture: 2DFT (repetition), Melodia [154] (time and pitch proximity), and harmonic/percussive source separation [42] (HPSS). The masks produced by each algorithm are shown in the middle of Figure2.14. The goal is to extract the vocals from the mixture. For HPSS, I show the mask that will extract the harmonic elements, and for the 2DFT and Melodia I show the mask for separating out the non-repetitive and melodic elements, respectively.

One can observe that the three algorithms make very different decisions on how to parse the auditory scene. The 2DFT algorithm produces a very sparse mask that definitely captures some of the vocals but leaves in a lot of the snare (the vertical lines seen in the mask). Melodia produces a much denser mask that is completely blank when the vocals are not there, and has a rich overtone structure when they are. Finally, HPSS produces a mask that grabs all of the harmonic material, leaving behind the percussive material. None of the algorithms have perfect output, as can be seen in their corresponding SDRs. Of the three algorithms, 2DFT has the best performance by itself, giving 2.94 dB on this mixture.

Formally, each primitive algorithm is a function f_P that takes a point in the mixture time-frequency representation $X(t, f)$ and maps it to a value between 0 and 1 (inclusive):

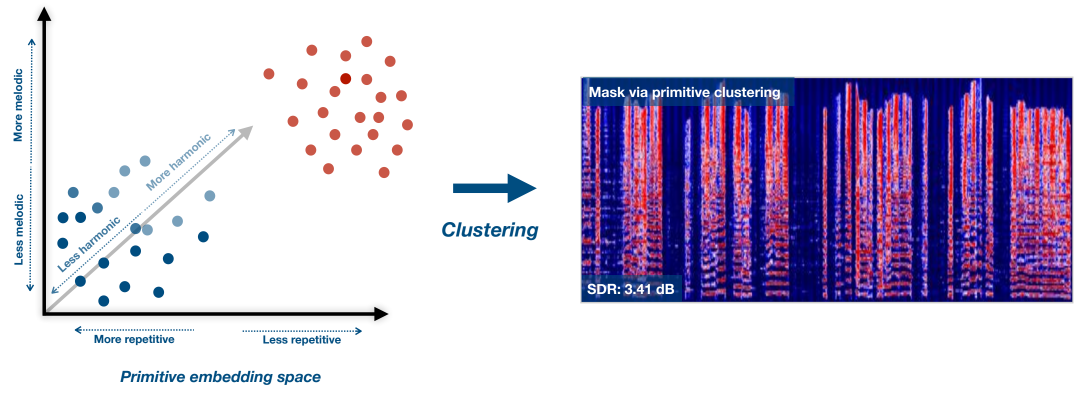

Figure 2.15. The primitive embedding space is clustered via K-Means to produce a new mask that takes into account the decisions of each primitive. The more each primitive agrees on how to classify a source, the stronger the value in the primitive clustering mask. The ensemble of the primitives produces a mask that has better separation (by SDR) than any of the primitives by themselves.

$$f_P(X(t,f)) \in [0,1]$$

Applying all of the primitives to each time-frequency point gives us the function $\mathcal{F}$, which maps each point to an N dimensional space, where N is the number of primitive algorithms applied to the mixture:

$$\mathcal{F}(X(t,f)) = [f_{P_0}(X(t,f)), f_{P_1}(X(t,f)), ..., f_{P_N}(X(t,f))] \in [0,1]^N$$

This mapping is visualized in Figure 2.14 for three primitive algorithms, resulting in a three dimensional space. To get intuition on how the primitive embedding space works, let's consider two points in that space: $[0]^N = [0,0,...,0]$ and $[1]^N = [1,1,...,1]$. These points are where all of the primitive algorithms agree on how to classify a time-frequency point. To interpret it in terms of Figure 2.14, points near $[0]^N$ are those where all three algorithms say that a point is repetitive, not harmonic (e.g. percussive), and not melodic. Points near $[1]^N$ are points that are not repetitive, are harmonic, and are melodic. Therefore, points near $[0]^N$ are likely to be accompaniment and points near $[1]^N$ are likely to be vocals. This is because accompaniment sources tend to be repetitive and non-melodic, where vocals tend to non-repetitive and melodic. Both sources can have harmonic or percussive elements, so this last primitive serves as a "tie-breaker" of sorts between all of the primitives.

To turn the embedding space into a soft mask that takes into account all of the decisions made by each algorithm, I use soft K-Means clustering [77]. The one trick I use is that instead of running K-Means to discover the means, I fix them to be points where all the primitives output either 0.0 or 1.0 for each time-frequency point: $\mu_0 = [0]^N$ and $\mu_1 = [1]^N$. Very few, if any, points are actually equal to these fixed points, as each primitive outputs a soft mask which has values between 0.0 and 1.0. Thus, it is more likely to get mask values that are near 0.0 or 1.0. When a point is near one of these fixed means, it means that all the primitives have output something close to 0.0 or 1.0, more strongly indicating that the point is likely to be accompaniment or vocals. The distance of every time-frequency point to these means is used to calculate calculate the soft mask for each source $M_k(t, f)$ at each point as follows:

$$M_k(t,f) = \frac{e^{-\beta \mathcal{D}(\mathcal{F}(X(t,f)),\mu_k)}}{\sum_j e^{-\beta \mathcal{D}(\mathcal{F}(X(t,f)),\mu_j)}} \tag{2.10}$$

where $\mathcal{D}(x, y)$ is the Euclidean distance between points x and y. This is familiarly the *softmax* function. It maps distances in the embedding space to values between 0 and 1. β is a tunable parameter that controls the *aggressiveness* of the softmax. In the normal setup of K-Means (i.e. hard K-Means), the cluster assignments are hard. If a point in the space is closer to one mean than another, even by a little bit, it is assigned to the closer mean in whole. In soft K-Means, points are softly assigned based on distance. As β increases, soft K-Means approaches hard K-Means. In primitive clustering, this has interesting consequences. If, for some time-frequency point, Melodia and HPSS both say that it belongs to the vocals, while 2DFT says it belongs to the accompaniment, then a high β would result in this point being assigned almost entirely to the vocals, rather than having its assignment reflect the disagreement between the primitives. The parameter β then strikes a balance between a "winner-take-all" strategy (high β) and a less aggressive strategy, where time-frequency points with disagreement are softly assigned accordingly to the different sources (low β).

Finally, an algorithm's importance to the final clustering decision can be decided simply by multiplying the mask values (e.g. features) produced by that algorithm for each time-frequency point by some scalar w. This increases its weight when computing the mask value via Equation 4.10. The weights can be decided in a fixed manner (e.g. by seeing the performance of each algorithm on a training set). I plan to explore the effect of weighting the primitives based on efficacy in future work. In this work, the weights are fixed to be equal.

The result of primitive clustering can be seen in Figure 2.15, where the primitive embedding space is clustered using K-Means with the fixed means described above and $\beta = 5$. The resultant mask is shown in the right part of the figure. The combination of the three primitives - for this example, harmonic/percussive,

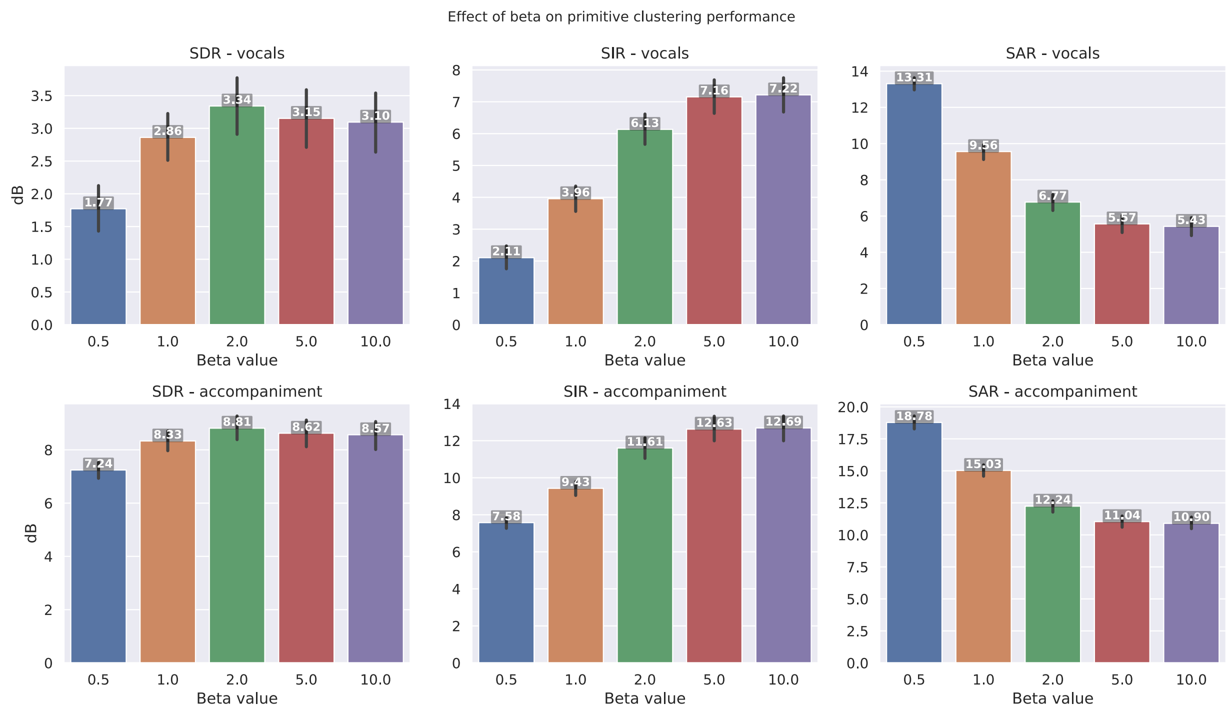

Figure 2.16. Primitive clustering with different values of β. As β increases, the soft masks approach binary masks, resulting in greater SIR at the cost of SAR. The SDR for primitive clustering peaks around $\beta = 2$ and diminishing returns for SIR starts around $\beta = 5$.

time and pitch proximity, and repetition - significantly out-performs any primitive by itself, producing a source estimate with a higher SDR of 3.41 dB. I now turn to testing and evaluating primitive clustering more extensively.

2.3.2. Experiments

As before, I tested my approach using the MUSDB dataset [173], which consists of 150 tracks with isolated vocals, drums, bass, and other stems. The goal is, given a mixture of vocals and accompaniment, to separate the vocals from the accompaniment. I use the same evaluation metrics as in Section 2.2.5.2: source-to-distortion ratio (SDR), source-to-interference ratio (SIR), and source-to-artifact ratio (SAR). My experiments are designed to answer two questions:

(1) How does the tunable parameter β affect the performance of primitive clustering?

(2) Does primitive clustering out-perform each primitive by itself?

I fixed my combination of primitive cues to be micromodulation, repetition, time and pitch proximity, and harmonic/percussive separation. These were selected because they all tend to isolate vocals from accompaniment, and they all rely on very different assumptions. Thus, a combination of them may be useful. The first two primitives use the 2DFT algorithm (Section **??**, Melodia is used for time and pitch proximity, and finally HPSS for harmonic/percussive separation. Algorithms like REPET and REPET-SIM were excluded as they perform worse than the 2DFT algorithm for source separation via repetition, as seen in Figure 2.12. Further, combining algorithms that leverage the same primitive in an auditory scene was found to not be helpful in preliminary experiments. This is possibly due to the fact that algorithms that rely on the same primitive make the same types of mistakes. For example, REPET, REPET-SIM, and 2DFT all put some of the snare drum into the estimate for the vocals. HPSS and Melodia, however, do not make this mistake. Therefore, combining multiple algorithms that make different kinds of mistakes because they rely on different primitives works best. Many combinations of algorithms are possible but this analysis is restricted to only the combination consisting of micromodulation, repetition, time and pitch proximity, and harmonic/percussive separation.

In Figure 2.16, I show the results of running primitive clustering on the MUSDB train dataset with $\beta = 0.5, 1, 2, 5, 10$. Note that I am using the training dataset here, as I did for all my previous parameter sweeps for other methods. This is to save the test set for later experiments and comparison between algorithms. As β increases, the masks produced by primitive clustering approach binary or hard masks. The harder the masks, the more that the interfering sources are suppressed by the method, resulting in higher SIR. However, this also produces more artifacts, resulting in lower SAR. As a result, this trade-off results in a peak in SDR at $\beta = 2$. However, $\beta = 5$ reduces the SDR by only a little bit while increasing the SIR by almost 1dB. Because of this, $\beta = 5$ is a good setting, providing an optimal trade-off between the evaluation measures.

In Figure 2.17, I show the results of running primitive clustering on the MUSDB test dataset as well as its input primitive algorithms: 2DFT (using micromodulation), 2DFT (using repetition), Melodia, and HPSS. Primitive clustering achieves a mean SDR on the test set of 2.93 dB, higher than the best competing input method 2DFT-R (which is 2DFT using repetition), which achieves 2.54 dB. Primitive clustering also achieves higher SIR and SAR than each of the input methods, indicating the usefulness of this simple approach to combining source separation algorithms.

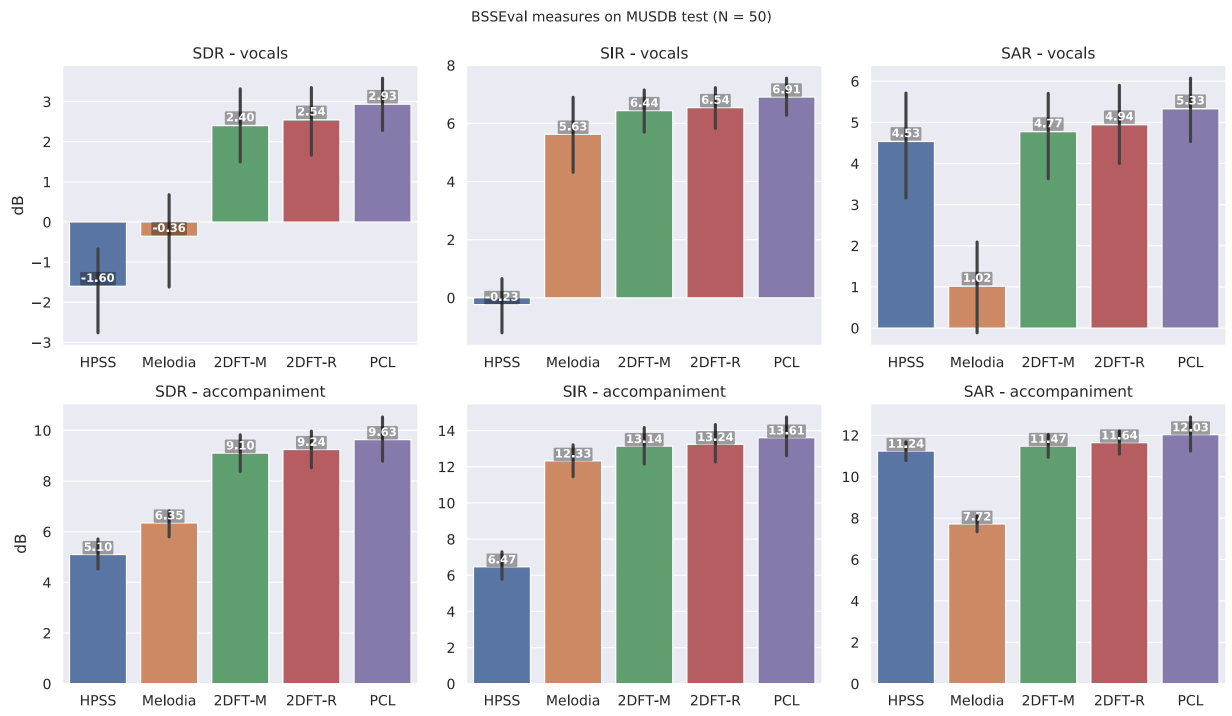

Figure 2.17. Performance comparison between primitive clustering (PCL) and the input methods used for primitive clustering on the MUSDB test set. 2DFT-M stands for 2DFT using micromodulation cues, while 2DFT-R stands for 2DFT using repetition cues. It can be seen that primitive clustering out-performs all of its input methods in terms of SDR, SIR, and SAR.

2.4. Conclusion

I have shown two of my contributions to primitive source separation. The first - 2DFT - was a novel method that leveraged repetition and micromodulation in an auditory scene to perform source separation. It did this by borrowing a simple technique - the 2D Fourier Transform - from image processing. In experiments, I showed it to be superior to competing unsupervised methods for the task of separating vocals from accompaniment in musical mixtures. I then turned to showing how different primitive algorithms can be used together such that the combination of them out-performs any single algorithm in isolation. My framework for doing this is called primitive clustering. In experiments, I showed it to out-perform all of its input methods on the task of separating vocals from accompaniment.

CHAPTER 3

Estimation of separation quality without access to ground truth

A hallmark of human audition is the ability to automatically, mostly unconsciously, know when a separation strategy is failing and switch to a different one [8, 156][1]. In this chapter, I present my work on estimating performance of source separation algorithms without access to ground truth (an individual recording of each source in the audio scene). This ability is important for deployment of modern source separation algorithms for applications like hearing aids. Imagine an intelligent hearing aid that uses some intelligent source separation algorithms to boost the level of important sources to the user. These algorithms will never be effective in 100% of scenarios. When they are ineffective, they may output wrong or even harmful audio to the user. If there was a mechanism that knew when an algorithm was failing to separate an auditory scene, it could improve its output by switching to better performing methods.

As an example, let's consider the problem of separating the vocals (singing voice) from a recording of a band performing a song. This is known as singing voice separation and this problem is well studied [103, 124, 173]. Many source separation algorithms that depend on a variety of primitives have been applied to this task. For instance, REPET [141] and REPET-SIM [144] assume that the mixture is made up of a repeating background source (often musical accompaniment) and a non-repeating foreground source (often the singing voice). DUET [147] and PROJET [44] assume each source comes from a unique position in space. Additionally, source separation of the singing voice can be done by leveraging harmonicity and pitch proximity to extract the main melody of a song [30, 32, 154].

Knowing how and when to use any of these algorithms requires an understanding of the mechanisms an algorithm uses and the assumptions it makes about the input mixture. This knowledge requirement is a block on widespread adoption of source separation technology by non-experts. A system that could make recommendations about what auditory situations are best handled by a particular algorithm would be of much use to spreading the adoption of audio source separation technologies.

More broadly, detecting that an approach is failing and switching to other methods is paramount in safety-critical applications. For example, a self-driving car could recognize that it is in a situation it will

[1]This chapter has supplementary audio examples: `https://pseeth.github.io/public/thesis/ch3.html`

perform poorly in and turn control over to a human driver, given it has a way to estimate its performance. Similarly, a smart hearing aid could recognize that it will not work in a certain situation and avoid piping poor audio to the wearer's ears. Especially as machine learning applications become more prevalent, being able to tell when they are not working is a requirement for deployment. To that end, several researchers have recently developed methods to estimate confidence when encountered with novel situations using ensembles, density estimation and generative adversarial networks. The basic idea of all of these methods is to detect when the input to a neural network is very dissimilar to the types of input the network was trained on.

DeVries et al. [25] does this by incorporating a measure of confidence that is learned alongside the actual labels for deep networks that perform image classification. Subramanya et al. [174] and Hendrycks et al. [61] achieve the same goal by borrowing techniques from metric learning. All of the images used to train the model belong to a distribution. If the input image is dissimilar to the images used to train the model, then it will be far away from the distribution of the training data. Using this approach, they can more reliably detect when models are being fed outlier images where it will not perform well. Liang et al. [99] tackles the same task by noticing that some perturbations of an input image that is similar to the training data have more predictable behavior when fed to the model the input images that are out-of-distribution.

My aim is to estimate the performance of a source separation algorithm without access to ground truth. While ground-truth-free estimation of source separation performance has not been extensively studied, there has been significant work in estimating errors in automatic speech recognition (ASR) [52, 64, 72]. The cues used in these works depend on particularities of speech and are not generalizable to source separation of arbitrary sources.

The most commonly used metrics for source separation quality estimation require access to ground truth separated sources. The source-to-distortion ratio (SDR) [93, 185] is, perhaps, the most widely used estimate of audio source separation quality. It requires access to the ground truth source to rate the quality of a separated source. In this chapter, I propose a method for estimating SDR without the need for ground truth separated sources.

The vast majority of source separation approaches leverage only a *single* cue to perform vocal extraction from a musical mixture. An algorithm that employs a single cue is only as robust as its cue. If the cue is absent, the algorithm produces poor results. However, as noted in Chapter 1, humans rely on many cues to parse complex auditory scenes. A system able to switch approaches as conditions change will be able to capitalize on the strengths of an ensemble of algorithms. Successful estimation of separation performance

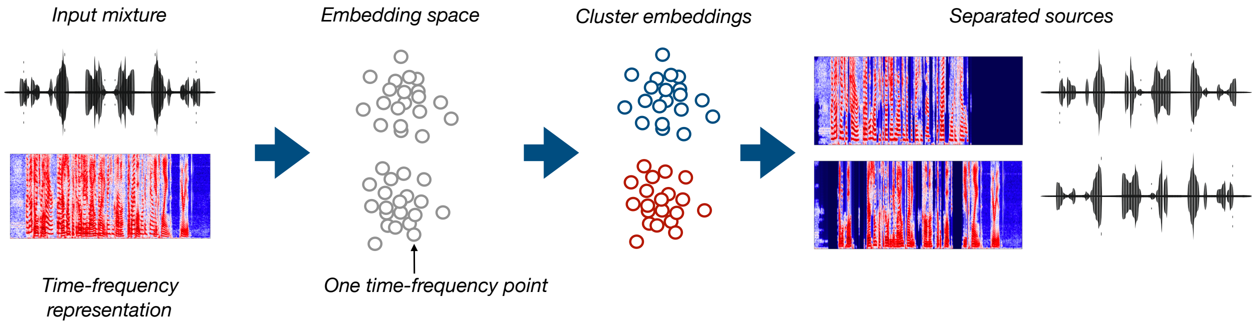

Figure 3.1. Overview of clustering-based source separation. Every point in a time-frequency representation is mapped to a feature vector in an embedding space through some method (e.g. deep learning, direction of arrival features, and so on). The points are then grouped via a clustering algorithm, like K-Means. The assignments produced by the clustering algorithm are then used to separate the sources.

will enable the creation of such a system. By estimating separation performance of multiple approaches over time, one can create a system that will switch dynamically between the approaches based on which has the highest estimated separation performance.

My approach to estimating performance can be applied to *any* clustering-based source separation algorithm, such as those based on direction of arrival [84, 147], deep learning [65], or the primitive clustering approach I proposed in Chapter 2. This approach works via an analysis of the feature space that is used to cluster and recover sources. My method differs from these others in several ways. It focuses on audio source separation, not image classification. It is not trying to classify input as out-of-distribution but rather estimate the quality of in-distribution data without ground truth. It does not leverage the characteristics of a large training set. In fact, it does not require anything to be trained and is not specifically tied to deep neural networks. Instead, it can be more broadly applied to any clustering-based separation method.

In this chapter, I present my work on estimating separation quality via cluster analysis. My method can estimate performance of a separation algorithm *without* the need to compare the separated sources to ground truth isolated sources. I first describe the method and then evaluate it across a variety of audio domains and across different clustering-based separation algorithms. This is a critical component for the overall work presented in this dissertation: learning computer audition models without access to ground truth. The computer audition models are learned from the output of primitives. However, primitives are brittle, and when they fail they tend to produce completely useless output. Avoiding learning from failure cases is done via the confidence measure presented in this chapter.

3.1. Clustering-based source separation

Many source separation methods are based on clustering. Each time-frequency point in a spectrogram is represented by some feature vector. The features can be anything - direction-of-arrival, harmonicity, or learned features. A clustering algorithm such as K-Means is then applied to the set of time-frequency points which are put in a space defined by the features for each time-frequency point, resulting in cluster assignments for each point. These assignments can either be hard assignments (e.g. as in traditional K-Means clustering), or soft assignments (e.g. as provided by soft K-Means or Gaussian Mixture Models). A soft assignment is the soft membership of a time-frequency point to some cluster and has a value between 0.0 and 1.0, rather than being only 0 (not a member) and 1 (a member), as in hard assignments. The assignments are then used as a time-frequency mask on the mixture spectrogram that will be used to recover the individual sound sources. Hard assignments result in a binary mask while soft assignments result in a soft mask. Generally, soft masking produces separated results of higher quality [75].

In Chapter 2, I covered one such clustering-based separation method that leveraged direction of arrival. Direction of arrival features - inter-aural phase delay (ϕ) and inter-aural level difference (δ) - are extracted for each time-frequency point in a mixture. This forms an embedding for each time-frequency point. Then each time-frequency point p is placed into an embedding space at location $\phi_p, \delta_p)$. K-means clustering is applied in the embedding space. Each cluster is presumed to represent a source. The underlying assumption is that all time-frequency points that share an inter-aural phase delay and level difference come from the same direction and therefore from the same source. This is just one example of clustering-based separation.

I described deep clustering [66] in Chapter 1, in which the features for every time-frequency point are learned using a deep network. The learned features are then clustered to separate sources, using K-Means. An overview of clustering-based separation algorithms can be seen in Figure 3.1. In Chapter 2, I proposed a new clustering-based separation algorithm called *primitive clustering*, where several primitive-based separation algorithms (e.g. repetition-based, melody-based) are run on a single mixture. Each algorithm makes a decision on how to assign every time-frequency point in the mixture. Their decisions are concatenated into an embedding for each time-frequency point. Clustering is then applied to the resultant embedding space, resulting in separated sources.

In every clustering-based algorithm, time-frequency points are mapped to an embedding space in which the clustering is performed. It follows that the structure of this embedding space may be related to the performance of the algorithm. For example, if the embedding space has a single natural cluster, then it is

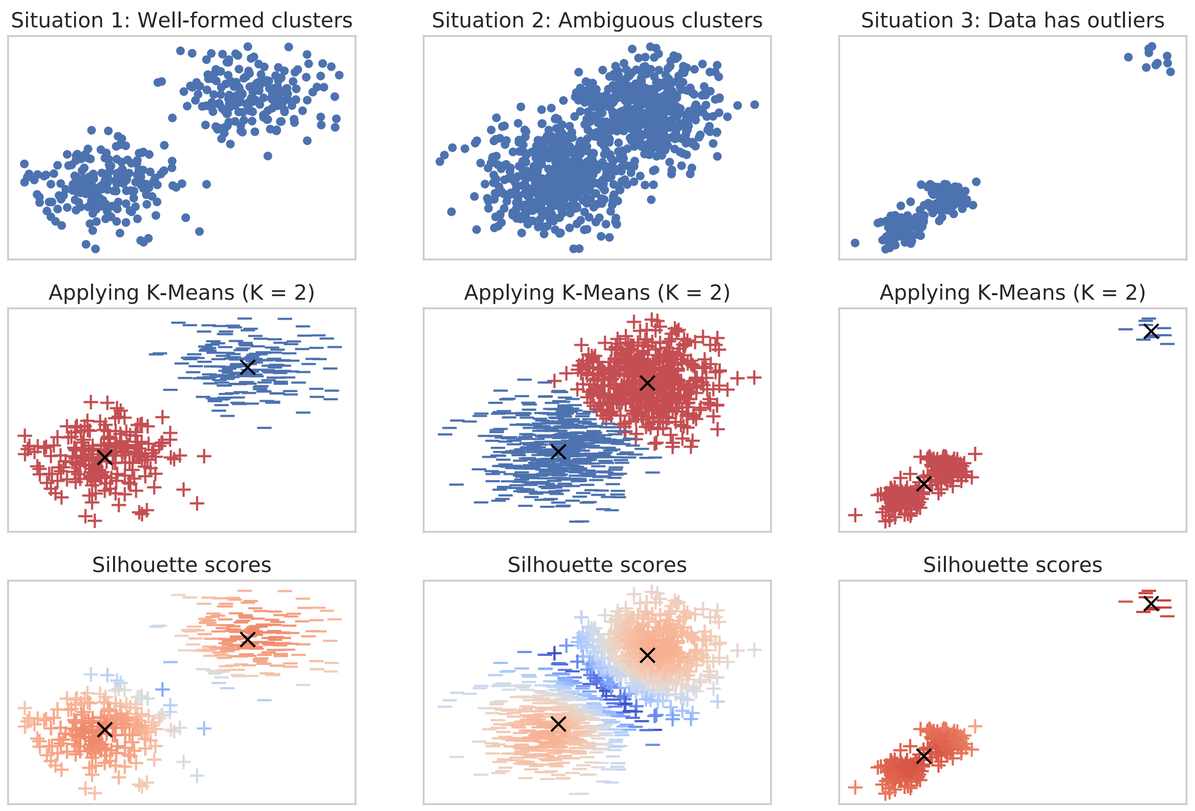

Figure 3.2. Clustering applied to different types of data. Top row: data to be clustered. Middle row: K-Means applied to the data. The red +'s indicate membership of one cluster while the blue −'s indicate membership to the other cluster. Bottom row: silhouette scores for each point given the clustering (red means a high silhouette score, and blue means a low silhouette score).

unlikely that any clustering algorithm would recover distinct sources, even if there are two audible sources in the mixture. In this case, the method used to construct the embedding space wasn't useful for clustering. If an embedding space has two natural dense clusters, as in the first column of Figure 3.2, then applying a clustering algorithm to the space is more likely to have useful output. However, applying clustering can fail for a variety of reasons, like if there are outliers in the data, as in the right-most column of Figure 3.2. By analyzing the embedding space, I can estimate the performance of any clustering-based source separation algorithm without the need to compare to ground truth sources. My approach has three components: the silhouette score, posterior strength, and the size of the smallest cluster. I now describe each of these components.

3.2. The silhouette score

The silhouette score is a way to measure the quality of labels produced by any clustering algorithm. Rousseeuw [151] introduced the score as a way to easily visualize the quality of a clustering alongside the output of that clustering. It produces a score for every point in a dataset. To compute, let us first assume you have a partition of your dataset $X = \bigcup_{k=1}^{K} C_k$ into K clusters. The basic idea is to compare each point's distance to all the points in its own cluster versus the distance to all the points in the other clusters. More formally, for a data point x in cluster C_k, compute:

$$a(x) = \frac{1}{|C_k| - 1} \sum_{y \in C_k, x \neq y} d(x, y) \tag{3.1}$$

This is the mean distance (using a distance function d) between x and all other points in C_k. Next, we compute the mean distance between x and all the points in the other clusters:

$$b(x) = \min_{j \neq i} \frac{1}{|C_j|} \sum_{y \in C_j} d(x, y) \tag{3.2}$$

The inner term computes the mean distance between a point $x \in C_k$ and points in some other cluster C_j. The min operator computes the minimum distance (e.g. the neighboring cluster to C_k). Finally, the silhouette of $x \in C_k$ is computed by:

$$s(x) = \frac{b(x) - a(x)}{\max(a(x), b(x))} \text{ if } |C_k| > 1 \tag{3.3}$$

If $|C_k| = 1$, then $s(x) = 0$. $s(x)$ ranges from -1 to 1. Negative values indicate that the point is classified into the wrong cluster - the neighboring cluster would be better, as $b(x) < a(x)$. Values close to 1 indicate that $a(x) \ll b(x)$ - the *intracluster* distance is much smaller than the *intercluster* distance. Values close to 0 indicate that $a(x) \approx b(x)$ - the distance of x to points in its own cluster is about the same as the distance of x to points in neighboring clusters. The silhouette score of each point x indicates the strength of its membership to the cluster that it is in.

Figure 3.2 shows three embedding spaces with K-Means clustering applied to each of them. In the bottom row, I show the silhouette scores for each sample. In the first situation, the silhouette score for most of the samples is high (colored red). A few points that fall on the edges of the clusters have lower silhouette scores (colored blue). In the second situation, the two clusters are much closer together, making the labels for many of the points in the middle far more ambiguous. The corresponding plot with the silhouette scores for this situation reflects this, with many more points colored blue. By taking the mean silhouette score across all of the points in the data, we can measure the overall quality of the clustering.

However, the silhouette score has two major issues. First, notice that in the third situation in Figure 3.2, the clustering is thrown off by a few outlier data points. There are two relatively well-formed clusters in the bottom left of the plot that are grouped as one cluster, while the outliers are grouped as their own cluster. This results in generally high silhouette scores (the inter-cluster distance is very high). Here, the silhouette score does not capture the failure that has occurred.

Second, to compute the mean silhouette score, one must compute $a(x)$ and $b(x)$ in Equation 3.3 for every single point in the dataset before one can compute the mean silhouette score. While this works for the data in Figure 3.2, it is intractable for the data I am concerned with - an embedding space containing every time-frequency point in a time-frequency representation. These spaces can easily contain one million or more points - a 3 minute audio waveform with a sampling rate 44100 Hz, converted into a time-frequency representation will contain 31 million points[2]! Computing the silhouette for every point in the corresponding embedding space is intractable. I offer a few solutions to overcoming these issues.

3.2.1. Computing the silhouette score via sampling

To solve the intractability issue of computing the silhouette, we can instead use sampling to compute the mean silhouette score. To do this, simply sample N points from the dataset and compute the silhouette of each point in the subset. The cluster assignments are computed on the full dataset. The silhouette is computed by comparing the N points in the subset to the other N points in the subset, making the computational cost of getting the mean silhouette score far more tractable, instead of using the full dataset. Similar sampling strategies have been employed to speed up other clustering-related problems [97, 203]. This makes the computation far more tractable - what required a $31,000,000 \times 31,000,000$ distance matrix now only requires an $N \times N$ distance matrix. Of course N should be chosen to be sufficiently large such that the sampled mean silhouette score is the close to the actual mean silhouette score. In practice, $N = 1000$ was sufficient to get accurate mean silhouette scores. $N = 1000$ was chosen as it seemed to work for short

[2]Using an STFT with a filter size of 2048 samples and a hop size of 512 samples

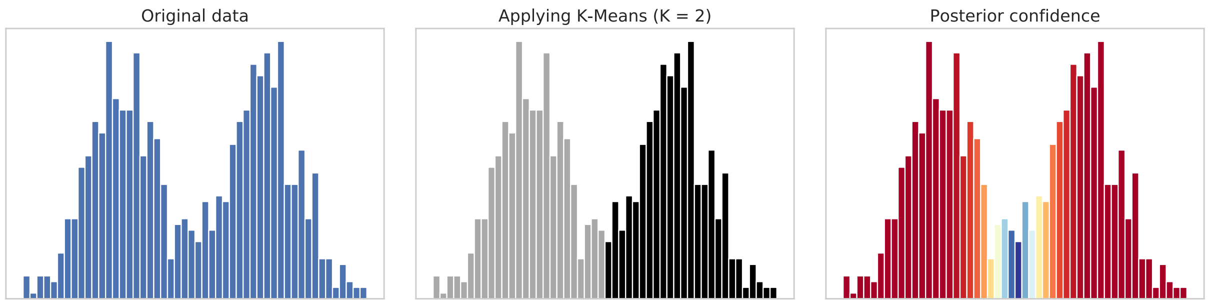

Figure 3.3. An illustration of the confidence measure that leverages the strength of the posteriors. On the x axis are the confidence values of points in the embedding space. On the y axis is the number of points in the embedding space at a specific value (e.g. a histogram). One can observe two clusters, indicated by the two modes in the histogram. Points that fall in between the clusters have unsure posteriors (roughly equal assignment) and thus low confidence (blue) while points that are well assigned to a cluster have high confidence (red).

mixtures, reflecting the actual silhouette score within some margin of error. Further, $N > 1000$ required excessive computational resources for little gain in accuracy. The final silhouette score is then calculated via:

$$S(X) = \frac{1}{N}\sum_{i}^{N} s(x_i) \tag{3.4}$$

where x_i are the points in a subset of size N of the full dataset X. This is the mean silhouette score across all of those points in the subset of size N. This component produces a single number that estimates the performance for the entire mixture. However, now we only have a single number, rather than a number for every point in the dataset. The next component, which computes the posterior strength, addresses this.

3.3. Posterior strength

The next component of the confidence measure is an analysis of the posterior assignments produced by the clustering algorithm. In clustering algorithms, every point is assigned to a cluster either in whole or in part, depending on whether the clustering uses hard assignments or soft assignments. Here, we specifically use soft K-Means for clustering. For every point x_i in a dataset X, soft K-Means produces $\gamma_{ik} \in [0, 1]$, which indicates the membership of the point x_i in some cluster C_k. γ_{ik} is also called the *posterior* of the point x_i in regards to the cluster C_k. The closer that γ_{ik} is to 0 (not in the cluster) or 1 (in the cluster), the more sure the assignment of that point. If, on the other hand, γ_{ik} is only slightly greater than $\frac{1}{K}$, then we do not have a lot of confidence that the point definitely belongs to that cluster. The goal of this confidence measure is to quantify how strongly a point x_i is assigned to any cluster. If it is weakly assigned to all clusters (the maximum posterior is around $\frac{1}{K}$), then the confidence measure should return 0. If it strongly assigned to some cluster (the maximum posterior is around 1.0), then the confidence measure should return 1. More formally, for a point x_i with corresponding γ_{ij} for $j \in [0, 1, ..., K]$, we compute:

$$P(x_i) = \frac{K(\max_{j \in [0,...,K]} \gamma_{ij}) - 1}{K - 1} \tag{3.5}$$

where K is the number of clusters, and $P(x_i)$ is the confidence measure for a particular point. I call this confidence measure *posterior strength*, as it captures how strongly a point is assigned to any of the K clusters. The equation maps points that have a maximum posterior of $\frac{1}{K}$ (equal assignment to all clusters) to a confidence of 0, and maps points that have a maximum posterior of 1 to a confidence of 1. It is easy to compute for every point in the space as we already have γ_{ij} from applying the clustering algorithm.

A visualization of this is shown in Figure 3.3. On the left is a histogram of original one-dimensional data which can be visually grouped into two clusters. In the middle is the application of K-Means clustering to this data, which discovers the two visible clusters. On the right is the strength of the posterior assignments using Equation 3.5 for each point in the dataset. Points that fall between the two natural clusters have low posterior assignments, indicated by the color blue. Points that do not fall in the middle have high posterior assignments, indicated by the color red.

3.3.1. Size of the smallest cluster

The third component of the confidence measure is looking at the size of the smallest cluster discovered by the clustering algorithm. The size of a cluster is determined by the fraction of all the data points assigned to the cluster. Often, clustering algorithms will fail due to a small cluster of outliers that are far away from the majority of the points in the data. In the third column of Figure 3.2, I show a case where clustering has failed because a small set of outliers has thrown off the K-Means algorithm. This confidence measure offers a simple way to detect when this has occurred. It is not always appropriate to use, as sometimes small clusters are to be expected, as in the problem of separating vocals from accompaniment. However, in other problems like separating two overlapping speakers that are likely talking the same amount, it is expected that all of the clusters (two, in this case) will have considerable size.

More formally, given a partition of a dataset X into clusters $X = \bigcup_{k=1}^{K} C_k$, we can compute the size of each cluster $|C_k|$. Then, we can compute:

$$R(X) = \frac{\min_{j \in [0,\ldots,K]} |C_j|}{|X|} \tag{3.6}$$

This is the size of the smallest cluster found expressed as a fraction of the total dataset size and is between 0 and 1. Note that incorporating this measure is *optional.* For some applications, it is not desirable as the sizes of the two clusters may naturally be unbalanced (e.g. vocals vs accompaniment in music mixtures). In the third column of Figure 3.2, the outliers are detected as a separate, very small cluster. This results in a measure close to 0 when computing Equation 3.6, resulting in low confidence according to this measure.

3.4. Combining the components

The three confidence measures I have presented - the silhouette score, the posterior strength, and the size of the smallest cluster - are combined to create a single confidence measure via a product. Specifically:

$$\text{CONFIDENCE}(x_i, X) = S(X) * P(x_i) * R(X) \tag{3.7}$$

All three measures are needed as they capture different aspects of the clustering:

(1) The silhouette score is an overall measure that captures how much separation exists between the clusters (intercluster distance) and how dense the clusters are (intracluster distance).
(2) The posterior strength captures the confidence we have in the assignment of every point in the space.
(3) The size of the smallest cluster allows us to catch cases when a clustering has been thrown off by outliers or resulted in a very small cluster.

The three measures are combined via a product because we want high confidence to only occur when all three measures have high confidence. This makes the confidence measure more strict in a way that other approaches to combining the measures might not, like averaging the three components. When we have high confidence, we can interpret this to mean there is a clustering with great intercluster distance and

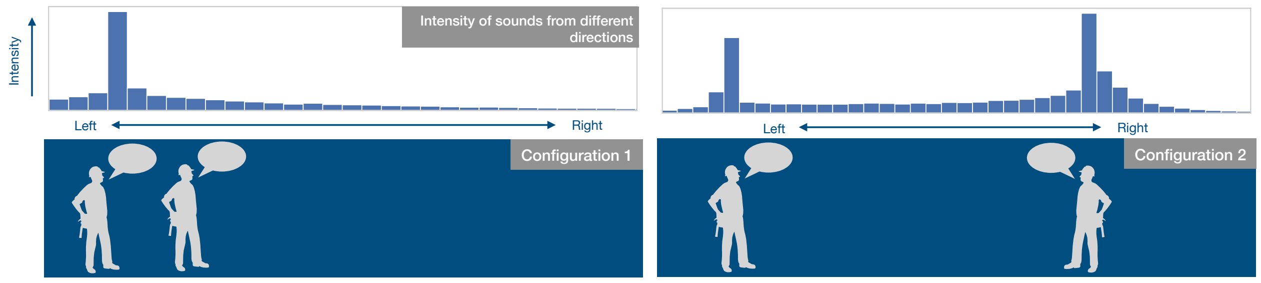

Figure 3.4. Two mixture configurations. *Configuration 1* is a mixture in which the two speakers are both coming from the left. *Configuration 2* is a mixture in which the two speakers are coming from different directions. The difference between the configurations can be seen in the direction of arrival features (top row) that are used to separate the mixture.

low intracluster distance (the silhouette score), the strength of assignment for every point is very high (the posterior strength), and that no spurious clusters have been found (the size of the smallest cluster).

Note that the confidence score is calculated for every time-frequency bin in Equation 3.7. To derive a single confidence score for the entire mixture, I simply take the mean of the confidence scores for the time-frequency bins in the mixture, with one important caveat. The mean is only taken across a subset of all of the time-frequency bins, namely the top one percent of all bins in a mixture by loudness. In a time-frequency representation with millions of points, running the confidence measure on a small set makes it tractable. The reasoning for selecting the highest-amplitude time-frequency points is that we care mostly about the assignment of loud time-frequency points is that louder time-frequency points have a greater perceptual effect on the separation quality, so the confidence we have in their assignments matters more.

3.5. Estimating performance of direction of arrival clustering

Recall the separation algorithm that used direction of arrival detailed in Section 2.1. The algorithm works well when sounds are coming from very different directions but poorly when sounds are coming from similar directions. Figure 3.4 shows two mixture configurations. In the configuration where the two speakers are located far from each other, there are two clear peaks in the corresponding visualization. In the other configuration, where the speakers are close, there is a single peak. When applying the confidence measure, we can see that the first configuration will result in high confidence while the second configuration will result in low confidence. This section is devoted to showing that this relationship between the confidence measure and ground truth performance holds across many mixtures, allowing me to estimate the performance of direction of arrival clustering without ground truth.

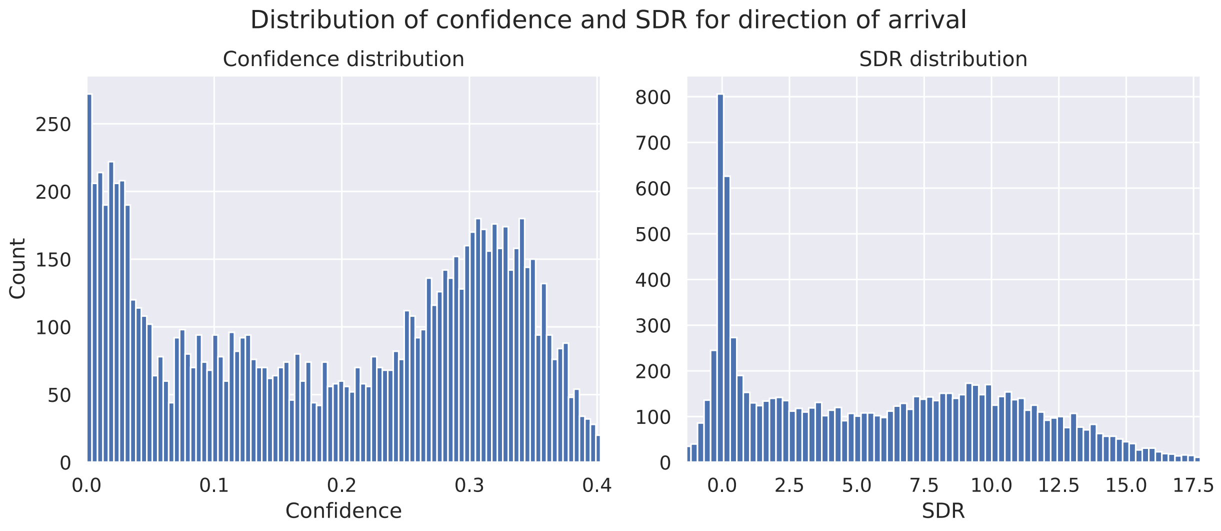

Figure 3.5. Distribution of confidence measure (i.e. predicted performance) and source-to-distortion (i.e. actual performance) for the direction of arrival separation algorithm as applied to 5000 mixtures in the validation set. On the right, we can see that the algorithm actually fails about half the time (the peak at 0 dB SDR). We also get a peak at 0.0 confidence.

The dataset used here is from the publicly available[3] spatialized version of the Wall Street Journal mix dataset with two speakers (wsj0-2mix) [65, 196]. This dataset is created by randomly mixing the speakers at random locations in synthetic rooms in reverberant and anechoic conditions. I use the both anechoic and reverberant versions of the dataset here. The speakers are panned at random (sometimes overlapping) angles. In the reverberant condition, the decay time is between 0.2 and 0.6 seconds. There are 20000, 5000, and 3000 two-speaker mixtures for training, validation, and testing.

Each mixture was recorded in 8 channels. Recall that the direction of arrival algorithm only operates on two of the eight channels. To construct a two-channel mixture to apply the direction of arrival algorithm to, I always select the first channel of the 8, and then pick a random other channel from the remaining 7 channels. I present results on the *validation* data - 5000 mixtures - from this dataset, so as to save the test data for experiments in Chapter 4. Performance of the separation algorithm is measured in terms of source-to-distortion ratio, as discussed in Section 2.2.5.2.

In a mixture from this dataset, there is no guarantee the direction of arrival separation method will work at all. In some mixtures, the speakers are coming from very different directions while in others they are either coming from similar directions or one speaker is much louder than the other. In the latter sort of

[3]`http://www.merl.com/demos/deep-clustering`

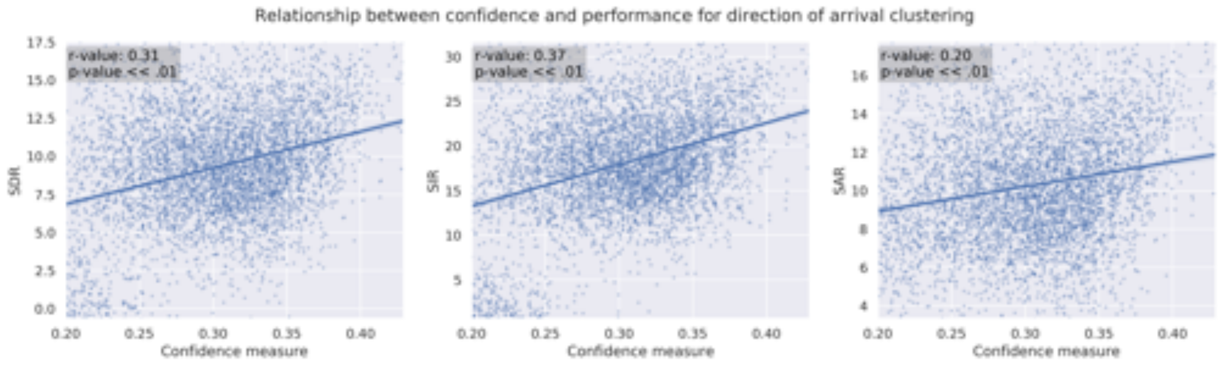

Figure 3.6. Relationship between the proposed confidence measure and three actual performance measures - SDR/SIR/SAR. Each dot represents a separated source from the WSJ0 spatialized anechoic dataset. The data here is filtered to only show the relationship for sources that obtained a confidence $>$ .2. This is because the distribution of performance for this dataset is bimodal. Here, I only show the regression on mixtures where the separation actually worked, so as not to be misleading. After filtering, there are 5050 remaining sources, and thus 5050 dots in these scatter plots. The blue line is the line of best fit found via linear regression. The p-values (measured via a Wald test [54]) and r-values for each regression are overlaid on each scatter plot.

situation, the direction of arrival algorithm will not work at all. Figure 3.5 shows the distribution of SDR (right) and the distribution of the confidence measure for the direction of arrival algorithm applied to each of the 5000 mixtures in the validation data. Getting an SDR around 0 dB indicates that the algorithm has failed to separate a mixture entirely. This occurs for around half the validation mixtures as is visible by the peak at 0 dB in the distribution of SDR shown in the figure. On the left in the figure, I show the distribution of confidence across those same mixtures. The overall confidence for each mixture is computed using the procedure described in Section 3.4.

We get a very similar looking distribution of confidence as we do for SDR, with a peak around 0.0 confidence, and a similar, albeit smaller peak, at the higher end (around 0.3 confidence and 10 dB SDR). The overall performance for the direction of arrival algorithm on this dataset is 5.84 dB SDR, 10.84 dB SIR, and 12.62 dB SAR. The next section shows that the distributions are not just similar but correlated.

Figure 3.6 shows the actual relationship between confidence and performance. Performance is shown along three measures - source-to-distortion ratio (SDR), source-to-interference ratio (SIR), and source-to-artifacts ratio (SAR). To more accurately show the relationship between confidence and SDR, I remove all mixtures that had below .2 in confidence, only showing the regression between the remaining points. The reasoning for this is that, since the distribution of confidence and SDR are bimodal, as seen in Figure 3.5, it is more informative to see the relationship on the mode where the separation actually worked. The regression

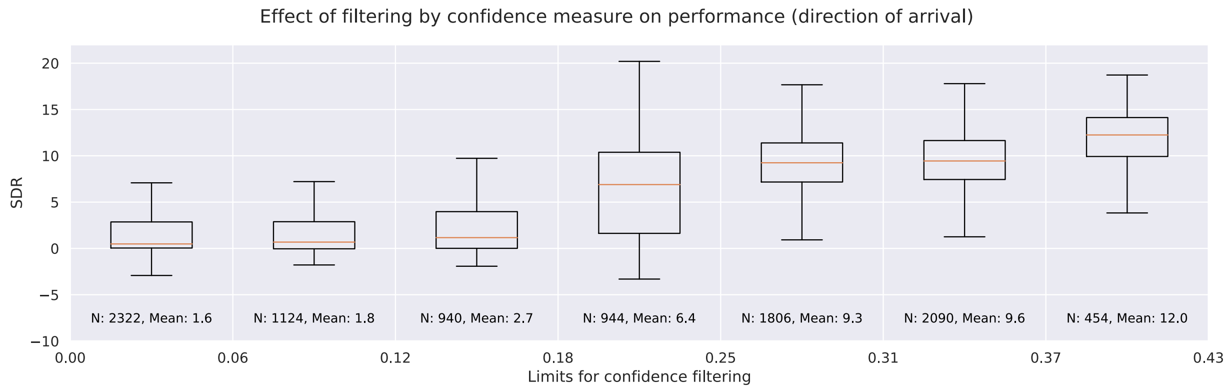

Figure 3.7. Filtering the separated sources using confidence. Each box-plot shows the distribution of the separated sources that had an associated confidence between the bounds on the x axis. For example, the first box plot on the left shows the distribution of SDR for sources that had confidence between 0.00 and 0.06. As the limits increase, the distribution moves upwards, obtaining sources with higher average SDR.

between confidence and SDR/SIR/SAR is statistically significant, obtaining a p-value $\ll .01$ and an R^2 value of .31 for SDR, .37 for SIR, and .20 for SAR. This shows that we can reliably predict performance for direction of arrival clustering using the proposed confidence measure.

Another way we can visualize the relationship between confidence and actual performance is by binning the separated sources using confidence. For example, I can take every separated source that has an associated confidence between 0.12 and 0.18 and show the distribution of SDR for those sources. I do this in Figure 3.7. As the bounds for binning the separated sources by confidence increase, the SDR distribution of the separated sources increases. The mean SDR across all of the mixtures is 5.84 dB. As we increase the confidence bounds for binning separated sources, we can find subsets of all of the separated sources that have very high separation quality. At the highest bin, where confidence is between .37 and .43, we find 454 separated sources that have a mean SDR of 12 dB! This is very close to oracle performance (using the ground truth source to construct a mask on the mixture) - 12.5 dB [195]. The confidence measure allows us to find high-quality separated sources in an unsupervised way by using it as a selection mechanism on a collection of separated sources. Separated sources that come from mixtures where the clustering algorithm had high confidence generally have high separation quality. This will become important in Chapter 4, where we use the confidence measure in different ways to bootstrap a deep source separation model.

So far, I have shown that the confidence measure can predict the performance of the direction of arrival algorithm in anechoic mixtures. It is well known that the performance of the direction of arrival separation algorithm is degraded when reverberation is added to the scene. Reverberation causes the direction of

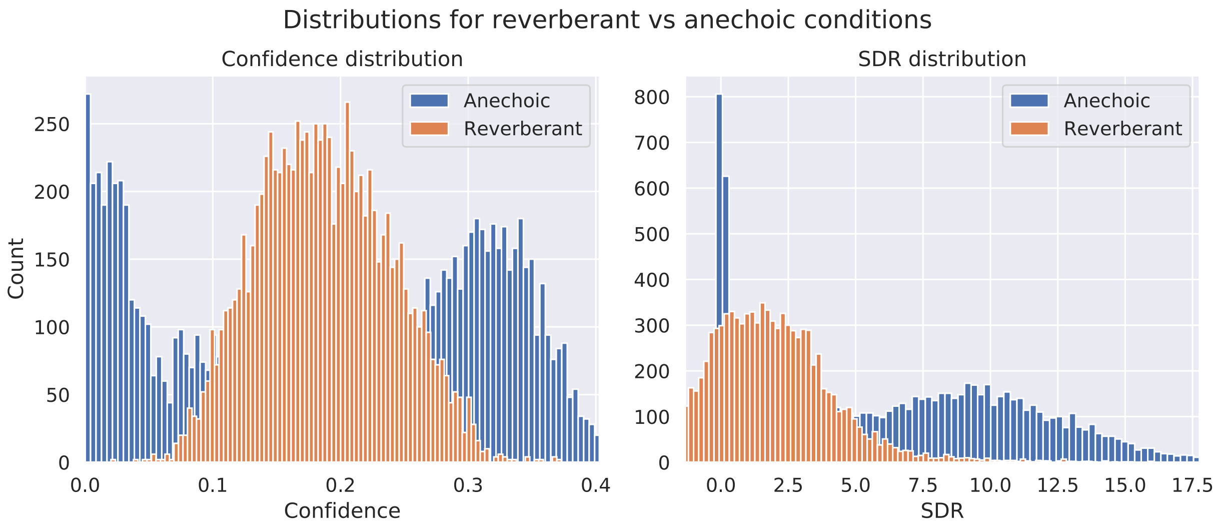

Figure 3.8. Comparison of distributions for SDR and confidence for reverberant (orange) vs anechoic (blue) conditions. The direction of arrival algorithm has difficulty with the reverberant condition, as can be seen on the right graph. This is also reflected by lower confidence for the reverberant mixtures, as seen on the left graph.

arrival features to blur slightly. Ideally, the confidence measure would be lower for reverberant conditions vs anechoic conditions, as increased reverberation results in lower performance. In Figure 3.8, I show the distribution of confidence and SDR for the reverberant data overlapped with that of the anechoic data. It can be seen, that outside of the lower mode where the algorithm has failed completely on the anechoic data, that the reverberant sources have lower confidence than the anechoic sources, reflecting the change in mixture conditions.

I have shown that the confidence measure correlates well the actual ground truth performance for a direction of arrival clustering algorithm. The relationship between confidence and SDR/SIR/SAR is statistically significant, as shown in Figure 3.6. I can also use the confidence measure as a way of discovering high-quality separated sources in an unsupervised way, as shown in Figure 3.7. The confidence measure can thus be used a reliable proxy for actual ground truth performance.

3.6. Estimating performance of primitive clustering

The confidence measure I have proposed can not only estimate the performance of direction of arrival clustering but other clustering-based separation algorithms as well. In this section, I show the relationship between the confidence measure and *primitive clustering*, which I introduced in Chapter 2, Section 2.3.1. Primitive clustering is a framework for combining multiple primitive source separation algorithms called primitive clustering. I showed that one such combination of primitive algorithms out-performed any of the input primitives in isolation.

Recall that primitive clustering constructs an embedding space where every time-frequency point is mapped to an embedding space using the soft-mask constructed via a primitive separation algorithm. For example, the harmonic/percussive separation algorithm constructs a soft mask for every point where 0.0 indicates that the time-frequency point belongs to a percussive source and 1.0 indicates that the time-frequency point belongs to a harmonic source. This is done for several primitive separation algorithms. In Chapter 2, I use micromodulation, repetition, harmonic/percussive timbre, and time and pitch proximity as the input primitives. Together, these map every point to a 4-dimensional embedding space. Here, I apply the confidence measure to this embedding space and show that it correlates to actual ground truth performance measured via SDR/SIR/SAR.

Recall that in the confidence measure there is a component that measures the size of the smallest cluster. While that works for speech mixtures, it is not relevant for music mixtures and is thus not used here. Only the first two components - the silhouette score and the posterior strength - are used. The reasoning for this is that small clusters are possible in musical mixtures, as the number of time-frequency bins that could belong to the vocals source is often much smaller than those that could belong to the accompaniment source. In some mixtures, there may be only a few seconds of vocals in the entire song!

I apply the same primitive clustering algorithm described in Chapter 2, Section 2.3.1 that consisted of four primitive algorithms. The four algorithms were based on micromodulation, repetition, time and pitch proximity, and harmonic/percussive timbre. I applied the algorithm to the mixtures in the MUSDB training set [173], which consists of 100 tracks with vocals, bass, drums, and other stems for each track. The bass, drums, and other tracks are combined to form the accompaniment. The task was again vocals/accompaniment separation. For each mixture, I obtained vocals and accompaniment estimates using primitive clustering and computed the overall confidence measure using the procedure described in Section 3.4, as well as the actual performance metrics - SDR, SIR, and SAR.

The correlation between the confidence measure and actual performance is shown in Figure 3.9. We observe a strong correlation between SDR, SIR, and SAR for the accompaniment, obtaining a p-value $\ll .01$ and an R^2 value of .56 for SDR, .62 for SIR, and .39 for SAR. We also see a correlation for the vocals sources for SDR and SIR, but with lower R^2 values. However, for vocals SAR, we see no statistically significant correlation.

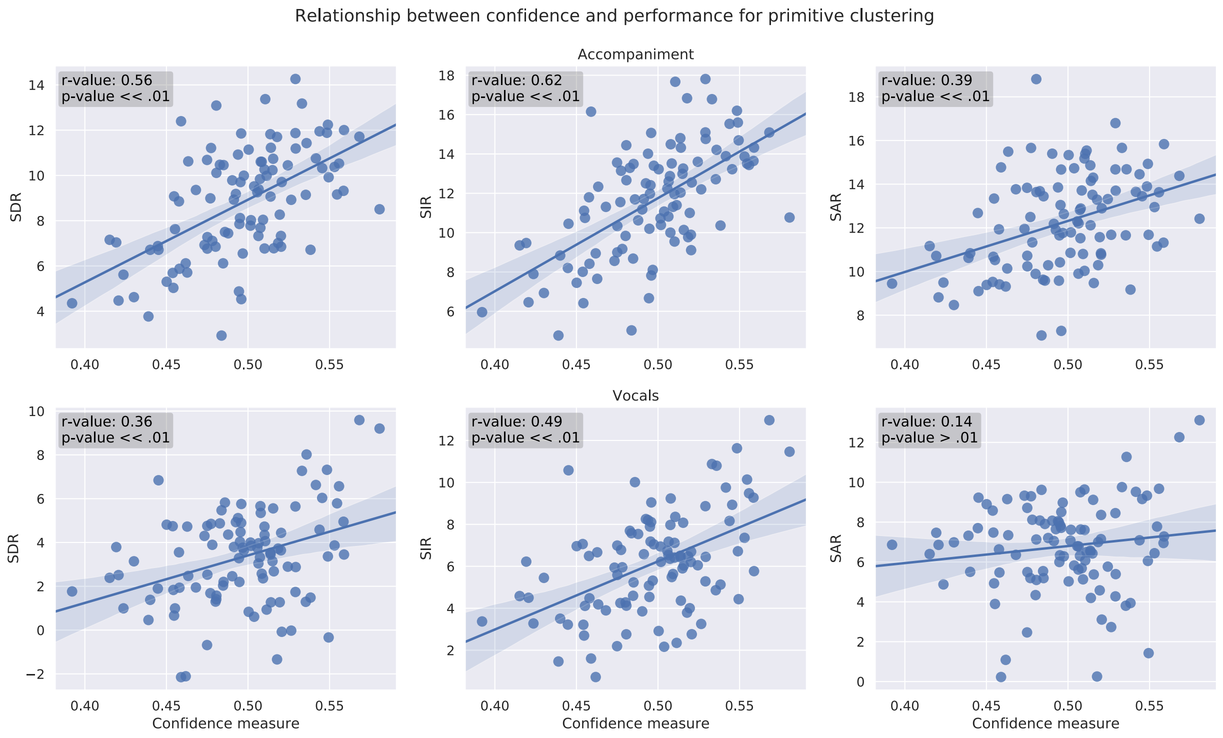

Figure 3.9. Relationship between confidence measure and actual performance for music mixtures, using primitive clustering to separate each mixture. There are 100 points in each plot, representing 100 separated sources that are either accompaniment (top row) or vocals (bottom row). We see a strong correlation between confidence and actual performance for accompaniment. We see a similar correlation for SDR and SIR for vocals, but no correlation for SAR. The blue line is the line of best fit found via linear regression. The p-values and r-values for each regression are overlaid on each scatter plot.

For primitive clustering, we can interpret the confidence measure in an interesting way. Recall that the K-Means is applied to primitive clustering to compute the masks. The means used are fixed and are the points where all of the primitive algorithms agree on how to label a time-frequency point. That is, they all agree that a time-frequency point belongs to the accompaniment (0 in the soft mask) or to the vocals (1 in the soft mask). When confidence is high, it means that both components of the confidence measure are high - the silhouette score indicates that the intercluster distance is high and the intracluster distance is low, and the posterior strength indicates that most of the points are assigned strongly to one cluster. Thus, the confidence measure is a proxy for how much the primitive separation algorithms agree on how to separate the auditory scene. When the confidence measure is high, the algorithms all agree with one another, and separation quality is high as seen in Figure 3.9. When it is low, the algorithms disagree with one another and separation quality is also low.

3.7. Estimating performance of deep clustering

Finally, the confidence measure can be applied to estimate, without ground truth to compare to, the performance of source separation based on deep clustering. Deep clustering, which I presented in Chapter 1, Section 1.6.3.1, works by learning a mapping from time-frequency points to a D dimensional embedding space where time-frequency points from the same source are near each other in the embedding space. The embedding for each time-frequency point is learned via a deep network. The deep network maps each point in the time-frequency representation ($T \times F$) to a D-dimensional embedding space ($T \times F \times D$. The D-dimensional embedding space is then clustered to recover the sources, under the assumption that a cluster represents a single sound source.

In this section, I apply the confidence measure to the deep embedding space learned by deep clustering networks. I show that the confidence measure correlates with performance for two deep clustering networks. One network is trained to separate mixtures of speakers and the other is trained to separate vocals from accompaniment in music mixtures. Despite how different these two audio domains are, the confidence measure can still be applied to both spaces to estimate the performance of deep clustering.

The first network we will deal with is a deep clustering network that is trained to separate vocals from accompaniment in musical mixtures. The network architecture is a stack of 4 bidirectional long short-term memory layers with 300 hidden units in both directions followed by a linear layer that maps each point to a 20-dimensional embedding. The input to the network is a log-magnitude short-time Fourier transform (STFT) with a filter length of 2048 and a hop length of 512. This input is very high-dimensional and also contains *linear* frequency spacing. To lower the input dimensionality to make the network more efficient, I project the STFT to a log-frequency spacing by using a mel-frequency projection [74]. The 2048 input frequencies are projected down to 300 mel frequencies. After this, the features are fed to a instance-normalization layer [180]. This is to shift the range of the inputs (which are between -80 and 0, due to the log-magnitude transformation) to a better range for the learning process. The instance-normalization layer makes it such that every example mixture has a mean of 0 and a variance of 1. This network architecture is standard in the deep clustering literature and is shown to achieve good performance on many different separation tasks [75, 104, 195].

The network is optimized using the weighted deep clustering loss function [195]:

$$\mathcal{L}_{\text{DC},W}(V,Y) = \|W^{1/2}(VV^T - YY^T)W^{1/2}\|_F^2 \tag{3.8}$$

$$= \sum_{i,j} w_i w_j [\langle v_i, v_j \rangle - \langle y_i, y_j \rangle]^2, \tag{3.9}$$

This loss function optimizes the output embeddings of the network to match the ground truth affinity of each time-frequency point. I trained the network for 100 epochs on 20000 15 second training mixtures that are constructed using ground truth sources from the MUSDB training dataset [173]. The training mixtures had a sample rate of 44100 Hz. During training, I select 400-frame excerpts from each mixture (about 4.6 seconds). I used the Adam [87] optimizer with a learning rate of 1e-3 and a batch size of 40. Validation loss was tracked on a separate set of mixtures. If the validation loss did not improve for 10 consecutive epochs, the learning rate was halved.

Figure 3.10 shows the relationship between the confidence measure and actual performance on the MUSDB test set. We observe a correlation between confidence and performance for the accompaniment sources, with significant p-values, and R^2 values of .46 for SDR, .56 for SIR, and .38 for SAR. We do not observe as strong of a correlation between confidence and SDR for vocals, however, but we do for SIR. We do not find any correlation between confidence and SAR for vocals, however, similar to the results for primitive clustering.

To show that the confidence measure also works for a deep clustering network that is trained for a different task, I trained a deep clustering network to separate speech mixtures. The setup of this network was very similar to that of the music separation network with a few notable differences. There is no mel-projection layer in this network. The filter length for taking the STFT was 512 (instead of 2048) and the hop length was 128 (instead of 512). The training mixtures came from the spatialized WSJ0-2Mix (2 speakers in every mixture) anechoic dataset described in Section 3.5. The 20000 mixtures had a sample rate of 8000 Hz. As the mixtures are multichannel, I selected only the first channel for training. I selected a 400-frame excerpt from each mixture during training. The learning rate was $1e-3$, the optimizer was Adam, and the batch size was 40. The network was trained for 50 epochs.

The relationship between the confidence measure and actual performance is shown in Figure 3.11. The regression between the confidence measure and performance is statistically significant across all three performance measures. This means that, using this confidence measure, we can reliably predict the performance

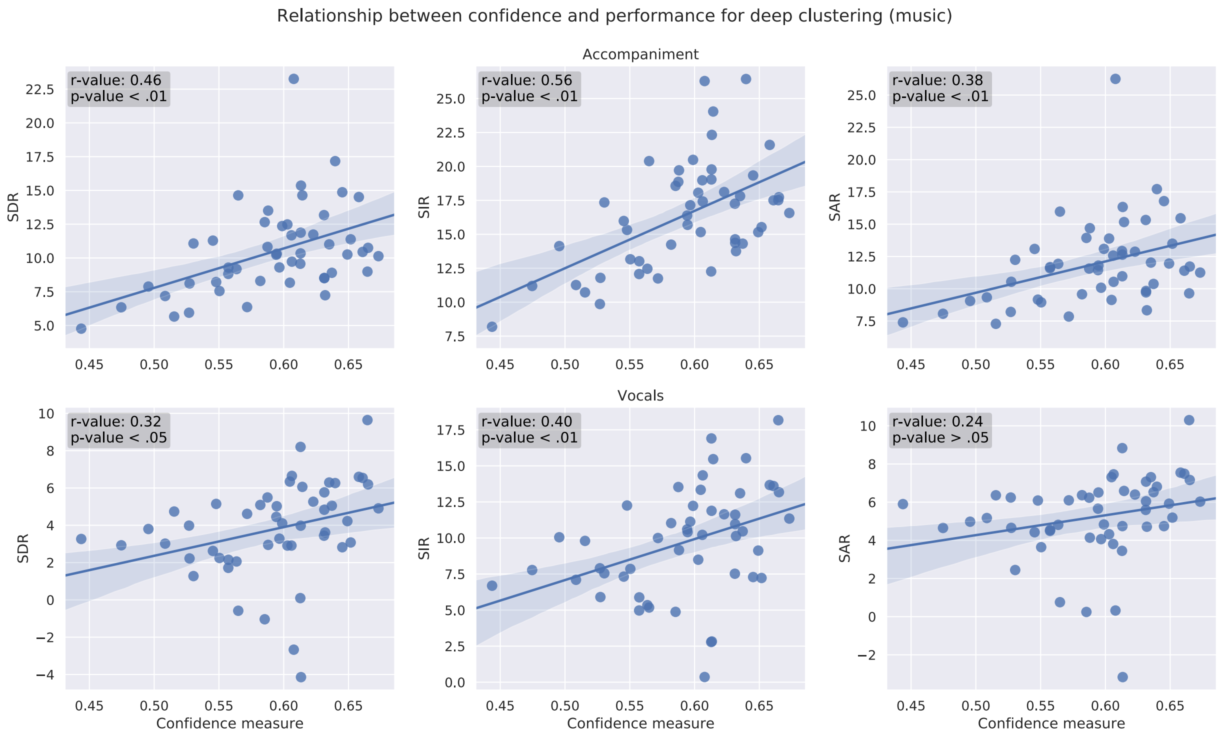

Figure 3.10. Relationship between confidence measure and actual performance for music mixtures, using deep clustering to separate each mixture. There are 50 points in each plot, representing 50 separated sources that are either accompaniment (top row) or vocals (bottom row). We see a correlation between confidence and actual performance for accompaniment. We see a similar correlation for SIR for vocals, but a weaker correlation for SDR, and no correlation for SAR. The blue line is the line of best fit found via linear regression. The p-values (measured via a Wald test [54]) and r-values for each regression are overlaid on each scatter plot.

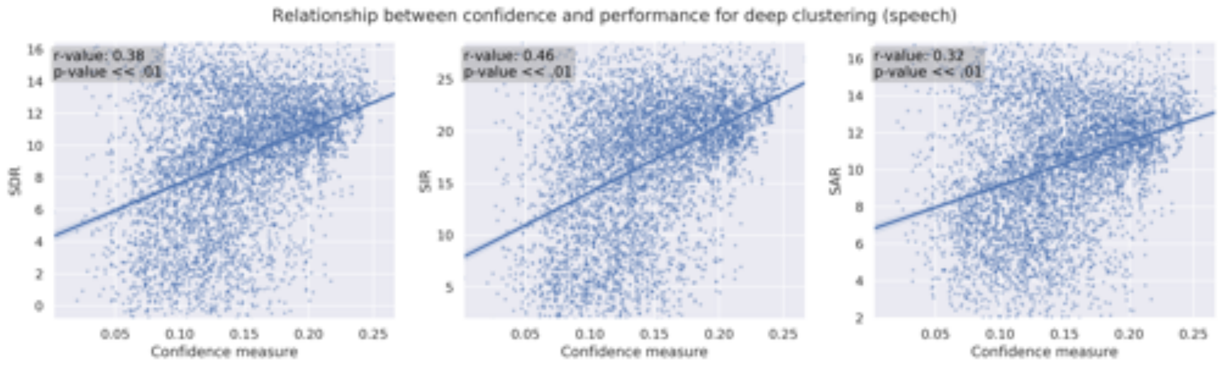

Figure 3.11. Relationship between the proposed confidence measure and three actual performance measures - SDR/SIR/SAR - for deep clustering. Each dot represents a separated source from the WSJ0 spatialized anechoic testing dataset ($N = 3000$). The blue line is the line of best fit found via linear regression. The p-values (measured via a Wald test [54]) and r-values for each regression are overlaid on each scatter plot.

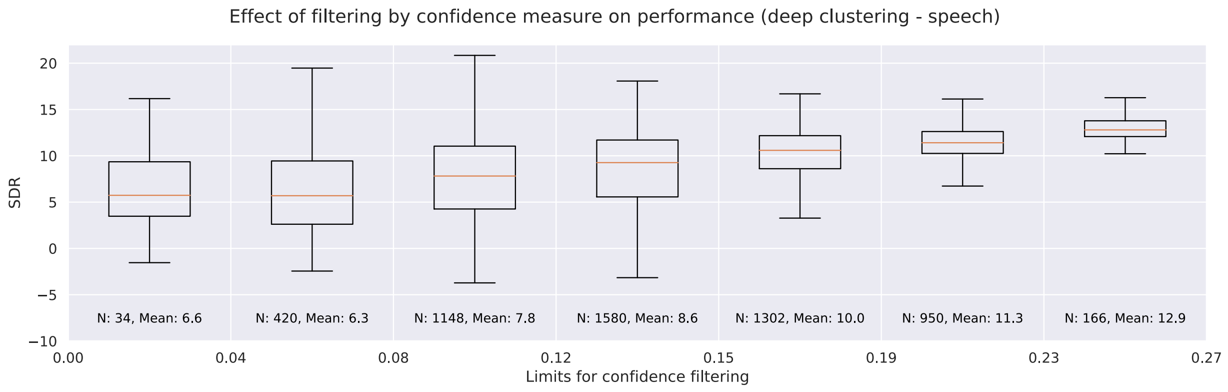

Figure 3.12. Filtering the separated sources using confidence. Each box-plot shows the distribution of the separated sources that had an associated confidence between the bounds on the x axis. For example, the first box plot on the left shows the distribution of SDR for 34 sources that had confidence between 0.00 and 0.04. As the limits increase, the distribution moves upwards, obtaining sources with higher average SDR.

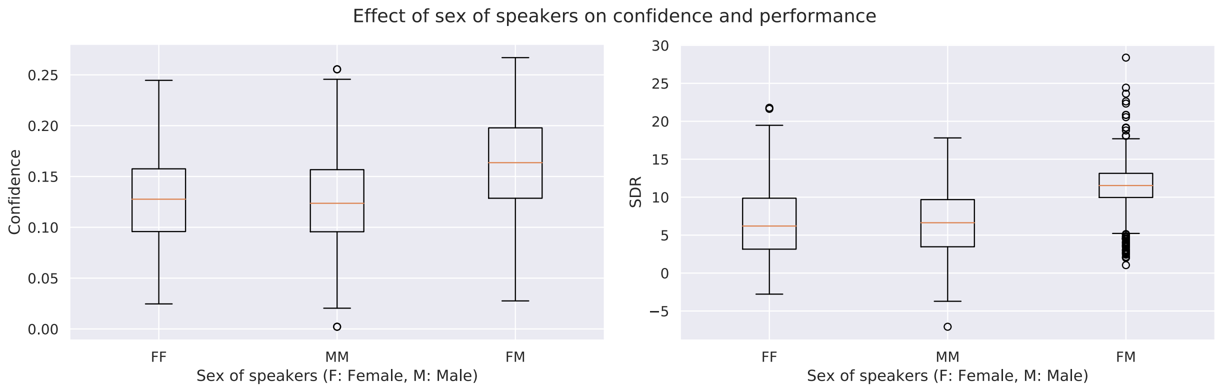

Figure 3.13. Visualizing confidence for different kinds of mixtures. FF indicates the mixture contains two female speakers. MM indicates the mixture contains two male speakers. FM indicates that the mixture contains one male and one female speaker. This last case is the easiest to separate using the deep clustering model. Accordingly, it has both higher confidence than same-sex mixtures as well as higher actual performance.

of a deep network in the wild. In Figure 3.12, I show how the confidence measure can be used to discover high-quality separated sources in an unsupervised way when applying the network to a large set of mixtures. We see that as we increase the limits of the confidence bins, the separation quality of the sources within those limits increases as well, further demonstrating the relationship between the confidence measure and actual performance. We can use the confidence measure to find a subset consisting of 950 sources of all of the 10000 separated sources - 2 for each mixture - that has a mean SDR of 11.3 dB SDR. This is much higher than the overall performance, which is 9.1 dB SDR.

What is the cause of the variation in performance in deep clustering networks? Clearly, there are mixtures it performs well on and mixtures that it performs poorly on, as seen in Figure 3.11. There are types of mixtures that are hard to separate and ones that are easy to separate in the speech dataset. My goal is to show that the confidence measure is sensitive to this, successfully identifying the easy mixtures versus the hard mixtures. One important aspect of a mixture that dictates the performance of deep clustering is the sex of the speakers in the mixture. Separating mixtures of two speakers where the two speakers are of different sex is easier than separating mixtures where the two speakers have the same sex. This is because mixtures of speakers of the same sex tend to have much more time-frequency overlap as the pitches of their voices are closer. Same-sex mixtures have been established as a mixture type that deep speech separation models struggle with [75]. Ideally, the confidence measure would be lower for same-sex mixtures than for different-sex mixtures. Figure 3.13 shows the distribution of the confidence measure as well as SDR for three mixture types: two female speakers, two male speakers, and one female speaker and one male speaker. The confidence measure for this last case is on average higher than the other two harder cases, as is the actual performance of the network.

I have shown that the confidence measure can estimate the performance of deep clustering networks that are trained for very different tasks - separating speech and separating vocals from accompaniment.

3.8. Conclusion

I have presented a method for estimating the performance of source separation methods without ground truth. This method worked by analyzing the embedding space used by several different clustering algorithms. The analysis resulted in a simple confidence measure that can be applied to any clustering-based separation algorithm. I showed that this confidence measure correlates well with actual performance across a variety of audio domains (speech and music), using a variety of different clustering algorithms - ones based on primitives and ones based on deep learning. This performance estimation method is important as it allows one to deploy a separation algorithm in the wild and continuously estimate its performance. If this technology were deployed in a hearing aid, we would be able to mediate between several different separation algorithms (e.g. a direction of arrival method, a deep clustering model trained for speech, a deep clustering model trained for music, and so on), selecting the one that is the best fit for the hearing aid wearer's situation (e.g. are they at a concert trying to hear the music or in a coffee shop or a crowded bar trying to hear their friend speak?). Finally, performance estimation is a crucial part of this dissertation, as it allows me to bootstrap deep learning models for source separation without access to ground truth.

CHAPTER 4

Putting it all together

Source separation systems based on deep learning are currently the most successful methods for separating recordings containing multiple concurrent sounds in underdetermined conditions, that is, where there are fewer channels than sources [173]. Traditionally, deep learning systems are trained on many mixtures (e.g., tens of thousands) for which the ground truth decompositions are already known. Since most real-world recordings have no such decomposition available, developers train systems on artificial mixtures created from isolated individual recordings. Although there are large databases of isolated speech, it is impractical to find or build large databases of isolated recordings for every arbitrary sound. This fundamentally limits the range of sounds that deep models can learn to separate.

The traditional learning procedure for these source separation models is in contrast to how humans learn to segregate audio scenes [8]: sources are rarely presented in isolation and almost never in "mixture/reference" pairs. One can argue that the brain is able to learn to separate sounds without having access to large datasets of isolated sounds. There is experimental evidence that the brain uses primitive cues (e.g., direction of origin of a sound) that are independent of the characteristics of any particular sound source to perform an initial segmentation of the audio scene [114]. The brain could use such cues to separate at least some scenes to some extent, and use that information to train itself to separate more difficult scenes.

In this chapter, I present my work on bootstrapping deep learning models for source separation without ground truth[1]. Bootstrapped deep learning models learn directly from mixtures, rather than via an artificial and unrealistic training process. Bootstrapping allows us to train from the vast quantities of unlabeled audio recordings that exist in the world, rather than from small sets of labeled, carefully curated audio recordings that are not available for every source type one would like to learn how to separate. I use algorithms that I developed that are based on primitives, described in Chapters 2 and 3, to accomplish this. The general outline of my approach is as follows:

(1) **Give each time-frequency points in the spectrogram a source label.** This is done by applying a clustering-based source separation algorithm to a set of mixtures (e.g. music recordings

[1]This chapter has supplementary audio examples: `https://pseeth.github.io/public/thesis/ch4.html`

from YouTube, or mixtures of people talking). The clustering-based algorithms I deal with are direction-of-arrival clustering and primitive clustering, as described in Chapter 2. The selected algorithm produces estimated source labels for every time-frequency point in each mixture.

(2) **Calculate the confidence we have in the labels.** Apply the confidence measure described in Chapter 3 to the embedding space associated with the selected clustering-based source separation algorithm for each mixture.

(3) **Use the estimated labels as ground truth for training a deep clustering network,** weighting the influence of training examples by the confidence we have in the labeling.

This is the gist of how bootstrapping deep computer audition models from unlabeled data (audio mixtures where the source decompositions are not known) works in my framework. In this chapter, I demonstrate three ways to instantiate this approach by leveraging a confidence measure in different ways.

4.1. Bootstrapping from a single cue: direction of arrival

The first scenario I deal with is bootstrapping a separation model from a single cue. In this case, I focus on the task of bootstrapping a deep clustering model for speech separation using a direction of arrival cue. The mechanics of the direction of arrival primitive were previously described in Chapter 2, Section 2.1, but are restated here.

4.1.1. Primitive separation method

To separate via direction of arrival, I use a simple blind source separation method that clusters time-frequency bins based on low-level spatial features present in stereo mixtures. The key idea is to exploit differences between the two channels to decide which time-frequency bins go with which source. First, transform the input stereo audio to a stereo complex spectrogram $X_{t,f}^{(c)}$ where c is the channel, t the time index, and f the frequency index. Then, extract the inter-channel phase difference (IPD) θ and the inter-level difference (ILD) $X^{\log}$:

$$\theta_{t,f} = \angle\Big(X_{t,f}^{(0)}\overline{X_{t,f}^{(1)}}\Big), \tag{4.1}$$

$$\phi_{t,f} = \frac{20\log_{10}\left(|X_{t,f}^{(0)}|\right)}{20\log_{10}\left(|X_{t,f}^{(1)}|\right)}. \tag{4.2}$$

These features are then clustered to form masks. To bias the clustering towards bins with significant energy, weight the computation of the means using the magnitude of each time-frequency bin. The posterior assignments are used as masks on the complex spectrogram, using soft K-Means [77]:

$$M_k(t,f) = \frac{e^{-\beta \mathcal{D}([\theta_{t,f}, \phi_{t,f}], \mu_k)}}{\sum_j e^{-\beta \mathcal{D}([\theta_{t,f}, \phi_{t,f}], \mu_j)}} \tag{4.3}$$

4.1.2. Incorporating confidence

Recall from Chapter 3 that the confidence measure is comprised of three components: the mean silhouette score for all of the time-frequency points, the posterior strength for each time-frequency point, and the size of the smallest cluster in the embedding space. These three measures are multiplied by each other to form an overall confidence measure for each time-frequency point x_i in the time-frequency representation X:

$$\text{CONFIDENCE}(x_i, X) = S(X) * P(x_i) * R(X) \tag{4.4}$$

Next, recall that the original objective function for deep clustering is as follows:

$$\begin{aligned} C(\theta) &= |\hat{A} - A|_F^2 && (4.5) \\ &= |VV^T - YY^T|_F^2 && (4.6) \end{aligned}$$

where θ is the parameters of a neural network, $\hat{A}$ is the affinity matrix constructed using the embeddings V for each time-frequency point as output by the neural network, and A is the target affinity matrix constructed using Y. F indicates the Frobenius norm. This loss function is then adapted [195] to incorporate weights for each time-frequency point as follows:

$$\begin{aligned} \mathcal{L}_{\text{DC},W}(V,Y) &= \|W^{1/2}(VV^T - YY^T)W^{1/2}\|_F^2 && (4.7) \\ &= \sum_{i,j} w_i w_j [\langle v_i, v_j \rangle - \langle y_i, y_j \rangle]^2, && (4.8) \end{aligned}$$

The weight matrix W is a diagonal matrix where each element w_i contains a non-negative value that represents the weight of the corresponding time-frequency point x_i. Wang et al. [195] sets each w_i to be the magnitude (loudness) of the corresponding time-frequency point $|x_i|$. This has the effect of making the training process focus on points that are louder during training, rather than silent points. I will refer to this from now on as *magnitude weighting*. Hershey et al. [65] sets each w_i to be the inverse of the size of the partition that x_i belongs to according to the time-frequency assignments Y. This has the effect of making the network focus equally on both classes in each mixture. If in one mixture, the source classes are unbalanced, as is the case in vocals vs accompaniment mixtures, this weighting scheme weights the points corresponding to the vocals higher than the accompaniment. I will refer to this scheme from now on as *class weighting*.

I propose an additional weighting scheme, called *confidence weighting*. This is done by using the CONFIDENCE equation in Equation 4.4. Each point x_i has corresponding confidence, which is between 0.0 and 1.0. I adjust the confidence measure slightly to incorporate a tunable parameter α as follows:

$$\text{CONFIDENCE}(x_i, X) = (S(X) * P(x_i) * R(X))^{\alpha} \tag{4.9}$$

This tunable parameter α raises the confidence measure to some power. If $alpha = 0$, then the weights are all set to 1, flattening the confidence across all the points. As α gets higher, points that have higher confidence become more and more important relative to low confidence points, causing the network to focus more and more on high confidence points during training. The setting of α is important to the bootstrapping process, as I will show in experiments later.

4.1.3. Experimental design

I demonstrate this approach to bootstrapping on a dataset which consists of speech mixtures. The goal of the following experiments is to answer the following questions:

(1) What is the effect that α has on the separation performance of the bootstrapped model?
(2) How does the performance of the bootstrapped model compare to the performance of the primitive used to train it?
(3) How does the bootstrapped network perform compared to a deep clustering network trained using ground truth labels?

The mixtures are constructed such that there are two speakers in each of them, which are panned to random spatial locations in the scene. This is the same dataset used in Chapter 3, Section 3.5 - the Wall Street Journal dataset, consisting of 20000 spatialized mixtures for training, 5000 mixtures for validation, and 3000 mixtures for testing. I use the anechoic data from that dataset here. I bootstrap a trained model from the direction of arrival primitive separation algorithm applied to these mixtures.

As mentioned in Chapter 3, it is important to recall that there is no guarantee the direction of arrival separation method will work at all. In some mixtures, the speakers are well separated spatially while in others they are either overlapping in spatial location or one speaker is much louder than the other. In these situations, the direction of arrival algorithm will not work at all.

I train a single-channel deep clustering network for speech separation. It is bootstrapped from the output of the direction of arrival algorithm in concert with the confidence measure. I generate estimated labels for each mixture in the training set by applying the direction of arrival algorithm with $K = 2$ for the K-Means clustering algorithm to a stereo mixture consisting of the first channel of each training mixture combined with a random other channel of the same mixture (there are 8 channels in total for each training mixture in the dataset). The parameters of the direction of arrival algorithm are as follows: the window size of the STFT (the time-frequency representation used here) is 512 samples, the hop length is 128 samples. Each mixture has a sampling rate of 8000 Hz.

Applying the direction of arrival algorithm to results estimated time-frequency labels that separate each training mixture into two sources. Only the first channel is used as input to the deep clustering network. It is important to realize that the deep clustering network and the direction of arrival algorithm do not have access to the same information. The deep clustering network only gets a single channel as input, whereas the direction of arrival algorithm gets two channels as input. This means that the deep clustering network does not get access to the same information that the direction of arrival algorithm gets access to. It cannot use spatial information to separate the speakers, so it must learn a different approach to separate the speakers.

For each mixture in the training set, I compute the confidence measure for each time-frequency point in each mixture. This results in a confidence weight that is incorporated into the training process as laid out in Equation 4.7. Finally, the class weighting and magnitude weighting schemes are also incorporated into the loss function by multiplying all three weighting schemes together. This makes the network focus on points that are loud and confident.

The deep clustering network consists of 4 bidirectional long short-term memory layers (BLSTM) with 300 hidden units in each direction (so 600 total), followed by a linear layer that projects each time-frequency point to an embedding of dimensionality $D = 20$. There is a sigmoid activation on the embedding. Each embedding is also constrained to have unit norm, as is standard for deep clustering networks [65]. Before the BLSTM layers, there is an instance normalization layer which standardizes each mixture's time-frequency representation to have zero mean and unit variance [180].

The network is trained for 50 epochs with a batch size of 40, with a total dataset size of 20000 training mixtures. This results in a total of 25000 iterations. I used the ADAM optimizer with a learning rate of 2e-4, and $\beta_1 = .9, \beta_2 = .999$ (the default β values for the PyTorch [132] implementation of the optimizer). Dropout - zeroing out a random fraction of the weights in a layer - with a value of 0.3 is applied to each BLSTM during training. Finally, the value of α in Equation 4.9 is set to be either $0.0, 0.4, 1.0, 2.0, 4.0$ or 8.0. This results in 6 bootstrapped networks, all with the same network architecture and that are all initialized with the same weights.

The upper baseline in terms of performance is a network that is trained using the actual ground truth labels for each training mixture. This network has the same configuration as the bootstrapped networks described above. The training parameters are identical. The only difference between the network trained with ground truth and the bootstrapped networks is the training data. All 7 networks (6 bootstrapped + 1 ground truth) were initialized with the same random seed (0), making the only differences be the value of α during training for the bootstrapped networks, and the training data for the ground truth network.

For evaluation, I use the scale-dependent evaluation metrics from [93]. These metrics are scale-dependent source-to-distortion ratio, source-to-interference ratio, and source-to-artifact ratio. The value and interpretation of these measures was discussed in Chapter 2, Section 2.2.5.2.

4.1.4. Results

There are a total of 7 trained deep clustering networks, 6 of which are bootstrapped from the direction of arrival primitive, and 1 which is trained using ground truth. Additionally, there is the direction of arrival primitive separation algorithm by itself, without training. Each of these networks, as well as the primitive, is evaluated on the testing data from the spatialized anechoic Wall Street Journal dataset.

Figure 4.1 shows the effect of α on the training process for each of the bootstrapped networks. As α increases, mixtures that have higher confidence are stressed in the cost function used to train the network. At

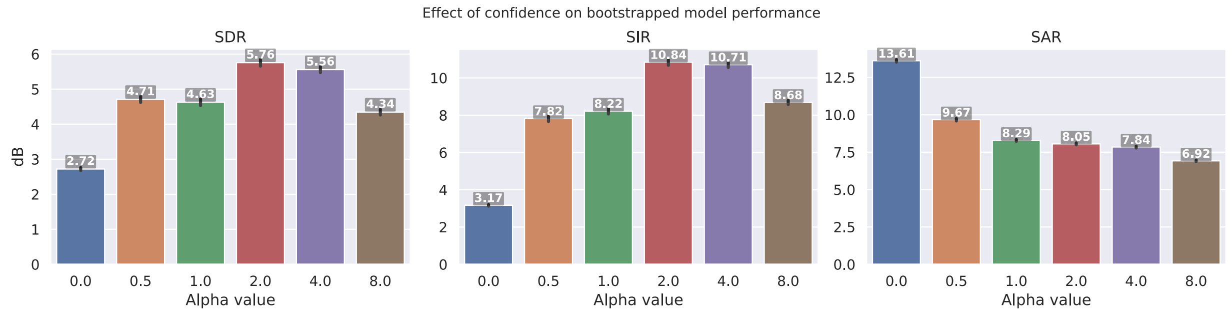

Figure 4.1. Performance of bootstrapped networks that are trained with different values of α, the weight of the confidence measure during training. As α increases, there is a trade-off between the quantity of the data and the quality of the labels used to train the model. Performance peaks at $\alpha = 2$.

$\alpha = 0$, it shows very poor performance, with low SDR, SIR, SAR. As α increases, we observe a peak at 2, and the diminishing performance thereafter. I conclude that α has a significant effect on the performance of the bootstrapped networks. However, settings of α that are too high or too low result in lower performance. To understand why this is, recall that deep learning networks require two things to perform well: high-quality training data *and* a great amount amount of training data [53]. As α increases, the quality of the data increases, but the quantity of the data decreases. This trade-off between quality and quantity leads to the trend in Figure 4.1. From the experiments in Chapter 3, Section 3.5, we know that the confidence measure and actual performance are correlated. Thus, incorporating the estimated performance of the clustering algorithm is important to the bootstrapping process, to ensure that the deep network gets enough data with sufficient quality to train.

Next, I compare the performance of the following approaches:

(1) The direction of arrival primitive by itself.

(2) The best bootstrapped model, which was trained with $\alpha = 2$.

(3) A random ensemble of the bootstrapped model and the primitive that is constructed as follows: for each test mixture, run the primitive as well as the bootstrapped model. Pick randomly between the two.

(4) A confidence-based ensemble of the bootstrapped model and the primitive: for each test mixture, run the primitive and the bootstrapped model. Compute the overall confidence measure on the direction of arrival embedding space, as well as the bootstrapped deep clustering embedding space. Pick whichever approach has higher overall confidence.

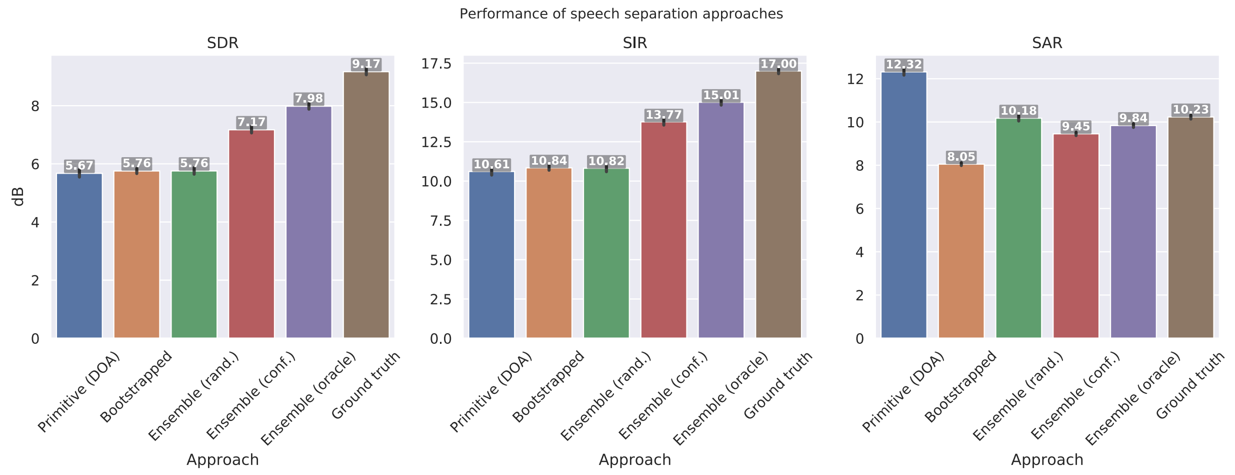

Figure 4.2. Performance of bootstrapping compared to other methods. The bootstrapped model performs on par with the primitive used to train it. However, the bootstrapped model can work in scenarios where the primitive fails, leading to an *ensemble* that outperforms either by itself. Ensemble (rand.) chooses randomly between the primitive and the bootstrapped model. Ensemble (conf.) chooses based on which method has higher confidence. Ensemble (oracle) is the upper bound on an ensemble approach.

(5) An oracle ensemble of the bootstrapped model and the primitive. For each test mixture, run both approaches. Pick whichever approach had higher ground truth performance (this is the upper baseline for an ensemble approach).

(6) The network trained with ground truth labels (upper bound on performance for any bootstrapped network).

The results for each of these approaches is shown in Figure 4.2. First, I found that the bootstrapped model performs about as well as the primitive separation approach used to provide ground truth for training it. However, recall that the primitive algorithm does not work when the speakers in a mixture are not spatially separated from one another. In these cases, the primitive utterly fails to separate the scene. However, since the bootstrapped model was trained on single-channel input it was not able to learn to use stereo cues to determine direction of origin (the cue used by the system that provided training examples. This is remarkable because the bootstrapped model learned something orthogonal to the primitive that trained it. It therefore learned an approach that allows it to separate single-channel mixtures, where it is not possible to determine direction of arrival. This is shown in Figure 4.3, where the performance of both the primitive and the bootstrapped network is shown for every mixture in the test. There are many points where the bootstrapped network significantly out-performs the primitive and vice versa.

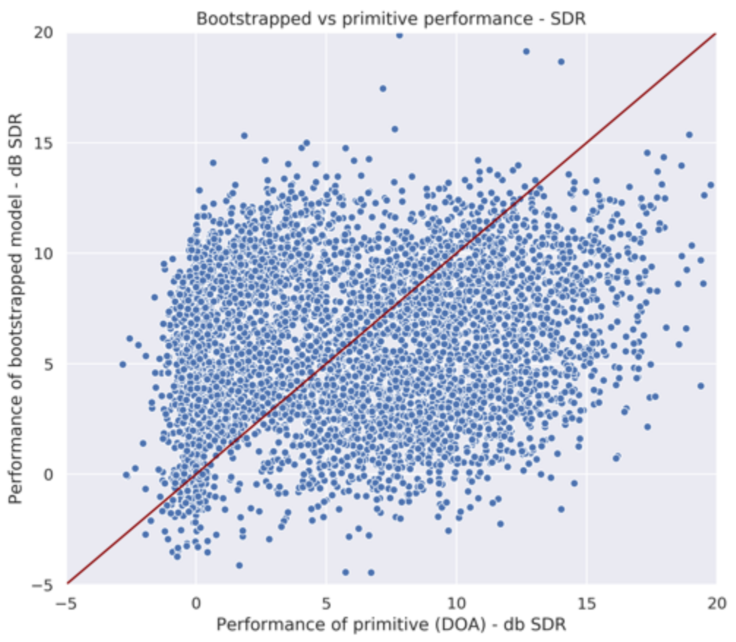

Figure 4.3. Performance of the best bootstrapped network ($\alpha = 2$) vs performance of direction of arrival primitive used to train that network on the test set. Every dot represents a mixture in the test set. Points on the red line have equal performance by both approaches. Points to the right of the line mean the primitive out-performed the bootstrapped network. Points to the left of the line mean the bootstrapped network out-performed the primitive.

Therefore, an ensemble approach is fruitful and this is shown in the data. The random ensemble does not out-perform either primitive by itself, but the confidence-based ensemble does, reaching an SDR of 7.1 dB, nearly 1.5 dB higher than either the primitive or the bootstrapped model by itself! This indicates that the confidence measure is useful for picking between approaches, even if those approaches have very different mechanics (direction of arrival clustering vs deep clustering). The confidence-based ensemble does fall short of the oracle ensemble, indicating that there are further possible improvements to be made in the confidence measure.

The network bootstrapped from direction-of-arrival information still falls short of the performance of the network trained on ground truth. However, a bootstrapped network is trained *without* access to ground truth data and can be trained using any mixture where the direction of arrival primitive will work (e.g. music mixtures). The confidence measure allows us to deploy this training process in the wild, collecting high-quality data in an unsupervised way. The networks can be trained to separate any arbitrary audio sources (not just speech), so long as the primitive has a chance of working on the in-the-wild mixtures. Finally, the training process outlined here makes for deep clustering networks that can adapt to new data, constantly learning from auditory scenes that are encountered in the wild. Given any stereo mixture, the direction of arrival algorithm could be run on it. The confidence measure can predict if the direction of arrival algorithm works. Then the network can be trained on the resulting labels, mediated by the confidence measure.

4.2. Bootstrapping via remixing

The next bootstrapping technique I will present is that of bootstrapping via *remixing*. In the previous technique, I learned a model directly from the same set of mixtures that the primitives were applied to. In this alternative approach, the idea is instead to create *new* mixtures from the sources separated by the primitive and train a network using the new mixtures, using the separated sources as "ground truth" during training. I present this bootstrapping approach on a different task from the previous one: separating vocals from accompaniment in music.

4.2.1. Primitive separation method

The primitive separation used here is *primitive clustering*. The details of primitive clustering are in Chapter 2, Section 2.3.1 but are also briefly restated here. Primitive clustering works by running multiple primitive-based algorithms on a single mixture and then combining their output. Each algorithm produces a soft time-frequency mask that separates a mixture in some way. Each primitive maps each time-frequency point is mapped to a number between 0 and 1. All of the primitives map each time-frequency point to a primitive embedding space, as shown in Figure 4.4.

To turn the embedding space into a soft mask that takes into account all of the decisions made by each algorithm, I use soft K-Means clustering [77]. Instead of running K-Means to discover the means, I fix them to be the points $\mu_0 = [0]^N$ and $\mu_1 = [1]^N$. I then use the distance of the primitive embedding of every time-frequency point to these means to calculate the soft mask for each source $M_k(t, f)$ at each point as follows:

$$M_k(t,f) = \frac{e^{-\beta \mathcal{D}(\mathcal{F}(X(t,f)),\mu_k)}}{\sum_j e^{-\beta \mathcal{D}(\mathcal{F}(X(t,f)),\mu_j)}} \tag{4.10}$$

where $\mathcal{D}(x, y)$ is the Euclidean distance between points x and y. This is familiarly the *softmax* function. It maps distances in the embedding space to values between 0 and 1. β, which I found to have an effect on the performance of primitive clustering, is set to 5.0 here.

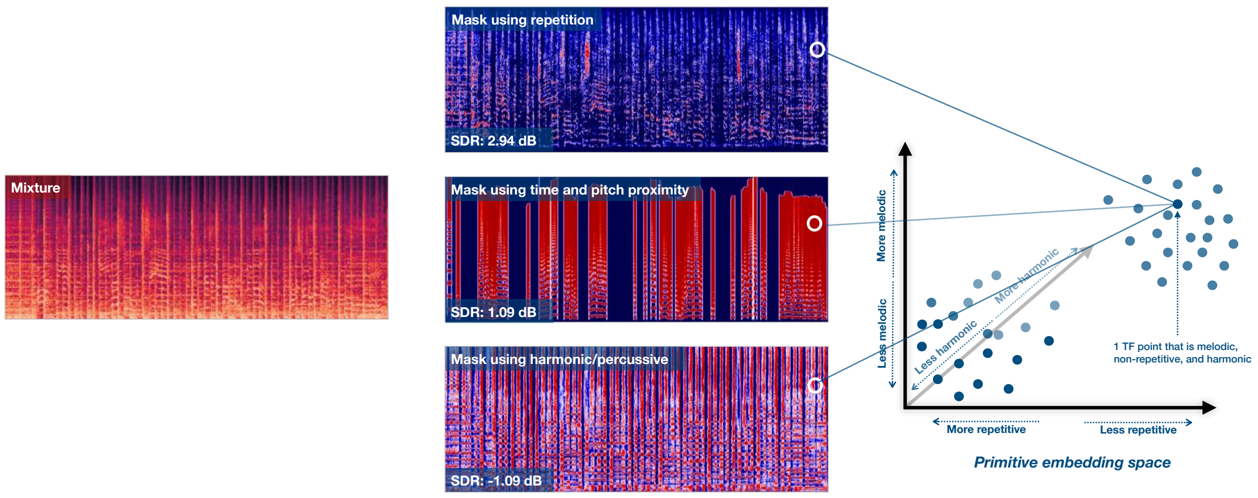
Mixture
Mask using repetition
SDR: 2.94 dB
Mask using time and pitch proximity
SDR: 1.09 dB
Mask using harmonic/percussive
SDR: -1.09 dB
More melodic
Less melodic
Less harmonic
More harmonic
1 TF point that is melodic, non-repetitive, and harmonic
More repetitive
Less repetitive
Primitive embedding space

Figure 4.4. The process for making a primitive embedding space. A set of primitive algorithms is run on the mixture. Each algorithm produces a mask with values between 0 (blue) and 1 (red) that indicates how it is segmenting the auditory scene. The performance (using source-to-distortion ratio - SDR) for each algorithm is shown. Together, the three masks map each time-frequency point to a 3D embedding space, shown on the right. The marked point was classified by the three primitives as melodic, not repetitive, and harmonic.

4.2.2. Incorporating confidence

In Chapter 3, I showed that the confidence measure is predictive of separation performance for primitive clustering. I will now show how to incorporate the confidence measure into a training process. First, a set of mixtures is separated using primitive clustering. Then for each mixture, the *overall* confidence is computed, rather than the confidence for every time-frequency point in every mixture. Finally, of the three components of the confidence measure, only two are used: the sampled silhouette score and the posterior strength. For the same reasons as in Chapter 3, the size of the smallest cluster measure is not used, as accompaniment and vocals are naturally unbalanced classes. The silhouette score is computed by sampling the silhouette score (with $N = 1000$) from the top percentile of time-frequency points by loudness for each mixture. The posterior strength score is computed by taking the mean of this same top percentile of time-frequency points. This produces a single overall confidence number for every mixture.

Next, we use the computed overall confidence measures to sift through the separated sources to find high-quality separated sources. Since we do not have access to the ground truth quality of the separated sources during the bootstrapping process (otherwise, we would be able to just use ground truth sources), we instead use the confidence measure, which I showed to be predictive of ground truth quality in Chapter 3. Figure 4.5 shows the result of sifting through the separated sources on the MUSDB train set ($N = 100$).

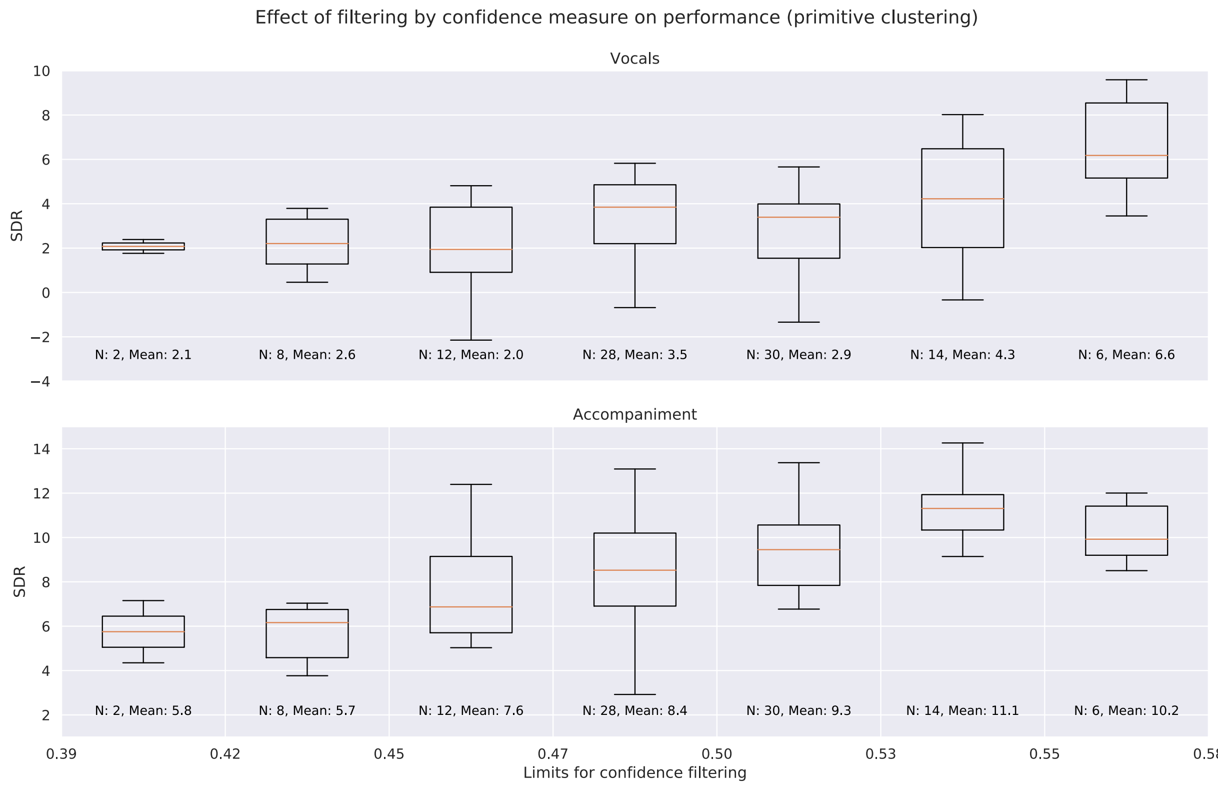

Figure 4.5. Filtering the separated vocals and accompaniment sources produced by primitive clustering using confidence on the MUSDB train set. Each box-plot shows the distribution of the separated sources that had an associated confidence between the bounds on the x axis. For example, the first box plot on the left shows the distribution of SDR for sources that had confidence between 0.39 and 0.42. As the limits increase, the distribution moves upwards, obtaining sources with higher average SDR.

As I increase the bounds on overall confidence for each box-plot, the distribution of actual performance (measured via source-to-distortion ratio) goes up.

The overall approach, then, is to first use the confidence measure to select separated sources on some large set of mixtures. Separated sources that have confidence below some tunable threshold τ can be discarded. The remaining separated sources are then remixed to create new mixtures which are used to train a deep clustering network. These remixes serve a few purposes. First, they augment the amount of data that is used to train the deep clustering model, as the remixes are diverse. For instance, given 1000 separated vocals and accompaniment sources, there are one million possible remixes. Without remixing, there are only 1000 possible mixtures that can be used. Second, the separated sources are separated via primitives, which are functions of characteristics of the original mixture (e.g. how repetitive it is, if there is percussion, and so on). The remixes are not likely to share those characteristics, as the vocals and accompaniment are unrelated sources. This forces the deep clustering network to learn a different way to separate the vocals from the accompaniment. Similarly, in the bootstrapping method considered in Section 4.1.3, the primitive separation method required two channels to compute. Therefore, to have the deep clustering network learn a different way to separate the sources, I made the input to the deep network single-channel. The remixing procedure is similar to this.

Finally, unlike the bootstrapping method in Section 4.1.3, where the confidence measure was incorporated during training, here the separated sources are used as if they are actually ground truth. The deep clustering objective function is applied normally, with magnitude and class weighting, and without confidence weighting. This puts the focus of method on simply using the confidence measure to obtain high quality sources that can be remixed. The two methods could be combined but this is saved for future work.

4.2.3. Experimental design

I demonstrate the bootstrapping via remixing approach on a dataset which consists of music mixtures. The goal of the following experiments is to answer the following questions:

(1) Does the bootstrapped deep clustering network out-perform the primitive source separation method used to train the bootstrapped network?
(2) What is the effect of selecting separated sources for remixing using the confidence measure?
(3) What is the effect of increasing the amount of available data that is used to create the mixtures?
(4) How does the bootstrapped network perform compared to a a deep clustering network trained from ground truth?

I use the MUSDB dataset, which consists of 150 tracks, 100 of which are used for training and 50 of which are used for testing. This dataset was previously used in Chapters 2 and 3. The task for the experiments in this section is, given a musical mixture, separate out the vocals from the accompaniment in the mixture. As the lower baseline, the primitive separation method is evaluated on the test set of MUSDB. The goal is for the bootstrapped deep clustering to out-perform this lower baseline.

Each deep clustering network has a similar architecture to the networks used in Section 4.1.3, using hyperparameter settings and an architecture that is commonly used in deep clustering literature [75, 105, 195]. The networks consist of a stack 4 BLSTM layers, followed by a linear layer that projects every time-frequency point to an embedding of size $D = 20$. The output of the linear layer is put through a sigmoid activation and is unit normalized. An instance normalization layer is used prior to the stack of BLSTM layers.

After the instance norm layer, there is a mel projection, which projects the input frequencies to a 300 mel frequency basis. This has the effect of transforming the linear frequency spacing of the input representation to log frequency spacing. This is commonly done for music separation networks [104], because mel-frequencies more closely reflect the important information in musical mixtures.

To create training mixtures for the *ground truth* network, I took the 100 training tracks from the MUSDB dataset and remixed them to create $20,000$ new mixtures. This was done by taking random 15 second snippets from some accompaniment source (the sum of the bass, drums, and other stems) and some vocals source in the dataset and remixing them at some random signal-to-noise ratio between -2.5 and 2.5 dB. I created 20000 15 second mixtures using this process. Each mixture had a sample rate of 44100 Hz, and an STFT was taken of each with a window length of 2048 and a hop length of 512.

The network was trained using sequences with a length of 400 frames (about 4.6 seconds) taken from each mixture in the training set. There are $20,000$ total mixtures, each 15 seconds long. A 400 frame sequence was taken at a random offset every time a particular mixture was used during training. One epoch was a single training pass over all $20,000$ mixtures. The network was trained for 50 epochs. I used the ADAM [87] optimizer with a learning rate of 2e-4, and $\beta_1 = .9, \beta_2 = .999$. Dropout with a value of 0.3 is applied to each BLSTM during training. The resultant network, trained with ground truth (i.e. perfectly isolated recordings of sound sources, prior their being mixed together), is the upper baseline for all the bootstrapped networks.

The *bootstrapped* networks had exactly the same network architecture as the ground truth network. The only difference between the training procedure for a bootstrapped network and training procedure for the ground truth network was the data from which it was trained. The training data for bootstrapping is generated by applying the primitive clustering method (described in Section 4.2.1) to a set of mixtures and remixing the resulting separated sources. The procedure for generating the separated sources for training a bootstrapped network is the same for all sets of mixtures - the only change is the underlying mixtures contained in each set. I will first describe the procedure, and then the various sets of mixtures I used.

For each mixture in the set, I first split it up into segments of length 30 seconds with 15 second overlap between segments. The loudness of each segment was computed via root mean square (RMS) of the amplitude of the time-series representation of the segment. If the RMS was below the 25th percentile of RMS across all the segments being processed, it was discarded. Otherwise, it was included. I applied the primitive clustering algorithm to each 30 second segment, resulting in a separated accompaniment and vocals source for each segment. Finally, I created a new mixture from the separated sources from different songs. This is done by choosing a 15 second excerpt from some separated accompaniment source and a 15 second excerpt from some separated vocals source and mixing them with a random signal-to-noise ratio chosen between -2.5 dB and 2.5 dB. The vocals and accompaniment sources can be from different songs. I repeated this procedure 20000 times to create 20000 mixtures. The separated sources that went into each mixture were used for training a deep clustering model via bootstrapping.

The separated sources used to create each new mixture can be selected via the confidence measure. Each separated source was extracted from some mixture using primitive clustering. There is an overall confidence measure with each separated source based on the mixture that the separated source came from. The confidence measure is predictive of the separation quality. Each separated source can then be included or excluded from the remixing process by using the value of the confidence measure on the mixture the separated source came from. Figure 4.5 shows how the ground truth quality (measured via SDR) of the separated sources generally improves as the value of the confidence measure increases. By computing the confidence measure across the entire set of mixtures, one can filter the separated sources used, excluding low-quality sources from the remixing process. I set fixed thresholds that are based on the distribution of the confidence measure across the entire set. Everything below a percentile τ is excluded from the remixing process. In my experiments, I show the relationship between selecting separated sources based on confidence and the performance of the bootstrapped network.

I experimented with three sets of mixtures. The first set only used the mixtures included in the MUSDB train set. There are 100 songs in the MUSDB train set. The procedure above is applied, resulting in 1008 30 second segments separated into vocals and accompaniment sources. The next set augments the MUSDB train set with additional music recordings that are downloaded from YouTube. The music recordings were gathered by downloading playlists from YouTube, resulting in an additional 831 songs across 4 genres - oldies, opera, pop, and rock. This resulted in $1008 + 8655 = 9663$ 30 second segments (1008 being from the MUSDB train set, and 8655 from YouTube) that are separated into vocals and accompaniment sources. The final set adds in the MUSDB test set, which consisted of 50 songs. The test set adds in an additional 560 30 second segments to the 9663 30 second segments, resulting in a total of 10223 segments. The performance of the bootstrapped network when using these three sets of mixtures shows the effect of increasing the amount

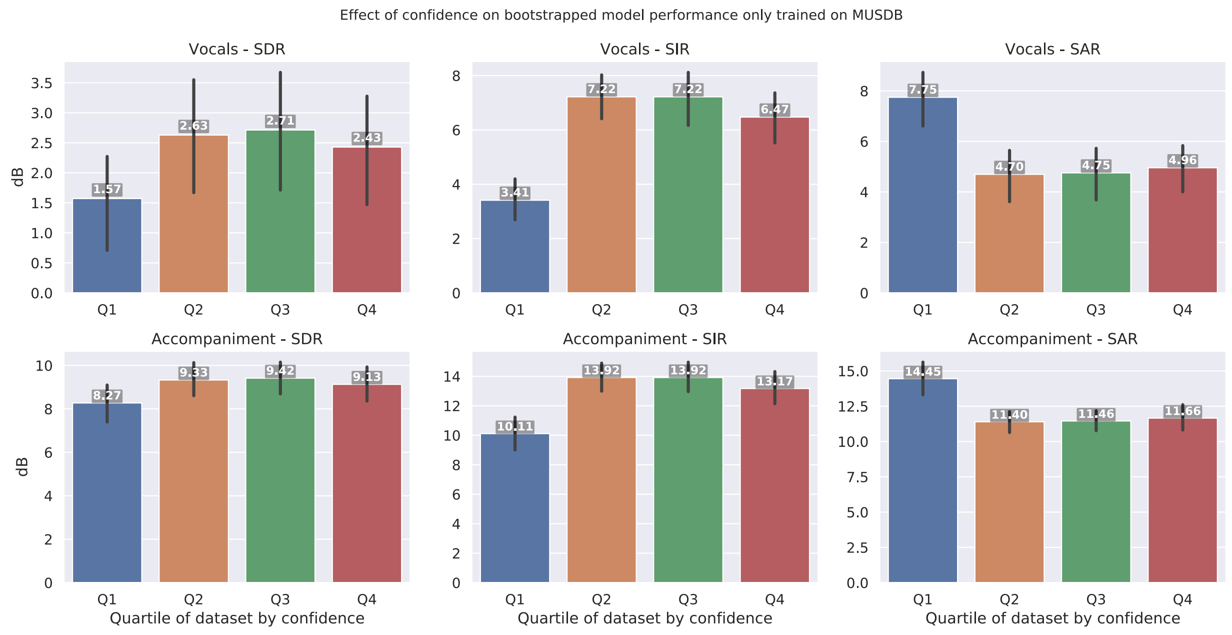

Figure 4.6. Performance of bootstrapping when the training mixtures are constructed with different quartiles of the training set, organized by the confidence measure. Q1 indicates that the network was bootstrapped from separated sources in the lowest quartile: sources that had a confidence below the 25th percentile of all confidence measures in the training set. Q2 indicates the next quartile, between 25th and 50th. Q3 (50th to 75th percentile), and Q4 (above 75th percentile) are defined similarly. We see that the network bootstrapped from the lowest quartile has very poor performance, indicating that the confidence measure is good at ruling out low-quality sources. Q2 and Q3 have similar performance, while Q4 has a slight degradation in separation quality.

of available data used to create the training data. Finally, separated performance is evaluated using scale-dependent source-to-distortion ratio, source-to-interference ratio, and source-to-artifacts ratio, as described in Section 2.2.5.2.

4.2.4. Results

I first investigated the effect of the selection procedure using the confidence measure. This is shown on the first set of mixtures, which only included the mixtures from the MUSDB train set. The MUSDB train mixtures are split into 1008 30 second segments, resulting in 1008 separated vocals sources and 1008 separated accompaniment sources. Each separated source has an associated confidence measure with it. We can sort the separated sources by the confidence measure and split them into quartiles. The first quartile contains all the sources that had a confidence below the 25th percentile. The second includes all the sources between the 25th and 50th quartile, and so on. Each quartile contains 252 30 second separated vocals and accompaniment

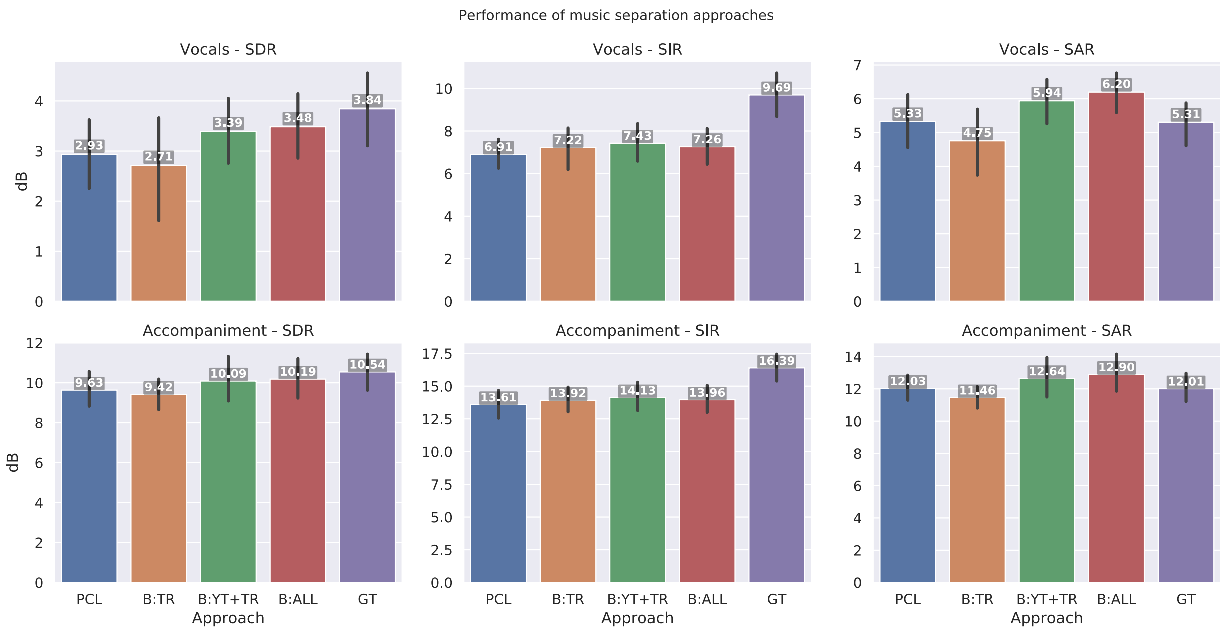

Figure 4.7. Performance of bootstrapping compared to other methods. The bootstrapped model significantly out-performs primitive clustering, which was used to train it, but falls short of the performance of the ground truth model in terms of SDR and SIR.

that are remixed into 20000 mixtures. These mixtures are then used to bootstrap 4 separate deep clustering networks, one from each quartile.

The results are shown in Figure 4.6. The performance of each quartile is marked Q1 (up to 25th percentile), Q2 (25th to 50th percentile), and so on. The performance of the bootstrapped network bootstrapped from the separated sources in the first quartile is far lower than the three other quartiles. This shows that bootstrapping a network from low-confidence sources will have a detrimental effect on separation performance. This network achieves a very low SDR of 1.57 dB, which is far below primitive clustering on its own, which achieved 2.93 dB SDR. Q3 was the best performing quartile, with 2.71 dB SDR. However, it achieved very similar performance to Q2, with the two having identical SIR. Finally Q4 achieves slightly lower performance at 2.43 dB SDR. These results indicate that the confidence measure is very good at ruling out low-quality sources for training a network but is less predictive for higher-quality sources.

I then investigated using larger datasets to bootstrap a deep clustering network. The other two sets of mixtures, which include the YouTube data as well as the MUSDB test set, are much larger than the MUSDB train set by itself, adding an additional 881 unique songs, resulting in an additional 9215 30 second segments that can be included in the remixing process. The results from the experiment detailing the effect of the different quartiles suggest that discarding the lowest quartile and using the upper three quartiles, which had similar performance, is a reasonable setting. Thus, everything below the 25th percentile is excluded from the remixing process in these networks ($\tau = 25$). Figure 4.2 shows the performance of the following approaches:

(1) PCL - Primitive clustering by itself. This is the performance of the unsupervised primitive clustering algorithm and is the lower baseline. The primitive clustering algorithm included micromodulation, repetition, time and pitch proximity, and harmonic/percussive timbre.

(2) B:TR - The bootstrapped network that is trained off remixes created using only the MUSDB train set using the third quartile, which was the best performing quartile as shown in Figure 4.6.

(3) B:TR+YT - The bootstrapped network that is trained off of remixes created using the second set of mixtures which included the MUSDB train mixtures as well as the additional 831 songs from YouTube, with $\tau = 25$.

(4) B:ALL - The bootstrapped network that is trained off of remixes created using the MUSDB train set, the YouTube data, and the 50 songs from the MUSDB test set, again with $\tau = 25$.

(5) GT: The deep clustering network that is trained using ground truth, generated using the MUSDB train set.

First, note that the bootstrapped models that are trained with the YouTube data (B:YT+TR and B:ALL) out-perform the primitive clustering method considerably, achieving considerably higher SDR, SIR, and SAR. Second, note that the amount of data that is used to create the mixtures has a considerable effect on the performance of the bootstrapped networks. The model that is trained with only mixtures created from the MUSDB train set has SDR lower than the primitive method. Increasing the amount of data using the YouTube mixtures increases the performance by .68 dB SDR. Using the test set in the remixing process further increases the SDR by .1 dB. The performance of the bootstrapped networks still falls short of the ground truth network. This gap in performance could perhaps be addressed by adding more data. Note that it is much easier to add training data for a bootstrapped network than for one trained on traditional ground truth separated sources, since any audio mixture can be used as training data.

4.3. Conclusion

In this chapter, I have presented a method for training deep clustering networks directly from mixtures without ground truth. The general procedure is to separate the mixtures using primitive separation methods. Then, the performance of the primitive separation method is estimated using an unsupervised confidence measure, developed in Chapter 3. Finally, the confidence measure is incorporated into the training process for a deep clustering network. I call this entire process *bootstrapping.* One way to instantiate this procedure was demonstrated using a single primitive source separation method - direction of arrival - applied to a set of speech mixtures. A model was then trained off the output of the primitive on each mixture, with the loss function mediated by the confidence measure. The training process thus focused on learning from separations that I was confident were correct. I show that the bootstrapping method can result in a network that performs on par with the primitive used to train it. However, the ensemble of the bootstrapped method and the primitive used to train it out-performed either approach by itself.

Another way to instantiate the bootstrapping procedure was demonstrated using a multiple primitive source separation method (primitive clustering) applied to a set of music mixtures. The training procedure for this method is different, as it instead uses the primitive separation method to create a large quantity of separated sources that can then be remixed into new sources. The confidence measure is used to sift through the separated sources so that the remixing procedure excludes sources that are likely to be low-quality. I was able to train a bootstrapped music separation model that out-performs the primitive source separation method directly from music mixtures drawn from MUSDB as well as YouTube.

The endless amount of audio mixtures that are available for download makes these bootstrapping techniques an exciting approach that can be used to learn how to separate any source (e.g. learning to separate trumpet by training from Miles Davis albums), given that the primitive separation algorithm works on at least some of the mixtures. Which mixtures the algorithm works on does not have to be decided by hand but can rather be predicted in an unsupervised manner by using the confidence measure.

CHAPTER 5

Conclusion

In this dissertation, I have proposed a method for training a deep learning model for audition without access to ground truth labels. My approach for training these models is follows. The system first applies primitive separation algorithms to an auditory scene. These primitives are inspired by human audition, such as our tendencies to group sources by spatial location, repeating vs not repeating, common fate, pitch and time proximity, and so on [8, 114, 116, 156, 160, 161]. It then estimates the confidence it has in the labels produced by the primitives. The goal is to focus the learning process for the deep network on learning the labels that are more likely to be correct. Finally, I train a network using the labels produced by the primitive audio source separation algorithms. The labels are mediated by the confidence measure so that the network avoids learning from noisy or incorrect labels. The result is a source separation model that can learn to segment the auditory scene without ground truth by bootstrapping its understanding of the auditory world using primitive unsupervised audio source separation algorithms.

To accomplish this, I created novel methods that advanced the state-of-the-art in primitive source separation, unsupervised estimation of performance of source separation algorithms, and unsupervised learning of separation models. In Chapter 2, I presented two major advances to primitive source separation. The first was the 2DFT algorithm, which can be effectively used to separate auditory scenes based on three primitives: frequency micro-modulation, repetition, and common fate. I showed that the 2DFT algorithm has similar behavior to humans when presented with the same auditory stimuli. It can separate repeating sounds from non-repeating sounds, recognize when a tone is being micromodulated in frequency, and groups moving tones from non-moving tones in auditory scenes. I showed that 2DFT out-performs all known competing primitive separation algorithms for the task of vocals separation from musical mixtures.

I then showed how different primitive algorithms can be used together such that the combination of them out-performs any single algorithm in isolation. My framework for doing this was called primitive clustering. In experiments, I showed it to out-perform all of its input methods on the task of separating vocals from accompaniment.

In Chapter 3, I presented a method that can estimate the quality of an audio separation produced by clustering-based separation algorithms without the need to compare to ground-truth isolated sound sources. The method works by analyzing the embedding space that is used in a clustering-based separation algorithm. The confidence measure is entirely unsupervised. In experiments, I showed that the confidence measure correlated well with actual ground truth performance across a variety of audio domains (speech and music) and with many different clustering based source separation algorithms, including ones based on primitives as well as ones based on deep learning. The performance estimation method is important as it allows one to deploy a separation algorithm in the wild and continuously estimate its performance. If this technology were deployed in a hearing aid, we would be able to mediate between several different separation algorithms (e.g. a direction of arrival method, a deep clustering model trained for speech, a deep clustering model trained for music, and so on), selecting the one that is the best fit for the hearing aid wearer's situation.

Finally, the methods presented in Chapters 2 and 3 were brought together with deep learning in the Chapter 4 to realize a computer audition model that embodies the opening quote of this dissertation:

> “ *The innate influences on [audition] should not be seen as being in opposition to principles of learning. The two must collaborate, the innate influences acting to 'bootstrap' the learning process.* ”
>
> Albert Bregman, *Auditory Scene Analysis*, p. 40, 1994

Finding ways to bootstrap the learning process was the central goal of this dissertation. The method I have proposed works by using a confidence measure in concert with a training procedure to learn a source separation model without access to any ground truth sources. In Chapter 4, I proposed and tested bootstrapping procedure for two tasks: learning to separate speakers that are talking simultaneously, and learning to separate vocals from accompaniment in music mixtures. For speech, I bootstrapped a model based on the direction of arrival primitive. The direction of arrival primitive separates by identifying what location sounds are coming from. Sounds coming from the left are separated from sounds coming from the right. For music, I bootstrapped a model from primitive clustering, an ensemble approach that combines multiple primitives in an effective way that I proposed in Chapter 2.

In both cases, I found that the bootstrapping process can lead to a significant boost in separation quality compared to the primitives by themselves, despite the fact that the networks are learned directly

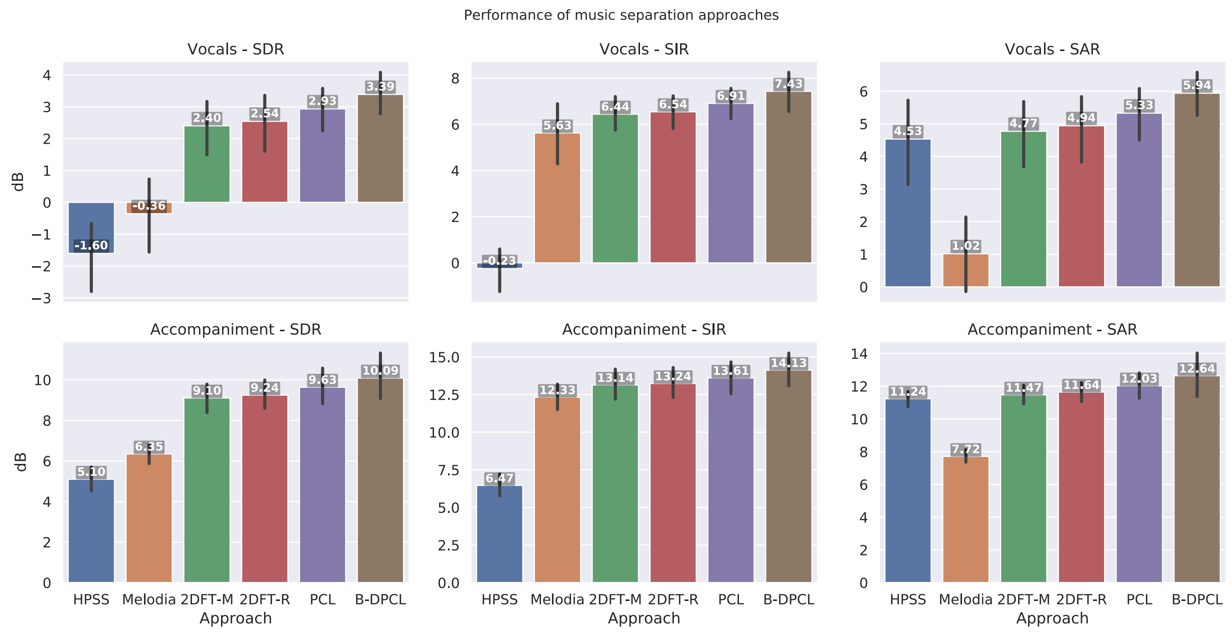

Figure 5.1. Comparison between music separation approaches. I started with a set of primitive algorithms: harmonic/percussive timbre (HPSS), time and pitch proximity (Melodia), micromodulation (2DFT-M), and repetition (2DFT-R). I presented a way to combine them, called primitive clustering (PCL). Primitive clustering out-performs all of the primitives that are in it. Finally, I bootstrapped a deep clustering network from primitive clustering, further improving the separation performance (B-DPCL).

from the primitives. In the case of bootstrapping from direction of arrival, I found that the ensemble of the bootstrapped network and the primitive out-performed either in isolation. The ensemble was done by applying the confidence measure to the embedding spaces of the two clustering algorithms - direction of arrival clustering and deep clustering. Whichever approach had higher confidence was chosen in the ensemble. In the case of bootstrapping from primitive clustering, I found that the bootstrapped networks significantly out-performed the primitive clustering by itself, resulting in separated sources with much higher quality.

Figure 5.1 shows the progression of vocals separation performance throughout this thesis. I started with a set of primitive-based separation algorithms, one of which I created - the 2DFT algorithm that can be used for repetition-based or micromodulation-based separation. The other primitives were harmonic/percussive timbre and time and pitch proximity. The best performing primitive was my proposed 2DFT algorithm, which used repetition for separation. It achieved 2.54 db SDR on a test set consisting of musical mixtures. Primitive clustering further improved the performance, achieving 2.93 dB SDR on the same test set. This is considerably higher than any primitive in isolation. Finally, a deep clustering network that was bootstrapped from primitive clustering out-performed primitive clustering, achieving 3.39 dB SDR.

5.1. Limitations

The greatest limitation the work presented in this thesis has is that the resultant networks are still not at the quality of a network trained from ground truth labels. There are ways forward that could improve the separation quality of the bootstrapped network that will be discussed in Section 5.2. This limitation is most likely due to the quality of the primitive separation algorithms that are used to bootstrap the network and the reliability of the confidence measure. To the first issue, perhaps better primitive separation algorithms could be developed to obtain better training data for the bootstrapped networks. To the second issue, the confidence measure is not perfectly correlated with ground truth performance. As seen in Chapter 3, the confidence measure has much better correlation for speech mixtures than it does for music mixtures. A more robust confidence measure could be easily swapped into the training procedure presented in Chapter 4. This would likely improve the performance of the bootstrapped networks greatly.

The direction of arrival algorithm that I bootstrap from in Chapter 4 is not at all resilient to reverberation. The performance of the algorithm is much worse on reverberant mixtures. I attempted to bootstrap off of reverberant mixtures, but it unfortunately does not work. The quality of the direction of arrival algorithm is just too low. Other, more sophisticated direction of arrival based algorithms [28, 110] could be used in place of this. These algorithms are still based on clustering, but with features that are more resistant to reverberation, so the same bootstrapping method could be applied.

5.2. Future work

There is quite a bit of room for future work, developing and improving the ideas presented throughout this thesis. Some of these I have already mentioned, such as improving the confidence measure or the quality of primitive separation algorithms. For example, the primitive clustering method presented in this work only incorporated monaural cues such as frequency micromodulation, repetition, and so on. It did not incorporate stereo cues. Integrating stereo cues into primitive clustering may improve separation quality and improve the performance of networks bootstrapped from primitive clustering.

A different way to improve separation quality is to use the bootstrapped networks as the separation algorithm to bootstrap a new network off of. By this I mean a collection of mixtures would separated using the bootstrapped network that was obtained via primitives. Because the bootstrapped network is based on deep clustering, the confidence measure can be applied again, as I showed in Chapter 3. Then, a new network can be bootstrapped from the bootstrapped network. I plan to explore this in future work.

Another possible direction for future work is to incorporate more primitives. Many primitives, such as "old plus new", were left out of this work. They could be incorporated into future versions of my approach. It is also possible to include primitives from other modalities, such as vision. It is well known that the visual and auditory senses are linked in the human brain [149]. A natural extension would be to incorporate separation methods that use audio-visual correspondence [207].

In prior work, that was not covered in this thesis, I developed an approach that estimated the performance of separation algorithms via a deep neural network [112]. The result was a deep network that could estimate the separation quality of a separated source very effectively. Such methods could be incorporated into future iterations of the bootstrapping method I have proposed in this thesis.

More broadly, there is a recent trend in the source separation literature to continuously obtain higher and higher evaluation numbers according to source-to-distortion ratio, source-to-interference ratio, and source-to-artifact ratio. This race to the top has pushed the state-of-the-art very far very quickly. However, performance has recently begun to saturate, as new deep separation methods beat prior separation methods by less than tenths of a decibel, and the output of these separation methods is almost indistinguishable from an oracle separator [173]. I believe there are two reasons for this. First, the evaluation measures we use are limited. Better evaluation measures should be used in the future, especially those that include a human in the loop [14, 15]. Second, the test sets we are using are being completely solved by deep learning methods. We need harder test sets that more accurately reflect the real world in order to keep making progress on separation problems.

The methods proposed here could be used to learn to separate sounds that we were not previously able to learn how to separate. Namely, sounds that we do not have large datasets consisting of recordings of that sound in isolation. This could be done by simply finding the appropriate mixtures that contain that sound and learning directly from them via the methods presented in this thesis. These methods could also be used to bootstrap a network from vast quantities of unlabeled mixtures which can then be fine-tuned for specific tasks like music separation, or even non-separation tasks like sound classification.

The bootstrapping techniques I have proposed could be extended to other domains, such as vision. An analogous problem to source separation in the visual field is image segmentation. In image segmentation, the goal is to assign every pixel in the image as belonging to or not belonging to specific objects. Clustering of pixels using various unsupervised methods has been explored for image segmentation, using features like color or edge detection. One could apply also deep clustering to this problem, as the difference between image pixels and time-frequency points does not matter to deep clustering. Only the features learned by deep clustering would matter. Once a clustering-based segmentation method (either deep or unlearned) is established, all of the same bootstrapping techniques proposed in this work could be applied.

I have shown that my confidence measure can effectively estimate the performance of networks that are designed specifically for audio source separation. However, the confidence measure is not specifically tied to anything about audio, but rather to facts about embedding spaces. This opens a door to estimating performance for many other tasks where a clustering can be applied. Clustering is a ubiquitous technique in machine learning and computer science and being able to estimate the performance of it would be very useful. Future work could explore how the confidence measure could be used for unsupervised performance estimation in other domains, like language, vision, and so on.

The large amount of audio mixtures that are available for download makes these bootstrapping techniques an exciting approach that can be used to learn how to separate any source (e.g. separating saxophone by using a set of music recordings from John Coltrane albums). I look forward to exploring the possibilities that are unlocked by the methods I have proposed here. A source separation system that can continuously improve its own performance in an unsupervised fashion, learning directly from its experience with the auditory world without any human intervention at all is now possible.

References

[1] Fausto Acernese, Angelo Ciaramella, Salvatore De Martino, Rosario De Rosa, Mariarosaria Falanga, and Roberto Tagliaferri. “Neural networks for blind-source separation of Stromboli explosion quakes”. In: *IEEE Transactions on Neural Networks* 14.1 (2003), pp. 167–175.

[2] Jonathan Allen. “Short term spectral analysis, synthesis, and modification by discrete Fourier transform”. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 25.3 (1977), pp. 235–238.

[3] Radu Balan and Justinian Rosca. “Source separation using sparse discrete prior models”. In: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on.* Vol. 4. IEEE. 2006, pp. IV–IV.

[4] Jon Barker, Emmanuel Vincent, Ning Ma, Heidi Christensen, and Phil Green. “The PASCAL CHiME speech separation and recognition challenge”. In: *Computer Speech & Language* 27.3 (2013), pp. 621–633.

[5] Juan Pablo Bello, Claudio Silva, Oded Nov, R Luke DuBois, Anish Arora, Justin Salamon, Charles Mydlarz, and Harish Doraiswamy. “SONYC: A system for the monitoring, analysis and mitigation of urban noise pollution”. In: *arXiv preprint arXiv:1805.00889* (2018).

[6] Emmanouil Benetos and Simon Dixon. “A shift-invariant latent variable model for automatic music transcription”. In: *Computer Music Journal* 36.4 (2012), pp. 81–94.

[7] Nancy Bertin, Roland Badeau, and Emmanuel Vincent. “Enforcing harmonicity and smoothness in Bayesian non-negative matrix factorization applied to polyphonic music transcription”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.3 (2010), pp. 538–549.

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

[9] Nicholas J Bryan and Gautham J Mysore. “Interactive refinement of supervised and semi-supervised sound source separation estimates”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on.* IEEE. 2013, pp. 883–887.

[10] 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.

[11] Nicholas J Bryan, Gautham J Mysore, and Ge Wang. “Source Separation of Polyphonic Music with Interactive User-Feedback on a Piano Roll Display.” In: *International Society for Music Information Retrieval (ISMIR)*. Curitiba, Brazil, November 4-8, 2013, pp. 119–124.

[12] Nicholas Bryan and Gautham Mysore. “An efficient posterior regularized latent variable model for interactive sound source separation”. In: *International Conference on Machine Learning*. 2013, pp. 208–216.

[13] J-F Cardoso. “Blind signal separation: statistical principles”. In: *Proceedings of the IEEE* 86.10 (1998), pp. 2009–2025.

[14] 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.

[15] 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.

[16] 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.

[17] Taishih Chi, Powen Ru, and Shihab A Shamma. “Multiresolution spectrotemporal analysis of complex sounds”. In: *The Journal of the Acoustical Society of America* 118.2 (2005), pp. 887–906.

[18] John M Chowning. “Frequency modulation synthesis of the singing voice”. In: *Current directions in computer music research*. MIT Press. 1989, pp. 57–63.

[19] John M Chowning. *Method of synthesizing a musical sound*. US Patent 4,018,121. Apr. 1977.

[20] John M Chowning. “The synthesis of complex audio spectra by means of frequency modulation”. In: *Journal of the audio engineering society* 21.7 (1973), pp. 526–534.

[21] Andrzej Cichocki, Rafal Zdunek, and Shun-ichi Amari. “New algorithms for non-negative matrix factorization in applications to blind source separation”. In: *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*. Vol. 5. IEEE. 2006, pp. V–V.

[22] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation.* John Wiley & Sons, 2009.

[23] Pierre Comon and Christian Jutten. *Handbook of Blind Source Separation: Independent component analysis and applications.* Academic press, 2010.

[24] CJ Darwin. "Perceiving vowels in the presence of another sound: A quantitative test of the "Old-plus-new" Heuristic". In: *Levels in Speech Communication: Relations and Interactions: A Tribute to Max WajskopAmsterdam, Elsevier* (1995), pp. 1–12.

[25] Terrance DeVries and Graham W Taylor. "Learning Confidence for Out-of-Distribution Detection in Neural Networks". In: *arXiv preprint arXiv:1802.04865* (2018).

[26] Jonathan Driedger and Meinard Müller. "Extracting Singing Voice from Music Recordings by Cascading Audio Decomposition Techniques". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* Brisbane, Australia, 2015, pp. 126–130.

[27] Jonathan Driedger, Meinard Muller, and Sebastian Ewert. "Improving time-scale modification of music signals using harmonic-percussive separation". In: *IEEE Signal Processing Letters* 21.1 (2014), pp. 105–109.

[28] Lukas Drude, Jahn Heymann, and Reinhold Haeb-Umbach. "Unsupervised training of neural mask-based beamforming". In: *arXiv preprint arXiv:1904.01578* (2019).

[29] Zhiyao Duan, Gautham J Mysore, and Paris Smaragdis. "Online PLCA for real-time semi-supervised source separation". In: *International Conference on Latent Variable Analysis and Signal Separation.* Springer Berlin Heidelberg. 2012, pp. 34–41.

[30] 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.

[31] Zhiyao Duan, Bryan Pardo, and Changshui Zhang. "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions". In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.8 (2010), pp. 2121–2133.

[32] J Durrieu, Bertrand David, and Gaël Richard. "A musically motivated mid-level representation for pitch estimation and musical audio source separation". In: *IEEE Journal of Selected Topics in Signal Processing* 5.6 (2011), pp. 1180–1191.

[33] Jean-Louis Durrieu, Gaël Richard, Bertrand David, and Cédric Févotte. "Source/filter model for unsupervised main melody extraction from polyphonic audio signals". In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.3 (2010), pp. 564–575.

[34] Mounya Elhilali. "Modulation Representations for Speech and Music". In: *Timbre: Acoustics, Perception, and Cognition.* Springer, 2019, pp. 335–359.

[35] Mounya Elhilali and Shihab A Shamma. "A cocktail party with a cortical twist: how cortical mechanisms contribute to sound segregation". In: *The Journal of the Acoustical Society of America* 124.6 (2008), pp. 3751–3771.

[36] Daniel PW Ellis and Ron J Weiss. "Model-based monaural source separation using a vector-quantized phase-vocoder representation". In: *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on.* Vol. 5. IEEE. 2006, pp. V–V.

[37] 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.

[38] Ariel Ephrat, Inbar Mosseri, Oran Lang, Tali Dekel, Kevin Wilson, Avinatan Hassidim, William T Freeman, and Michael Rubinstein. "Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation". In: *arXiv preprint arXiv:1804.03619* (2018).

[39] Hakan Erdogan and Takuya Yoshioka. "Investigations on Data Augmentation and Loss Functions for Deep Learning Based Speech-Background Separation." In: *Interspeech.* 2018, pp. 3499–3503.

[40] 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.

[41] 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.

[42] Derry Fitzgerald. "Harmonic/percussive separation using median filtering". In: (2010).

[43] Derry FitzGerald, Matt Cranitch, and Eugene Coyle. "Extended nonnegative tensor factorisation models for musical sound source separation". In: *Computational Intelligence and Neuroscience* 2008 (2008).

[44] Derry Fitzgerald, Antoine Liutkus, and Roland Badeau. “Projection-based demixing of spatial audio”. In: (2016).

[45] Brendan Fox and Bryan Pardo. “Towards a model of perceived quality of blind audio source separation”. In: *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE. 2007, pp. 1898–1901.

[46] 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.

[47] 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.

[48] 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.

[49] Jonathan Fritz, Shihab Shamma, Mounya Elhilali, and David Klein. “Rapid task-related plasticity of spectrotemporal receptive fields in primary auditory cortex”. In: *Nature neuroscience* 6.11 (2003), p. 1216.

[50] Aviv Gabbay, Ariel Ephrat, Tavi Halperin, and Shmuel Peleg. “Seeing through noise: Visually driven speaker separation and enhancement”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 3051–3055.

[51] Olivier Gillet and Gaël Richard. “Transcription and separation of drum signals from polyphonic music”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 16.3 (2008), pp. 529–540.

[52] Larry Gillick, Yoshiko Ito, and Jonathan Young. “A probabilistic approach to confidence estimation and evaluation”. In: *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*. Vol. 2. IEEE. 1997, pp. 879–882.

[53] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[54] Christian Gourieroux, Alberto Holly, and Alain Monfort. “Likelihood ratio test, Wald test, and Kuhn-Tucker test in linear models with inequality constraints on the regression parameters”. In: *Econometrica: journal of the Econometric Society* (1982), pp. 63–80.

[55] Emad M Grais and Mark D Plumbley. “Single channel audio source separation using convolutional denoising autoencoders”. In: *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE. 2017, pp. 1265–1269.

[56] Jinyu Han, Gautham Mysore, and Bryan Pardo. “Audio imputation using the non-negative hidden markov model”. In: *Latent Variable Analysis and Signal Separation* (2012), pp. 347–355.

[57] Jinyu Han and Bryan Pardo. “Improving separation of harmonic sources with iterative estimation of spatial cues”. In: *Applications of Signal Processing to Audio and Acoustics, 2009. WASPAA'09. IEEE Workshop on*. IEEE. 2009, pp. 77–80.

[58] Jinyu Han and Bryan Pardo. “Reconstructing completely overlapped notes from musical mixtures”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE. 2011, pp. 249–252.

[59] Yushen Han and Christopher Raphael. “Informed Source Separation of Orchestra and Soloist.” In: *ISMIR*. 2010, pp. 315–320.

[60] Marko Helen and Tuomas Virtanen. “Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine”. In: *Signal Processing Conference, 2005 13th European*. IEEE. 2005, pp. 1–4.

[61] Dan Hendrycks and Kevin Gimpel. “A baseline for detecting misclassified and out-of-distribution examples in neural networks”. In: *arXiv preprint arXiv:1610.02136* (2016).

[62] 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.

[63] Wolfgang Herbordt and Walter Kellermann. “Adaptive beamforming for audio signal acquisition”. In: *Adaptive Signal Processing*. Springer, 2003, pp. 155–194.

[64] Hynek Hermansky, Ehsan Variani, and Vijayaditya Peddinti. “Mean temporal distance: predicting ASR error from temporal properties of speech signal”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 7423–7426.

[65] 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.

[66] 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.

[67] 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.

[68] Geoffrey E Hinton. "Connectionist learning procedures". In: *Artificial intelligence* 40.1-3 (1989), pp. 185–234.

[69] Po-Sen Huang, Scott Deeann Chen, Paris Smaragdis, and Mark Hasegawa-Johnson. "Singing-voice separation from monaural recordings using robust principal component analysis". In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE. 2012, pp. 57–60.

[70] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. "Joint optimization of masks and deep recurrent neural networks for monaural source separation". In: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 23.12 (2015), pp. 2136–2147.

[71] Po-Sen Huang, Minje Kim, Mark Hasegawa-Johnson, and Paris Smaragdis. "Singing-Voice Separation from Monaural Recordings using Deep Recurrent Neural Networks." In: *ISMIR*. 2014, pp. 477–482.

[72] Po-Sen Huang, Kush Kumar, Chaojun Liu, Yifan Gong, and Li Deng. "Predicting speech recognition confidence using deep learning with word identity and score features". In: *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE. 2013, pp. 7413–7417.

[73] Aapo Hyvärinen and Erkki Oja. "Independent component analysis: algorithms and applications". In: *Neural networks* 13.4 (2000), pp. 411–430.

[74] Satoshi Imai. "Cepstral analysis synthesis on the mel frequency scale". In: *ICASSP'83. IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 8. IEEE. 1983, pp. 93–96.

[75] Yusuf Isik, Jonathan Le Roux, Zhuo Chen, Shinji Watanabe, and John R Hershey. "Single-channel multi-speaker separation using deep clustering". In: *arXiv preprint arXiv:1607.02173* (2016).

[76] Katsutoshi Itoyama, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. "Instrument Equalizer for Query-by-Example Retrieval: Improving Sound Source Separation Based on Integrated Harmonic and Inharmonic Models." In: *ISMIR*. 2008, pp. 133–138.

[77] Anil K Jain. "Data clustering: 50 years beyond K-means". In: *Pattern recognition letters* 31.8 (2010), pp. 651–666.

[78] Rajesh Jaiswal, Derry FitzGerald, Dan Barry, Eugene Coyle, and Scott Rickard. "Clustering NMF basis functions using shifted NMF for monaural sound source separation". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE. 2011, pp. 245–248.

[79] Tzyy-Ping Jung, Scott Makeig, Colin Humphries, Te-Won Lee, Martin J Mckeown, Vicente Iragui, and Terrence J Sejnowski. "Removing electroencephalographic artifacts by blind source separation". In: *Psychophysiology* 37.2 (2000), pp. 163–178.

[80] Christian Kaernbach. "The memory of noise". In: *Experimental psychology* 51.4 (2004), pp. 240–248.

[81] Hirokazu Kameoka, Takuya Nishimoto, and Shigeki Sagayama. "A multipitch analyzer based on harmonic temporal structured clustering". In: *IEEE Transactions on Audio, Speech, and Language Processing* 15.3 (2007), pp. 982–994.

[82] Walter Kellermann. "Beamforming for speech and audio signals". In: *Handbook of signal processing in acoustics*. Springer, 2008, pp. 691–702.

[83] Minje Kim. "Collaborative Deep Learning for Speech Enhancement: A Run-Time Model Selection Method Using Autoencoders". In: *arXiv preprint arXiv:1705.10385* (2017).

[84] Minje Kim, Seungkwon Beack, Keunwoo Choi, and Kyeongok Kang. "Gaussian mixture model for singing voice separation from stereophonic music". In: *Audio Engineering Society Conference: 43rd International Conference: Audio for Wirelessly Networked Personal Devices*. Audio Engineering Society. 2011.

[85] Minje Kim and Paris Smaragdis. "Adaptive denoising autoencoders: A fine-tuning scheme to learn from test mixtures". In: *International Conference on Latent Variable Analysis and Signal Separation*. Springer. 2015, pp. 100–107.

[86] Minje Kim and Paris Smaragdis. "Collaborative audio enhancement using probabilistic latent component sharing". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 896–900.

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

[88] S Kırbız and B Gunsel. "Perceptually weighted non-negative matrix factorization for blind single-channel music source separation". In: *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE. 2012, pp. 226–229.

[89] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[90] Rajath Kumar, Yi Luo, and Nima Mesgarani. "Music Source Activity Detection and Separation Using Deep Attractor Network". In: *Proc. Interspeech 2018* (2018), pp. 347–351.

[91] Stephen Lakatos. "A common perceptual space for harmonic and percussive timbres". In: *Perception & psychophysics* 62.7 (2000), pp. 1426–1439.

[92] 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.

[93] Jonathan Le Roux, Scott Wisdom, Hakan Erdogan, and John R Hershey. "SDR-half-baked or well done?" In: *arXiv preprint arXiv:1811.02508* (2018).

[94] Daniel D Lee and H Sebastian Seung. "Learning the parts of objects by non-negative matrix factorization". In: *Nature* 401.6755 (1999), pp. 788–791.

[95] Te-Won Lee. "Independent component analysis". In: *Independent Component Analysis*. Springer, 1998, pp. 27–66.

[96] Simon Leglaive, Umut Şimşekli, Antoine Liutkus, Roland Badeau, and Gaël Richard. "Alpha-stable multichannel audio source separation". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017, pp. 576–580.

[97] Mu Li, Xiao-Chen Lian, James T Kwok, and Bao-Liang Lu. "Time and space efficient spectral clustering via column sampling". In: *CVPR 2011*. IEEE. 2011, pp. 2297–2304.

[98] Yipeng Li and DeLiang Wang. "Singing Voice Separation from Monaural Recordings." In: *ISMIR*. vol. 176. Citeseer. 2006, p. 179.

[99] Shiyu Liang, Yixuan Li, and R Srikant. "Enhancing the reliability of out-of-distribution image detection in neural networks". In: *arXiv preprint arXiv:1706.02690* (2017).

[100] Frank R. Lin, John K. Niparko, and Luigi Ferrucci. “Hearing Loss Prevalence in the United States”. In: *JAMA Internal Medicine* 171.20 (Nov. 2011), pp. 1851–1853.

[101] Antoine Liutkus, Derry Fitzgerald, Zafar Rafii, Bryan Pardo, and Laurent Daudet. “Kernel additive models for source separation”. In: *Signal Processing, IEEE Transactions on* 62.16 (2014), pp. 4298–4310.

[102] Antoine Liutkus, Zafar Rafii, Roland Badeau, Bryan Pardo, and Gaël Richard. “Adaptive filtering for music/voice separation exploiting the repeating musical structure”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE. 2012, pp. 53–56.

[103] 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.

[104] 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.

[105] 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).

[106] Yi Luo, Zhuo Chen, and Nima Mesgarani. “Speaker-independent speech separation with deep attractor network”. In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 26.4 (2018), pp. 787–796.

[107] 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.

[108] Richard Lyon. “Computational models of neural auditory processing”. In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP’84.* Vol. 9. IEEE. 1984, pp. 41–44.

[109] Paul Magron, Roland Badeau, and Antoine Liutkus. “Generalized Wiener filtering for positive alpha-stable random variables”. In: (2016).

[110] Michael I Mandel, Ron J Weiss, and Daniel PW Ellis. "Model-based expectation-maximization source separation and localization". In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.2 (2009), pp. 382–394.

[111] Ethan Manilow, Prem Seetharaman, and Bryan Pardo. "The Northwestern University Source Separation Library". In: *Proceedings of the 19th International Society of Music Information Retrieval Conference (ISMIR 2018).* 2018.

[112] 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.

[113] Aaron S Master. "Bayesian two source modeling for separation of N sources from stereo signals". In: *Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on.* Vol. 4. IEEE. 2004, pp. iv–iv.

[114] Josh H McDermott, D.P.W. Ellis, and E.P. Simoncelli. "Empirical derivation of acoustic grouping cues from natural sound statistics". In: *Association for Research in Otolaryngology, Annual Meeting* (2011).

[115] Josh H McDermott and Eero P Simoncelli. "Sound texture perception via statistics of the auditory periphery: evidence from sound synthesis". In: *Neuron* 71.5 (2011), pp. 926–940.

[116] Josh H McDermott, D. Wrobleski, and A. Oxenham. "Recovering sound sources from embedded repetition". In: *Proceedings of the National Academy of Sciences* 108.3 (2011), pp. 1188–1193.

[117] Brian McFee and Daniel PW Ellis. "Better beat tracking through robust onset aggregation". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE. 2014, pp. 2154–2158.

[118] M McVicar, R Santos-Rodriguez, and T De Bie. "Learning to separate vocals from polyphonic mixtures via ensemble methods and structured output prediction". In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on.* IEEE. 2016, pp. 450–454.

[119] Shariq Mobin, Brian Cheung, and Bruno A. Olshausen. "Convolutional vs. Recurrent Neural Networks for Audio Source Separation". In: *CoRR* abs/1803.08629 (2018). arXiv: `1803.08629`. URL: `http://arxiv.org/abs/1803.08629`.

[120] Todd K Moon. "The expectation-maximization algorithm". In: *IEEE Signal processing magazine* 13.6 (1996), pp. 47–60.

[121] Gautham J Mysore, Paris Smaragdis, and Bhiksha Raj. "Non-negative hidden Markov modeling of audio with application to source separation". In: *Latent variable analysis and signal separation.* Springer, 2010, pp. 140–148.

[122] Andrew Ng et al. "Sparse autoencoder". In: *CS294A Lecture notes* 72.2011 (2011), pp. 1–19.

[123] Aditya Arie Nugraha, Antoine Liutkus, and Emmanuel Vincent. "Multichannel music separation with deep neural networks". In: *Signal Processing Conference (EUSIPCO), 2016 24th European.* IEEE. 2016, pp. 1748–1752.

[124] N. Ono, Z. Rafii, D. Kitamura, N. Ito, and A. Liutkus. "The 2015 Signal Separation Evaluation Campaign". In: *International Conference on Latent Variable Analysis and Signal Separation (LVA/ICA).* vol. 9237. Latent Variable Analysis and Signal Separation. Liberec, France, Aug. 2015, pp. 387–395.

[125] Alexey Ozerov and Cédric Févotte. "Multichannel nonnegative matrix factorization in convolutive mixtures for audio source separation". In: *IEEE Transactions on Audio, Speech, and Language Processing* 18.3 (2009), pp. 550–563.

[126] Alexey Ozerov, Cédric Févotte, Raphaël Blouet, and Jean-Louis Durrieu. "Multichannel nonnegative tensor factorization with structured constraints for user-guided audio source separation". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on.* IEEE. 2011, pp. 257–260.

[127] Alexey Ozerov, Antoine Liutkus, Roland Badeau, and Gaël Richard. "Informed source separation: source coding meets source separation". In: *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2011 IEEE Workshop on.* IEEE. 2011, pp. 257–260.

[128] Alexey Ozerov, Emmanuel Vincent, and Frédéric Bimbot. "A general flexible framework for the handling of prior information in audio source separation". In: *IEEE Transactions on Audio, Speech, and Language Processing* 20.4 (2012), pp. 1118–1133.

[129] Constantinos B Papadias and Arogyaswami J Paulraj. "A constant modulus algorithm for multiuser signal separation in presence of delay spread using antenna arrays". In: *IEEE signal processing letters* 4.6 (1997), pp. 178–181.

[130] Bryan Pardo, Antoine Liutkus, Zhiyao Duan, and Gael Richard. "Applying Source Separation to Music". In: *Audio Source Separation and Speech Enhancement.* 2018.

[131] Lucas C Parra. "Temporal models in blind source separation". In: *Adaptive processing of sequences and data structures.* Springer, 1998, pp. 229–247.

[132] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. "Automatic differentiation in pytorch". In: (2017).

[133] Roy D Patterson. "Auditory images: How complex sounds are represented in the auditory system". In: *Acoustical Science and Technology* 21.4 (2001), pp. 183–190.

[134] Roy D Patterson, Mike H Allerhand, and Christian Giguere. "Time-domain modeling of peripheral auditory processing: A modular architecture and a software platform". In: *The Journal of the Acoustical Society of America* 98.4 (1995), pp. 1890–1894.

[135] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. "The matrix cookbook". In: *Technical University of Denmark* 7.15 (2008), p. 510.

[136] Erin M Picou and Todd A Ricketts. "An Evaluation of Hearing Aid Beamforming Microphone Arrays in a Noisy Laboratory Setting". In: *Journal of the American Academy of Audiology* 30.2 (2019), pp. 131–144.

[137] Fatemeh Pishdadian and Bryan Pardo. "Multi-resolution Common Fate Transform". In: *Transactions on Audio, Speech, and Language Processing* (2018). DOI: 10.1109/TASLP.2018.2878616.

[138] Fatemeh Pishdadian, Bryan Pardo, and Antoine Liutkus. "A Multi-resolution Approach to Common Fate-based Audio Separation". In: *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* New Orleans, LA, USA, March 5-9, 2017.

[139] Reinier Plomp. "The ear as a frequency analyzer". In: *The Journal of the Acoustical Society of America* 36.9 (1964), pp. 1628–1636.

[140] Zafar Rafii, Zhiyao Duan, and Bryan Pardo. "Combining rhythm-based and pitch-based methods for background and melody separation". In: *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)* 22.12 (2014), pp. 1884–1893.

[141] Zafar Rafii and B. Pardo. "Repeating pattern extraction technique (REPET): A simple method for music/voice separation". In: *Transactions on Audio, Speech, and Lang. Processing* 21.1 (2013), pp. 73–84.

[142] Zafar Rafii and Bryan Pardo. "Degenerate unmixing estimation technique using the constant Q transform". In: *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on.* IEEE. 2011, pp. 217–220.

[143] Zafar Rafii and Bryan Pardo. “Music/Voice Separation Using the Similarity Matrix.” In: *ISMIR*. 2012, pp. 583–588.

[144] Zafar Rafii and Bryan Pardo. “Online REPET-SIM for real-time speech enhancement”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 848–852.

[145] Bhiksha Raj and Paris Smaragdis. “Latent variable decomposition of spectrograms for single channel speaker separation”. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, 2005*. IEEE. 2005, pp. 17–20.

[146] Kamisetty Ramamohan Rao, Do Nyeon Kim, and Jae Jeong Hwang. *Fast Fourier transform-algorithms and applications*. Springer Science & Business Media, 2011.

[147] Scott Rickard. “The DUET blind source separation algorithm”. In: *Blind Speech Separation* (2007), pp. 217–241.

[148] Scott Rickard, R. Balan, and J. Rosca. “Real-time time-frequency based blind source separation”. In: *Proc. ICA*. 2001, pp. 651–656.

[149] Lawrence D Rosenblum, Mark A Schmuckler, and Jennifer A Johnson. “The McGurk effect in infants”. In: *Perception & Psychophysics* 59.3 (1997), pp. 347–357.

[150] Andrew Rouditchenko, Hang Zhao, Chuang Gan, Josh McDermott, and Antonio Torralba. “Self-Supervised Segmentation and Source Separation on Videos”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019, pp. 0–0.

[151] Peter J Rousseeuw. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis”. In: *Journal of computational and applied mathematics* 20 (1987), pp. 53–65.

[152] Powen Ru and Shihab A Shamma. “Representation of musical timbre in the auditory cortex”. In: *Journal of New Music Research* 26.2 (1997), pp. 154–169.

[153] John C Russ and Roger P Woods. *The image processing handbook*. 1995.

[154] 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).

[155] Mikkel N Schmidt and Rasmus K Olsson. “Single-channel speech separation using sparse non-negative matrix factorization”. In: *Ninth International Conference on Spoken Language Processing*. 2006.

[156] Andrew Schwartz, Josh H McDermott, and Barbara Shinn-Cunningham. "Spatial cues alone produce inaccurate sound segregation: The effect of interaural time differences". In: *The Journal of the Acoustical Society of America* 132.1 (2012), pp. 357–368.

[157] Prem Seetharaman and Bryan Pardo. "Simultaneous Separation and Segmentation in Layered Music". In: *Proceedings of the 17th International Society for Music Information Retrieval Conference.* 2016, pp. 495–502.

[158] Prem Seetharaman, Fatemeh Pishdadian, and Bryan Pardo. "Music/Voice Separation Using the 2D Fourier Transform". In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA 2017).* New Paltz, NY, USA, October 15-18, 2017.

[159] 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.

[160] Shihab Shamma. "Encoding sound timbre in the auditory system". In: *IETE Journal of research* 49.2-3 (2003), pp. 145–156.

[161] Shihab A Shamma, Mounya Elhilali, and Christophe Micheyl. "Temporal coherence and attention in auditory scene analysis". In: *Trends in neurosciences* 34.3 (2011), pp. 114–123.

[162] *ShotSpotter.* URL: `https://shotspotter.com/`.

[163] Umut Şimşekli, Antoine Liutkus, and Ali Taylan Cemgil. "Alpha-stable matrix factorization". In: *IEEE Signal Processing Letters* 22.12 (2015), pp. 2289–2293.

[164] 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.

[165] 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.

[166] Paris Smaragdis, Bhiksha Raj, and Madhusudana Shashanka. "Supervised and semi-supervised separation of sounds from single-channel mixtures". In: *International Conference on Independent Component Analysis and Signal Separation.* Springer. 2007, pp. 414–421.

[167] Paris Smaragdis, Madhusudana Shashanka, Bhiksha Raj, and Gautham J Mysore. "Probabilistic factorization of non-negative data with entropic co-occurrence constraints". In: *International Conference on Independent Component Analysis and Signal Separation*. Springer. 2009, pp. 330–337.

[168] Paris Smaragdis and Shrikant Venkataramani. "A Neural Network Alternative to Non-Negative Audio Models". In: *arXiv preprint arXiv:1609.03296* (2016).

[169] Martin Spiertz and Volker Gnann. "Source-filter based clustering for monaural blind source separation". In: *Proceedings of the 12th International Conference on Digital Audio Effects*. 2009.

[170] Pablo Sprechmann, Alexander M Bronstein, and Guillermo Sapiro. "Real-time Online Singing Voice Separation from Monaural Recordings Using Robust Low-rank Modeling." In: *ISMIR*. 2012, pp. 67–72.

[171] 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).

[172] Fabian-Robert Stöter, Antoine Liutkus, Roland Badeau, Bernd Edler, and Paul Magron. "Common fate model for unison source separation". In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 126–130.

[173] Fabian-Robert Stöter, Antoine Liutkus, and Nobutaka Ito. "The 2018 Signal Separation Evaluation Campaign". In: *International Conference on Latent Variable Analysis and Signal Separation*. Springer. 2018, pp. 293–305.

[174] Akshayvarun Subramanya, Suraj Srinivas, and R Venkatesh Babu. "Confidence estimation in Deep Neural networks via density modelling". In: *arXiv preprint arXiv:1707.07013* (2017).

[175] Johan Sundberg. "Acoustic and psychoacoustic aspects of vocal vibrato". In: *Vibrato* (1995), pp. 35–62.

[176] Ilya Sutskever, Geoffrey E Hinton, and Graham W Taylor. "The recurrent temporal restricted boltzmann machine". In: *Advances in neural information processing systems*. 2009, pp. 1601–1608.

[177] Fabian J Theis, Carlos G Puntonet, and Elmar W Lang. "A histogram-based overcomplete ICA algorithm". In: *The 4th International Symposium on Independent Component Analysis and Blind Signal Separation*. Nara, Japan, April, 2003, pp. 1071–1076.

[178] Tero Tolonen. "Methods for separation of harmonic sound sources using sinusoidal modeling". In: *Audio Engineering Society Convention 106*. Audio Engineering Society. 1999.

[179] Stefan Uhlich, Franck Giron, and Yuki Mitsufuji. “Deep neural network based instrument extraction from music”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 2135–2139.

[180] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. “Instance normalization: The missing ingredient for fast stylization”. In: *arXiv preprint arXiv:1607.08022* (2016).

[181] Barry D Van Veen and Kevin M Buckley. “Beamforming: A versatile approach to spatial filtering”. In: *IEEE assp magazine* 5.2 (1988), pp. 4–24.

[182] Shrikant Venkataramani, Jonah Casebeer, and Paris Smaragdis. “End-to-end source separation with adaptive front-ends”. In: *2018 52nd Asilomar Conference on Signals, Systems, and Computers*. IEEE. 2018, pp. 684–688.

[183] Shrikant Venkataramani and Paris Smaragdis. “End-to-end Source Separation with Adaptive Front-Ends”. In: *arXiv preprint arXiv:1705.02514* (2017).

[184] Emmanuel Vincent. “Musical source separation using time-frequency source priors”. In: *IEEE Transactions on Audio, Speech, and Language Processing* 14.1 (2006), pp. 91–98.

[185] 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.

[186] Emmanuel Vincent and Xavier Rodet. “Underdetermined source separation with structured source priors”. In: *International Conference on Independent Component Analysis and Signal Separation*. Springer. 2004, pp. 327–334.

[187] Emmanuel Vincent, Hiroshi Sawada, Pau Bofill, Shoji Makino, and Justinian P Rosca. “First stereo audio source separation evaluation campaign: data, algorithms and results”. In: *International Conference on Independent Component Analysis and Signal Separation*. Springer. 2007, pp. 552–559.

[188] 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.

[189] Tuomas Virtanen, A Taylan Cemgil, and Simon Godsill. “Bayesian extensions to non-negative matrix factorisation for audio signal modelling”. In: *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2008, pp. 1825–1828.

[190] Tuomas Virtanen and Anssi Klapuri. "Separation of harmonic sound sources using sinusoidal modeling". In: *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on.* Vol. 2. IEEE. 2000, pp. II765–II768.

[191] Tuomas Virtanen and Anssi Klapuri. "Separation of harmonic sounds using linear models for the overtone series". In: *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on.* Vol. 2. IEEE. 2002, pp. II–1757.

[192] Paul Joseph Walmsley. "Signal separation of musical instruments". In: *Signal Processing and Communications* (2000).

[193] 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.

[194] Yuxuan Wang and DeLiang Wang. "Towards scaling up classification-based speech separation". In: *IEEE Transactions on Audio, Speech, and Language Processing* 21.7 (2013), pp. 1381–1390.

[195] 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.

[196] Zhong-Qiu Wang, Jonathan Le Roux, and John R Hershey. "Multi-Channel Deep Clustering: Discriminative Spectral and Spatial Embeddings for Speaker-Independent Speech Separation". In: *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* 2018.

[197] Richard M Warren, James A Bashford, Jeffrey M Cooley, and Bradley S Brubaker. "Detection of acoustic repetition for very long stochastic patterns". In: *Perception & psychophysics* 63.1 (2001), pp. 175–182.

[198] István Winkler, Susan Denham, Robert Mill, Tamás M Bőhm, and Alexandra Bendixen. "Multistability in auditory stream segregation: a predictive coding view". In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* 367.1591 (2012), pp. 1001–1012.

[199] Guy Wolf, Stéphane Mallat, and Shihab Shamma. "Rigid motion model for audio source separation". In: *IEEE Transactions on Signal Processing* 64.7 (2016), pp. 1822–1831.

[200] John F Woodruff, Bryan Pardo, and Roger B Dannenberg. "Remixing Stereo Music with Score-Informed Source Separation." In: *ISMIR.* 2006, pp. 314–319.

[201] John Woodruff and Bryan Pardo. “Using pitch, amplitude modulation, and spatial cues for separation of harmonic instruments from stereo music recordings”. In: *EURASIP Journal on Advances in Signal Processing* 2007.1 (2006), p. 086369.

[202] Hiroaki Yamajo, Hiroshi Saruwatari, Tomoya Takatani, Tsuyoki Nishikawa, and Kiyohiro Shikano. “Evaluation of blind separation and deconvolution for convolutive speech mixture using SIMO-model-based ICA”. in: (2003).

[203] Donghui Yan, Ling Huang, and Michael I Jordan. “Fast approximate spectral clustering”. In: *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2009, pp. 907–916.

[204] Delia Fano Yela, Sebastian Ewert, Derry FitzGerald, and Mark Sandler. “Interference Reduction in Music Recordings Combining Kernel Additive Modelling and Non-Negative Matrix Factorization”. In: *arXiv preprint arXiv:1609.06210* (2016).

[205] Ozgur Yilmaz and Scott Rickard. “Blind separation of speech mixtures via time-frequency masking”. In: *IEEE Transactions on signal processing* 52.7 (2004), pp. 1830–1847.

[206] Yichi Zhang and Zhiyao Duan. “Imisound: An unsupervised system for sound query by vocal imitation”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 2269–2273.

[207] Hang Zhao, Chuang Gan, Andrew Rouditchenko, Carl Vondrick, Josh McDermott, and Antonio Torralba. “The sound of pixels”. In: *arXiv preprint arXiv:1804.03160* (2018).