DATA-DRIVEN PITCH CORRECTION FOR SINGING

Sanna Catherine de Treville Wager

Submitted to the faculty of the University Graduate School

in partial fulfillment of the requirements

for the degree

Doctor of Philosophy

in the Luddy School of Informatics, Computing, and Engineering,

Indiana University

March 2021

Accepted by the Graduate Faculty, Indiana University, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Doctoral Committee

Minje Kim, Ph.D.

Daniel J. McDonald, Ph.D.

Christopher Raphael, Ph.D.

Donald S. Williamson, Ph.D.

Date of Defense: 01/20/2021

ACKNOWLEDGEMENTS

I cannot thank enough all those who have supported me throughout the PhD. First and foremost, I would like to express my gratitude to my advisor, Prof. Minje Kim, for drawing me out as a researcher, helping me grow my career, always being excited about music-related projects, and spoiling the lab with delicious food. Prof. Christopher Raphael introduced me to the field of music informatics, sharing his passion for the field from when he taught my first undergraduate computer science class. My interactions with Chris have taught me how to address complex and hard-to-define research problems. Prof. Daniel McDonald brought Kalman filters to life for me during a research rotation. Prof. Donald Williamson provided an essential perspective on deep learning as applied to audio and speech that has been very helpful.

My colleagues in the SAIGE lab and, more generally, in the Luddy School of Informatics, Computing, and Engineering, were a constant source of friendship and research discussions. I would like to thank Sunwoo Kim for his endless curiosity; Aswin Sivaraman for helping me prepare figures in the early hours of the morning; Kai Zhen for his huge support in singing for my subjective tests; Haici Yang and David Badger—though we only overlapped for a year—for the insights they brought to the lab. Kahyun Choi provided inspiration for how to finish a thesis remotely and contributed to making my time at school fun. I would also like to thank Lijiang Guo for being a great cubicle neighbor, and Liang Chen, Yupeng Gu, Rong Jin, and Yucong Jiang for many interesting discussions about music. Thanks also to the many other faculty, students, and staff at Indiana University Bloomington, who provided inspiration for my work and made my studies enjoyable. In particular, I would like to thank Prof. David Crandall for serving on my qualifying committee, and Prof. Predrag Radivojac for teaching me crucial skills in machine learning. Finally, Prof. Paris Smaragdis from University of Illinois at Urbana-Champaign has been a terrific academic grandfather.

My internships at Google, Smule, Amazon, and Spotify helped me advance both as a researcher and a software engineer, and provided new mentors and friends. I would like to thank Glenn

Kasten, Andy Hung, John Shimmin, George Tzanetakis, Cheng-i Wang, Stefan Sullivan, Perry Cook, Shiva Sundaram, Aparna Khare, Sebastian Ewert, Keunwoo Choi, Simon Durand, and all my other teammates, for the wonderful opportunities I had to work with them. Thank you also to Tarun Pruthi and Trausti Kristjansson for their support as I completed my thesis while starting my first full-time job on the Amazon Lab126 audio team.

I would also like to thank the many friends who made my time at Indiana University Bloomington very special and fun, and helped me make it through the challenges inherent to getting a PhD. Let me mention Minju Kim and her ideas for exciting road trips, and Ruth Osbrink who became a friend in undergrad. Ruth has been a constant source of adventures around town and an encouragement as I transitioned from music performance to music informatics. Warm thanks and gratitude to all the Bloomington friends for the companionship and support.

My family's belief in me has been a big help in making it to the end of the degree. In particular, thank you to my parents for spoiling me with great food throughout my studies and being willing to listen to my complaints at any time. Thank you also to my brother Stefan and sister-in-law Julie for their helpful insights and for taking me on long hikes during the months when I was finalizing the thesis. Finally, I would like to thank the extended family for the wonderful support and encouragement that they have provided.

Sanna Catherine de Treville Wager

DATA-DRIVEN PITCH CORRECTION FOR SINGING

In this thesis, I introduce a data-driven algorithm for estimating and applying pitch corrections to a vocal performance recording. Existing pitch correction systems usually map the vocals pitch to a discrete set of values, such as the notes in the equal-tempered scale. The proposed algorithm instead treats pitch as a continuous parameter. Continuous pitch representation enables the algorithm to apply to a wide variety of musical traditions, regardless of the scales and pitch variation patterns used in these traditions.

The algorithm uses both the vocal and backing tracks as references for its pitch correction predictions. It is built around a deep neural network model, which includes convolutional layers for detecting patterns in the audio signals, and recurrent layers for processing the song sequentially. The model is trained on karaoke performances selected for accuracy. It is exposed both to incorrect intonation, for which it learns a correction, and intentional pitch variation, which it learns to preserve.

This thesis includes arguments for favoring a data-driven approach to digital music processing over a model-based approach, prioritizing expressivity over control and interpretability. The proposed algorithm shows promising performance on real-world singing.

Minje Kim, Ph.D.

Daniel J. McDonald, Ph.D.

Christopher Raphael, Ph.D.

Donald S. Williamson, Ph.D.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Digital music builds on a rich music tradition in the acoustic realm, but also differs from it by nature: Instead of directly reaching our ears, digital music first needs to be represented as data that a computer can process. Our listening experience depends on the quality of the representation. Design and engineering decisions lie behind the representation, from the sound wave itself—stored as a sequence of numbers—to musical structure, including melody, harmony, and rhythm. In this thesis, I introduce an automatic pitch correction algorithm for singing voice. The pitch representation in the algorithm is designed to capture the complex way we experience pitch and manipulate it when singing. This thesis work builds on studies of how physical measures of pitch, such as frequency and timbre, map to what we hear through a complex psychoacoustic process influenced by our musical upbringing. This complex process results in us either considering a note to be "in tune" or not.

Music making with family, friends, and the wider community is one of the universally enjoyed activities across the world. In recent years, digital options have emerged in applications such as Smule, Spotify, Cadenza, YouTube, and TikTok. These enable people to share their recordings with their peers, and provide tools for collaboration and audio processing that only exist in the digital realm. These apps use post-processing, which involves editing the track after it has been recorded to improve the quality. Automatic pitch correction to make a performance sound in tune is one type of post-processing.

Why not just leave the recording unaltered? As anyone who has tried to photograph a beautiful landscape has experienced, capturing the natural colors, beauty, and atmosphere of a scene is challenging. Image post-processing might actually make the outcome better capture reality or, alternatively, bring to the viewer's attention aspects of the landscape that could otherwise be easily missed. In the same way, post-processing of audio can enhance the listening experience: The

listener is hearing the performance through the recording and playback pipeline, outside of the original acoustic setting. A performance heard recorded versus live can be a substantially different musical experience.

The process of producing a digital recording can range from professional to recreational. Professional digital recording involves the use of professional audio-editing software, high-end microphones, and recording spaces with good acoustics, all of which require considerable time and resources. It results in a polished and high-quality result. The recreational kind often involves multipurpose recording devices such as smartphones, and typically involves little in the way of time and resources. Music apps can serve as a platform for recreational music making. Apps can provide a meaningful musical opportunity. Users of Smule, Inc, a singing app, have reported taking a ten-minute break in their workday to sing, and shared with the company how that short singing break brings them joy and reduces their stress. Without being willing to invest time and money in editing their recording or having the resources or training to do so, recreational musicians may still benefit from using post-processing tools that are built into the app. These apply automatic pitch correction or add effects such as reverb. Post-processing tools do not replace professional audio editing, but are recognized as improving the quality of the recording with remarkably little effort or cost. Although post-processing tools do not make a recording sound professional, they can make it more pleasing to a listener. A parallel can be seen when a person uses a grammar-checker to improve the quality of their writing, even when writing simple text such as an email. The ability of the grammar-checker to remove minor errors produces a more polished result, which can make a big difference, especially given the high stakes of sharing content in a recorded format.

1.1 Challenges in data-driven automatic pitch correction

I illustrate the complexity of the task of automatic pitch correction by describing the difference between a musical score and a performance. In a musical score, a melody for singing voice is typically notated as a sequence of notes of discretized lengths and pre-defined pitches. The simplicity of the symbolic representation leaves considerable scope for variation in the singer's interpreta-

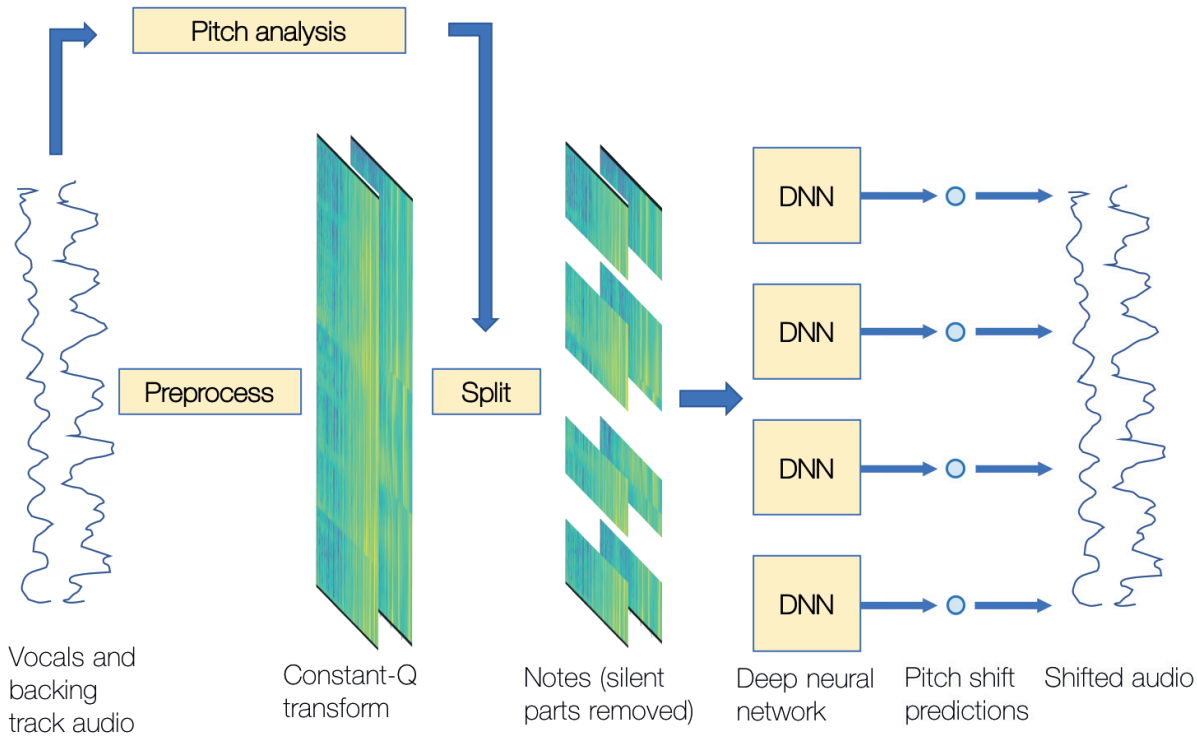

Figure 1.1: Pitch correction algorithm overview. The algorithm requires audio for the vocals and backing track (also called accompaniment) on separate audio tracks. It applies the constant-Q transform to the audio to generate time-frequency representations. It then splits the data into individual notes based on the measured singing frequency. It discards silent sections in the audio. Next, for every note, it predicts a pitch correction shift using a deep neural network (DNN) trained on real-world singing examples. Finally, in the post processing phase, it applies the shifts to every note and combines them along with silent sections to construct the corrected track.

tion. Even when a vocalist follows the general contour of the score, the singing voice actually varies continuously due to expressive gestures such as pitch bending, vibrato, and other variations coming from musical tradition, personal preference, and randomness. A singer's deviation from the symbolic score is sometimes large, as described in Chapter 2.

In existing commercial systems for automatic pitch correction, vocal track notes are typically shifted to be centered around pitches in a user-defined score, or mapped to the closest pitch among the twelve equal-tempered scale degrees (e.g., [1]). This approach reliably corrects out-of-tune singing. The downside is that it also removes intentional nuances whenever pitch deviates from the symbolic score.

The algorithm in this thesis represents pitch as a continuous parameter instead of a discretized

value. This continuous representation enables it to preserve the nuanced variations of sung pitch while applying corrections. Furthermore, the algorithm makes its predictions purely on the audio content of the vocals and backing track (also called accompaniment), without relying on a musical score. This leaves room for the singer to sing a different melody, as is common when harmonizing or improvising.

Making a singer's pitch track sound more in tune without relying on a symbolic score is not straightforward. The algorithm must behave like a human listener with a moderate level of musical understanding. Such a listener can often detect the out-of-tune notes and predict the amount and direction of the pitch shift required to bring the note back in tune, all without requiring access to the score. The algorithm in this thesis uses a Deep Neural Network (DNN) to make predictions. The DNN is trained on real-world singing examples, from which it learns patterns of intonation that help it make accurate predictions.

Training data for automatic pitch correction is hard to find. The first challenge involves collecting examples of in-tune singing. Publicly available datasets typically mix performances of all levels of singing, in tune and out of tune. The in-tune recordings need to be extracted to form a smaller training dataset. The second challenge—if training the DNN in a supervised manner—is to design data pairs where the input is out of tune, the target is in tune, but the signals are otherwise identical. Such pairs don't occur naturally, making data synthesis a viable approach. Synthesizing out-of-tune singing from an in-tune performance, however, requires defining a de-tuning algorithm, which is a challenge in itself. Chapters 2 and 3 illustrate the difficulty of representing pitch as a set of features that can be measured and manipulated. Chapter 4 delves into the numerous decisions around feature design.

1.2 Overview

The algorithm introduced in this thesis predicts pitch corrections of solo singing performances in a data-driven manner. It predicts note-wise pitch shifts from the relationship between the respective spectrograms of the singing and backing track. It outputs the amount and direction of the pitch

shift expected to bring the note back in tune. The pitch shift predictions are constant by note, and continuous in frequency. In post-processing, the algorithm applies the shifts to each note, as shown in Figure 4.3. It does not require access to the musical score, which makes it usable even when the singer improvises or harmonizes in a performance.

The algorithm is designed to utilize information similarly to the human ear, basing corrections on information found in the audio, such as the level of perceived musical harmony and context in time. It is a DNN trained on patterns in real-world singing examples from a dataset of 4,702 amateur karaoke performances selected for good intonation [2]. The model is trained on both incorrect intonation, for which it learns a correction, and intentional pitch variation, which it learns to preserve. The design of the algorithm is based on the empirically derived conceptualization of musical intonation described in Section 2.2.2. Given its flexibility and preservation of nuance, it is adaptable to different musical traditions, regardless of the scales and pitch-related practices used in these traditions. It is a step towards a model-free automatic pitch correction system, the use of which is justified in Chapter 3.

The proposed DNN architecture includes convolutional layers for feature extraction followed by Gated Recurrent Units (GRUs) for sequential processing. The algorithm shows promising performance on the real-world score-free singing pitch correction task. To the best of my knowledge, this is the first data-driven approach to correcting singing voice pitch based on the harmonic content of the backing track.

1.3 Scope and limitations

The algorithm—designed for amateur singing—is intended for situations where a singer wishes to apply simple post-processing—for example, on their smartphone—without using professional audio-editing software. It requires that the vocals and backing track be separate, and for the vocals to be monophonic and free of noise. The current algorithm is limited to post-processing: Adapting it for real-time processing is left to future work.

The assumptions on which the algorithm is based are strong and might not be accurate. First,

the backing track is assumed to have clearly identifiable pitches—a chord progression—that serves as a reference for the vocals. Second, only past and current musical content is needed for making a prediction. the DNN processes a performance as a sequence in time, and does not access data from the future. Third, each note is corrected by a constant amount, and this is assumed to produce an accurate sounding result. Fourth, the dataset used to train the algorithm is assumed to be in-tune enough for this prototype, despite consisting of amateur singing, which sounds different from professional singing.

1.4 Contributions

Contributions in this thesis include:

- A data-driven algorithm for predicting pitch corrections directly based on the time-frequency content of the vocals and backing tracks instead of based on a symbolic music score.
- Continuous representation of pitch, which enables the algorithm to preserve pitch nuances, and to incorporate concepts from psychoacoustics, physics of sound, and cultural practices regarding musical intonation.
- A technique for generating pairs of training examples where one is in tune and the other is out of tune. This involves de-tuning in-tune performances to synthesize out-of-tune singing. De-tuning is based on random sampling from a HMM [3].
- Adaptability to any musical culture for which training data is available.
- An adaptation of a DNN architecture for pitch detection—including convolutional and Gated Recurrent Unit [4] layers—to the task of automatic pitch correction. The convolutional layers are used to extract features, while the GRU layers process the music signal sequentially.
- The "Intonation" dataset of in-tune singing performances, including the time-frequency magnitude transformation of the backing tracks and other metadata.

- A commentary on transforming this algorithm into a system that is usable in practice, even given the fact that not all of its corrections will result in higher accuracy.
- A small-scale subjective listening test, where the deep pitch correction network is shown to provide convincing adjustments when the backing track includes clear reference pitches.

1.5 Outline

Chapter 2 provides a musical background for the algorithm design, placing its development in the context of the discussion of musical intonation that has occurred over millenia. It surveys two standard conceptualizations of musical intonation, relevant definitions, and describes musical cultures that use intonation in different ways. This chapter also describes Antares Auto-Tune, one of the music-industry standards for pitch correction, discussing its advantages and disadvantages, and what can be learned from it when developing a data-driven pitch-correction algorithm.

Chapter 3 discusses the technical background. It provides an overview of music representation in software and compares model-driven approaches to data-driven approaches, including how they affect control, interpretability, and expressivity of the programs. It concludes with reasons behind the choice of a deep neural network for the task of automatic pitch correction.

Chapter 4 provides a technical presentation of the proposed algorithm. It starts with an overview of related work in music information retrieval, deep learning, and audio signal processing. It then describes in detail the proposed algorithm. This includes note boundary detection techniques, pitch de-tuning techniques, model architecture choices, and the experimental configuration.

Chapter 5 describes how we collected the "Intonation" dataset and details about the dataset. It indicates how the clustering technique used to collect the data can be used to generate datasets for other tasks where the target is subjective—as it is in the case of musical intonation. It also describes shortcomings related to genre bias in the current dataset, and how this issue can be fixed in future work.

Chapter 6 describes results both on the synthesized test set and the real-world dataset. It in-

cludes a comparison of the results using pitch deviation histograms. It also includes a comparison of the pitch behavior in the various datasets used in this thesis work. The quantitative analysis is followed by a qualitative listening test that provides insights into the way the model works. The analysis indicates that the proposed approach is more reliable than a conceptually and computationally simple baseline. It also indicates that the proposed approach effectively utilizes pitch content in the backing track.

Chapter 7 concludes the thesis. It summarizes the proposed automatic pitch correction algorithm, describes its current limitations, and how these might be addressed in future work. It places this thesis work into the broader context of music information retrieval.

CHAPTER 2
MUSICAL INTONATION: BUILDING ON AUTO-TUNE AND MILLENIA OF MUSIC HISTORY

Theories of musical intonation serve as a background to the proposed automatic pitch correction algorithm. This chapter starts with definitions of intonation and related terms such as frequency, pitch, and interval. It then introduces two standard conceptualizations of musical intonation. The first is based on precise mathematical formulas involving ratios, while the second is based on empirical observations, involving complex interactions between physics of sound, psychoacoustics, and musical culture. The relative merits of these two conceptualizations have been debated for millenia. The selection of conceptualization has profound implications for how a musical culture develops, as can be seen by exploring European classical music, Indian Raga, Blues, and 21^{st} century pop music. This chapter then uses Antares Auto-Tune, one of the music industry standards for pitch correction—and the first of its kind—to exemplify the application of ratio-based intonation. It relates the underlying assumptions about singing in tune that informed its design to the way it is used by musicians, both amateur and professional. Auto-Tune has emerged as being suitable for some musical styles—even creating opportunities for new ways of making music—but not suitable for others. A review of how Auto-Tune is used—sometimes in surprising ways—and some of its less positive results provides insight into what an empirical conceptualization of intonation can contribute to the musicality of automatic pitch correction.

2.1 What is musical intonation?

Discussing the concept of musical intonation requires that we first define *frequency*, *period*, and *pitch*. Hass describes these concepts in the context of acoustics [5].

> Some sound waves are periodic, in that the change from equilibrium (average at-

> mospheric pressure) to maximum compression to maximum rarefaction back to equilibrium is repetitive. The 'round trip' back to the starting point just described is called a cycle. [5, Ch. 1, Sec. 4–5]

The common measurement unit for frequency is in cycles per second or simply cps. One hertz equals 1 cycle per second. Our focus here will be periodic sound waves: They are the ones that cause a listener to hear a pitch (like a singing voice) instead of a noise (like consonants in speech or ocean waves).

Parncutt *et al.* define pitch by comparing it to frequency:

> In general, [pitch] is not the same as [frequency]. We use the term *pitch* in the psycho-acoustic sense of the experience of how high or low a tone sounds. It is a purely subjective parameter—an experience of the listener that can depend on several different physical parameters." [6, p. 477]

The Cambridge Online Dictionaries (6/12/2020) define musical intonation as the degree to which the notes of a piece of music are played or sung exactly in tune, e.g., "The violinist had good intonation, and a wonderful pure tone." It defines *in tune* as: singing or playing notes that are at the right pitch (= level) or that agree with others being sung or played.

The Dictionaries' definition implies that musical intonation can be experienced only when two or more tones are played together, either simultaneously or in succession. The relationship between two tones is referred to as an *interval*.

Unlike the definition above, which focuses on the listeners' perspective, Parncutt *et al.* define intonation by musicians' actions: "the real-time adjustment of (fundamental) frequencies in music performance." [6, p. 477] They elaborate this definition as it relates to limits of auditory perception, inharmonicity of musical tones, pitch as a subjective value versus frequency as an objective metric, and complexity and subjectivity in music.

Precisely and comprehensively defining musical intonation would be challenging. First, the subjective component makes it difficult to directly measure intonation without feedback from listeners. Second, the relationship between frequency and pitch is complex, as described in section

2.2.2. Third, a performer dynamically adjusts intonation over time. In some musical contexts such as jazz improvisation, a musician can transform locally bad intonation into globally good intonation. Bassist Victor Wooten famously teaches this to students. He uses an exercise where a student who plays a note that sounds out of key slides it over a half step in either direction, which then sounds good. "You're never more than a half step away from a right note," he says. "It keeps [the students] from being afraid of being wrong. And if they are wrong, it's only the note, not everything else. The music doesn't have to stop." [7]

2.2 Measuring musical intonation

Despite being challenging to define precisely, the concept of musical intonation has been the subject of research and debate for millenia. Developing a way of measuring intervals, or the relationship between tones, is key to studying intonation. This section introduces two contrasting conceptualizations: ratio based and empirically derived. My own training is in the culture of Western music, so that is my primary focus. I will, however, also make a brief foray into intonation systems from other traditions to underline the far-reaching possibilities that emerge from a richer understanding of intonation and its implications for tools like autotuning.

2.2.1 Ratio-based measures of intonation

A natural approach to measuring an interval is to calculate the ratio between the two fundamental frequencies. Fundamental frequency is the lowest frequency of the waveform. Use of *fundamental* frequency is an important distinction in natural waveforms, which are usually periodic at many different values. Fundamental frequency often roughly corresponds to the pitch the listener perceives, as the next section describes in more detail.

Pythagorean and just intonation are closely related ratio-based systems that define intervals as in tune, or *pure*, when the length of their shared periodicity is minimized. An octave has a ratio of 2:1, and a fifth a ratio of 3:2, making these two intervals the most pure after the unison, which has a ratio of 1:1. Pythagorean intonation is attributed by legend to the mathematician Pythagoras

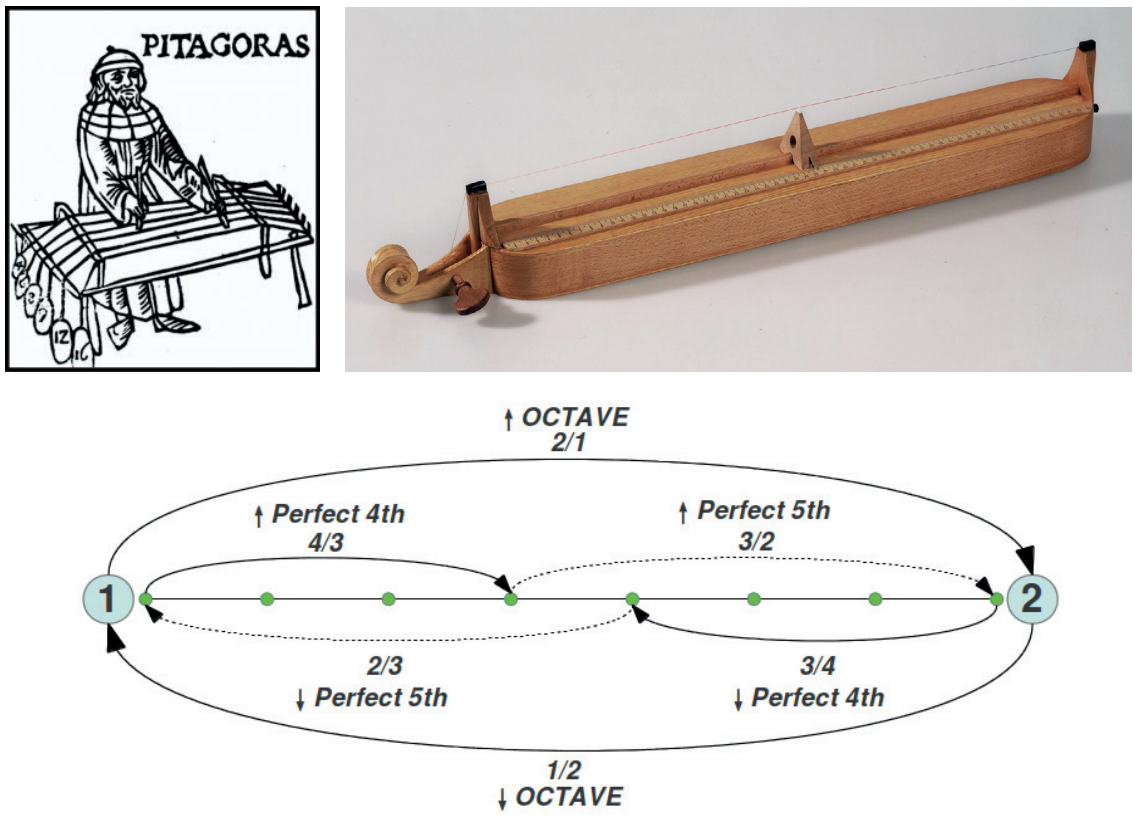

Figure 2.1: Illustrations of Pythagoras' technique using a monochord to generate musical intervals. Pressing down a string at specific distances produces substrings with lengths whose ratios to the full string length are 2:1 for an octave, 3:2 for a fifth, etc... The intervals can be heard by plucking the string. The mathematician is depicted in (a). The image was found in a book by Middle Ages music theorist Franchino Gaffurio (1492–1480?) [8]. (b) is a modern-day replica of the monochord instrument [9]. (c) shows the exact locations where the string should be pressed to produce an octave, a perfect fifth, and a perfect fourth.

(sixth century B.C.), along with the invention of the monochord, a one-stringed instrument used by Pythagoreans and acoustical scientists up through the Middle Ages [10, Ch. 2, p. 2 and Ch. 4, p. 42]. Figure 2.1 shows depictions of Pythagoras using a monochord, a modern-day replica of a monochord, and a chart showing how to produce pure musical intervals.

In Pythagorean intonation, every interval is constructed from octaves and fifths, and is formulated as $2 * *a3 * *b : 1$, where a and b are whole numbers. This leads to the concern of large ratios when using intervals beyond the fifth and the fourth. For example, a major third is $81/64 : 1$. Just intonation addresses these concerns by adding powers of 5. This enables a smaller ratio for the major third, namely, $4/5$ [6].

Just intonation was used in Europe during the Renaissance, as shown in writings by composer and music theorist Gioseffo Zarlino's *Istitutioni harmoniche* (1558). Zarlino was an advocate of using as many pure intervals as possible [11]. In fact, when music is complex enough that there are multiple intervals occurring at the same time—for example, in polyphonic music—it is often mathematically impossible to make all intervals pure. Keeping intervals pure becomes even more difficult on fixed-pitch instruments such as keyboards, where it is not possible to adjust a tone's frequency based on the interval that is being played. Zarlino defended Just intonation against another tuning system—equal temperament—that was advocated for by other theorists such as Franchino Gafurius (1451-1524) and Faber Stapulensis (1455-1537). Equal temperament involved replacing pure intervals with equally spaced ones, though the ratio between them is irrational, approaching $2^{\frac{n}{12}}$, where n is a whole number [11]. They represented a practical compromise that often sounded quite good. The equal-tempered scale became more established in the late 18th century [12].

In contemporary Western music, Pythagorean, Just, and Equal-tempered scales remain prominent. The Equal-tempered scale is commonly used for tuning fixed-pitch instruments, but Pythagorean and Just systems are commonly taught to musicians who play continuous-pitched instruments such as violin or who sing. New understanding of the physics of sound provides incentive to use pure intervals. Pitched waveforms are represented as sine tones at the fundamental frequencies. Intervals are represented as their sums. Small-integer ratios minimize *roughness* and *beats*: periodic oscillations in amplitude caused by constructive and destructive interference of the signals, which are considered undesirable.

2.2.2 Empirically derived conceptualization of intonation

An alternative to the ratio-based conceptualization of musical intonation, as old as its counterpart, is the empirical approach. Parncutt *et al.* refer to Aristoxenus' three books entitled "Elements of Harmony", where he

> argued on the basis of musical experience and intuition that the basic elements

> of musical structure—intervals, scales, tuning, melody—do not depend on arithmetic proportions, as the Pythagoreans claimed, but on what we today would call auditory psychology: processes of auditory perception, cognition, memory, and recall. To understand music, we have to perceive it. To understand the musical effect or function of an interval, we have to listen to it, not make abstract calculations. To understand how melodies work, we have to perceive, remember, and reproduce them. Musicians are not aware of ratios as they perform melodies. Interval sizes vary on a continuous scale and do not generally correspond to mathematically idealized ratios [6, p. 475].

Parncutt *et al.* argue in favor of the empirical approach. This section summarizes their key arguments.

First, studies measuring interval sizes in Western tonal musical performances on continous-pitch instruments such as voice or violin find much variety in musical interval sizes both above and below the ratio-based and equal-tempered intervals. For example, Devaney *et al.* find such deviations in polyphonic choral music performed by professional-level musicians [13]. More generally, studies tend to report normal, unimodal distributions around the twelve equal divisions of the octave. The octave that is divided into twelve is slightly larger than 2:1. The theoretical Just and Pythagorean variants are found to lie well within the distributions. The studies also show that performers tend to stretch larger intervals, compress smaller intervals, and play sharper if they are soloists. Researchers hypothesize that performers might exaggerate intonation patters for stylistic purposes.

Second, physical, fundamental frequency and perceived pitch differ most of the time. One reason for this is nonlinearity in the cochlea's spectrum analysis. However, perceived pitch also changes based on intensity, register, timbre, and masking effects when multiple sounds are played simultaneously. Perceived pitch can differ from frequency as much as two semitones, where one semitone is a twelth of an octave.

Third, rich timbres, inharmonicity—the fact that the overtone frequencies are often not exact integer ratios of the fundamental frequency—and randomization of phase via overlapping of the

direct sound and its reflections, make most beats and roughness inaudible in practice.

Fourth, there are physical limits to how precisely the human auditory system can detect pitch, for example, if a note is short. Physical limits also apply to how precise a performer can be.

Fifth, a performer's pitch can drift over time, either on purpose or through random variation. This means that the ratio between frequencies is changing over time, not locked into a specific value.

Parncutt *et al.* conclude that

> [m]usical intervals are not ratios, nor are they magical mathematical entities. They are learned, approximate, perceptual distances. They emerge from a multigenerational perceptual-historical process, mediated by the physical properties of musical tones and the physiological and psychological properties and limitations of the human auditory system.

2.2.3 Outcomes in music

The conceptualizations of musical intonation described above played an significant role in forming musical language and conventions. This section provides a few examples of how musicians' conceptualization of intonation led them to compose and perform music in a certain way. It is by no means comprehensive, but serves as background for considering how the design of music technology might affect what kind of music its users are able to make with it or will choose to make.

Ancient Greek philosophy as a basis for Western tonality

The Ancient Greek concept of pure intervals together with musical-event abstraction formed the basis of Western Classical tonality. Burkholder *et al.* describe how "[f]or Pythagoras and his followers, numbers were the key to the universe, and music was inseparable from numbers." Pure intervals were linked to *harmonia*, the unification of parts in an orderly whole.

> Through this flexible concept [...] Greek writers perceived music as a reflection of the order of the universe. [14, Ch. 1, p. 13]

Burkholder *et al.* describe how the history of music in medieval Europe is built on these concepts. Music in the church in medieval Europe was influenced by Greek philosophy in music theory. The church adopted the concept of perfect intervals relating to God [14, Ch. 2, p. 22]. In Gregorian chant, a *parallel organum* consists of a principal voice chanting the main melody, and of an *organal voice* that moves in exact parallel motion a fifth below. The Fifth was considered consonant, perfect, and beautiful, and superior to other intervals. The very simple structure of parallel fifths, however, led singers to innovate by experimenting with other intervals and breaking from perfect parallel motion. This move away from exact parallel motion opened the door to polyphony, which grew more complex over time and formed the basis of Western tonal harmony [14, Ch. 5, p. 86].

Burkholder *et al.* also describe how Aristoxenus, in his *Harmonic Elements*,

> distinguishes between *continuous* movement of the voice, gliding up and down in speech, and *diastematic* [...] movement, in which the voice moved between sustained pitches separated by a discrete intervals. A melody consists of a series of notes, each on a single pitch; an interval is formed between two notes of different pitch; and a scale is a series of three or more different pitches in ascending or descending order. Such seemingly simple definitions established a firm basis for Greek music and all later music theory. [14, Ch. 1, p. 15]

Both the restriction of musical intervals according to Pythagoras' conceptualization of intervals as ratios and the abstraction of pitch variations into notes according to Aristoxenus were simple conceptualizations of complex musical events. This simplification led to the ability to build complex harmonic structures and polyphony such as one finds in Beethoven's symphonies from a discrete set of musical elements. The equal-tempered scale was a further simplification of the elements, sacrificing small pitch nuances to enable increasingly complex harmonic modulations.

Classical Indian theory

Classical Western Tonality can be contrasted to musical cultures that prioritize nuances in pitch and rhythm over harmonic modulations, choosing to use a more complex set of musical elements. Classical Indian theory provides one such example.

Ramanna writes about *Ragas*, or scales:

> In principle, an octave can be divided into any number of parts, but musical value of a raga restricts these to a maximum of only seven out of altogether 22 intervals. [...] The frequency ratios of the *Swaras* (notes) play the most important part in the creation of a raga or a scale. [15, p. 897]

Classical Indian music theory varies by region. Ramanna writes about the Venkatamakhin scheme, where

> the [...] octave is divided into 7 Swaras of different intervals in 72 different ways. The 72 combinations [form] the basic Ragas from which other Ragas can be recognized. Though some appear identical, they differ in the way they are played.[15, p. 898] [...] The change of tonic in a piece of Karnatic music is strictly forbidden. This has, as, perhaps, come about due to the fact that any change of tonic requires a retuning of all the notes in a non-even temperament system. [15, p. 899]

Arnold, writing from a Western perspective, proposes that

> La théorie musicale ancienne de l'Inde, exposée integralement dans les premiers traités sanscrits sur la musique, et qui s'intéresse à la notion de consonance, aux échelles musicales et à leurs arrangements systématiques préservant les différences enharmoniques entre les positions tonales des notes, constitue la formulation la plus incisive du système d'intonation juste qui ait jamais été proposée. [Ancient Indian music theory, integrally formulated in the first Sanscrit treatises on music, that focuses on concepts of consonance, musical scales and their systematic arrangements preserving

enharmonic differences between the notes' tonal positions, constitutes the most incisive formulation of the Just intonation that has ever been proposed.] [my translation from French 16, p. 11].

Blues

Blues is a musical genre developed by African Americans in the early 20th century. Blues became by the 1960s one of the most important influences on popular music in the United States [17]. Palmer writes that "[t]he African music from which the blues ultimately derived came to what is now the Southern United States with the first African slaves. These Africans had belonged to a number of different tribal and linguistic groups, each of which had its own musical traditions." [18, p. 25] Some traditions had for many centuries had contact with Berber and Arab cultures in the North of the Saharan desert. The vocal music in these areas reflected the Middle East's tendency for long, tortuous melodic lines and formal solo singing. In other Sub-Saharan cultures, music tended to consist of group singing in call and response form with multiple overlapping melodies and percussion orchestras playing dazzlingly complex polyrhythms. Harmony was also used: "Not the periodic resolving Harmony of European music but the parallel Melodies song a third, fourth, or fifth away from each other." [18, p. 27] Alper explains that, as slaves were not allowed to sing music from their homelands, they incorporated their own performance practices into the musical forms that they were permitted to perform. [19]. Ultimately, what developed was a set of template chord progressions that were repeated during a piece. Twelve-bar blues is a widely used template [19]. These chord progressions use periodic resolving harmony of European music. However, the chord progression is not the main focus of the music, but rather provides a predictable structure on which instrumentalists and singers can improvise complex melodies or rhythms, or add personal nuances such as vocal intonation. These aspects of Blues relate closely to the African style. One nuance Blues is notorious for is intentional "off-key" singing [20], which is an example of a musical practice that does not fit into the Just or Pythagorean definition of intonation, but rather in the empirical, cultural conceptualization described by Aristoxenus.

21st century Pop music

Peres analyzes pop songs released between 2011 and early 2016 using the concept of *sonic syntax*. He defines sonic syntax as "a musical grammar that relies on manipulation of timbre, sonic density (the presence and amplitude of frequencies across the sonic spectrum at any given moment), and rhythmic intensity." [21, p. 2] Sections and subsections of a song function as sonic setup, buildup, or peak. Electronic dance music was the first popular genre to use sonic syntax as the driving structural feature. Some other recent pop music has adopted a similar syntax, with the consequence of tonality becoming a secondary component in musical structure. Peres describes how this musical innovation in pop was based on developments in audio techonology. "Modern technology allows record producers nearly unlimited control over timbre, which they have used to create new forms of musical expression." [21, p. 36] The ability to control timbre and small nuances leaves room for the development of musical subcultures and expressive nuances. Section 2.3 describes how Auto-Tune is commonly used in contemporary music as an effect, and can be considered a subculture. Given the close relationship between intonation and timbre, the question arises of how a differently designed system might be used by musicians.

2.2.4 Conceptualization of intonation in this thesis

While ratio-based conceptualization of musical intervals has permitted the development and understanding of sophisticated musical structures, it has neglected subtle but important nuances that emerge as a key source of musicality in many genres. My objective is to explore how an empirical approach to intonation as described in Section 2.2.2 makes space for reincorporating these nuances.

The scope of this thesis is limited to intonation of singing voice in relation to a backing track, or recorded accompaniment. It focuses on intonation as the adjustment of (fundamental) frequencies in music performance so that the pitches in the voice "agree" with the content in the backing track. It defines this agreement not via the proximity of the intervals to ratios of fundamental frequencies belonging to a discrete, pre-defined set, but based on proximity to patterns found in the given

musical tradition. This approach to working with musical intonation is built on the assumption that musical traditions have developed based on physical properties of sound and psychoacoustics, providing either purity or roughness in intervals, which provide a stimulating and expressive sound to the listeners.

2.3 Antares Auto-Tune

This section summarizes the functionality of Auto-Tune so that we can understand how it models musical intonation. It starts with a brief history of Auto-Tune for context.

2.3.1 A brief history

For two decades, Antares Auto-Tune has been the world standard for professional pitch correction. The way that other available tools (e.g., Melodyne) model pitch closely resembles that of Auto-Tune.

Andy Hildebrand founded Jupiter Systems, which later became Antares Audio Technology, in 1997. He got his PhD in electrical engineering at the University of Illinois, with a focus on signal processing. His full-time job involved signal processing on seismic data for oil exploration. In an interview in 2016 with Eckard [22], he wrote:

> Around 1995 I was at a trade show, it was me and a couple partners, and we were with a person who was distributing our products. His wife was there, and we were talking about what products would be interesting to do next. His wife said, "Well, Andy, why don't you make me a box that would have me sing in tune?" I looked around at the table, and everyone just stared at their lunch plates, they didn't say a word.
>
> So I thought, "boy, that's a lousy idea." About eight or nine months into the year, I'd gone to work for a different project, and I came back to that idea, I said, "you know, that's pretty straightforward to do, I'll do that." At the same trade show a year later I had producers ripping it out of my hands.

In the interview, Hildebrand described Western music as having a long history of innovations, and placed Auto-Tune into that timeline. He expressed surprise at the fact that artists often did not use the Auto-Tune software as intended, to discretely fix a singer's pitch, but instead used extreme settings that produce a robotic effect.

2.3.2 How does Auto-Tune work?

The Auto-Tune Pro Manual [1] describes the program's functionality in detail. This section focuses on the features that are relevant to the thesis. Auto-Tune features two modes of operation—*Auto Mode*, which is optimized for automatic (optionally, real-time) adjustments, and *Graph Mode*, which provides a user interface for precise, manual editing of the pitch and timing. Auto Mode is most relevant as it is designed to be used in a similar context to the proposed program. Graph Mode enables the user to be as musically refined and nuanced as they wish to be, but is not automatic and requires more extensive use of an interface.

Auto-Tune in Auto Mode takes as input a well isolated, monophonic sound source. It continuously adjusts the input pitch towards a target pitch. The target pitch is the closest scale tone as determined by the current scale settings. The default scale is the chromatic, equal-tempered scale, but the user can customize the set of notes that is used by specifying the key and the scale. These can also be automatically detected using MIDI. Furthermore, the user can customize the (fixed) frequency of every note.

One of the most important parameters in Auto-Tune is the *Retune Speed*, which controls how rapidly the pitch correction is applied to the incoming audio. The units are milliseconds. If set to zero, the pitch is immediately shifted to the target pitch, completely suppressing any vibrato or deviations in pitch. A setting between 10 and 50 milliseconds is commonly used for producing a more natural-sounding effect. There is always a tradeoff between remaining close to the scale and preserving pitch variation. Some additional functionalities help address unwanted artifacts. One is the *Humanize* function, which adjusts Retune Speed based on the length of the note. The Retune Speed is reduced on longer notes to prevent a static pitch, but kept fast for the short notes so that

the melodic contour is accurate. Another is *Flex-Tune*. This control helps note transitions be less abrupt by only applying pitch correction when the performer approaches the target note. Controls also exist to amplify existing vibrato or to add a synthesized one. Finally, controls exist to preserve the singer's formant—their spectral shaping that results from the acoustic resonance of their vocal tract—or even change it to sound like they have a longer or shorter throat. Voices are put into five categories: Soprano, Alto/Tenor, Bass, Instrument, Bass Instrument. This categorization also helps preserve a natural formant.

2.4 Auto-Tune's ratio-based conceptualization of musical interval

Auto-Tune's model of pitch originates directly from the ratio conceptualization of intervals introduced in Section 2.2.1. We can conjecture a set of assumptions that underlie the way the program applies corrections. The first assumption is that perceived pitch and fundamental frequency can be treated as equivalent. This assumption is evident from the fact that Auto-Tune is described as a pitch-correction program instead of a fundamental-frequency-correction program [1], meaning that the corrections apply at the perceptual level. A second assumption is that a note's proximity to a small set of frequencies—by default, the twelve notes in the equal-tempered scale—is considered accurate and, by extension, in tune. A third assumption is that masking and interference from the backing track does not provide essential information for the pitch corrections: Auto-Tune only processes the vocal track without referring to the backing track. A fourth assumption is that the center pitch in every note remains constant throughout, except during note transitions at the beginning and end. Pitch bending is not explicitly modeled.

2.4.1 Modeling tradeoffs and negative reception

While the simple model behind Auto-Tune makes it possible to design a program that consistently produces reliable results, it is worth pondering what nuances are lost in the process.[1] First, while

[1] Parncutt *et al.* recommended that "Computer scientists [...] resist the temptation to develop and implement models of musical structure based on frequency ratios." I personally spent much of my early PhD thinking about how such a model could be developed, but found the task difficult. I synthesized four-part harmony with intervals based on just or

the concept of intervals as ratios is useful, shifting frequencies so that the ratios of their fundamental frequencies exactly correspond to the small number of acceptable values removes both desired and erroneous pitch variations without discrimination. One can argue that, in the process, part of the social phenomenon that is music is lost: the part of music that consists of a shared auditory culture, where artists and listeners learn to appreciate specific pitch patterns in specific contexts, and become part of a musical community. Second, results of psychoacoustics are ignored in favor of theoretical models of musical intonation that do not correspond to what proficient musicians have been shown to do in practice. Third, the richness of vocal timbre and of interaction between the vocals and backing track is ignored.

Fourth, many musical practices and even cultures are left out. Early reception of the Auto-Tune program showed concern for this reason. As described in *Encyclopaedia Britannica*,

> Critics noted that Auto-Tune had disrupted a long and important history of intentional "off-key" singing, a technique that was particularly prevalent in the blues, a style that became the backbone of much of popular music in the present day. Blues singers traditionally played with pitch to express feelings such as longing or yearning, to punch up a nastier lyric, or to make the lyrics sound more suggestive. For example, Mick Jagger's vocals on *Sweet Virginia,* from the seminal Rolling Stones album *Exile on Main Street*, were sung almost totally flat for the purpose of achieving a certain effect." [20]

There is also no modeling of Aristoxenus' diastematic pitch movement, of tortuous melodies as found in Arab-influenced music, for example, or of "off-key" singing.

A visual analogy

I personally find that it is easy to get lost while discussing aesthetics in the context of music, so I provide a crude analogy. Human height approximately follows a multimodal normal distribution. Suppose somebody determines that a good height for a human corresponds to one of these modes,

Pythagorean intonation and only produced dissonant results.

and develops technology to make all people have one of these heights. The result would be that many people would appear to be of usual height when encountered individually. Encountering a group of people of identical height, however, would be startlingly unnatural.

Negative contemporary opinions

Among amateur singers, Auto-Tune used in the fully automatic mode on apps such as Smule produces a robotic effect not always popular among users. One Smule user writes about a plug-in that includes Auto-Tune: "Please, whatever you do, don't use Popstar/Super Pop on a song that is not meant for [Auto-Tune]. Nothing kills the mood faster [...]." [23] This comment indicates that the effect is so strong that users omit using it unless a song sounds good with the obviously audible effect.

Though this thesis focuses on automatic pitch correction for amateur musicians, it is interesting to consider how professional artists adopted Auto-Tune. As described in more detail in Section 2.4.2, Auto-Tune has become a popular effect among professional singers. One can contrast Miley Cyrus's Auto-Tuned voice in *Party in the USA* at 0:09 and 0:43 to her natural voice *The Backyard Sessions - "Jolene"*[2]. It is not uncommon to hear criticism regarding "excessive use of Auto-Tune", e.g., [24], making singing voice sound overly processed.

2.4.2 Adoption by professional artists and positive reception

The initial goal of Auto-Tune was to automatically help people sing more in tune. *Sunday Morning* by Maroon 5 or *Toxic* by Britney Spears both used Auto-Tune in the originally expected manner. A listener might not notice the effect unless they are aware of it and put some thought into it.

Others turned the settings to extreme values to develop new musical effects. An early example is *Buy U a Drank (Shawty Snappin')* by T-Pain (feat. Yung Joc), starting at 0:02. Recent examples in R&B include *Die For You* by The Weeknd, contrasting heavy Auto-Tune at 1:24 with a rich voice at 0:28, and *Work* by Rihanna and Drake, at 0:09 and 1:00, where the latter example makes

[2] `https://youtu.be/wOwblaKmyVw`, accessed 06/20/2020

her voice sound like a synthesized instrument. In rap, one example is *What The Price* by Migos. It uses different settings in overlapping voices, at 0:17, 0:26, and 0:44. In electropop, *Anything Could Happen* by Ellie Goulding uses different settings at 0:00, 0:08, and 2:28. To hear a song with Auto-Tune versus without, Ke$ha's *Die Young* can be heard as the original production or *Deconstructed*, with her natural voice. In Electronic Dance Music, Daft Punk's *Harder, Better, Faster, Stronger* uses Auto-Tune at 1:37.

2.5 How millenia of music theory and the design of Auto-Tune inform this thesis

Auto-Tune is an early system for automatic pitch correction. Its invention and commercialization has led to widespread usage, development of new musical effects used in multiple genres, and positive and negative reactions. Auto-Tune created new musical opportunities, but also left out important musical practices and nuances from many traditions. The analysis of the assumptions behind the design of Auto-Tune and how they affected the way the program works and is used by musicians provides a useful background for this thesis. The algorithm proposed in this thesis is designed based on the empirical conceptualization of musical intonation. It strives to incorporate and even encourage use of subtle pitch nuances. Chapter 3 describes the process of designing a computer program for automatic pitch correction.

While this chapter has provided ample reason to consider an empirical approach to automatic pitch correction, or to develop a more complex model of intonation, one question arises. Does moving towards an empirical representation of musical intonation force us to discard an elegant mathematical and physical theory? Parncutt *et al.* argue that this is not the case.

> Physics is often thought of as an 'exact science' because it is dominated by mathematical theory. But mathematics is also an excellent tool for dealing with inexactness. Music theory is often considered mathematical and therefore exact. In fact, the quantities considered in applied mathematics vary along a spectrum from very exact to very approximate. The number ratios that correspond to musical intervals also lie somewhere along that spectrum."

CHAPTER 3

TOWARDS DATA-DRIVEN AUTOMATIC PITCH CORRECTION

The previous chapter provided a musical background for designing an automatic pitch correction system based on an empirical perspective on musical intonation. This chapter addresses the practical and technical aspects of implementing such a system. It focuses on the current state of music and audio representation in software, and on the state of data science with respect to complex problems such as music.

One choice that needs to be made is how to represent music and audio data in the program. Should the program process thousands of audio samples per second, or abstractions such as notes? This chapter provides an overview of recent advancements in computation power and machine learning that make possible rich music representations.

A second choice that needs to be made is whether to adopt a model driven or a data-driven approach. This chapter compares the two approaches, and then describes a continuum of approaches between model driven and data driven. A model-based approach begins from understanding, which requires abstraction and simplification of the task to make it sufficiently understandable to be modeled. A data-driven approach begins from data and searches for meaningful patterns, without a guarantee of complete understanding.

This chapter discusses successful examples of both types, showing how they contribute to greater understanding in audio and related fields. It also considers how a data-driven approach differs from one that is model driven in the automatic pitch correction context. It relates Pythagorean and Just ratio-based conceptualizations of musical intonation to a model-driven approach, and argues that Antares Auto-Tune is model-driven. It then lays out the difficulties inherent in formulating an empirically derived approach. This discussion contributes to the wider theme of the trade-offs controllability, expressivity, and interpretability in a given program. Trade-offs between the three are often required. Examples are provided that illustrate how model-driven approaches

tend to provide more control and interpretability, in contrast to data-driven approaches, which can be more expressive when applied to complex tasks. This chapter considers how important these features are in the case of automatic pitch correction and what type of program might be appropriate for the task. It argues for the importance of carefully considering all three features in designing an automatic pitch correction tool, and sets the stage for the development of such a tool.

3.1 Music representation in software

Technical writer Jaron Lanier wrote a decade ago about the invention of the MIDI:

> One day in the early 1980s, a music synthesizer designer named Dave Smith casually made up a way to represent musical notes. It was called MIDI. His approach conceived of music from a keyboard player's point of view. MIDI was made of digital patterns that represented keyboard events like "key-down" and "key-up".
>
> That meant it could not describe the curvy, transient expressions a singer or a saxophone player can produce. It could only describe the tile mosaic world of the keyboardist, not the watercolor world of the violin. But there was no reason for MIDI to be concerned with the whole of musical expression, since Dave only wanted to connect some synthesizers together so that you would have a larger palette of sounds while playing a single keyboard. In spite of its limitations, MIDI became the standard scheme to represent music in software. Music programs and synthesizers were designed to work with it, and it quickly proved impractical to change or dispose of all the software and hardware. MIDI became entrenched, and despite Herculean efforts to reform it on many occasions by a multi-decade-long parade of powerful international commercial, academic and professional organizations, it remains so. [25, p. 7]

A decade later, the state of music and audio technology has evolved. While MIDI remains a common tool, alternative audio technologies such as phase vocoding and deep learning have become available alternatives. Section 3.4 provides examples of programs that deploy them. These

technologies make it possible to process audio at the level of the spectrogram frame or sample, preserving as many nuances as desired. In later sections, I provide examples of such technologies. Discretization of music as a sequence of events in MIDI also relates discretization of pitch, which extends to discretization of musical intervals in the ratio conceptualization of musical intonation discussed in Section 2.2.1. All provide a simple and elegant framework for structuring music, but risk discarding nuances. The choice of representation is at the discretion of the the programmer or designer. As described in more detail in later sections and chapters, the automatic pitch correction system proposed in this thesis combines some MIDI-like discretization with a continuous-valued pitch shift representation, seeking to retain a reasonable level of controllability and interpretability while making more space for expressivity.

3.2 Model-driven approaches

A common debate in multiple research areas including audio and music is whether to use a model driven or a data-driven approach for making predictions. This section explores the model-driven approach and reviews its advantages and limitations.

When developing a model of musical intonation, one needs to explicitly define a set of variables that affect how in tune an interval sounds. One also needs to define the function that maps the variables' values to pitch correction predictions. One needs to understand the process well enough to organize the variables and functions, or simplify the process to a level of abstraction where one is able to organize it. The assumptions and the variables one chooses to include in the model can stem from one's place in time and space, and values about the world [26]. In fact, "multiple models for the same target system do not generally stand in a deductive relationship, as they often contradict each other." [27]

One might ask, is a model worth using even though it does not capture the full complexity of the real world? The Stanford Encyclopedia of Philosophy entry on models in science describes how

> models are vehicles for learning about the world. Significant parts of scientific inves-

> tigation are carried out on models rather than on reality itself because by studying a model we can discover features of, and ascertain facts about, the system the model stands for. [...] Learning about a model happens in two places: in the construction of the model and in its manipulation (Morgan 1999). There are no fixed rules or recipes for model building and so the very activity of figuring out what fits together, and how, affords an opportunity to learn about the model. Once the model is built, we do not learn about its properties by looking at it; we have to use and manipulate the model in order to elicit its secrets. [27]

Furthermore, the entry describes how models typically provide a distorted or idealized representation of their targets, or at least one that is only approximately true. Elgin argues that learning occurs not despite, but because, of models being false [28].

The Encyclopedia entry provides reasons for why models help us organize our knowledge of the world and confront it with reality. Given this capability, models are useful for building more sophisticated understanding over time, and can even lead to theory development [27].

Pythagorean and Just conceptualizations of musical intervals as ratios can be formulated as a model: Good intonation is produced between two notes—when they are played consecutively or simultaneously and both are audible—by shifting their fundamental frequencies so that their ratio is one of a set of specified values: 1:1, 2:1, 3:2, etc..., because this minimizes shared period length, and therefore roughness and beats. This elegant model functions at a high level of abstraction. For example, it applies to all musical contexts and all timbres. Furthermore, it can be confronted with reality via empirical studies of musical intonation, leading to scientific knowledge. Finally, this model has provided the basis for much learning—such as the development of sophisticated Classical Western tonality—despite the fact that it does not correspond to results of empirical studies.

3.2.1 Auto-Tune as a model

Section 2.4 described Auto-Tune in detail through the lens of a ratio-based conceptualization of musical intervals, making it a model-driven program. In practice, implementing a ratio model is often mathematically impossible, as described in section 2.2.1. To fix this issue, Auto-Tune moved from Pythagorean and Just intonation to the equal-tempered scale, a compromise that makes the model usable in practice, while sacrificing some of the purity of the intervals.

3.2.2 The proposed system as a model

When building a model based on psychoacoustics and on results from empirical studies of intonation, the number of variables increases dramatically. This section lists some possible features that might be relevant to determining whether a note is likely to sound in tune as judged by a listener. The list is not comprehensive, but provides a general idea.

The first set of features relates to the music. It includes the genre of the piece being performed (including stylistic choices common for the genre, such as pitch bending), backing track instrumentation (number of instruments, timbre, relative amplitude), global and local tempo, duration of the given note, harmonic, and melodic context.

The second set of features relates to the singer. It includes vocal type (soprano, alto, tenor, bass), vocal characteristics (timbre, breathiness), fundamental frequency characteristics (vibrato rate, vibrato depth, and jitter [29]), vocal training (years of training, genre the singer was trained in, e.g., classical, jazz, rock), general musical training (not related to singing), vocal ability (range, stability, expressiveness), personal aesthetics, amount of vibrato, attack-decay-sustain-release envelope of the given note, musical choices, and successful and undesired outcomes during the note and earlier in the performance.

The third set of features relates to the listener. It includes musical background (musical genre familiarity and affinity, culture, musical training, hearing ability, subjective preference), audio quality (live, analog, digital, compression, sample rate, bit depth, speaker quality, number of channels), room acoustics (reverberance, distance from the source to the listener), perceived intonation

accuracy of previous notes, relationship of listener to performer (self, teacher, parent, fan, audience member who bought an expensive ticket).

More generally, the theory should include available knowledge in areas such as the physics of sound, psychoacoustics, cochlea structure, and masking.

Many of these features are hard to quantify or measure objectively. Their interactions are complex. This increases the challenge of developing a model that encompasses the psycho-cultural aspect of musical intonation. These challenges motivate our consideration of a data-driven approach.

3.3 Data-driven approaches

Taking a data-driven approach makes it possible to incorporate a large number of variables without fully understanding how they interact, and therefore to build a program that is better aligned with the real world than what we are explicitly able to explain. Section 3.4 discusses this trade-off between understanding and expressivity. Moving from a model driven to a data-driven approach has often ultimately led to improvement in modeling, and vice versa. An example of the former is the model-driven Scale-Invariant Feature Transform (SIFT) from computer vision, which inspired an encoder-decoder based feature extractor with a similar structure to SIFT, but learned abstract parameters from the data, which produced higher accuracy [30]. An example of the latter, from natural language processing, is the invention of a linguistically motivated, count-based approach to word vector embedding [31], which was inspired by `word2vec`, a deep learning model that had outperformed previous model-driven approaches [32]. These examples illustrate how a data-driven approach can help move beyond arbitrary decisions such as SIFT feature extraction parameters, and can also demonstrate through its high accuracy how well a model has the potential to perform if formulated properly, inspiring the design of better models.

This toggling between model driven and data-driven approaches relates to how we interact with music, both in its acoustic and digital forms. Music is not considered something that we as humans understand deeply or can model, despite the mature field of music theory and the existence

of multiple sub-models and sub-theories that address various aspects of how music works. This has not stopped humanity from developing musical instruments and making music. Development of models of harmony has lead to innovation by composers, and when composers have broken "the rules"—for example, when European Medieval singers added less pure intervals to their musical vocabulary of octaves and fifths—this led to new models. Continuing this tradition in music technology might lead to new means of musical expression, new developments in music theory, or both of the above.

3.4 Developing music technology: Control, interpretability, and expressivity

When developing music technology, one needs to make choices regarding how much control the programmer has over the output, how interpretable the program's actions are, and how complex functions the program can express. These features tend to come at the expense of one another. This section describes the three features and considers which ones should be prioritized for the proposed automatic pitch correction system.

3.4.1 Control

Programmers usually provide a computer with exact, step-by-step instructions. The computer then executes these exactly, with no ability to deal with ambiguity. When designing music technology, a natural, naive approach is to provide the computer precise instructions for what to output under a comprehensive list of circumstances. This approach provides exact control over the output. However, formulating this set of instructions is developing a fully deterministic model. As described in section 2.2.1, this tends to result in simplifications and loss of richness. One can note that Auto-Tune is one example of a program that provides full control over the output.

Developments in machine learning and deep learning have provided a framework for inserting randomness into music technology—both into model and data-driven approaches. Specifically, they utilize computers' capacity to generate random numbers and to follow statistical distributions. They also provide a framework for computers to detect or learn patterns in data. Incorporating

randomness into music technology removes some control. Music Plus One [33], a musical accompaniment system, provides an example. The accompaniment track dynamically adjusts tempo based on a soloist's actions. The adjustments are based on real-time score matching using a hidden Markov model and future note timing prediction based on a Kalman filter-like model. Though the structure of these models is precisely determined by the programmer, and therefore controlled, the actual values adopted during a performance are out of the programmer's control. The result is much more dynamic than an alternative, where the programmer would have enforced specific tempi under specific circumstances.

Other machine learning programs provide less control than Music Plus One. Non-negative Matrix Factorization (NMF) [34]—used, for instance, for processing magnitude spectrograms—is one such example. A NMF program represents a matrix as a set of basis vectors and activations. It iteratively minimizes the error between the input and reconstruction. The outcome is partially under the programmer's control, as the program enforces desired characteristics such as non-negativity. However, the model can converge to multiple different results, and this aspect cannot be predetermined. A DNN is by default even less controllable. The programmer can design the input and target data to the program and train the model to output data that is similar to the target, but has little control over the weights that the model learns. The programmer also has little control over what the program will output given previously unseen input data. Using a DNN, however, does not necessarily result in full loss of control. Examples in later sections illustrate how a programmer can use knowledge about a task to structure and regularize a DNN in a way that restores some amount of control.

3.4.2 Interpretability

A program is considered to be interpretable when weights that it learns can be explained, that is, connected to understandable features such as fundamental frequency or spectrogram amplitude. A linear model is one example of such a program. A program whose weights cannot be explained but rather take on an abstract meaning is often referred to as a "black box" program.

Interpretability is often desirable when developing machine or deep learning models, as described by Molnar [35]. Interpretability makes it easier to understand why the program fails under certain conditions. This leads to the ability to debug and to build safety measures. Interpretability also can be used to ensure that only relevant features are used for predictions, building reliability and addressing problems such as bias against minority populations. Furthermore, it can build trust among users, especially when stakes are high, as observed in the medical context. Finally, it makes it possible to ask *Why* instead of *What*, leading towards development of new theories [35].

Interpretability declines in importance when stakes are low, or when the problem has been sufficiently studied to provide confidence that provided insights are useful even though the weights remain abstract. This holds, for example, in the case of optical character recognition, which has gained acceptance in spite of its lack of interpretability through extensive practical experience and correction of shortcomings over time.

One example of such a problem is optical character recognition, where there is enough practical experience with the model and shortcomings have been addressed over time. Furthermore, even in the case of models that are not directly interpretable, other techniques are being developed to methodically detect and remove bias, as in [36].

3.4.3 Expressivity

A third concept useful for characterizing a program for music technology is its expressivity. The model's structural properties determine which functions it is able to compute. Higher expressivity can come at the expense of some control and interpretability. Wager [37] provides an example in the context of causal inference, describing how machine learning can be used to avoid extraneous modeling assumptions. Approaches using easily interpretable models (e.g., linear regression) often make strong but powerful assumptions, and these assumptions may not be scientifically or methodologically motivated. A partially linear model is more expressive and enables addressing more complex situations such as treatment heterogeneity, which would have been outside the family of functions expressible by the linear model. Its parameters can be estimated using generic

machine-learning tools. [37]

In the case of DNNs, complexity of the computed function has been shown to grow exponentially with depth [38]. Even a single-layer network can theoretically express any function as long as it contains enough weights. DNNs represent the quintessential black-box model: They make possible expressiveness, but at the cost of being notoriously difficult to control or interpret. *Conv-TasNet* is one example of such a network [39]. It is designed for source separation and takes as input the time-domain audio signal without any pre-determined feature extraction such as a Short-Time Fourier Transform (STFT). A time-domain signal is harder to interpret than a time-frequency signal, where separation of the signal into its frequency components corresponds to the way the human auditory system processes sound. Trained people might even decipher music or speech from a spectrogram [40, Ch. 3, p. 41]. In the time domain, many audio signals that look quite different can sound the same to the listener, because the only difference is the phase offset [41]. *Conv-TasNet* replaces the STFT with multiple convolutional filtering layers that are trained to produce an optimal feature extraction for the task of source separation. The large number of filtering layers are hard to interpret, especially because of non-linear activation functions applied to the outputs, such as parametric rectified linear units—a ramp function where the negative side is modified to have small nonzero weights [42]. After these layers of filtering, the signal is processed in an unknown embedding space, which is again useful for the task but hard to interpret.

Though more expressive models are not always fully interpretable, increasing interpretability is an active research area. The boundary between understandable and black box models is not fixed. Examples of techniques that increase understanding involve examination of what deep convolutional neural networks learn by visualizing inputs that maximize the activation of the filters [43]; restructuring of convolutional neural networks as probabilistic models [44]; and research on interpretable parameters in reinforcement learning [45].

3.4.4 Setting priorities in the proposed system

How important is it for the proposed system to be controllable, interpretable, and expressive?

Section 3.3 briefly argued that a high level of control over exactly what should happen musically under a comprehensive list of circumstances is not necessary for music making. That might actually be misaligned with how even top musicians perform. Though a top musician has technical mastery of their musical instrument, much randomness can occur during a performance, and the musician is reacting spontaneously to other musicians' actions and other factors such as the mood they are in. Such reactions are not fully controlled by the musician. Furthermore, the musician may not be able to comprehensively interpret which factors led to specific successful or unsuccessful artistic decisions in a performance.

Thus, the objective for the automatic pitch correction algorithm proposed here is to increase expressivity relative to existing approaches while maintaining reasonable levels of controllability and interpretability. The proposed system strives to predict pitch shifts in the context of physics of sound, psychoacoustics, and musical traditions. Section 3.2.2 described the large number of moving parts in the empirical approach. Given empirical studies of intervals in musical performances, the proposed approach should predict pitch corrections on a continuous scale instead of mapping frequency to a discrete set of values. It should apply to musical genres and cultures that do not use the equal-tempered scale. Furthermore, given that it builds on psychoacoustics, it should utilize the full vocals and backing track timbres to make predictions instead of the fundamental frequency extracted from the vocals. These goals result in an automatic pitch correction system that takes more complex data as input and outputs more complex predictions. It likely requires a very expressive system.

The proposed system can still use a time-frequency transformation as a pre-processing step instead of directly processing the time-domain audio signal. It also uses MIDI-like abstractions of notes, processing notes instead of audio samples. These design decisions prevent undesirable artifacts from occurring within a note. They increase control, and decrease expressiveness.

3.4.5 What happens when the pitch correction fails?

When working with a model that is not fully controllable or interpretable, it is important to consider what types of erroneous output or bias the program might produce, consider how to prevent it, and determine how much to prioritize minimizing these wrong outputs compared to expressivity. An example where avoiding wrong outputs is the top priority is the design of self-driving cars, where a physical risk is involved, or a credit-score predictor, where bias can impact an individual's economic status.

The effect of an automatic-pitch-correction error depends on the context in which the program is used. The context for the proposed program is as a post-processing tool for an amateur, karaoke-style performance. If the user applies the plug-in, and a note gets worse, one can imagine two negative outcomes. One is that the user notices the mistake and decides not to use the plug-in, even if it might make other notes sound more in tune. Another is that a user with low confidence in their ability to hear pitch assumes the program is correct, and as a result feels less confident or accepts a bad result. However, a solution to both negative outcomes would be to design the program to be interactive, and to make clear to the user that, as a machine-learning-based algorithm, it sometimes makes mistakes. The interaction would, for example, let the user listen to "before" and "after" segments and decide note by note whether or not to accept the result. Optionally, the user could even adjust the prediction. Framed in this way, the program might actually serve as an ear-training tool that lets singers refine their ability to hear subtle pitch shifts and how these affect intonation. With excellent design, detecting mistakes might actually make the program more fun to interact with and musically stimulating than a program that enforces its model of in-tune singing on the user's performance and "considers" itself to always be correct. Ideally, the program could even learn from user feedback to avoid repeating mistakes and become tailored to individual preferences and styles.

As described in later chapters, bias can be an issue for this program, especially in the context of less-common musical genres. Training a model on pop will likely make it unusable for Classical Indian music or blues because the singing styles are very different. Just as singers develop a style

within a specific musical style, and build on a specific musical tradition, the model should be trained on the appropriate dataset. It is also important to include a variety of vocal timbres and even accents because vowel formant affects timbre.

What becomes clear is that, in deploying the proposed automatic pitch correction algorithm, it is important to involve professional designers and/or music theorists or musicologists to ensure a positive user experience and to make sure that musical genres and singing styles are properly represented.

3.4.6 The usefulness of deep learning for automatic pitch correction

In the field of audio processing, deep learning has proven to be a technology that lets us represent audio in a richer manner than abstract notes—as in MIDI—processing audio at the level of the sample or spectrogram bin. The software and hardware developed for deep-learning tasks are powerful enough to process audio data in its full complexity. Deep-learning methods have been shown to produce impressive and state-of-the-art results on ill-defined audio tasks such as generating natural-sounding speech for home assistants [46], pitch detection [47] or dereverberation [48]. The first example, *Wavenet* is a deep generative model of raw audio waveforms of speech. It was shown to reduce the gap with human performance by over 50% compared to the state of the art in Text-to-Speech synthesis algorithms. The model is fully probabilistic and autoregressive. Oord *et al.* write that it is able to produce natural-sounding speech based on its training on tens of thousands of samples of audio per second. The DNN structure in this case provides the ability to work efficiently at the level of the audio sample on speech, which is a complex signal. The second example uses a deep convolutional network to extract features from a time-frequency representation of polyphonic audio. Extracting pitch information when there are multiple instruments is challenging, but the large number of filtering layers provides the model the ability to extract useful information from a complex signal. The third example is based on the *Wavenet* mentioned above. It takes as input reverberant audio, and outputs a dryer version of the signal. In this case, the DNN provides the ability to work with raw audio, not discarding much of the richness of the

signal despite removing room reflections.

The automatic pitch correction task has similar goals to the above examples. It aims to extract frequency and interval information from vocals and backing tracks, and to process complex audio and music data. The success of DNNs in audio provides a reason for considering using one for the given task.

CHAPTER 4

THE DATA-DRIVEN PITCH CORRECTION ALGORITHM

This chapter provides details of the algorithm [49]. It begins with related work, and then moves to the neural network structure. It also covers the data preparation steps—including pitch de-tuning and feature extraction—followed by the training configuration and the experimental setup.

4.1 Open-source repository

I released an implementation of the algorithm in Python using the Pytorch framework for deep learning [50]. The repository is available at `https://github.com/sannawag/autotuner`[1]. Users have the option of training the model on their own dataset or of downloading the parameters of the model that provided the best test results. The repository also includes code for a baseline automatic pitch correction algorithm introduced in 6.2, and an implementation of the Time-Domain Pitch-Synchronous Overlap and Add (TD-PSOLA) algorithm [51]. A standalone implementation of TD-PSOLA is available at `https://github.com/sannawag/TD-PSOLA`. The dataset used to train the model is available upon request via `https://ccrma.stanford.edu/damp`. Note that the CQT parameters used in the published dataset are different from those referred to in this chapter.

4.2 Related work

The first commercial pitch-correction technique, Antares Auto-Tune [52], is also one of the most commonly used. Section 2.3 describes how it is designed. Auto-Tune measures the fundamental frequency of the input monophonic singing recording, then re-synthesizes the pitch-corrected audio signal. In recent work on continuous score-coded pitch correction [53], as in Auto-Tune, each

[1]Note that the algorithm used to be named *Deep Autotuner*, but I have modified the name when possible to avoid confusion with the trademarked term *Auto-Tune* by Antares.

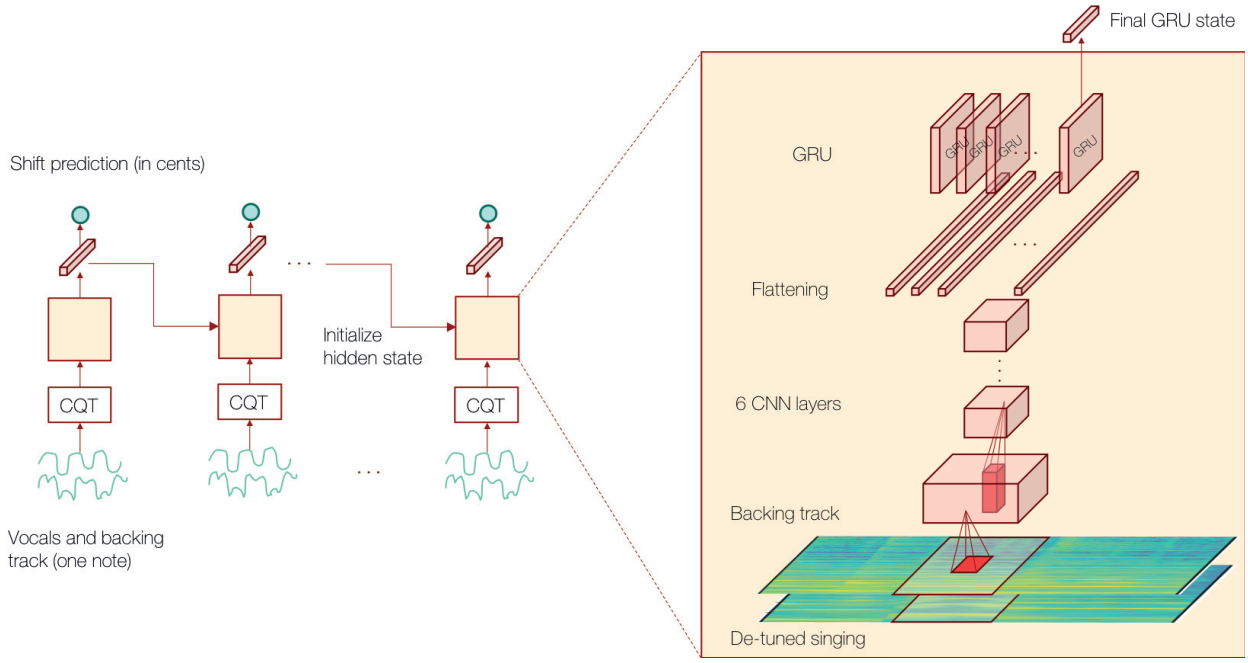

Figure 4.1: Program overview. The program processes one note at a time, and predicts a constant shift for the note's duration. The proposed DNN architecture includes convolutional layers for feature extraction followed by GRUs for sequential processing.

vocal note is pitch shifted to the nearest note in a user-input set of pitches (scale) or to the note in the score if it is known. The default musical scale is the equal-tempered scale, in which each pitch p belongs to the set of MIDI pitches $[0, 1, ..., 127]$ and its frequency in Hertz is defined as $440 * 2^{\frac{p-69}{12}}$. Some users prefer a finer resolution and include more than twelve pitches per octave, or use intervals of varying sizes between pitches. In every case, the fundamental frequency is discretized to a small set of values, around which every note is shifted to be exactly centered. Hence, the pitch shifts tend to ignore a singer's intentional expressive gestures and might not easily apply to musical traditions with different scales or more fluidly varying pitch. The proposed algorithm accommodates a variety of frequencies by letting the fundamental frequency take any value along a continuous scale, and by shifting every note by a constant without modifying internal pitch variation.

Other recent approaches to pitch correction include style transfer. Style-transfer-based work modifies amateur performances to mimic a professional-level performance of the same song. Luo *et al.* propose to match the pitch contour of the professional-level performance while preserving

the spectral envelope of the amateur performance [54]. Meanwhile, Yong and Nam propose to match both the pitch and amplitude envelopes [55]. The approach in this thesis is similar in the sense that it also uses features gathered from high-quality performances [2]. However, it does not necessitate a "target" performance of the same song during testing. Instead, it learns from many in-tune singing voice examples and their backing tracks, and then generalizes to unknown songs, while preserving the original singer's style.

4.2.1 Music information retrieval

While only a few algorithms exist for pitch correction, Music Information Retrieval (MIR) research on related tasks such as pitch detection provides a useful background for this thesis. Gomez *et al.* [56] provide an overview of recent developments in deep learning for singing processing, ranging from pitch detection to singing separation and synthesis. Pitch detection is particularly relevant to automatic pitch correction. Pitch detection algorithms, like the algorithm in this thesis, aim to extract harmonic patterns from the audio. In the case of pitch detection, the target pitches are often manually labeled.

Bittner *et al.* introduce a fully convolutional DNN for polyphonic pitch detection and transcription. The input is the magnitude Harmonic Constant-Q Transform (HCQT) of the audio. The CQT is a time-frequency transformation suitable for a convolutional neural network, which I also use in this thesis. It can be contrasted to the Fourier transform, which has linearly spaced center frequencies $f_n = n * \frac{SR}{N}$, where n is the frequency bin index, SR is the sampling rate, and N is the dimension of the transformation space. The CQT instead has logarithmically spaced center frequencies $f_j = f_{min} * 2^{\frac{j}{b}}$ where f_{min} is a pre-defined minimum frequency and b determines the number of bins per octave. The fact that the center frequencies are logarithmically spaced results in the audio representation being translationally invariant, meaning that shifting a musical interval up or down will not change the distance between the harmonics in the CQT. This enables a Convolutional Neural Network (CNN) filter to discover harmonic patterns across the full range of frequencies. Another advantage of the CQT representation is that its resolution resembles that of

the human auditory system, with high resolution in the lower frequencies and wider bins in the higher frequencies. The downside of using CQT is that it cannot benefit from the Fast Fourier transform optimization, so is computationally expensive, $\mathcal{O}(N^2)$ instead of $\mathcal{O}(N \log(N))$. Bittner *et al.* add more resolution by computing multiple, overlapping CQTs, each starting at a different frequency. This technique is called HCQT. The CNN structure includes four lower layers with small filters of dimension 5×5 or 3×3 to detect small-scale patterns. The fifth layer, instead, uses a filter that spans an octave of audio. This layer increases the relevant receptive field of each output state—the context in the input HCQT it can access—without needing to make the network very deep. The sixth layer uses a 1×1 filter to combine all the learned features and output a pitch activation map [47]. The DNN proposed in this thesis utilizes CNN layers whose structure is closely based on this network. Since the pitch correction task is sensitive even to a small amount of pitch shift, I also choose to use the CQT for its finer resolution in the lower frequencies. I do not use HCQT as it is too computationally expensive.

Basaran *et al.* add a GRU layer [4, 57] to the pitch detection CNN described above in order to model the sequential nature of [58] audio and music signals. The network estimates the main melody in polyphonic audio signals in the CQT representation. I include a GRU in the proposed algorithm.

4.2.2 Deep learning

Research in the broader field of deep learning also provides a useful background for the given task. One challenge with the pitch correction DNN is that its depth makes it hard to train. Wager *et al.* improve the performance of a DNN with multiple layers by first training the lower layers on a smaller task, then initializing the corresponding the full network with the trained weights. The network—designed for automatic speech recognition—has a similar structure to the one proposed in this thesis, with linear lower layers for feature extraction followed by recurrent layers for sequential processing [59]. I use a similar type of pre-training of a smaller version of the DNN in the experiments described in this chapter.

Wavenet is a highly expressive model that can synthesize or transform sound at the level of the sample [46]. The *Wavenet*-based dereverberation network introduced in [48] demonstrates that it is able to learn a transformation for a highly complex task. The network is trained in an adversarial manner [60] to help the results retain the nature of the original signal. It provides inspiration for moving to a sample-by-sample model from the current note-by-note model. Though this thesis does not include experiments using such a model, the concept of moving to a more fine-grained representation is worth investigating.

4.2.3 Audio signal processing

Monophonic pitch detection and transposition techniques provide tools for feature extraction and post-processing. I use the pYIN algorithm [61] for pitch detection in this thesis. pYIN is used as a benchmark for measuring frequency in monophonic music signals [29]. Unlike other state-of-the-art algoritms, including CREPE [62], its resolution is not limited to a margin such as 20 cents. While this precision is suitable for MIR tasks such as music recommendations, in the case of musical intonation, 20 cents can make the difference between being in tune or out of tune. The pYIN algorithm, like the pitch detection algorithm used for Antares Auto-Tune, is based on autocorrelation for periodicity detection. Autocorrelation alone is not always reliable: It might find a stronger periodicity at a harmonic—for example, the octave—or choose a maximum periodicity at value 0, when the signal is not shifted. The pYIN algorithm is based on the YIN pitch detection algorithm [63], which adds steps after the autocorrelation computation to reduce error from 10 per cent to 0.5 per cent. These steps include a weighting of the autocorrelation output to reduce periodicity at harmonics, and a cumulative normalization that discourages selection of the 0-lag periodicity without a need for an arbitrary threshold. It also includes linear interpolation to further refine the pitch estimate. The pYIN algorithm applies a HMM to the output of YIN, using a set of candidate periodicities instead of selecting the maximum one. This results in a smoother output. Both YIN and pYIN output 0 when they do not detect a period, making the output suitable for voice activity detection and note boundary analysis. The precision of the pitch measurement combined

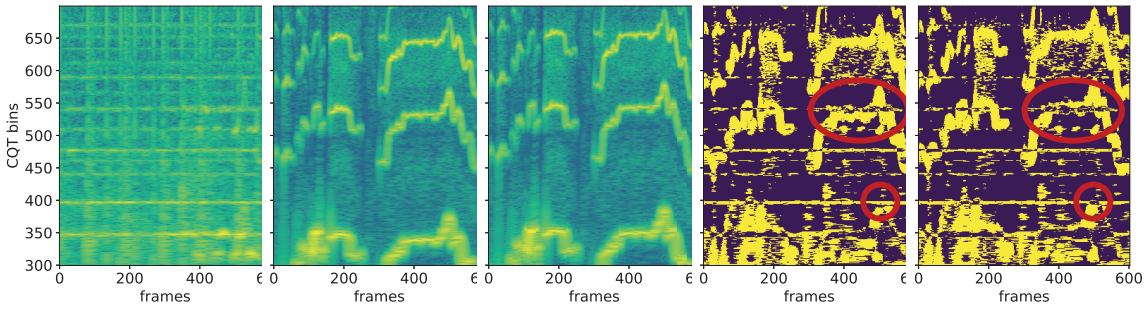

Figure 4.2: CQT of the vocals and backing tracks computed using Librosa [64]. The plot focuses in on frequency bins 300 through 700 out of 1024 for better visibility. (a) shows the CQT of the backing track. The horizontal lines are due to constant pitches, which indicates that a chord is being played. (b) and (c) show the CQT of the vocals before and after the correction, respectively. (d) and (e) show the superposed vocals and backing track before and after corrections. The CQTs are binarized by the mean of their amplitude, which makes the louder components stand out for visibility (see Section 4.3.6). In this example, we see that the correction shifted the pitch of the vocals up and centered it around the desired harmonics of the backing track (red circles).

with the voice activity detection make pYIN ideal for the automatic pitch correction task. As described in 4.3.1, the HMM used in pYIN inspires the HMM explored in this thesis for detecting note boundaries.

Another useful tool is TD-PSOLA, a pitch shifting algorithm [51]. I use it in the post-processing phase, applying the shifts to the audio. Similarly to YIN, it starts by detecting periodicity in audio. It then splits the audio signal into individual periods, and shifts these slightly in time to produce the effect of shorter or longer periods. It uses cross-fading to avoid clipping. TD-PSOLA is suitable for the task of applying pitch corrections because it produces a natural sounding result. Unlike other pitch-shifting techniques such as resampling, it does not modify the structure of the waveform except at the edges of audio windows. This minimizes changes to the formant—the harmonic structure of the sound—so that the timbre is not modified along with the pitch.

4.3 The proposed algorithm

The proposed model takes as input two CQTs—the monophonic vocal track's and the backing track's (also called accompaniment)—and outputs pitch shift predictions. The DNN struc-

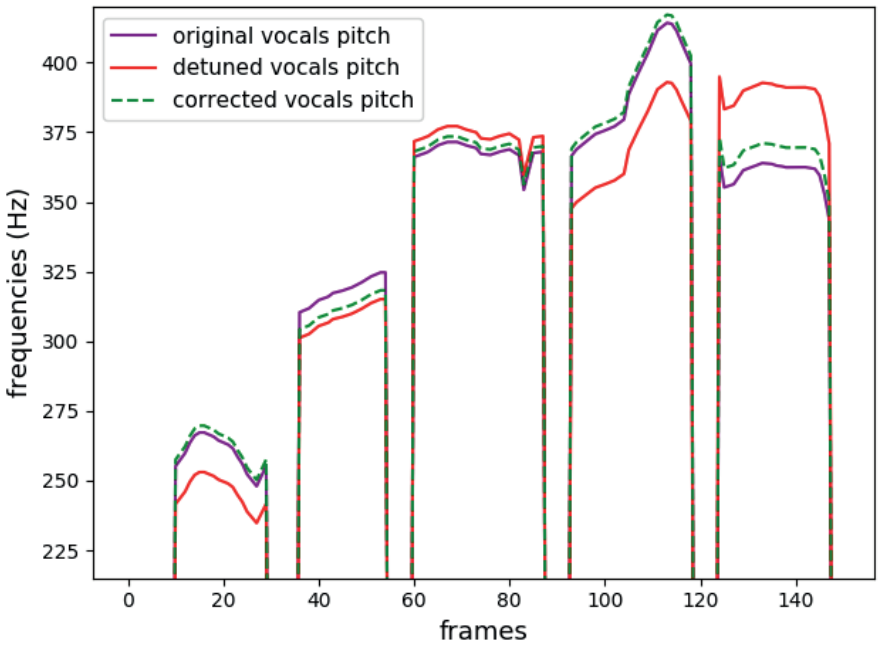

Figure 4.3: Training technique for the model using synthesized in tune versus out-of-tune data pairs. The program first detunes the original singing. As a result, the measured pitch moves from the purple line to the red line. The deep neural network takes as input the detuned signal, and predicts shifts that will restore the original pitch. The result of the predicted corrections is in green.

ture is built on the assumption that the backing track has clearly identifiable pitches—a chord progression—which serve as a reference for the vocals. The program uses the harmonic alignment between the vocals and backing tracks to make its predictions. Figure 4.2 shows the CQT of vocals and backing track excerpts of a few notes before and after applying predicted pitch corrections. In the excerpt, the backing track pitches are mostly constant, meaning that a chord is likely being held. After the correction, the vocals appear to be more closely aligned with the backing track, as can be seen in a CQT combining the two tracks.

The model is trained in a supervised manner. Training data consists of pairs of performances that are identical except for the vocals pitch. The backing track remains fixed, as it is when the singer performs in a karaoke setting. The input-target pairs, while required to train the model, are difficult to come across naturally. Hence, I synthesize them by detuning high-quality singing performances to construct the input signals, and then train the DNN to predict the shifts that recover

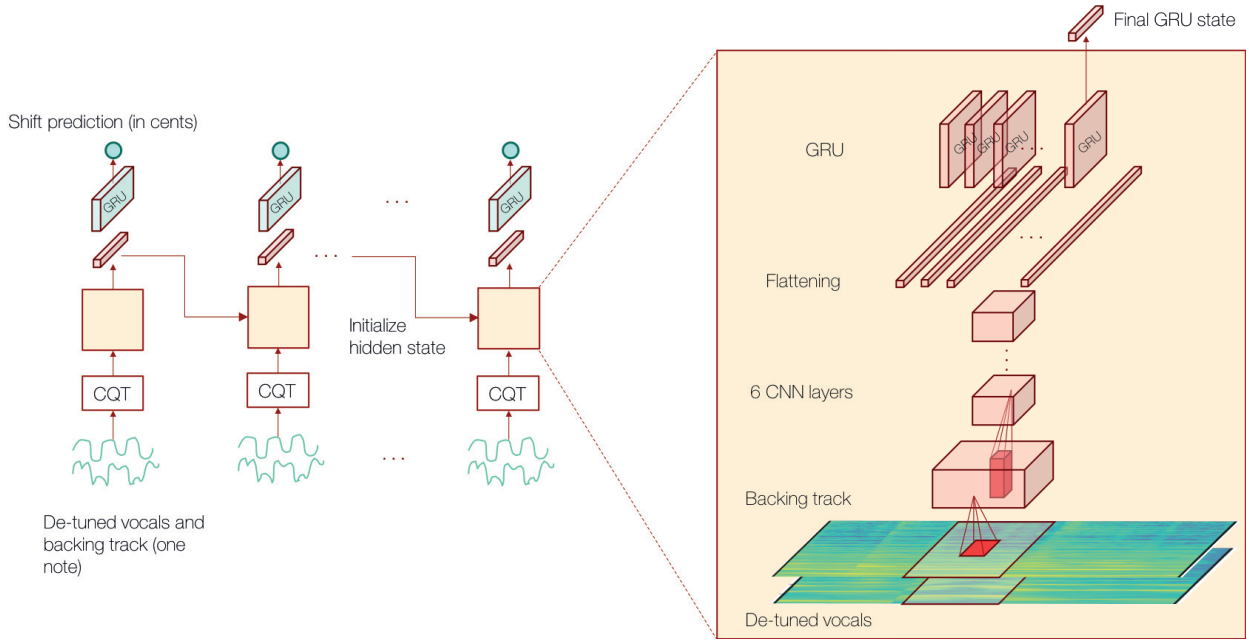

Figure 4.4: Model architecture with extension layer. A GRU sequentially processes the outputs for each note from the original DNN and is followed by a linear layer that outputs note-wise shifts.

the original performances. Section 4.3.6 describes the de-tuning process.

4.3.1 Note-by-note processing

The network corrects the pitch of each note by shifting all frames included in it by a constant. This approach is based on the assumption that every note is a single musical event, and that its accuracy can be improved by shifting it as a unit. The first step to processing the performance note by note is to detect the note boundaries. I choose not to use a musical score, first, because this makes the program usable in the many situations where no score is available, second, because this avoids inconsistent note boundaries due to alignment errors or improvisation. I tested three different approaches to score-free detection of note boundaries. Their respective outputs can be visualized in Figure 4.6.

The first note parsing technique, which produced the best result, was to define every transition silence as a note boundary. To this end, I analyzed the vocals pitch using the frame-wise pYIN algorithm, implemented as a Vamp plugin in [65]. The frame-wise pitch is set to 0 for unvoiced frames, which makes it possible to easily treat transitions between voiced and unvoiced frames

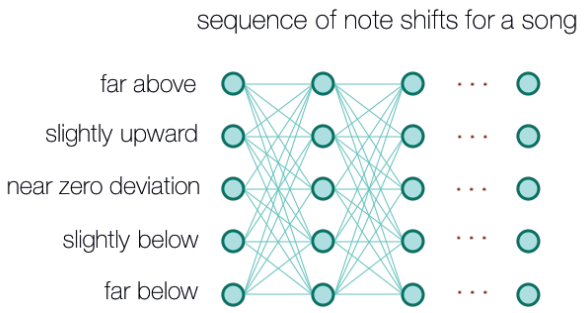

Figure 4.5: Note de-tuning Hidden Markov Model. The approximate de-tuning amount per note is defined in the hidden states. The exact de-tuning is sampled from the state using a Gaussian distribution,.

as note boundaries. A small amount of smoothing was required to avoid glitches in the case where a single frame had value 0. The advantage of this technique is that it minimizes artifacts during shifting. Any discontinuities arising due to pitch shifting a section of an audio recording are insignificant because the audio is silent at the discontinuity points. The disadvantage is that this note parsing technique fails to split notes when they are connected, though this is common in *legato* melodies. This means that if one part of a legato passage is out of tune and another part isn't, a shift that would correct the out-of-tune part would de-tune the remaining part.

The second approach, which turned out to be too error-prone for the task of automatic pitch correction, was to use the note-wise pYIN plugin available as an extension to the original frame-level pitch detection algorithm algorithm. This program is useful for melody detection, but I found that it did not always start and end notes at exactly the right frame for minimizing discontinuity. It often left small gaps between the end of one note and the beginning of the next one, making it difficult to determine what to do with these unaccounted frames. It also tended to be rather sensitive, converting pitch bending into two discrete notes. Any undesirable splitting of notes can have a significant impact on the ability of the model to perform reliably well. If a part of a note is separated from the rest and shifted in a different direction, the unnatural result can be displeasing enough to make a whole excerpt sound bad to the listeners.

The third approach assigns note boundaries to the vocal track using a Gaussian HMM applied to the frame-wise pYIN pitch contour. This approach combines the reliable output of pYIN with

a customized note parsing technique. The hidden states in the HMM are the equal-tempered scale frequencies within the range of given performance, $440 * 2^{\frac{p-69}{12}}$, where $p \in [21, 108]$, the standard MIDI range. 0 is included as a state for silence or unvoiced frames. The HMM standard deviations map to the difference in pitch between each equal-tempered scale frequency, leaving much room for the pitch in a note to vary from the center mean. The transition matrix is set to 0.001 everywhere except in the diagonal, which makes each row sum to 1. The starting probabilities are uniformly distributed. Fitting this model assigns a *note* state to each frame and provides boundaries both between *legato* notes and between unvoiced and voiced sections. The potential shortcoming of this approach is that it relies on the definition of a discretized scale. I note that it would be possible to use a more fine grained scale if working with a musical style that is not based on the equal-tempered scale—for example, use 22 subdivision of the octave for Classical Indian music. I used the equal-tempered scale here because the dataset I worked with was mostly made of popular music that used that scale.

4.3.2 Neural network structure

I trained two different DNN architectures. The second is an extension of the first, designed to include more temporal context across notes. The first version of the network consists of six stacked convolutional layers followed by a GRU layer. The network architecture is illustrated in Figure 4.1. The last output of the GRU is fed to a dense layer that predicts a single scalar output, the note-wise pitch shift. The convolutional filters pre-process the CQT, reducing its dimensionality while also extracting abstract features. Next, the GRU—from which the network only uses the last output—reduces the representation of a variable-length note to a fixed-length vector. Finally, the dense layer predicts the pitch shift in the approximate range of -1 to 1, which is mapped up to a semitone in either direction, or -100 to 100 cents. A semitone corresponds to one note shift on a piano.

The activation after each convolutional layer is the Rectified Linear Unit (ReLU) [42]. The DNN uses a linear activation for the prediction, to prevent having the model converge to favoring larger shifts. For example, the hyperbolic tangent function might have tended to move values closer

Table 4.1: The proposed network architecture.

	Conv1	Conv2	Conv3	Conv4
#Filters/Units	128	64	64	64
Filter size	(5, 5)	(5, 5)	(3, 3)	(3, 3)
Stride	(1, 2)	(1, 2)	(2, 2)	(1, 1)
Padding	(2, 2)	(2, 2)	(1, 1)	(1, 1)
	Conv5	Conv6	GRU	Linear
#Filters/Units	8	1	64	1
Filter size	(48, 1)	(1, 1)		
Stride	(1, 1)	(1, 1)		
Padding	(24, 1)	(0, 0)		

to -1 or 1 and away from 0.

The GRU recurrent structure is a way for the model to analyze the singer's note contour, which can last from a split second to multiple seconds, while smoothing over unvoiced or noisy sections. This is crucial because the algorithm is expected to rely on aligning harmonics, which only occur in pitched sounds. Another advantage of using the GRU is that the hidden state output by one note can initialize the hidden state for the following note, passing along some information about the previous notes. Even when using the simplest possible detuning model, which shifts every pitch by an independent amount, we can assume that some information from past notes (e.g. from the backing track) is useful. The GRU is unidirectional, meaning that it is only exposed to past musical events. I assume this is sufficient information, as a performing musician can adjust intonation without hearing future events.

The extended version of the DNN is designed to enable the model to include more temporal context. As shown in Figure 4.4, the final dense layer is replaced by a second GRU that takes as input the sequence of note representations output one-by-one by the first GRU. It finally applies a dense layer to the full sequence to output a song-level prediction sequence. This version has the potential to reach farther back in time, and include information from the chord progression in the backing track and long-term melodic patterns in the vocals. The downside of this model design is that the DNN is deeper, which makes it more difficult to train.

Table 4.1 displays the structure of the proposed network without the extension. The feature

extraction layers are convolutional, which is common for image processing. The input to the model is in spectrogram format, which resembles an image, except for the fact that its meaning is different along the time and frequency axes. In image processing, dimensions reduction techniques like max pooling are common. These techniques treat the x and y axes in the same way, which we wish to avoid. To preserve the frequency patterns axis, the proposed approach instead uses strides of two only in the time axis in three of the convolutional layers. This method was shown successful for the task of learning latent representations for speech generation and transformation in [66]. The third layer also includes a stride along the frequency axis, but this occurs only in one layer to not lose too much information. The fifth convolutional layer has a filter of size 48 in the frequency domain, which captures frequency relationships in a larger range of the CQT, as done in [47] and [66]. The error function is the Mean-Squared Error (MSE) between the pitch shift estimate and ground truth over the full sequence of notes. The MSE corresponds to the error in cents using the formula $|\text{cent error}| = 100 * \sqrt{\text{MSE}}$.

Applying the predicted pitch corrections

Once the program has output pitch correction predictions, these are applied to the vocals in post processing. The TD-PSOLA algorithm provides a natural sounding output with few artifacts. The shifting is constant across note, and subtle cross-fading is applied between legato notes—in between which there is no silence—to avoid glitches at the boundaries.

4.3.3 Post processing versus real time

The neural network introduced in this thesis is not explicitly designed to work in real time. First, the algorithm is designed for post-processing plug-ins such as are found in music apps like Smule. Second, the task of automatic pitch correction is challenging enough even given abundant computation time, so I choose to leave real-time applications to future work.

Even in its current form, though, the model might be adaptable to near-real-time processing as it only uses information from previous notes to make a prediction for the current note. Even

within a note, it processes one frame at a time in order. The challenge would be to make the data pre-processing—involving pitch detection and feature extraction—fast enough.

4.3.4 Dataset

During my internship with Smule, Inc, a company that offers a singing app for smartphone, my team and I constructed a training dataset by deriving from the "Intonation" dataset [2], which is assumed to be a collection of in-tune singing voice tracks. A detailed description of the dataset, including instructions on how to access it, is available in Chapter 5. The 4702 separate vocal tracks in the dataset are mostly of Western popular music, collected at Smule for good intonation. While browsing the dataset, I also discovered a few tracks of Blues; Western Classical music; Latin, Japanese, and Indian popular music, Country; and Rock. The songs are mostly based on the equal-tempered scale, but contain a wide variety of pitch deviation patterns from the scale. I discuss the measured pitch distribution in Chapter 5. While these real-world recordings contain some artifacts, no particular signal processing—e.g. denoising or filtering—has been applied to them. Each recording contains one minute of a performance, starting 30 seconds into the song. Although they are assumed to be in tune, this is not always exactly the case as the users are not necessarily professional singers. Overall, the sung pitch is sounds quite accurate and aligns reasonably well in timing and in pitch with the known musical score. Note when compared with the intended pitch. Hence, we can treat this paper as a proof of concept. The model can be trained on professional singing for best results.

Based on the metadata for each track indicating the backing track and user index, the dataset is split into 4561 training performances, 49 validation performances, and 64 test performances. The training set contains 709 backing tracks performed by 3468 different users, while the validation set is with 17 tracks sung by 43 users and the test set is with 16 sung by 62. There is no overlap in the backing tracks across the three sets. Overlap exists in the singer ID between the training and validation sets, but not with the test set.

4.3.5 The detuning process

As introduced in Section 4.3, the DNN is trained in a supervised manner. An input data sample consists of an out-of-tune performance, and the target consists of the note-wise shifts that should be applied to the vocals track to make it sound in tune. This type of pair is difficult to come across, unless one manually labels the corrections for every note in hundreds or in thousands of performances. The proposed approach to constructing training examples is to synthesize de-tuned examples. Singing performances from the "Intonation" dataset are de-tuned by shifting every note up or down and recording the amount of shift as the target. The synthetic pitch deviations are limited to approximately one semitone (100 cents) in either direction, a larger interval than the standard score-free approach of snapping to the nearest pitch, which limits the shift to 50 cents. In practice, it prevents errors in cases where the required shift is greater than 50, but can lead to degradation of the prediction accuracy on a too badly detuned input.

To detune the training data, the program shifts the magnitude CQT up or down. This is expected to not produce too noticeable artifacts that the program could learn instead of the pitch relationships. The one issue is formant shifting, but this is not a big concern when only shifting by $\pm$100 cents or less. I experimented with shifting the training data using TD-PSOLA, but this did not produce noticeably better convergence, and increased the computational complexity due to the need to compute autocorrelation instead of simply shifting a matrix.

De-tuning distributions

The de-tuning process described in the previous section requires assigning a distribution to the random shifts. Choosing a proper distribution involves balancing exposing the model to a wide variety of errors with ensuring it also is exposed to small deviations. A too spread distribution risks causing the model to frequenly apply large corrections, and produce noticeable errors.

I experimented with two distributions. The first is the random uniform distribution in the range [-100, 100] cents, adjusted to the logarithmic scale of cents so that the shift of the CQT spectrogram is linear. Random uniformly distributed shifts are very simple to implement, and provide the model

with plenty of examples of a wide variety of shifts, ranging from zero—which is useful, because singers are not likely to sing every note out of tune—to large ones. The downside is that this detuning technique is not based on real intonation patterns in out-of-tune singing. Furthermore, it is based on the strong and likely incorrect assumption that the detuning for every note is independent from that of other notes.

A Gaussian HMM addresses both of the shortcomings of the random uniform distribution in a simple manner. HMM states represent deviation levels, while the Gaussian distribution provides variance. The HMM structure is illustrated in Figure 4.5. The HMM is trained on real-world singing examples in the publicly available MIR-1K dataset of karaoke performances [67]. The proposed HMM outputs deviations in cents from the equal-tempered scale. These deviations are represented as the difference between the equal-tempered frequency output by the HMM used to parse notes, described in Section 4.3.1, and the median of the measured frame-wide pYIN pitch. As before, the equal-tempered scale can be replaced with any other division of the octave. The proposed model uses five hidden states, but the number is arbitrarily chosen and can be replaced by a different value. Sampling from this HMM produces sequences of pitch deviations that are based on real-world deviations and are not completely independent across notes.

To address the issue that all deviations will be 50 cents or less, because the distance to the nearest scale degree will never be greater than this, the MIDI pitch is moved by a semitone 5 per cent of the time across the median measured pitch, increasing the range to 100 cents. 5 per cent was another arbitrary choice, but the use of HMM is still expected to produce a slightly more accurate pitch behavior representation than the random uniform distribution.

The parameters learned from the MIR-1K dataset are in Table 4.3.5. The means show a state very close to the equal-tempered scale, at 3, two that are offset by a few cents—one in each direction—and two that are more than a quarter tone away. I note that singers were very likely to start close to an equal-tempered scale degree, and almost never started far off. They also didn't tend to stay far off, as the transition probabilities from a larger deviation to a larger deviation are

μ	σ	p_{start}	p_{trans}
$\begin{bmatrix} 52 \\ 13 \\ 3 \\ -8 \\ -73 \end{bmatrix}$	$\begin{bmatrix} 31 \\ 25 \\ 16 \\ 25 \\ 24 \end{bmatrix}$	$\begin{bmatrix} 0.1 \\ 9 \\ 76 \\ 15 \\ 0.1 \end{bmatrix}$	$\begin{bmatrix} 4 & 30 & 30 & 35 & 1 \\ 13 & 27 & 30 & 24 & 5 \\ 24 & 17 & 22 & 15 & 23 \\ 16 & 25 & 28 & 23 & 7 \\ 3 & 36 & 30 & 30 & 0 \end{bmatrix}$

Table 4.2: The de-tuning Gaussian HMM parameters fitted to the MIR-1K dataset. The first column shows the means, or hidden states, and the second column shows the standard deviations. The final two columns show the start and transition probabilities. All parameters are in cents, a logarithmic measure, and rounded to the nearest integer, except for zeros, which are set to 0.1 to show that no transition had zero probability.

the smallest.[2] One should note that these parameters include the occasional shifts of the note across the median pitch. Despite this modification, the distribution is slightly less spread than the random uniform option, as shows in Chapter 6, Figure 6.2.

The HMM model pitch deviation is not the most complex model that could be used for the task. It could be replaced by a RNN that uses additional information, such as absolute frequency—given that a singer might be more likely to sing sharp or flat based on the register—or spectral information. One could go another step further and synthesize out-of-tune singing using a WaveNet and/or adversarial training.

4.3.6 Data pre-processing details

The audio signals are normalized, then transformed using the CQT. The CQT covers 5.5 octaves with a resolution of 16 bins per semitone. The lowest frequency is 125 Hz. The top and bottom 16 bins are used as a buffer for pitch shifting, then truncated so that every input has dimension 1024. The frame size spans 92 ms and the hop size 11 ms. The vocals and backing track CQTs form two input channels to the neural network. In previous work, I experimented with using a third channel that would help bring out the contrast between the first two channels. I binarized the two CQT

[2] I later realized that the pitch detection was based on mixed MIR-1K signals, which included the accompaniment. This resulted in a more spread and noisy distribution, but I believe this larger spread was useful for training the model to address larger shifts.

spectrograms using the mean modulus as a threshold, a technique used in computer vision [68]. I then took the bitwise disagreement of the two matrices based on the expectation that the in-tune singing voice, better aligned with the backing track, would cancel out more harmonic peaks than the out-of-tune tracks. Figure 4.2 illustrates the three channels. It includes the binarized CQT before and after shifting though the third channel was ultimately left out, because it helps visualize the shift. Surprisingly, though the convergence with the third channel was better for multiple epochs, the loss with two channels ultimately dropped below its counterpart. The difference took long enough to appear that I only discovered it because I had left two models training for additional time despite having already concluded that the third channel improved the results.

4.4 Experimental configuration

4.4.1 Training setup

The program uses the Adam optimizer [69]. It processes one note at a time as a minibatch of seven differently pitch-shifted versions. It does not include batch normalization because the different versions of the same note are not *i.i.d.* When using the second GRU layer that bases the prediction on the sequence of notes, the outputs for each note are stored, then input to the GRU.

The program applies gradient clipping [70] with a threshold of 100. It reports validation loss every 500 songs and save the model with the best result along with the latest one.

4.4.2 Initialization

The convolutional parameters are initialized using He [42], and the GRU hidden state of the first note of every song is initialized using a normal distribution with $\mu = 0$ and $sd = 0.0001$. The hidden state of the note sequence GRU is initialized in the same way. When using the note sequence GRU, the weights from the note-by-note model can optionally be used to initialize the lower layers. These lower layers can also be fixed for an epoch before being trained all the network parameters.

Experiment settings				
Note parsing	**De-tuning**	**Learning rate**	**Extension**	**Initialization**
Silence	Uniform	0.00001	No	He, Gauss
Silence	HMM	0.000005	No	He, Gauss
HMM	HMM	0.00001	No	He, Gauss
Silence	Uniform	0.000005	Yes	pre-trained, Gauss
Silence	HMM	0.000005	Yes	He, Gauss
Silence	HMM	0.000005	Yes	pre-trained, Gauss

Table 4.3: The *Note parsing* column indicates whether the note boundaries were assigned based on silent pYIN frames or based on state changes in the HMM assigning a scale degree to each frame. The *De-tuning* column indicates whether the de-tuning distribution was random uniform or sampled from the HMM trained on MIR-1K. The *Extension* refers to whether the song-level GRU is added to the model architecture. Finally, the *Initialization* column provides the distributions used to initialize the parameters, and whether the feature extraction layers were initialized using pre-trained parameters from the model without extension.

4.4.3 Experiments

In this thesis, I report test results on a set of different configurations, listed in Table 4.4.3. Note that I only report configurations that showed the strongest convergence. First, I compare note parsing techniques and de-tuning techniques when training the smaller version of the model. I examine various learning rates. For the best performing models, I add the song-level GRU extension to check whether the deeper neural network performs better. I train the extended model either from scratch or initializing feature extraction parameters with the values learned for the smaller model. In the latter case, I freeze the pre-trained weights for one epoch. This technique was shown successful in previous work, e.g., [59]. For each model, I report results either with learning rate 0.00001 or 0.000005, based on which setting converged best. Other learning rates did not converge.

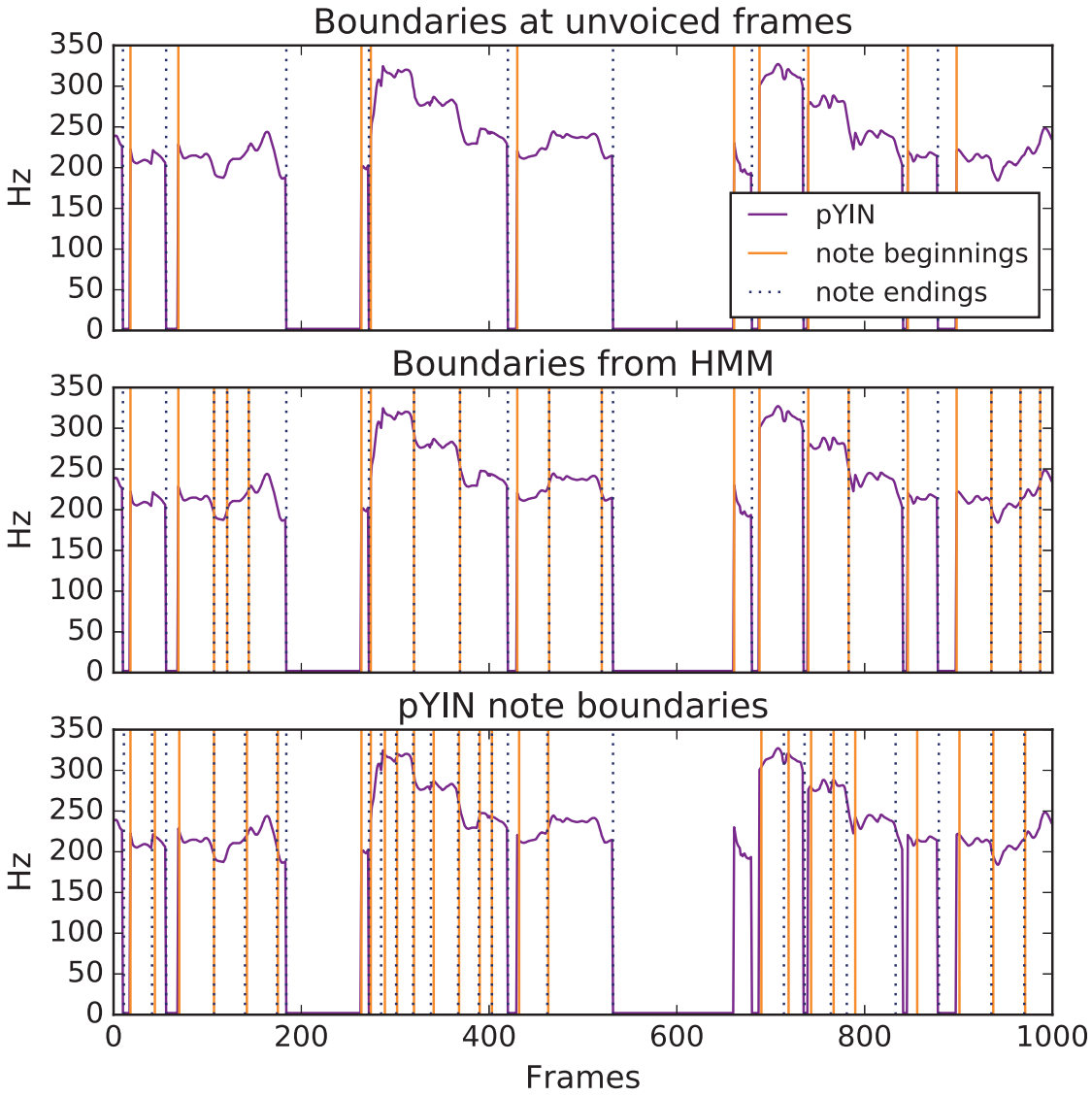

Figure 4.6: Comparison of three different note boundary detection outputs for an excerpt from *Attention* by Charlie Puth. The purple line shows the pYIN frame-wise pitch contour. The vertical lines show the note boundaries. The first and second approaches use the frame-wise pYIN pitch output. The first approach assigns note boundaries at the beginnings and ends of unvoiced sections. The second approach fits a Gaussian HMM to the pitch contour, and uses the hidden state sequence of equal-tempered scale frequencies along with unvoiced frames to assign boundaries. The third one uses the pYIN note-wise output. In this example, the unvoiced frame approach fails to split some *legato* passages into individual notes and the pYIN note approach is the most sensitive, assigning the largest number of notes. Sometimes this is musically relevant: for example, the lyrics in the three-step descending sequence around frames 300 to 400 are "knew-that-I, knew-that-I, knew-that-I", splitting each step into three musical events. The pYIN note detection detects these boundaries. However, it misses some notes—for example, the first note after frame 600—and its boundaries are not exactly aligned with the frames that switch between being voiced and unvoiced—for example, in multiple locations between frames 800 and 900.

CHAPTER 5

"INTONATION": A DATASET OF QUALITY VOCAL PERFORMANCES REFINED BY SPECTRAL CLUSTERING ON PITCH CONGRUENCE

This chapter describes the collection process of the "Intonation" dataset used in this thesis[1]. The "Intonation" dataset consists of amateur vocal performances with a tendency for good intonation, collected from Smule, Inc. The dataset can be used for music information retrieval tasks such as automatic pitch correction, query by humming, and singing style analysis. It is publicly available.[2] I describe a semi-supervised approach to selecting the audio recordings from a larger collection of performances based on intonation patterns. The approach can be applied in other situations where a researcher needs to extract a subset of data samples from a large database. A comparison of the "Intonation" dataset and the remaining collection of performances shows that the two have different intonation behavior distributions. I also analyze the "Intonation" dataset to check whether its amateur performances of mostly Western popular music show similar tendencies to those described in studies in Chapter 2.

5.1 Datasets for music research

Useful datasets have been made available for certain research topics in the fields of music information retrieval and audio. These include sound event detection [71], source separation [72], and recommendations [73]. Sometimes, though, the best dataset available for a topic is huge and difficult to process. A large collection of audio recordings is available, but the recordings with suitable characteristics for the given analysis form a smaller subset of the dataset. The filtering process to extract the desired samples can be labor intensive, requiring that the researcher select the samples with the desired features, which may or may not be labeled and can be hard to model. One way to

[1]This work was supported by the internship program at Smule, Inc.

[2]The dataset and detailed description of the contents are available upon request via `https://ccrma.stanford.edu/damp`.

approach this selection process is to automate it using feature engineering and clustering.

This chapter provides an example of a semi-automatic process for the task of searching through a large database of amateur karaoke performances for samples with a tendency for good musical intonation. The need for this task arose when designing a machine-learning model to predict pitch correction. The task requires selecting performances that were in tune enough but not those that were out of tune or contained little singing. Note that this task requires quantifying the concept of singing in tune. As described in Chapters 2 and 3, the concept is not obvious to model directly. A semi-supervised approach makes it possible to avoid creating an explicit definition of in tune. We first extracted musical intonation features from each performance, then applied spectral clustering to them and subjectively choose clusters that sound in tune by listening to samples from each. We also introduced the resulting dataset and an analysis of the intonation tendencies of its performances. Though I present this approach for the automatic pitch correction task, it can be adapted to other tasks, datasets, and features.

5.2 Related work

5.2.1 Automatic pitch deviation analysis

Automatic analysis of musical intonation behavior has also been performed in other contexts. For example, Nichols *et al.* [74] described an approach to discovering talented singers on YouTube based on features extracted mostly from the audio. One of the main features they chose consisted of a pitch deviation histogram, which characterizes intonation behavior of a full performance in a low dimension. Given that the performances were typically not associated with a musical score and that the singing was mixed with the accompaniment and other background sounds, the authors built the histogram from the STFT amplitude peaks. A singer who sings flat should have a histogram skewed to the left, and an active vibrato will cause values to spread. The proposed feature extraction task is different from [74] because, as described below, the proposed model has access to the musical scores of the vocals and because the audio sources are separated. One can, therefore, apply a standard pitch detection algorithm to each vocal track and compare the results to the musical score.

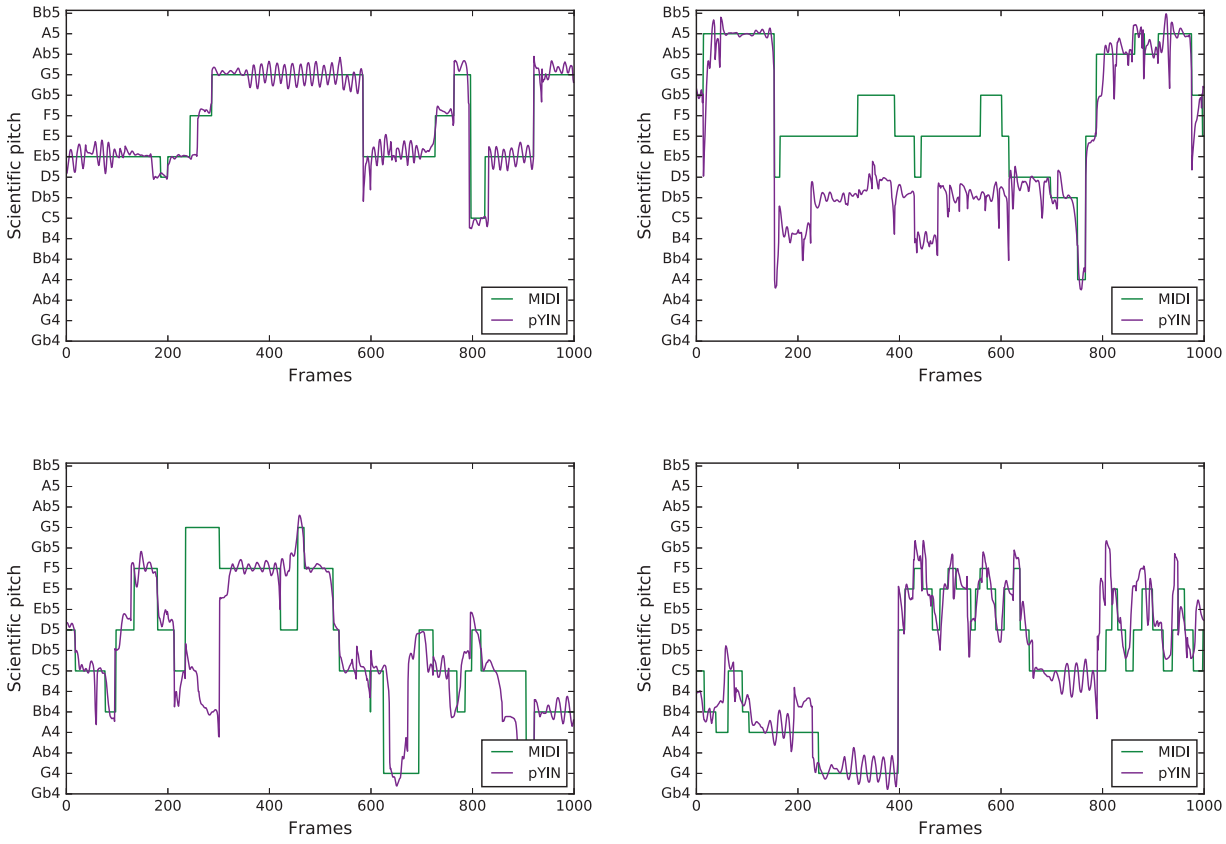

Figure 5.1: Singing pitch analysis of sample performances with aligned MIDI. Two are in the clusters selected for "Intonation" dataset (top), two in the remaining clusters (bottom). Much can be learned about the individual performances. The top two appear more tightly aligned to the expected pitch, though the second plot contains harmonization at a major third below the musical score. The vibrato in the first plot is particularly smooth, a sign of an advanced singer. The third plot shows frequent deviation from the score, while the fourth shows deviation at the beginning and the end but accuracy in the middle, along with a smooth vibrato. Still, it is difficult visually determine from this data format whether a performance sounds in tune.

Comparison of performance pitch and musical score is also used by [75] in the context of a tool for musical performance visualization.

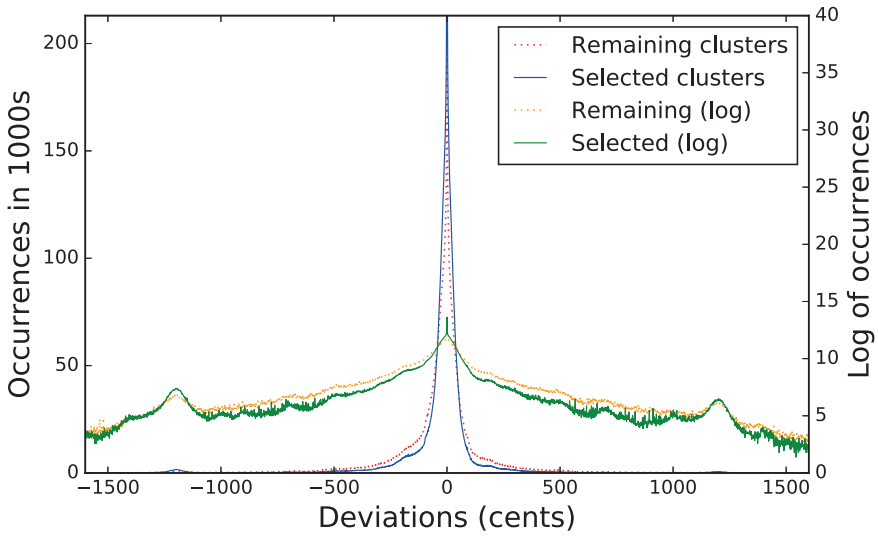

Figure 5.2: Global histograms of singing pitch deviations from the expected MIDI pitch in cents summed over 4702 performances in the "Intonation" dataset and 4702 in the remaining clusters. The plot is truncated at the top for readability. Scaled log histograms make more noticeable the small peaks at 1200 cents in both directions, due to octave deviations, common among singers. There is also, interestingly, a larger number of deviations between 100 and 300 cents in the negative direction than in the positive direction.

Figure 5.3: Comparison of positive and negative deviation counts for cents ranging from 1 to 100 (omitting 0) for both datasets. In both groups, negative deviations are more common than positive ones. The "Intonation" dataset deviations are more concentrated around zero.

5.3 Data collection and feature extraction

We collected solo vocal tracks of karaoke performances from a very large database. The first step was to filter for performances where singers used a headset—avoiding incorporating noise from the backing track into the recording. Given that we had access to a musical MIDI score of expected pitches, we also used a simple heuristic to filter for performances that were aligned enough with the score to exclude scenarios such as people speaking instead of singing. We kept this heuristic lenient enough that in-tune performances where the singer used harmonization (sang different pitches than the expected melody) or made other intentional deviations from the MIDI track wouldn't be excluded. This pre-filtering provided 14403 performances.

The next step was to summarize intonation patterns of a performance using a low-dimensional set of features. The procedure is shown in Figure 5.4 for two example performances. We first compared the singing pitch to the expected pitch in the MIDI score. We computed the singing pitch using the pYIN algorithm [61] on one minute of audio, starting at 30 seconds to avoid silence, with one sample (frame) per 11 milliseconds. pYIN has a high frequency resolution because it is based in the time domain and refines results using linear interpolation. Resolution is crucial for musical intonation, where a few cents difference can determine whether a pitch sounds in or out of tune. We shifted the MIDI score by a global constant to the octave nearest to the singing pitch, which can differ based on gender, age, and vocal type. We then computed the frame-wise absolute values of the difference in cents $\left|1200 * log_2 \frac{f_1+\epsilon}{f_2+\epsilon}\right|$ between the performance and MIDI score. Of this set of values, we kept the differences less than or equal to 200 cents, equivalent to two semitones, in order to focus the analysis on intonation behavior when the singer was close to the expected pitch. Larger differences could be due to many reasons, ranging from misalignment of notes in time to harmonization, and might add undesired noise to the distributions.

Finally, we summarized these variable-length sequences of frame-wise differences in a fixed, low-dimensional representation. We generated a random sample of 10,000 differences with replacement for every performance and kept 31 evenly spaced quantiles. This empirically chosen

number is large enough to effectively summarize the characteristics of the distribution but produces a low enough dimensionality for clustering.

5.4 Spectral clustering

As suggested by the studies described in Chapter 2, an advanced singer might produce both smaller and wider pitch deviations, due to a pronounced vibrato or expressive variations such as pitch bending, time shifting, or harmonization, than a singer who sings close to the musical score but is slightly off pitch. For this reason, selecting performances based on a simple metric like average distance of singing pitch from the score would not have been suitable. A semi-supervised approach also made it possible to avoid directly modeling the concept of being in tune. The approach clustered performances based on features generated from the deviations. The choice of which cluster to keep was based on listening to samples from each.

Spectral clustering was applied to the summarized performances using the signless Laplacian matrix as the adjacency graph [76]. This graph is based on selecting nearest neighbors (50 in this case). In practice, the program clustered approximately 5000 songs at a time into 3 or 4 clusters, depending on which value produced better Newman modularity [77]. I then listened to 50 samples from every cluster and subjectively determined the intonation of every performance by evaluating it as in tune, "neutral", out of tune. Consistently, one cluster produced distinctly good results with roughly 75 per cent of the songs classified as in tune and many of the remaining songs being classified as neutral rather than out of tune, while the other clusters had only a small percentage of performances classified as in tune.

Keeping the samples from the selected clusters resulted in the "Intonation" dataset of 4703 performances. Though not every performance is in tune and not every performance in remaining clusters is out of tune, a majority of in-tune performances in this dataset suffices for many machine-learning applications.

5.4.1 Genre, bias, and related challenges

An undesirable outcome of this clustering approach was that musical genres were clustered along with intonation patterns. As described in Chapter 2, different musical traditions and sub-traditions often have diverging traditions regarding musical intonation. In the Smule samples, a majority were of Western popular music, and the in-tune clusters tended to consist mostly of such music. I had planned to use performances from other clusters for testing, but realized that many of the performances were country, where singers deliberately might wish to sing flat, and the automatic pitch correction model trained on a different genre might not apply. As this was an intern project, I do not currently have the ability to update the clustering technique, and have to focus on the effects of the proposed algorithm on Western pop music. In future work, if implementing this program for real-world use, I would first separate performances by genre, then apply spectral clustering within the genre. This improvement requires advanced genre classification, but this field is growing and improving quickly in the context of music recommendation systems.

5.5 Analysis

The quality of the dataset is difficult to measure without a subjective listening test. At this point, we do not attempt to directly show that the "Intonation" dataset performances have better intonation than those in the remaining clusters. Instead, we show a difference in the intonation behavior distributions in the two collections. In order to compare samples of the same size, we analyzed the full "Intonation" dataset of size 4702 and a randomly selected a sample of the same size of performances from the remaining clusters.

5.5.1 Data pre-processing for analysis

We computed the frame-wise differences between singing pitch and MIDI score similarly to the way described in Section 5.3. Unlike before, we retained the sign instead of taking the absolute value in order to know whether the pitch was sharp or flat. We also kept all deviations instead

of discarding those larger than 200 cents: At the analysis stage, we are interested in intonation characteristics across the whole performance, including the larger deviations due to harmonization, expressive deviations, or inaccuracy.

To minimize misalignment before computing the deviations, we applied dynamic time warping [78] to better align the MIDI and singing pitch tracks. This algorithm stretches both signals in time in a way that minimizes the total sum of distances between the two. We used the algorithm as described in [79] and implemented in [64]. To avoid distorting the pitch track, we forced the algorithm to apply most time warping to the MIDI, which consists of straight lines. We discarded frames where either the musical score or pitch tracks were silent in order to only consider active frames in the analysis. Figure 5.1 shows four example performances after the initial processing. The top two are from the selected clusters and the bottom two from the remaining clusters.

5.5.2 Pitch deviation histogram

We compared the sequences of frame-wise pitch deviations from the selected clusters to those from the remaining clusters. Similarly to [74], we computed histograms of the deviations from the equal-tempered MIDI score summed over all performances in each group, normalizing them to have the same total counts. Figure 5.2 shows that the "Intonation" dataset deviations are more concentrated very close to 0 than those in the remaining clusters. The same can be observed at other harmonization peaks, ± 1200 cents (an octave) and other values in between, indicating more intentional harmonization and less accidental deviation. There is also, interestingly, a higher concentration of counts between 100 and 300 cents especially in the negative direction. This dataset is of amateur singing, so there is also a chance that some singers would be singing flat. However, this clearly visible peak might be due to intentional harmonization and expressive suspensions.

5.5.3 Pitch deviation probabilities

We examined whether we could find intonation tendencies like those described in Section 2.2.2. Unlike in the data used in the cited studies, the backing tracks are fixed recordings, so all pitch

Results from "Intonation" dataset (4702 performances)		
Cents range	Negative/positive deviation ratio	Var
1 to 2	0.500	0.001
2 to 16	0.506	0.001
1 to 100	0.532	0.002
100 to 300	0.727	0.002
Results from other performances (9701 performances)		
Cents range	Negative/positive deviation ratio	Var
1 to 2	0.500	0.001
2 to 16	0.509	0.001
1 to 100	0.541	0.002
100 to 300	0.700	0.002

Table 5.1: Probability estimates of negative versus positive frame-wise deviations of singing pitch from the equal-tempered MIDI score, computed using bootstrapping. The analysis was performed within different ranges of interest. When the deviation is less than 100 cents, the singer did not sing a different note. We found a particularly strong tendency towards negative deviations in the range of 100 to 300 cents.

adjustments happen in the voice. This can affect the pitch deviation distributions. In Figure 5.3, we examine deviations within 100 cents because a larger deviation corresponds a different note. Both collections tend towards negative deviations, but the tail is lighter in the selected clusters.

We quantify this result by estimating the probability of negative versus positive deviations within various absolute deviation thresholds using bootstrapping [80] with 10000 iterations, as shown in Table 5.1. We choose ranges of cents that are of interest when comparing theoretical musical intervals generated using the equal temperament versus other intonation systems (e.g., Pythagorean or Just intonation, described in the cited studies). Use of other intonation systems would explain deviations of 2 to 16 cents. We first examine the ratio of deviations less than 2 cents. As expected, a probability of 0.5 shows no significant preference for flat versus sharp intonation. Within 2 to 16 cents, we get 0.51. However, the largest probabilities occur at larger values, 300 cents. We cannot determine whether this deviation is a desirable effect or due to an unknown factor. The tendencies are observed in both collections.

5.6 Dataset description and applications

The "Intonation" dataset contains the full unmixed and unprocessed vocal tracks of 4702 performances. It consists of 474 unique arrangements by 3556 singers. It also contains the pYIN pitch analysis and multiple backing track features for the range of 30 to 90 seconds: constant-Q transform, chroma, mel-frequency cepstrum coefficients, root mean square error, and onset, all computed using the Librosa [64] package. Metadata of the performances is included. The dataset has applications ranging from the study of singing style in the context of karaoke performances, with optional study of user meta-data, to machine learning. For example, the vocal tracks can be used for informed source separation, an approach similar to separation by humming, described in [81] and [82]. Similarly, the dataset can be used for training a query-by-humming system, in a similar way to [83]. The vocal pitch tracks and backing track features can be used to study automatic pitch correction applications trained on real-world singing and develop a proof-of-concept model for vocal pitch correction [49].

5.7 Summary

We present a semi-automatic process for the task of searching through a large database of amateur karaoke performances for samples with a tendency for good musical intonation. The approach can be applied in other situations where a researcher needs to extract a subset of data samples from a large database. We show that the set of collected performances has a different intonation behavior distribution than the set of remaining performances. The resulting public dataset, "Intonation", of 4702 performances is available on the Stanford CCRMA DAMP website. The "Intonation" dataset can be used for music information retrieval applications like query-by-humming systems. Analyzing the dataset, we find that pitch deviations between the measured singing pitch and the MIDI score are more often negative than positive, implying that singers more often choose lower frequencies, use them unintentionally by singing flat, or decorate pitch contours with flat sections.

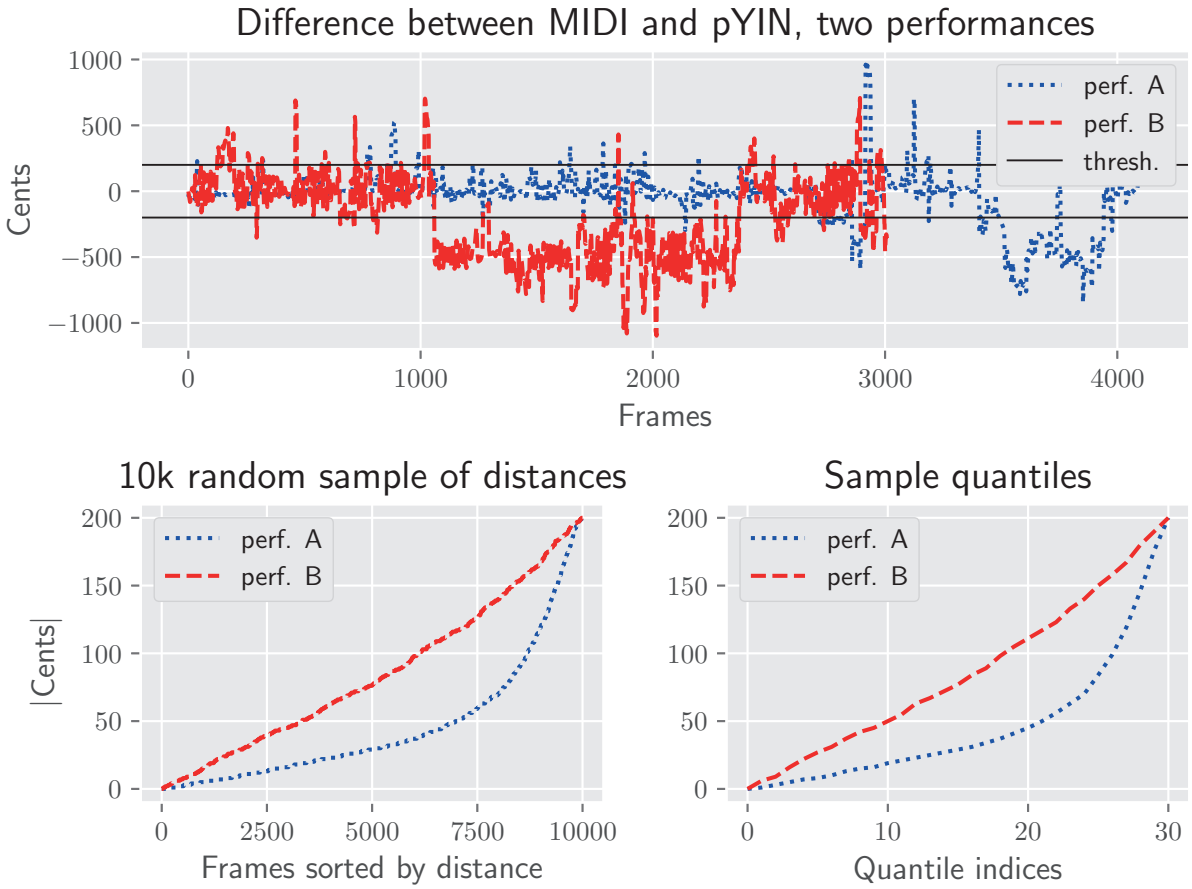

Figure 5.4: Data pre-processing steps for two example performances. The blue performance was selected for the "Intonation" dataset and the red performance was not. The first plot shows the frame-wise differences in cents between the measured singing pitch and equal-tempered MIDI score. We computed the absolute values of these differences and discarded those whose deviation was larger than 200 cents. The second plot shows random samples of 10,000 from the frame-wise difference lists, sorted by distance. The blue curve shows less deviation from the expected pitch than the red. The third plot shows 31 quantiles summarizing the curve in the second plot in a lower dimension.

CHAPTER 6

EFFECT OF DEEP PITCH CORRECTION ON PITCH DISTRIBUTION AND ON THE SUBJECTIVE LISTENING EXPERIENCE

In this chapter, I compare the results of the experiments listed in Section 4.4. I first examine the outputs on artificially de-tuned test data, and then on real-world karaoke examples from the MIR-1K dataset [67]. Based on histograms of the pitch deviations of the different outputs and their comparisons to the ground-truth Intonation dataset distribution, I choose one configuration. I use this configuration for an informal subjective listening test.

6.1 Experiments on the synthesized test set

The validation loss curves for each experimental configuration are displayed in Figure 6.1. For readability, the MSE is converted to absolute cents. The first three configurations did not use the song-level GRU extension, while the remaining three did. The last two configurations have different loss curves from the others because they were initialized using pre-trained weights. The initial large loss in these was due to the randomly initialized layers in the extension. One should note that the losses for different configurations are usually not comparable: the way the notes boundaries are assigned and the de-tuning techniques will all affect the value. How does error in cents correspond to musical accuracy? For perspective, 20 cents is the margin of error for some pitch detection algorithms [62]. For intonation purposes, I think that 20 cents can sound out of tune, but starts to be acceptably close. An average error of 20 cents, which would include large errors, where the model corrected in the wrong direction, would be a highly accurate result. Even 30 cents could be considered a decent result for the same reason.

The first two models trained for a longer time than the rest. When I tested them on real-world data, I was not happy with the audio result, and decided to test the other models at an earlier stage in training. Test results for the third model, which was trained on 223,000 songs instead of up to

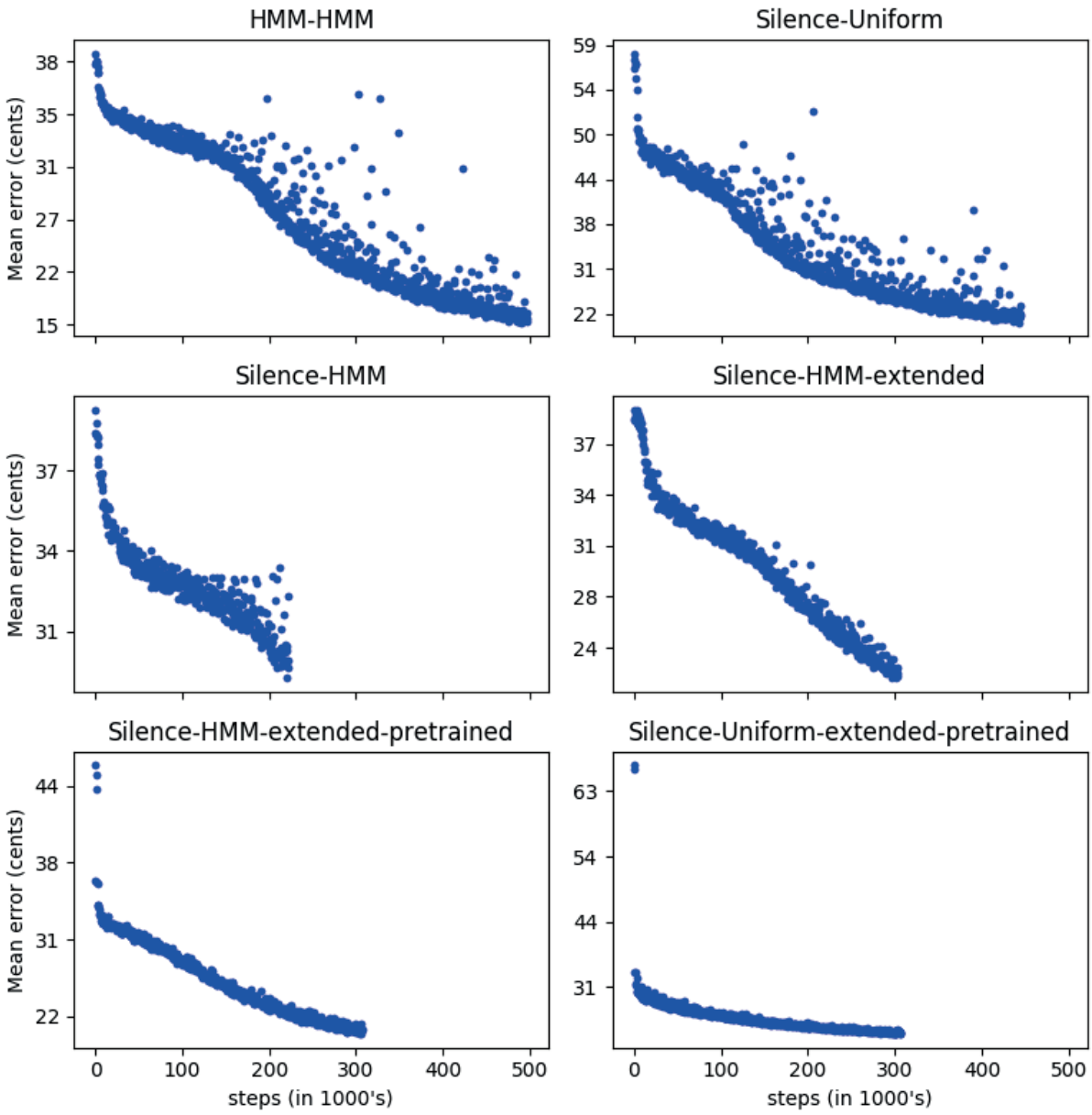

Figure 6.1: Validation losses for the respective configurations. The subtitle refers, first, to the note parsing technique, second, to the de-tuning technique, and then whether the model includes the song-level GRU extension layer and whether the model was initialized using pre-trained weights.

498,000 songs, turned out to produce superior results. In later sections, I describe how I determined this in a more objective manner. What is striking is that the validation loss in all curves has not converged. Additionally, some models reached very small MSE values. For example, the "HMM-HMM" model, which used HMMs both to assign note boundaries and for de-tuning, reached 0.03,

which corresponds to 17 cents average error. In practice, though, models that reached the lowest validation loss did not produce the best results on real-world data. I hypothesize that the models became too specialized for their specific configurations.

To compare the quality of the various configurations, I generated histograms of the pitch behavior in the results. These are displayed in Figures 6.2 and 6.3 for the synthesized test examples and the real-world MIR-1K dataset, respectively. The histograms show the pitch distributions before and after corrections. The pitch behavior is measured by generating the HMM described in Section 4.3.1 to assign an equal-tempered scale degree to each frame of audio. I then measured the signed difference in cents between the equal-tempered scale degree and pYIN frequencies assigned to each frame. A wider histogram would indicate that more frames of audio are centered at pitches far from any equal-tempered scale degree. While this metric is arbitrary, it allows for comparison between different datasets.

In order to interpret these histograms, I also compared them to histograms of the input data, which consisted either of de-tuned data or of the MIR-1K dataset, histograms from the Intonation dataset ground truth, and those from professional performances from a small, separate dataset, DSD-100 [84] of 150 tracks. The histograms are displayed in Figure 6.4. In Figure 6.2, we see that the input data de-tuned using a random uniform distribution was more spread than the data de-tuned using the HMM. The output histogram that stands out is the third one, "Silence-HMM", which is more symmetric than the others and less spread. The same configuration produces the least spread results in Figure 6.3. The reference datasets in Figure 6.4 show that the Intonation ground truth has a distribution similar to the output of the "Silence-HMM" model. This result made me choose the "Silence-HMM" model for the subjective listening test.

Figure 6.4 also displays the limitations of using the Intonation dataset as ground truth. The professional-level DSD-100 dataset has a narrower spread than the Intonation dataset, showing how different the pitch distribution is between amateur singers selected for accurate pitch, and professionals. The "Silence-HMM" model is able to shift the pitch deviation distribution of both synthesized and real-world data so that its corrected distribution matches that of the ground truth.

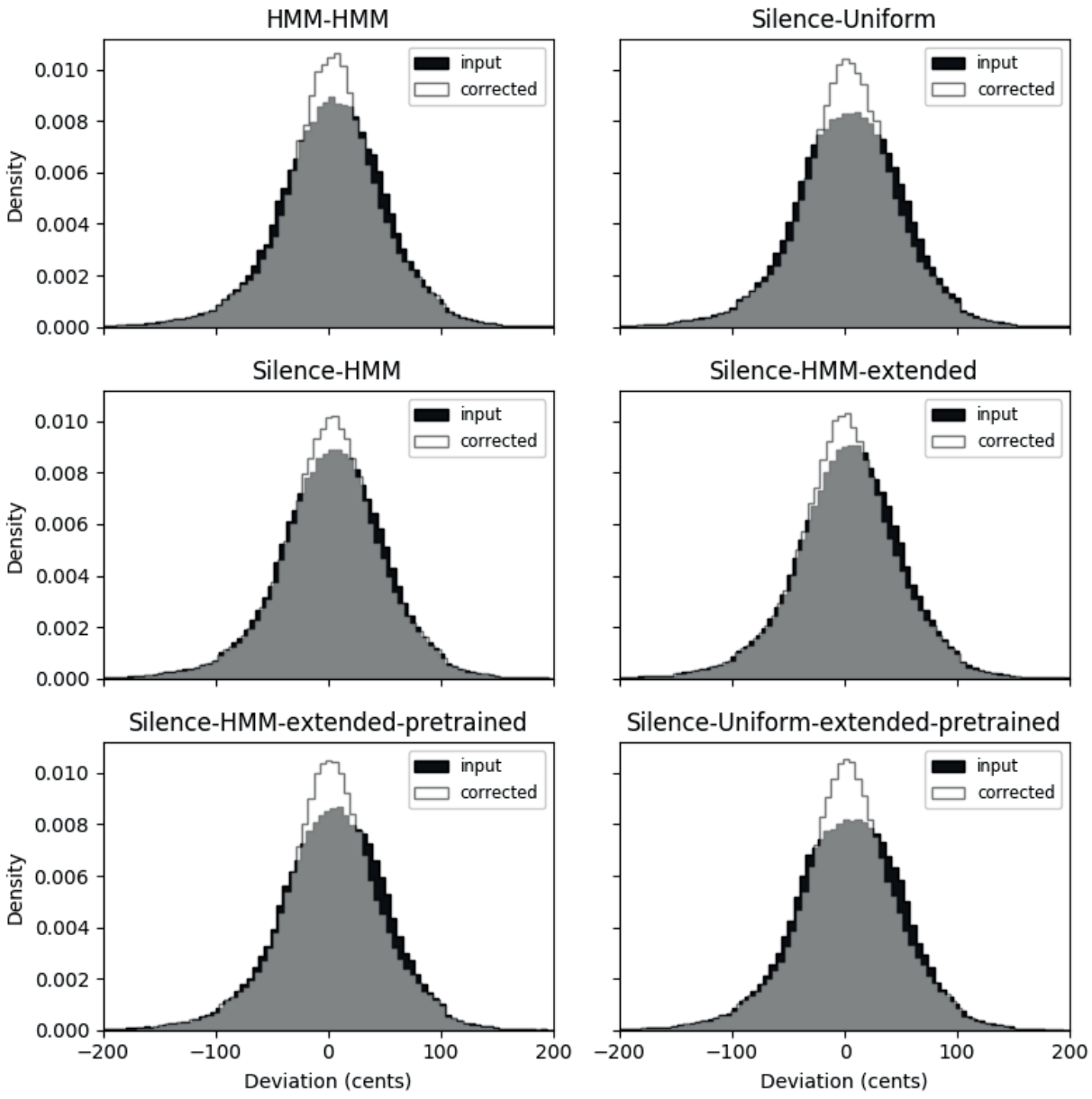

Figure 6.2: Pitch deviation histograms of the synthesized test data before and after corrections. Input data de-tuned using a random uniform distribution was more spread than the data de-tuned using the HMM. The third output histogram, "Silence-HMM" stands out as being the most symmetric.

Without professional-level data, though, it will not produce an outcome that has a distribution like the DSD-100 dataset.

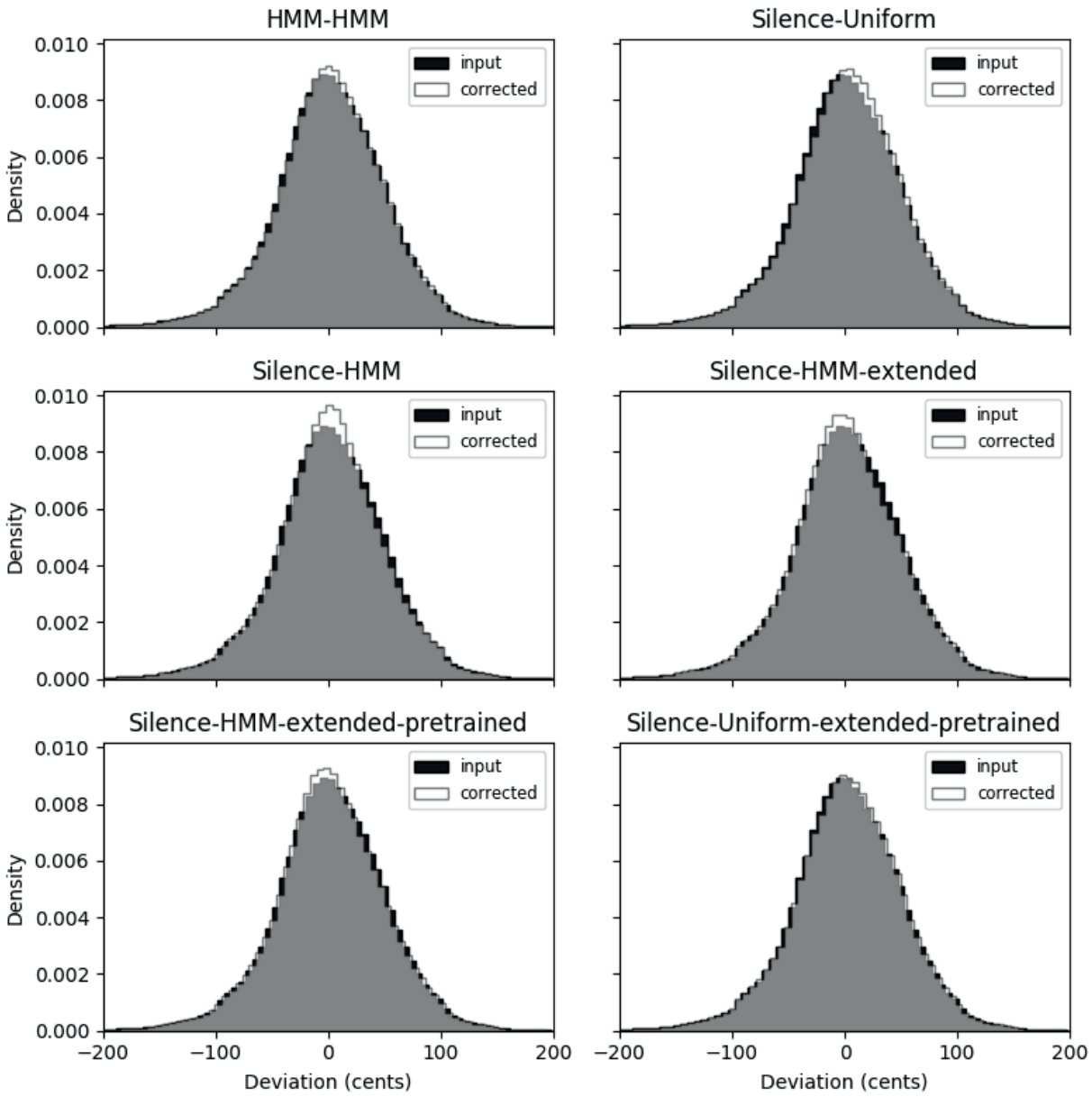

Figure 6.3: Pitch deviation histograms of the real-world MIR-1K data before and after corrections. The third output histogram, "Silence-HMM", again stands out as being the most symmetric.

6.2 Subjective listening test

I conducted a subjective listening test to qualitatively assess the pitch correction algorithm's performance. As listening material, I selected ten twenty-second samples from the MIR-1K original dataset, which I considered musically interesting. I had not heard the pitch correction results in the

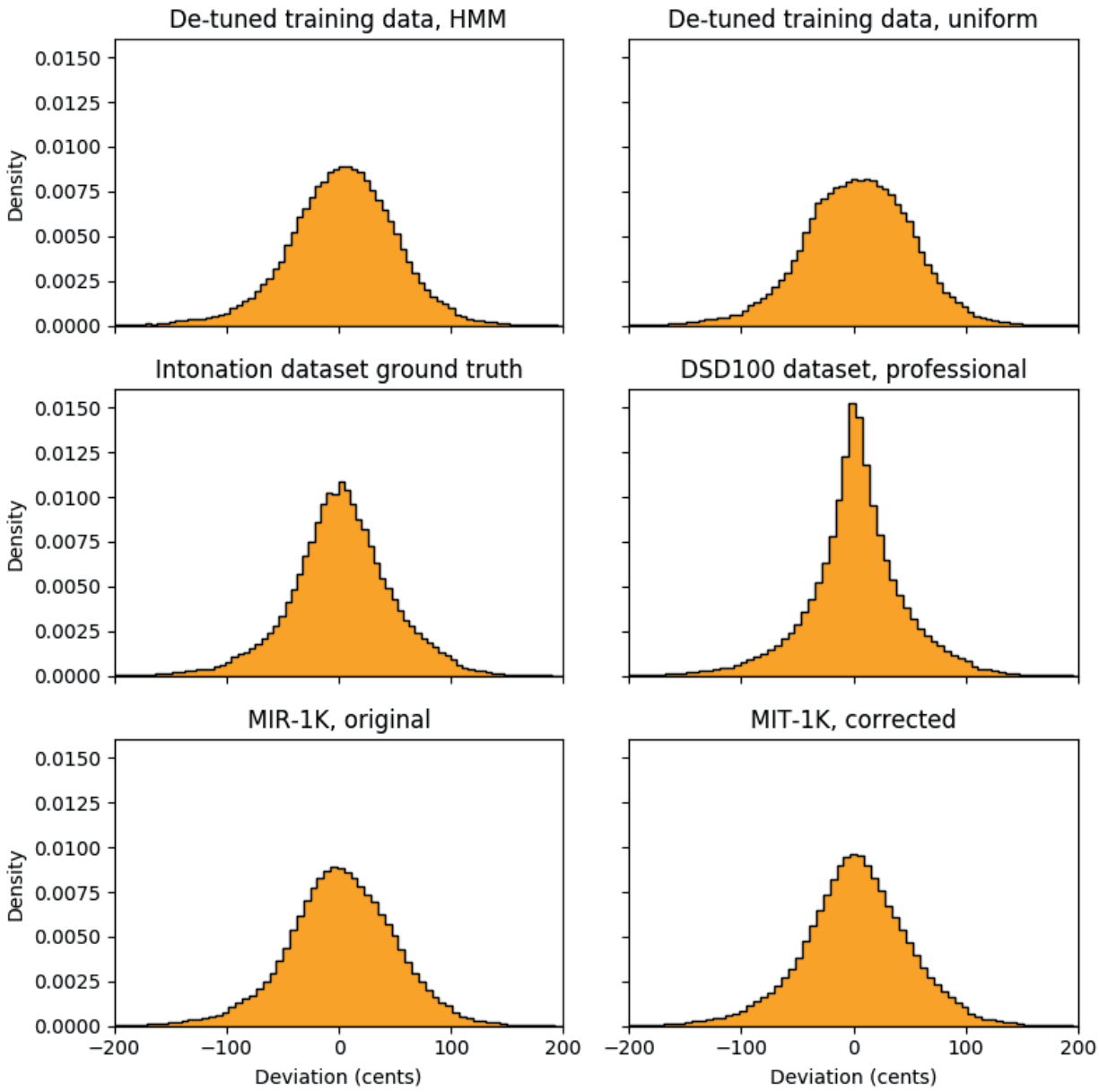

Figure 6.4: Pitch deviation histograms of various datasets. The first row displays artificially de-tuned training data. Note that it appears quite similar to the MIR-1K real-world data before corrections, shown in the bottom left subplot. The left subplot in the middle row shows the distribution of the Intonation ground truth. This resembles the histogram of the MIR-1K data after corrections using the "Silence-HMM" model, shown in the bottom right subplot. The middle-right subplot shows the distribution of a small, professional dataset. It looks strikingly different from the rest.

performances before when selecting the samples. I generated pairs comparing either the baseline performance to the original, or the corrected to the original.

I chose twenty seconds as a sample duration because this sample length gave listeners an idea

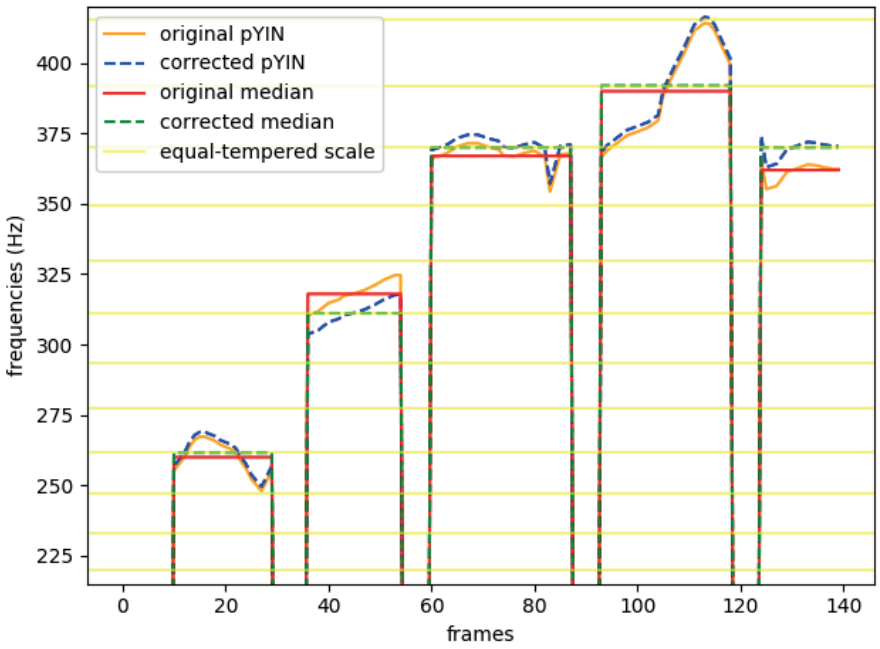

Figure 6.5: Example of the baseline algorithm's corrections. The algorithm determines note boundaries in the same way as the selected model, assigning them where silence occurs. It computes the median of each note, and shifts it so that its new median is the nearest equal-tempered scale degree.

of how the algorithm performed on a full musical phrase. Any ambiguity due to the sample length could be resolved by adding a comment. The sample names and the musical reasons behind the choices are in Table 6.2. One surprise was that some of the samples included soft MIDI drones that played the melody as a reference for the singer. This type of drone was nearly nonexistant in the "Intonation" dataset used to train the model.

I generated a baseline algorithm to address the question of whether a simple pitch correction technique could achieve comparable or superior results with significantly less time and resources. The baseline retains many the characteristics of the proposed model: it shifts each note by a constant to retain a natural sound, and assigns note boundaries where the pYIN pitch detection algorithm assigned silence. Its pitch corrections, however, are based on the equal-tempered scale, which simplifies it but also risk limiting its scope. For each note, it computes the median pitch from the pYIN—which should be the longest note's median in legato sections treated as a single note. It then shifts the note so that its new median is the nearest equal-tempered scale degree.

Subjective test samples from MIR-1K		
Sample name	**Start second**	**Description**
Ariel 1	0	Slightly off throughout; One prominent note is off by a semitone.
Ariel 2	23	Sharp throughout
Annar 4	3	In tune
Amy 7	27	An audible MIDI drone is playing the melody. Some notes are on pitch, others are off.
Davidson 4	6	Very low voice. Pitch is mostly close but slightly off. A very soft drone is playing the melody many octaves higher.
Bobon 4	2	Melody has some very long notes; Pitch is often off, usually flat, often by about 50 cents
Geniusturtle 4	9	Many notes are off, one by a semitone. There are many sustained notes and many jumps. A drone is playing the melody.
Stool 4	0	The pitch is off overall but never very far; The melody has many jumps; there is an audible MIDI drone
Fdps 4	17	Slow tempo, jumps in the melody, a couple notes are off by more than 50 cents
Jmzen 4	0	Very far off overall

Table 6.1: Description of the audio samples from MIR-1K used for the subjective listening test, and their diverse characteristics that test the program under varying conditions

Figure 6.5 shows the behavior of the baseline algorithm.

For both algorithms, the pitch shifts applied to the audio were computed using TD-PSOLA. The only difference between the outputs is the amount of shift. In a subjective test conducted in previous work, [49], I asked listeners who compared the original audio to the corrected audio which version sounded more natural. Listeners did not hear a difference between samples, which indicates that TD-PSOLA doesn't produce a significant amount of artifacts that would make it difficult to rate the quality of the algorithm only based on musical intonation instead of audio quality.

Listeners were amateur or professional-level musicians. Each listener was assigned four samples: two pairs comparing the baseline outputs to the original, and two comparing the corrected outputs to the original using the "Silence-HMM" model. The listeners did not know which algorithms were used in each pair. They were asked to indicate which performance they considered more accurate in terms of pitch, and could optionally leave comments. Tables 6.2 and 6.2 shows the results and anonymized comments for the baseline and corrected samples, respectively. N participants provided a total M responses.

Analysis

The results indicate at a qualitative level that the proposed approach was preferable to the baseline. In half of the samples, listeners quite confidently chose the corrected version over the original, and in some of the remaining ones the choice was difficult. This indicates that the proposed approach didn't tend to make the performances worse, even in cases where it failed to provide solid corrections. The fact that listeners preferred it half of the time over 20 seconds indicates that it didn't make serious errors too often.

It is striking to notice that many of the samples where the proposed approach was selected with confidence included a melody reference drone. As mentioned above, drones were nearly nonexistant in the "Intonation" dataset. The fact that the model performed well on these samples indicates that it might have learned how to effectively use reference pitches in the backing track,

Sample name	Score	Comments
Ariel 1	0-2	"Not easy to hear the difference. Both are mostly in tune"; "Difficult. Notes at 6 and 16 seconds are out of tune in both samples. The note between 9 and 11 might be better in the original. The original is better, neither is great."
Ariel 2	0-3	"Both mostly in tune"; "Very close, both pretty much in tune"
Annar 4	2-0	N/A
Amy 7	1-1	- "Baseline gets a mild preference. The original gets an overall score of 9/10"
Davidson 4	0-1	N/A
Bobon 4	0-2	"Both are pretty bad, but the original might be somewhat more in tune". "Baseline is so out of tune that it is difficult to figure out what the melody is, but one can perhaps make more sense of the corrected version"
Geniusturtle 7	2-0	N/A
Stool 4	0-2	"Strong preference for original. The baseline gets score 7/10"
Fdps 4	0-2	N/A
Jmzen 1	1-1	"Both are moderately out of tune, also the singer might not always hit the right note; now one's ears might be too tired"; "Very close, the original is better (e.g., on second 7)."

Table 6.2: Baseline versus original. Some listeners provided comments. These are summarized in this table, and randomized label names are replaced with the actual labels, unknown to the listeners.

Sample name	Score	Comments
Ariel 1	0-2	N/A
Ariel 2	0.5-0.5	“No preference”
Annar 4	0-1	N/A
Amy 7	1-0	N/A
Davidson 4	3-0	“Both are very accurate, very difficult to tell the difference”; “Both are pretty much in tune, a note at second 5 might be better in the corrected version”; “Strong preference for corrected version, the original gets overall score 9/10”
Bobon 4	1-1	N/A
Geniusturtle 7	3-0	“Corrected is clearly better”; “Corrected sounded clearly more in tune”; “Corrected is slightly better, original gets score 6/10”
Stool 4	3-0	“Corrected is clearly better”; “Notes at seconds 4 and 9 are better in corrected”
Fdps 4	1-1	“Original sounded clearly more in tune, starting from a pretty weak base in corrected version” “Both are moderately out of tune - one’s ear’s might be getting a bit tired at this point”
Jmzen 1	0-2	N/A

Table 6.3: Corrected versus original. Some listeners provided comments. These are summarized in this table, and randomized label names are replaced with the actual labels, unknown to the listeners.

and that the reference melody made the solution more clear. This would mean that the model is behaving as expected—basing its corrections on the backing track pitch content—instead of learning, for example, a pitch distribution or "scale", and mapping the vocals pitches to it while ignoring the backing track. This could also mean that the model depends on the pitch content being closely related to the melody.

The fact that the model without extension resulted in the histograms that most resembled the ground-truth data indicates that the extension layer was either useless for predictions or caused instability in training, despite pre-training of the feature extraction layers. One can hypothesize whether the broad temporal context provided by the extension layer is necessary for musical intonation, or whether pitch adjustments only require the local context, provided by the hidden state of the GRU, initialized with the last state of the previous note.

CHAPTER 7
CONCLUSION

In this thesis, I introduce a novel data-driven algorithm for estimating and applying pitch corrections in a monophonic vocal track, while using the backing track (also called accompaniment) as a reference. The deep neural network used to predict pitch corrections is exposed to both incorrect intonation, for which it learns a correction, and intentional pitch variation, which it learns to preserve. It represents pitch as a continuous value rather than a discrete set of notes. It does not rely on a musical score, thus allowing for improvisation and harmonization in the singing performance. Results on a convolutional neural network with a gated recurrent unit layer indicate that spectral information in the backing and vocal tracks is useful for determining the amount of pitch correction required at the note level.

The fact that musical pitch is not represented as a discrete set of notes also makes the algorithm applicable to musical traditions that differ from Western pop music with respect to the scales and pitch variation patterns used. The network in this thesis was trained on Western popular music, but could also be trained on music from other genres and cultures.

The amateur "Intonation" dataset used as ground truth in this thesis has a very different pitch distribution from that of a dataset built from professional performances. The results described in this thesis are thus prototypical in nature. A challenge for future work is to obtain professional-level data to permit more accurate prediction.

The current model is built on some strong assumptions. First, the backing track is assumed to have clearly identifiable pitches—a chord progression—that serve as a reference for the vocals. Second, a note's pitch is assumed to be correctable by shifting the full note by a constant, which minimizes changes to the original performance. The modifications preserve pitch nuances and timbre, ensuring a natural sounding result that preserves the singer's style. As these assumptions are relaxed in future work, the performance of the model should improve in the current context,

and the contexts to which the model can apply will expand. The model can also be developed to address other aspects of the performance, such as rhythm.

The algorithm now first predicts the amount by which singing should be shifted in pitch, then applies the shift in post processing. An objective for future work can be to extend the model so that it directly predicts the pitch-shifted signal in an end-to-end model.

Future work will benefit from collaboration with human-computer interaction designers, music theorists, and musicologists. A machine-learning-based algorithm can always make mistakes, and these mistakes can be addressed over time from the feedback that these collaborations will provide.

More generally, the work presented here contributes to the recent field of music information retrieval, which lies in the intersection of music and technology. The automatic pitch correction algorithm introduced here illustrates how recent technological advances can be used in the service of music. It also exemplifies the use of music technology to incorporate millenia of music theory into artistic expression and development.

Digital apps represent a primary way to interact actively with music, and they are often a first step for people getting started with making music. An app experience that is positive and is perceived as leading to musical growth may increase the probability that the user evolves to become a musician.

CHAPTER 8

GLOSSARY

CNN Convolutional Neural Network. 42, 43

CQT Constant-Q Transform. xiii, 40, 42, 43, 45, 46, 49, 51, 53, 55, 56

DNN Deep Neural Network. xiii, xiv, 4, 5, 6, 33, 35, 38, 39, 41, 42, 43, 45, 46, 47, 49, 50, 53

GRU Gated Recurrent Unit. xi, xiii, xiv, xv, 5, 6, 41, 43, 47, 49, 50, 56, 57, 70, 71, 81

HCQT Harmonic Constant-Q Transform. 42, 43

HMM Hidden Markov Model. xi, xiv, xv, 6, 44, 45, 48, 49, 54, 55, 57, 58, 71, 72, 73, 74, 75, 78

MIDI Musical Instrument Digital Interface. xi, xiv, xv, 27, 28, 36, 38, 41, 49, 54, 61, 62, 63, 65, 66, 67, 68, 69, 76

MIR Music Information Retrieval. 42, 44

MSE Mean-Squared Error. 51, 70, 71

NMF Non-negative Matrix Factorization. 33

pYIN Probabilistic YIN. xi, xiv, 44, 45, 47, 48, 54, 57, 58, 68, 72, 76

ReLU Rectified Linear Unit. 49

SIFT Scale-Invariant Feature Transform. 31

STFT Short-Time Fourier Transform. 35, 60

TD-PSOLA Time-Domain Pitch-Synchronous Overlap and Add. 40, 45, 51, 53, 78

REFERENCES

[1] Antares Audio Technologies. *Auto-Tune Pro: Manual*. https://www.antarestech.com/mediafiles/documentation_records/10_Auto-Tune_Live_Manual.pdf. [Online; Accessed June 10th, 2020]. 2018.

[2] S. Wager et al. "Intonation: A Dataset of Quality Vocal Performances Refined by Spectral Clustering on Pitch Congruence". In: *Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019.

[3] L. R. Rabiner. "Readings in Speech Recognition". In: 1990. Chap. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, pp. 267–296.

[4] J. Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *NeurIPS Workshop Deep Learning*. 2014, pp. 166–170.

[5] Jeffrey Hass. *Introduction to Computer Music: Vol. 1*. https://cmtext.indiana.edu/toc.php. [Online; Accessed February 13th, 2021]. 2019.

[6] R. Parncutt and G. Hair. "A Psychocultural Theory of Musical Interval: Bye Bye Pythagoras". In: *Music Perception: An Interdisciplinary Journal* 35.4 (2018), pp. 475–501.

[7] F. Villano. *Victor Wooten's Music and Nature Camps*. https://makingmusicmag.com/victor-wootens/. [Online; Accessed June 14th, 2020].

[8] F. Gafori. *Theorica Musice Franchini Gafuri Laudensis ([Reprod.])* https://gallica.bnf.fr/ark:/12148/bpt6k58171q.f36. [Online; Accessed February 13th, 2021]. 1492.

[9] Mount Salvation Salton Sea (Pinterest). *Monochord*. https://www.pinterest.com/pin/786300416163830080/. [Online; Accessed June 30th, 2020].

[10] P. Weiss and R. Taruskin. *Music in the Western World*. Cengage Learning, 2007.

[11] D. P. Goldman. “A New Look at Zarlino’s Theory and its Effect on his Counterpoint Doctrine”. In: *Theory and Practice* 16 (1991), pp. 163–177.

[12] The Editors of Encyclopaedia Britannica. *Equal Temperament*. https://www.britannica.com/art/equal-temperament. [Online; Accessed June 11th, 2020]. 2019.

[13] J. Devaney, J. Wild, and I. Fujinaga. “Intonation in Solo Vocal Performance: A Study of Semitone and Whole Tone Tuning in Undergraduate and Professional Sopranos”. In: *Proc. Int. Symp. Performance Science*. 2011, pp. 219–224.

[14] J. P. Burkholder, D. J. Grout, and C. V. Palisca. *A History of Western Music: 8th Edition*. WW Norton & Company, 2019.

[15] R. Ramanna. “The Structure of Raga Music”. In: *Current Science* 68.9 (1995), pp. 897–916.

[16] W. J. Arnold. “L’Intonation Juste dans la Théorie Ancienne de l’Inde: Ses Applications aux Musiques Modale et Harmonique”. In: *Revue de Musicologie* (1985), pp. 11–38.

[17] The Editors of Encyclopaedia Britannica. *Blues*. https://www.britannica.com/art/blues-music. [Online; Accessed July 1st, 2020]. 2020.

[18] R. Palmer. *Deep Blues*. Penguin Group USA, 1981.

[19] G. Alper. “How the Flexibility of the Twelve-bar Blues Has Helped Shape the Jazz Language”. In: *College Music Symposium*. Vol. 45. 2005, pp. 1–12.

[20] Lessley Anderson and Karen Sparks. *Auto-Tune*. https://www.britannica.com/topic/Auto-Tune-1996668. [Online; Accessed June 6th, 2020]. 2014.

[21] A. Peres. “The Sonic Dimension as Dramatic Driver in 21st-Century Pop Music.” PhD thesis. 2016.

[22] G. Eckard. *How an Oil Engineer Created Auto-Tune and Changed Music Forever*. https://www.vice.com/en_us/article/bmaj4d/how-an-oil-engineer-created-auto-tune-and-changed-music-forever-interview-creator. [Online; Accessed February 13th, 2021]. Feb. 25, 2016.

[23] *Smule blog: Tuesday Tips–What Effects Should I Use*. https://blog.smule.com/tuesday-tips-what-effects-should-i-use. [Online; Accessed July 07, 2020].

[24] R. Katz. *Artists Should Stay Acoustic, Auto-Tune Too Often Artificial and Overused*. https://theblackandwhite.net/36566/opinion/blogs/artists-should-stay-acoustic-auto-tune-too-often-artificial-and-overused/. [Online; Accessed June 17th, 2020]. Feb. 28, 2014.

[25] Jaron Lanier. *You Are Not a Gadget: A Manifesto*. Vintage, 2010, pp. 7–8.

[26] S. B. Bacharach. "Organizational Theories: Some Criteria for Evaluation". In: *Academy of Management Review* 14.4 (1989), pp. 496–515.

[27] R. Frigg and S. Hartmann. "Models in Science". In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. [Online; Accessed February 13th, 2021]. Metaphysics Research Lab, Stanford University, 2020.

[28] C. Z. Elgin. *True Enough*. MIT Press, 2017.

[29] J. Devaney. "New Metrics for Evaluating the Accuracy of Fundamental Frequency Estimation Approaches in Musical Signals". In: *Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 531–535.

[30] L. Zheng, Y. Yang, and Q. Tian. "SIFT Meets CNN: A Decade Survey of Instance Retrieval". In: *Trans. Pattern Analysis and Machine Intelligence*. Vol. 40. 5. IEEE, 2017, pp. 1224–1244.

[31] J. Pennington, R. Socher, and C. Manning. "Glove: Global Vectors for Word Representation". In: *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543.

[32] T. Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *Int. Conf. Learning Representations, (ICLR)*. 2013.

[33] C. Raphael. “Music Plus One and Machine Learning”. In: *Proc. Int. Conf. Machine Learning (ICML)*. 2010, pp. 21–28.

[34] D. D. Lee and H. S. Seung. “Algorithms for Non-negative Matrix Factorization”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2001, pp. 556–562.

[35] C. Molnar. “Interpretable Machine Learning”. In: `https://christophm.github.io/interpretable-ml-book/`. Lulu Press, Inc., 2019. Chap. Importance of Interpretability.

[36] H. Jiang and O. Nachum. “Identifying and Correcting Label Bias in Machine Learning”. In: *AISTATS*. 2020.

[37] S. Wager. *Machine Learning for Causal Inference*. Presented at California Center for Population Research Seminar, UCLA. 2019.

[38] M. Raghu et al. “On the Expressive Power of Deep Neural Networks”. In: *Proc. Int. Conf. Machine Learning (ICML)*. Vol. 70. JMLR. 2017, pp. 2847–2854.

[39] Y. Luo and N. Mesgarani. “Conv-TasNet: Surpassing Ideal Time–frequency Magnitude Masking for Speech Separation”. In: *Trans. Audio, Speech, and Language Processing (TASLP)*. Vol. 27. 8. IEEE, 2019, pp. 1256–1266.

[40] C. Raphael. *Lecture Note: Music Information Processing: Audio*. `http://www.music.informatics.indiana.edu/courses/I547/notes_I547.pdf`. [Online; Accessed February 13th, 2021].

[41] J. Engel et al. “DDSP: Differentiable Digital Signal Processing”. In: *Int. Conf. Learning Representations*. 2019.

[42] K. He et al. “Delving Deep into Rectifiers: Surpassing Human-level Performance on Imagenet Classification”. In: *Proc. IEEE Int. Conf. Computer Vision (ICCV)*. 2015, pp. 1026–1034.

[43] Z. Qin et al. “How Convolutional Neural Networks See the World - A Survey of Convolutional Neural Network Visualization Methods”. In: *Mathematical Foundations of Computing* 1 (2018), pp. 149–180.

[44] A. B. Patel, M. T. Nguyen, and R. Baraniuk. “A Probabilistic Framework for Deep Learning”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2016, pp. 2558–2566.

[45] A. Verma et al. “Programmatically Interpretable reinforcement learning”. In: *Proc. Int. Conf. Machine Learning (ICML)*. 2018, pp. 5045–5054.

[46] A. van den Oord et al. “WaveNet: A Generative Model for Raw Audio”. In: *ISCA Speech Synthesis Workshop (SSW)*. 2016, pp. 125–125.

[47] R. M. Bittner et al. “Deep Salience Representations for f_0 Estimation in Polyphonic Music”. In: *Int. Society for Music Information Retrieval Conf. (ISMIR)*. 2017, pp. 63–70.

[48] J. Su, Z. Jin, and A. Finkelstein. “HiFi-GAN: High-Fidelity Denoising and Dereverberation Based on Speech Deep Features in Adversarial Networks”. In: *INTERSPEECH*. 2020.

[49] S. Wager et al. “Deep Autotuner: A Pitch Correcting Network for Singing Performances”. In: *Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 246–250.

[50] A. Paszke et al. *Pytorch: An Imperative Style, High-Performance Deep Learning Library*. [arXiv preprint arXiv:1912.01703]. 2019.

[51] F. Charpentier and M. Stella. “Diphone Synthesis Using an Overlap-add Technique for Speech Waveforms Concatenation”. In: *Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*. Vol. 11. IEEE. 1986, pp. 2015–2018.

[52] Antares Audio Technologies. *Auto-Tune Live: Owner’s Manual*. `https://www.antarestech.com/mediafiles/documentation_records/10_Auto-Tune_Live_Manual.pdf`. [Online; Accessed March 20th, 2018]. 2016.

[53] S. Salazar et al. *Continuous Score-coded Pitch Correction*. US Patent 9,147,385. Sept. 2015.

[54] Y. Luo et al. “Singing Voice Correction Using Canonical Time Warping”. In: *Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 156–160.

[55] S. Yong and J. Nam. “Singing Expression Transfer from One Voice to Another for a Given Song”. In: *Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 151–155.

[56] E. Gómez et al. *Deep Learning for Singing Processing: Achievements, Challenges and Impact on Singers and Listeners*. [arXiv preprint arXiv:1807.03046]. 2018.

[57] K. Cho et al. “Learning Phrase Representations Using RNN Encoder-decoder for Statistical Machine Translation”. In: *Conf. Empirical Methods in Natural Language Processing (EMNLP)*. 2014.

[58] D. Basaran, S. Essid, and G. Peeters. “Main Melody Extraction with Source-filter NMF and CRNN”. In: *Int. Society for Music Information Retrieval Conf. (ISMIR)*. 2018.

[59] S. Wager et al. “Fully Learnable Front-End for Multi-Channel Acoustic Modeling using Semi-Supervised Learning”. In: *Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2020, pp. 6864–6868.

[60] I. Goodfellow et al. “Generative Adversarial Nets”. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014, pp. 2672–2680.

[61] M. Mauch and S. Dixon. “pYIN: A Fundamental Frequency Estimator Using Probabilistic Threshold Distributions”. In: *Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pp. 659–663.

[62] Jong Wook Kim et al. “CREPE: A Convolutional Representation for Pitch Estimation”. In: *Int. Conf. Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018.

[63] A. De Cheveigné and H. Kawahara. “YIN, a Fundamental Frequency Estimator for Speech and Music”. In: *The Journal of the Acoustical Society of America* 111.4 (2002), pp. 1917–1930.

[64] B. McFee et al. “LibROSA: Audio and Music Signal Analysis in Python”. In: *Proc. Python in Science Conf.* 2015.

[65] C. Cannam, C. Landone, and M. Sandler. “Sonic Visualiser: An Open-source Application for Viewing, Analysing, and Annotating Music Audio Files”. In: *Proc. Int. Conf. Multimedia*. ACM. 2010, pp. 1467–1468.

[66] W. Hsu, Y. Zhang, and J. Glass. “Learning Latent Representations for Speech Generation and Transformation”. In: *INTERSPEECH*. 2017.

[67] C. Hsu and J. R. Jang. “On the Improvement of Singing Voice Separation for Monaural Recordings Using the MIR-1K Dataset”. In: *Trans. Audio, Speech, and Language Processing (TASLP)*. Vol. 18. 2. IEEE, 2009, pp. 310–319.

[68] M. Sezgin and B. S. “Survey over Image Thresholding Techniques and Quantitative Performance Evaluation”. In: *Journal of Electronic imaging* 13.1 (2004), pp. 146–166.

[69] D. P. Kingma and J. Ba. *Adam: A Method for Stochastic Optimization*. [Computing Research Repository (CoRR), abs/1412.6980]. 2015.

[70] R. Pascanu, T. Mikolov, and Y. Bengio. “On the Difficulty of Training Recurrent Neural Networks”. In: *Int. Conf. Machine Learning*. 2013, pp. 1310–1318.

[71] A. Mesaros, T. Heittola, and T. Virtanen. “A Multi-device Dataset for Urban Acoustic Scene Classification”. In: *Proc. Detection and Classification of Acoustic Scenes and Events Workshop*. 2018, pp. 9–13.

[72] A. Liutkus et al. “The 2016 Signal Separation Evaluation Campaign”. In: *Int. Conf. Latent Variable Analysis and Signal Separation (LVA/ICA)*. Grenoble, France, 2017, pp. 323–332.

[73] T. Bertin-Mahieux et al. “The Million Song Dataset”. In: *Int. Society for Music Information Retrieval Conf. (ISMIR)*. Vol. 2. 9. 2011, p. 10.

[74] E. Nichols et al. “Automatically Discovering Talented Musicians with Acoustic Analysis of YouTube videos”. In: *Int. Conf. Data Mining (ICDM)*. IEEE. 2012, pp. 559–565.

[75] K.A. Lim and C. Raphael. “Intune: A System to Support an Instrumentalist’s Visualization of Intonation”. In: *Computer Music Journal* 34.3 (2010), pp. 45–55.

[76] M. Lucińska and S.T. Wierzchoń. “Spectral Clustering Based on K-Nearest Neighbor Graph”. In: *IFIP Int. Conf. Computer Information Systems and Industrial Management*. Springer. 2012, pp. 254–265.

[77] M.E.J. Newman. “Modularity and Community Structure in Networks”. In: vol. 103. 23. National Academy of Sciences, 2006, pp. 8577–8582.

[78] D.J. Berndt and J. Clifford. “Using Dynamic Time Warping to Find Patterns in Time Series”. In: *KDD Workshop*. Vol. 10. 16. 1994, pp. 359–370.

[79] M. Müller. “Music Synchronization”. In: *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*. Berlin, Heidelberg: Springer, 2015, pp. 131–141.

[80] B. Efron and R. J. Tibshirani. *An Introduction to the Bootstrap*. CRC Press, 1994.

[81] P. Smaragdis and G.J. Mysore. “Separation by “Humming”: User-guided Sound Extraction from Monophonic Mixtures”. In: *Workshop Applications of Signal Processing to Audio and Acoustics (WASPAA)*. IEEE. 2009, pp. 69–72.

[82] A. Liutkus et al. “Informed Audio Source Separation: A Comparative Study”. In: *Proc. European Signal Processing Conf. (EUSIPCO)*. IEEE. 2012, pp. 2397–2401.

[83] A. Huq, M. Cartwright, and B. Pardo. “Crowdsourcing a Real-world Online Query by Humming System”. In: *Proc. Sound and Music Computing Conf. (SMC)*. 2010.

[84] A. Liutkus et al. “The 2016 Signal Separation Evaluation Campaign”. In: *Proc. Int. Conv. Latent Variable Analysis and Signal Separation (LVA/ICA), Liberec, Czech Republic*. Cham: Springer International Publishing, 2017, pp. 323–332.

Sanna Wager

Education

2014-2021 Ph.D. in Informatics, Concentration: Music Informatics, Indiana University Bloomington (USA).
2014-2018 M.S. in Computer Science, Indiana University Bloomington (USA).
2009-2014 B.M. in Bassoon, Minor: Mathematics, Indiana University Bloomington (USA).
2011-2012 Studies in Bassoon, Hochschule Luzern Musik (Switzerland).
2006-2009 Maturité Fédérale, Gymnase Auguste Piccard (Switzerland).

Professional Experience

2020- Applied Scientist, Audio and Music, Lab126, Amazon, Inc.
2019-2020 Research Scientist Intern, Music Intelligence Group, Spotify S.A.
2019 Applied Scientist Intern, Alexa Speech, Amazon, Inc.
2018 Audio Engineering Intern, Smule, Inc.
2017 Software Engineering Intern, Android Audio, Google, Inc.
2014-2016 Teaching Assistant, Informatics and Computing, Indiana University Bloomington.

Publications and talks

publ. SW, George Tzanetakis, Cheng-i Wang & Minje Kim. **Deep Autotuner: A Pitch Correcting Network for Singing Performances**. *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.
SW, Aparna Khare, Minhua Wu & Shiva Sundaram. **Fully Learnable Front-End for Multi-Channel Acoustic Modeling Using Semi-Supervised Learning**. *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, 2020.
SW, George Tzanetakis, Stefan Sullivan, Cheng-i Wang, John Shimmin, Minje Kim & Perry Cook. **Intonation: a Dataset of Quality Vocal Performances Refined by Spectral Clustering on Pitch Congruence**. *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, 2019.
SW & Minje Kim. **Collaborative Speech Dereverberation: Regularized Tensor Factorization for Crowdsourced Multi-Channel Recordings**. *Proc. IEEE European Signal Processing Conf. (EUSIPCO)*, 2018.
SW, Liang Chen, Minje Kim & Christopher Raphael. **Towards Expressive Instrument Synthesis Through Smooth Frame-by-Frame Reconstruction: From String to Woodwind**. *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, 2017.

talks Kasten, Glenn & SW. **Learning the Pulse: Statistical and ML Analysis of Real-Time Audio Performance Logs**. *Audio Developer Conference, London, UK*, 2017.
Miksza, Peter, Kevin Watson, Kai Zhen & SW. **Relationships Between Experts Subjective Ratings of Jazz Improvisations and Computational Measures of Melodic Entropy. In Data Analysis Phase** *Improvising Brain III: Cultural Variation and Analytical Techniques Symposium, Atlanta, GA*, 2017.
Christopher Raphael & SW. **Example Application of Approximate Principal Component Analysis to Reconstruction of Low-Quality or De-Soloed Recordings by Imputing Spectrum Components to the Audio** in: Daniel McDonald. **Approximate Principal Components Analysis of Large Data Sets**. *Department of Statistics, Indiana University Bloomington*, 2014.