Guitar Tablature Transcription using a Deep Belief Network

by

Gregory D. Burlet

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

Music transcription is the process of extracting the pitch and timing of notes that occur in an audio recording and writing the results as a music score, commonly referred to as *sheet music*. Manually transcribing audio recordings is a difficult and time-consuming process, even for experienced musicians. In response, several algorithms have been proposed to automatically analyze and transcribe the notes sounding in an audio recording; however, these algorithms are often general-purpose, attempting to process any number of instruments producing any number of notes sounding simultaneously.

This work presents a transcription algorithm that is constrained to processing the audio output of a single instrument, specifically an acoustic guitar. The transcription system consists of a novel note pitch estimation algorithm that uses a deep belief network and multi-label learning techniques to generate multiple pitch estimates for each segment of the input audio signal. Using a compiled dataset of synthesized guitar recordings for evaluation, the algorithm described in this work results in a 12% increase in the f-measure of note transcriptions relative to a state-of-the-art algorithm in the literature. This thesis demonstrates the effectiveness of deep, multi-label learning for the task of guitar audio transcription.

I am a great believer in luck.
The harder I work, the more of it I seem to have.

—Coleman Cox, 1922.

Acknowledgements

I would like to acknowledge and express my gratitude to those individuals who have uploaded manual tablature transcriptions to the Ultimate Guitar website. In the interest of space these people will go unnamed; however, their contribution to the datasets compiled in this thesis is substantial.

I have been fortunate to complete this work in the company of extremely bright individuals who work in the Software Engineering Research Lab. Their feedback and brainstorming provided much insight into the problems encountered in this work.

Special thanks are owed to my supervisor, Abram Hindle, whose passion for audio rivals my own. His advice and guidance throughout the course of this work has been invaluable.

This work could not have been completed without the love and encouragement of my fiancée, the unwavering support of my parents, and my coffee maker.

This research was supported by an Alberta Innovates Graduate Student Scholarship provided by Alberta Innovates Technology Futures and a Graduate Student Scholarship provided by the Government of Alberta.

Table of Contents

List of Tables

List of Figures

List of Acronyms

DBN	deep belief network
DFT	discrete Fourier transform
GPU	graphics processing unit
HMM	hidden Markov model
MFCC	Mel frequency cepstral coefficient
MIDI	musical instrument digital interface
MIR	music information retrieval
MIREX	music information retrieval evaluation exchange
NMF	non-negative matrix factorization
RBM	restricted Boltzmann machine
RTFI	resonator time-frequency image
STFT	short time Fourier transform
SVM	support vector machine
XML	extensible markup language

Chapter 1

Introduction

The task of music transcription involves the transformation of an audio signal into a music score that informs a musician which notes to perform and how they are to be performed. This is accomplished through the analysis of the pitch and rhythmic properties of an acoustical waveform. In the composition or publishing process, manually transcribing each note of a musical passage to create a music score for other musicians is a labour-intensive procedure [40]. Manual transcription is slow and error-prone: even notationally fluent and experienced musicians make mistakes, require multiple passes over the audio signal, and draw upon extensive prior knowledge to make complex decisions about the resulting transcription [7].

In response to the time-consuming process of manually transcribing music, researchers in the multidisciplinary field of music information retrieval (MIR) have summoned their knowledge of computing science, electrical engineering, music theory, mathematics, and statistics to develop algorithms that aim to automatically transcribe the notes sounding in an audio recording. The problem of automatic music transcription is often characterized as the "holy grail in the field of music signal

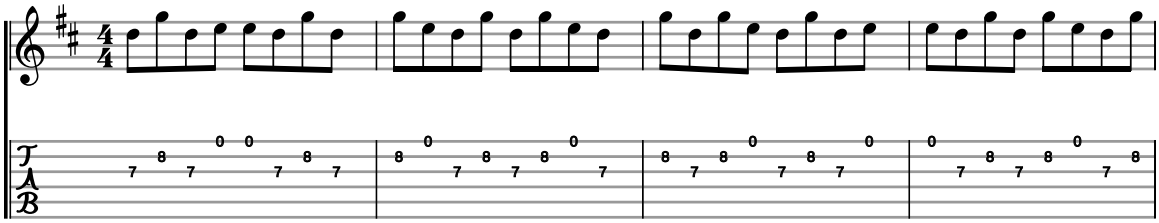
analysis" [8]. The difficulty of the problem increases as more instruments and notes sound simultaneously.

Although the automatic transcription of monophonic (one note sounding at a time) music is considered a solved problem [8], the automatic transcription of polyphonic (multiple notes sounding simultaneously) music "falls clearly behind skilled human musicians in accuracy and flexibility" [55]. In an effort to reduce the complexity, the transcription problem is often constrained by limiting the number of notes that sound simultaneously, the genre of music being analyzed, or the number and type of instruments producing sound. A constrained domain allows the transcription system to "exploit the structure" [68] and consequently reduce the difficulty of transcription. This parallels systems in the more mature field of speech recognition where practical algorithms are often language, gender, or speaker dependent [47].

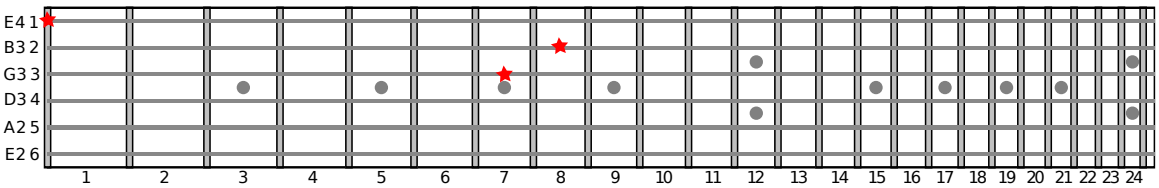
Automatic guitar transcription is the problem of automatic music transcription with the constraint that the audio signal being analyzed is produced by a single electric or acoustic guitar. Though this problem is constrained, a guitar is capable of producing chords of six notes simultaneously, which still offers a multitude of challenges for modern transcription algorithms. The most notable challenge is the estimation of the pitches of notes comprising highly polyphonic chords, occurring when a guitarist strums several strings at once.

Yet another challenge presented to guitar transcription algorithms is that a large body of guitarists publish and share transcriptions in the form of tablature rather than common Western music notation. Therefore, automatic guitar transcription algorithms should also be capable of producing tablature. An example of guitar tablature below its corresponding common Western music notation is presented in Figure 1.1a. Guitar tablature is a symbolic music notation system with a six-line staff representing the strings on a guitar. The top line of the system represents the

highest pitched (smallest diameter) string and the bottom line represents the lowest pitched (highest diameter) string, as shown in Figure 1.1b. A number on a line denotes the guitar fret that should be depressed on the respective string.[1] Modern tablature notation does not explicitly display rhythmic information such as the duration of notes; however, rhythmic information is often implicitly communicated using the relative spacing between symbols on the staff. Moreover, guitar tablature does not explicitly display the pitch of notes, but rather describes the gestures to apply to the guitar to produce a certain pitch. As an additional note, guitar transcriptions in common music notation are performed an octave lower than what is written.



(a) A system of modern guitar tablature for the song "Weird Fishes" by Radiohead, complete with common Western music notation above.



(b) A standard guitar fretboard. Stars denote the string and fret combinations required to perform the notes in the above music score.

Figure 1.1: An example system of music displayed in both tablature and common Western music notation, complete with an illustration of how the tablature is realized on a guitar fretboard.

Arranging tablature is challenging because the guitar is capable of producing the same pitch in multiple ways. Therefore, a "good" arrangement is one that is *biomechanically easy* for the musician to perform, such that transitions between notes

[1] Refer to Appendix A for definitions of common guitar terminology.

do not require excessive hand movement and the performance of chords require minimal stretching of the hand [42].

The development of an algorithm that is capable of generating reliable tablature transcriptions from guitar recordings would have an impact on both academia and industry. In academia, a solution to the constrained problem of automatic guitar transcription would offer a template algorithm that could be applied to instruments other than guitar. An algorithm that performs well on this task would likely perform well on related tasks in the field of MIR such as query by humming, melody transcription, chord recognition, and audio fingerprinting. A solution to this constrained problem would also provide a foundation for the solution to the unconstrained problem of automatic music transcription. Should future research design an algorithm that is capable of reliably separating the audio signals produced by individual instruments amongst a sound scape of multiple instruments performing simultaneously—a problem called *sound source separation*, which mimics the natural phenomenon known as the *cocktail party effect* [25]—the transcriptions of individual instruments can be combined to solve the larger problem at hand. In industry, a solution to the problem of automatic guitar tablature transcription is immediately commercializable in the form of a transcription application. Existing applications include *Melodyne* [23], musical video games such as *Rocksmith* [114], *WildChords* [124], *GuitarBots* [123], or educational guitar lesson applications such as *Yousician* [125] that offer feedback to the guitarist in training. Regarding existing guitar education software, these solutions involve knowing what notes should be performed prior to receiving an audio signal, thus simplifying the transcription problem.

The fact that researchers have been searching for a solution to the problem of automatic music transcription for the last four decades lends credence to the difficulty of this task. Although an accurate and robust transcription algorithm has yet to

be discovered, incremental gains in transcription accuracy have been reported over the last several decades [7]. Recently, however, the MIR research community has collectively reached a plateau in automatic music transcription system accuracy. In a paper addressing this issue, Benetos et al. [8] stress the importance of extracting expressive audio features and moving towards context-specific transcription systems. Also addressing this issue, Humphrey et al. [49, 48] propose that effort should be focused on audio features generated by deep neural networks instead of hand-engineered audio features, due to the success of these methods in other fields such as computer vision [61]. The aforementioned literature provides motivation for investigating the viability of applying deep machine learning architectures to the problem of automatic guitar transcription.

1.1 Project Overview

With these motivations in mind, the objective of this thesis is to develop an algorithm for automatic guitar tablature transcription using a deep belief network and evaluate not only the quality of transcriptions produced, but also the benefits and detriments to using such an algorithm for transcription. The hypothesis is that, analogous to their performance in other fields, deep belief networks will outperform other proposed machine learning or digital signal processing transcription algorithms.

To this end, an automatic guitar tablature transcription algorithm has been implemented. A deep belief network has been trained to transform features of audio signal segments to a list of pitches sounding during the audio waveform. A series of HMMs then postprocess the list of pitches to determine the most probable onset (starting time) and offset (ending time) of each note in the audio recording. An onset detection algorithm is then used with a finer-grained time resolution to adjust the time

of note onsets. Common Western music notation is generated from the note estimates and performable guitar tablature is arranged using a graph-search algorithm.

Another contribution of this thesis is the compilation of a new ground-truth dataset for training and evaluating guitar tablature transcription algorithms. The dataset consists of guitar recordings that are synthesized from crowdsourced, manually transcribed guitar tablature collected from `www.ultimate-guitar.com`. The dataset has been made publicly available to encourage future research on automatic guitar transcription. [2]

1.2 Thesis Organization

This thesis is organized as follows: the following chapter provides a detailed literature review of automatic music transcription and guitar tablature arrangement algorithms. Chapter 3 presents the developed guitar tablature transcription algorithm. This is followed by Chapter 4, which outlines the hypotheses and experiments conducted to quantitatively evaluate the proposed guitar transcription algorithm. The results of this evaluation are presented and discussed in Chapter 5. Chapter 6 outlines possible threats to validity of the conducted research. Finally, Chapter 7 concludes the work with a discussion regarding the merits of the proposed algorithm and outlines future research directions.

[2] `https://archive.org/details/DeepLearningIsolatedGuitarTranscriptions`

Chapter 2

Literature Review

Automatic guitar transcription algorithms aim to extract the notes being performed by a guitar in an audio recording and write the notes in a symbolic music notation such as common Western music notation or tablature notation. Formally, the process of automatic guitar tablature transcription can be defined as a function that accepts an audio recording of a guitar as input and outputs symbolic music notation. In more detail, this function consists of two composite functions: a polyphonic transcription function, which converts an audio recording of a guitar to a list of note events, and a guitar tablature arrangement function, which converts the list of note events to a sequence of string and fret combinations written as tablature. In even more detail, the polyphonic transcription function can be further decomposed into two composite functions: pitch estimation, which estimates the pitches of note events, and temporal estimation, which estimates the onset (start) times and offset (end) times of notes. This workflow is summarized in Figure 2.1. Though the majority of transcription algorithms perform pitch estimation before temporal estimation, some work has explored the estimation of note temporal attributes before pitch [22, 37].

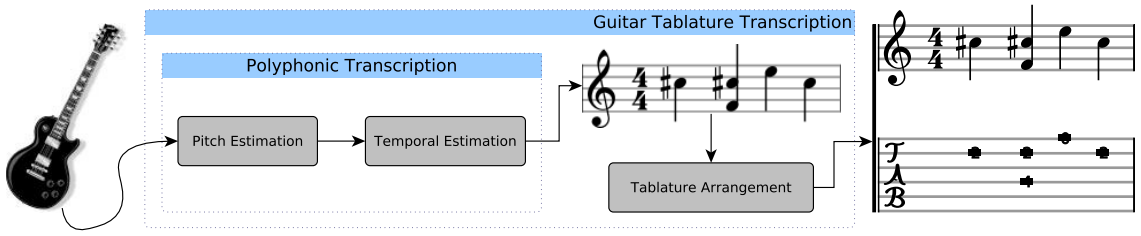


Figure 2.1: A high-level overview of the components of a guitar tablature transcription algorithm.

This chapter will review the fundamental concepts and algorithms that are used for the task of automatic guitar tablature transcription. The structure of this literature review resembles the workflow presented in Figure 2.1: Section 2.2 describes algorithms used for estimating the pitch, onset time, and duration of note events in an audio recording. Section 2.3 provides an overview of proposed methods to arrange tablature from a sequence of note events. Section 2.4 provides a detailed overview of the inner-workings of deep belief networks, followed by a review of multi-label learning algorithms in Section 2.5.

2.1 Digital Audio Preface

Before discussing algorithms used for the task of music transcription, several concepts in the field of digital signal processing, acoustics, and psychoacoustics must be reviewed. These terms include *frequency*, *fundamental frequency*, *harmonics*, *pitch*, *onset time*, *offset time*, *low-pass filter*, *high-pass filter*, *band-pass filter*, *spectrogram*, and finally, signal *downsampling* and *decimation*.

The first term to discuss is *frequency*. Frequency is a property of a signal that describes how quickly it oscillates and is measured in cycles per second (Hz). Building upon this definition, *fundamental frequency* f_0 is defined as the lowest frequency of a periodic waveform and is measured in cycles per second (Hz). Fundamental frequency

is the reciprocal of the fundamental period T_0, which measures the elapsed time between cycles of an audio waveform. Further building upon this definition is the concept of a *harmonic*. Harmonics f_k occur at integer multiples of the fundamental frequency, such that $f_k = kf_0, \forall\ k \in \mathbb{N}$.

The fundamental frequency and *pitch* of a note are interrelated. Pitch is the perceptual interpretation of frequency. Pitch is an attribute of sound that humans try to associate with all incoming acoustical signals [70]. A high frequency sound is perceived as a high pitch and a low frequency sound is perceived as a low pitch. In Western music notation, the nomenclature for pitch is a letter from A–G, optionally followed by an accidental (sharp or flat), that is then followed by an octave number. An *octave* consists of 12 pitches, called *semitones*. A pitch that is one octave higher than another is double the frequency of its counterpart. Following an equal-tempered scale, the pitch $A4$ is assigned a frequency of 440Hz and taking k steps in pitch away from $A4$ results in a frequency of $440 \cdot 2^{k/12}$Hz. Specifically, equal temperament refers to a system of tuning where consecutive pitches are separated by the same interval. To gain perspective, the lowest pitch on an 88-key piano is $A0$ (27.5Hz) and the highest pitch is $C8$ (4186Hz). The lowest pitch of a 24-fret guitar in standard tuning is $E2$ (164.8Hz) and the highest pitch is $E6$ (1318.5Hz). There are several other guitar tunings that affect the range of pitches capable of being performed by a guitar that will not be discussed here.

Pitch is not the only attribute of a note event. A note event also has temporal attributes such as *onset time* and *offset time*. The onset time of a note event marks the moment that the note begins, also called the *attack phase* of the note. Similarly, the offset time of a note event marks the moment that the note can no longer be heard. The duration of a note event is the difference between the offset time and the onset time.

Note events may occur alone, in which case they are simply called *notes*. A musical passage consisting of several notes in succession is called *monophonic*. Note events may also occur simultaneously, in which case they are called *chords*. A musical passage containing chords is called *polyphonic* if there is rhythmic independence between parts, or *homophonic* if there is no rhythmic independence between parts. In terms of music theory, the term *polyphony* refers to music with two or more simultaneous lines in counterpoint, which outlines chords. For the purposes of this thesis, however, the term *polyphony level* will refer to the number of notes that occur simultaneously. Polyphonic audio signals are the result of more than one instrument playing notes at the same time or a single instrument playing a chord consisting of several notes sounding simultaneously.

A typical audio recording consists of several notes that span a wide range of frequencies. Filters may be applied to the audio signal in order to modify the frequency content of the signal. A *low-pass filter* allows frequencies below a threshold to pass while attenuating frequencies above the threshold. Conversely, a *high-pass filter* allows frequencies above a threshold to pass while attenuating frequencies below the threshold. Combining these concepts, a *band-pass filter* defines a range of frequencies that are allowed to pass and attenuates all other frequencies.

The *spectrogram* of an audio signal is a standard digital signal processing analysis tool that is used to transform a signal from the time domain to the frequency domain. The spectrogram, often referred to as the short time Fourier transform (STFT), partitions the input audio signal into sequential "frames" or "windows" of audio samples, which are then converted from the time domain to the frequency domain using the discrete Fourier transform (DFT). The purpose of this transformation is to compute the magnitude of each frequency component of the audio signal.

Another common digital signal processing concept is *downsampling*. Downsampling is the process of removing samples from a digital audio signal. For example, if a signal is downsampled by a factor of four, the new signal is composed of every fourth sample of the original signal. This process directly affects the *sampling rate* of the digital audio signal, which describes the number of times the corresponding analog signal is sampled per second.

A more robust method of downsampling a signal is *decimation*. Decimation involves two steps: low-pass filtering followed by downsampling. The purpose of the low-pass filter, which allows low frequencies to pass up to a cut-off frequency, is to remove the effects of aliasing that occur when downsampling. According to the Nyquist-Shannon sampling theorem, the highest frequency encoded by a digital signal with sampling rate f_s Hz is $f_s/2$ Hz, often referred to as the *Nyquist frequency*. Therefore, when downsampling the audio signal by a factor of $m \in \mathbb{N}^+$, the cut-off frequency for the low-pass filter is set to $f_s/2m$ Hz. Downsampling involves taking every m^{th} sample of the original signal to form the new signal.

2.2 Polyphonic Transcription

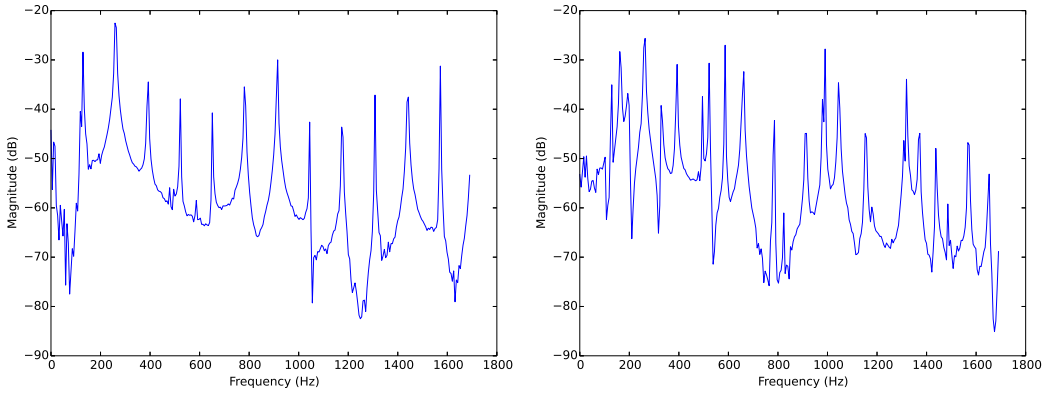
Automatic music transcription algorithms vary in complexity. Certain algorithms are only capable of transcribing monophonic musical passages in which one instrument plays a single note at a time, but perform with high accuracy [26, 64, 106]. Increasing in complexity, polyphonic transcription algorithms attempt to transcribe notes from an audio signal in which several notes sound simultaneously. Although monophonic transcription is a subproblem of polyphonic transcription since the transcription of a chord involves the transcription of its embedded notes, monophonic transcription algorithms use simpler and more robust analysis techniques that exploit the property

that only one note sounds at a time [80]. Therefore, polyphonic transcription algorithms must employ more complex audio analysis algorithms to accommodate more complex polyphonic audio signals.

The definition of harmonics provides insight into why the problem of polyphonic transcription is significantly more difficult than monophonic transcription. Recall the definition of *harmonics*, described in Section 2.1, which are frequencies that occur at integer multiples of the fundamental frequency. When a maximum of one note sounds at a time, the fundamental frequency of the note can easily be estimated by observing the evenly spaced frequencies of the harmonics in the audio waveform and selecting the lowest. When multiple notes sound at once, the harmonics of each fundamental frequency overlap and it is difficult to attribute a harmonic to a specific note and uncover the underlying fundamental frequencies.

To visualize this phenomenon, one can observe the *magnitude spectrum* of an audio signal containing a single note versus a mixture of notes. According to Joseph Fourier's theory, any waveform can be represented by a summation of sinusoids, each having a specific amplitude and phase. This sinusoidal decomposition of a digital audio signal is calculated using the DFT. The magnitude spectrum displays the frequency domain of an audio signal along the x-axis and the magnitude (weight) of each frequency along the y-axis. The magnitude spectrum of a pluck of the note $C3$ on a guitar is displayed in Figure 2.2a. Note that peaks in the spectrum occur at roughly integer multiples of the fundamental frequency of 130.8Hz. The magnitude spectrum of a strum of the C major chord, consisting of five notes sounding simultaneously with the note $C3$ at its root, is displayed in Figure 2.2b; conversely, there is no obvious structure in the frequency domain.

The majority of automatic music transcription systems proposed in the literature are offline algorithms that process entire audio recordings of instruments performing.



(a) Frequency spectrum of a pluck of the note $C3$ on a guitar.

(b) Frequency spectrum of a strum of the C major chord on a guitar.

Figure 2.2: Comparison of the frequency spectrum of a pluck versus a strum on an acoustic guitar with steel strings.

Due to the hardships encountered by offline transcription systems, little research has been conducted on the more difficult problem of realtime transcription, though it has been attempted [38]. Moreover, the majority of algorithms are *causal*: a signal processing term signifying that the transcription of note events at time t_0 only depends on the prior audio signal samples $x[t]$, $\forall\, t \in \{0, 1, \ldots, t_0\}$. Certainly, if an automatic music transcription system is operating on a performance occurring in realtime, the algorithm must be causal. On the other hand, non-causal transcription algorithms "cheat" by considering properties of the audio signal that occur after the current point of analysis, making realtime transcription impossible.

Given an input audio signal, polyphonic transcription algorithms output a list of note event estimates, each having a pitch, onset time, and duration. A review of the multitude of different algorithms developed for note pitch and temporal estimation are provided in the following sections.

2.2.1 Note Pitch Estimation

Algorithms for estimating the pitch of a note event are reviewed in this section. More emphasis is placed on machine learning algorithms for pitch estimation rather than digital signal processing algorithms in order to provide better context for the work presented in this thesis.

2.2.1.1 Digital Signal Processing Algorithms

Of the hundreds of algorithms proposed to estimate the pitch of notes sounding in an audio recording, the majority are digital signal processing algorithms. Digital signal processing is a method of signal analysis or transformation that focuses on the manipulation of a sampled analog waveform where each sample is stored as a sequence of bits on the computer using a method known as *pulse-code modulation*.

Polyphonic transcription systems that use digital signal processing algorithms operate by estimating the fundamental frequency of notes sounding in an audio recording and then transforming these frequencies into pitch estimates. When algorithms estimate the fundamental frequency of a note in an audio recording, often it does not match the frequency of a pitch exactly. In this case, the frequency is quantized to that of the nearest pitch, assuming an equal-tempered system. For example, a frequency of 257Hz is between the pitches $B3$ (246.9Hz) and $C4$ (261.6Hz) and is therefore assigned a pitch of $C4$ because $|257 - 261.6| < |257 - 246.9|$.

The first attempt at polyphonic transcription imposed strict constraints on the input audio signals that could be processed [73]: the recording must have less than three voices (layers of notes or instruments); notes must be longer than 80ms; and the fundamental frequency of a note may not collide with the harmonics of another note. The latter constraint addresses the fundamental problem of polyphonic transcription

and constructs a domain without this property to ensure accurate transcription. Nevertheless, this work was an important first step that stimulated future research in automatic music transcription, which has since sought to alleviate these constraints. Several approaches to the problem, outlined in the following sections, have been proposed in the literature.

Salience Methods

Algorithms that subscribe to the salience method of pitch estimation operate by transforming the magnitude spectrum of the input audio recording to accentuate the fundamental frequencies through an analysis of the structure of harmonics. The first practical instance of this method in the literature is the fundamental frequency salience function proposed by Klapuri [57]:

$$s(f) = \sum_{h=1}^{H} \alpha(f,h)|X(hf)|, \tag{2.1}$$

such that H is the number of harmonics to consider, α is an empirically constructed function to weight each harmonic, and $X(f)$ is the DFT of the input audio signal at a specific frequency f. The salience function calculates the additive energy of harmonics for each frequency. Fundamental frequency estimates are selected by considering global and local maxima of this salience function.

Another technique that has gained traction in the music information retrieval community is the development of an alternative analysis tool to the STFT called the resonator time-frequency image (RTFI) [128]. The RTFI also offers an analysis of the frequencies present in an audio signal over time, but allows analysis at uniformly or logarithmically spaced frequencies. To estimate the pitch of notes, Zhou et al. [129, 130] apply several filters to the RTFI in order to emphasize fundamental frequencies in

audio recordings of pianos. This algorithm received the best transcription results on piano recordings in the music information retrieval evaluation exchange (MIREX) competition: an annual evaluation of MIR algorithms using identical datasets and metrics.[1] Since its inception, the RTFI has been used in other recent polyphonic transcription systems [5].

Iterative and Joint Estimation

A leading method in the literature for multiple fundamental frequency estimation is iterative estimation [55]. Iterative fundamental frequency estimation algorithms first transform the time-domain input audio signal to the frequency domain using the DFT. Then a predominant fundamental frequency is selected through an analysis of the frequency domain; the magnitude spectrum of the selected fundamental frequency and its harmonics are estimated and then subtracted from the original frequency spectrum, essentially removing the note from the audio signal. The process repeats itself until no fundamental frequencies remain in the residual frequency spectrum. Klapuri [57] used the salience function presented in Equation 2.1 to estimate the predominant fundamental frequency in audio recordings created by mixing instrument samples of individual notes to create recordings of various polyphony levels. This iterative estimation algorithm was considered state-of-the-art in the MIR commmunity for quite some time, with fundamental frequency estimation errors ranging from 9% at a polyphony level of one to 37% at a polyphony level of six when the polyphony level was provided to the algorithm.

Klapuri [57] also proposed a joint multiple fundamental frequency algorithm. Joint multiple fundamental frequency estimation algorithms take a more holistic approach to pitch estimation. Fundamental frequencies are selected that, together, best account for

[1] `www.music-ir.org/mirex`

the observed magnitude spectrum of the input acoustic signal. Fundamental frequencies are often selected by assigning a score to combinations of fundamental frequency candidates using the physical properties of harmonic sounds and the combination with the highest score is selected [119, 121, 122]. In practice, the aforementioned algorithm produces relatively accurate transcriptions, with an error rate of approximately 38% on the MIREX polyphonic transcription piano dataset [24, 120]. However, due to the joint estimation procedure, this method is considerably slower than iterative estimation methods.

The previously reviewed pitch estimation algorithms consider harmonics at integer multiples of the fundamental frequency; however, for stringed instruments this is not exactly correct. The *inharmonicity phenomenon* is an acoustic phenomenon, caused by a variety of physical factors, which results in high-frequency harmonics to be translated upwards in frequency according to the formula

$$f_h = h f_0 \sqrt{1 + \alpha(k^2 - 1)}, \tag{2.2}$$

such that h is the number of the harmonic, f_0 is the fundamental frequency, and α is the inharmonicity factor [39]. The estimation of pitch for stringed instruments, such as the guitar, should account for this acoustical phenomenon.

Considering the inharmonicity phenomenon for the transcription of stringed instruments, Emiya et al. [36, 37] applied a joint multiple fundamental frequency estimation algorithm on synthesized piano recordings. The estimation algorithm outputs the fundamental frequencies that jointly maximize the likelihood function, calculated by considering Equation 2.2. On the MIREX polyphonic piano transcription dataset, the algorithm noted only 2% marginal gains in transcription accuracy relative to algorithms that did not consider the inharmonicity phenomenon.

Human Audition Modelling

While reflecting on the current state of automatic music transcription research, Anssi Klapuri, an expert in the field, advocated a research push towards further investigation and modeling of the human auditory system [55]:

> The problem is really not in finding fast computers but in discovering the mechanisms and principles that humans use when listening to music. Modeling perception is difficult because the world in which we live is complex and because the human brain is complex [and] combines a large number of processing principles and heuristics. We will be searching for them for years, perhaps even decades ...

The problem of polyphonic transcription can be recast as the problem of sound source separation, whereby chords consisting of multiple notes are deconstructed and individually transcribed using robust and accurate monophonic pitch estimation algorithms. With respect to human audition, this process is referred to as *auditory scene analysis*, which involves the separation of sounds that are mixed in both the time and frequency domain into their respective sources [17]. Perhaps more widely known, this natural phenomenon has been referred to as the *cocktail party effect*, which describes the ability of humans to parse and interpret a single sound source amongst a large body of acoustic noise [25]. Computational models of this natural process have been implemented using clustering of harmonics while considering cues used by humans for source separation, such as *timbre* identification, the inharmonicity phenomenon (Equation 2.2), and harmonic locations [52] or through more statistical oriented models such as a Gaussian mixture model [51, 74].[2] With a transcription accuracy error of approximately 46% on the MIREX polyphonic piano dataset, the

[2] *Timbre* refers to several attributes of an audio signal that allows humans to attribute a sound to its source and to differentiate between a trumpet and a piano, for instance. Timbre is often referred to as the "colour" of a sound.

human-audition inspired transcription system proposed by Nakano et al. [74] fell significantly behind other evaluated transcription systems.

Other human-audition inspired methods include models of the human auditory and periphery systems that contribute to the perception of pitch, where the function of the middle and inner ear are modelled by a set of band-pass filters [56, 71, 97]. Taking a different approach, Davy and Godsill [30] interpret acoustic waveforms as a summation of sinusoids plus some residual noise to form a statistical signal processing model that mimics the function of the basilar membrane: an element of the human inner ear responsible for interpreting acoustic waveforms and converting them to electrical signals. However, the proposed Bayesian algorithm for estimating the parameters of each contributing sinusoid requires searching an intractable solution space, making the technique practically infeasible.

2.2.1.2 Machine Learning Algorithms

The evolution of algorithms for note pitch estimation parallels that of optical character recognition. In the infancy of the problem, hand-coded logic and hand-engineered edge detectors were the foundation of these character recognition algorithms. Machine learning algorithms later became the de facto standard, processing pixels or other features extracted from images of characters or digits in order to classify their contents [31]. Similarly, pitch estimation algorithms exclusively used digital signal processing techniques but are gradually losing share to machine learning classification algorithms. Formally, the task of classification involves the discovery of a function $f : \phi \mapsto y$ that maps a feature instance ϕ to a class label y.

Digital signal processing algorithms are still being proposed for pitch estimation because machine learning algorithms are starved for sufficient amounts of ground-truth data to properly classify pitches of notes. This is the primary reason why the majority

of available instrument transcription datasets are piano recordings, because a Yamaha Disklavier—an acoustic piano that is mechanically operated by solenoids—is capable of generating real acoustic recordings that are time-aligned with note annotations. There is currently a lack of ground-truth datasets for other instruments such as the guitar.

Machine learning algorithms approach the problem of pitch estimation as a pattern recognition problem in which audio features are input to a classifier that labels the pitches present in each segment of audio. Several different machine learning algorithms have been applied to the problem of pitch estimation, presented in the following sections.

Hidden Markov Models

A HMM is a probabilistic graphical model that estimates a sequence of hidden states from a sequence of observations [15, 86]. In the case of note pitch estimation, the hidden states are combinations of pitches capable of being produced by an instrument and the observations are acoustic features of an input audio recording. Interestingly, the state transition matrix of the underlying Markov chain acts as a musicological model that statistically governs how notes and chords transition between each other in music. An emission distribution defines the probability of observing the acoustic features given that the model is in a specific state (note or chord). An insurmountable issue with modeling the states as potential combinations of notes is that the size of the state space becomes computationally intractable to process. The first and only application of an HMM to pitch estimation was performed by Raphael [92], who sought to transcribe polyphonic piano recordings. To combat the large number of states, the level of polyphony was limited to four, the pitch range of the piano was limited from

$C2$–$F6$, and heuristics were used to prune the search space. The algorithm yielded 61% transcription accuracy on recordings of movements from Mozart piano sonatas.

Non-negative Matrix Factorization

A currently trending approach to polyphonic transcription in the MIR research community is non-negative matrix factorization (NMF) algorithms, which are typically used for dimensionality reduction in other fields. NMF involves the decomposition of a matrix $X \in \mathbb{R}_{\geq 0}^{M \times N}$ into the product of two matrices $W \in \mathbb{R}^{M \times K}$ and $H \in \mathbb{R}_{\geq 0}^{K \times N}$ such that $X \approx WH$. The matrices W and H are trained by alternating the optimization of each matrix with respect to an objective function that measures the difference between X and its approximation WH [117].

In the context of polyphonic transcription, the non-negative matrix X is the STFT of the input audio recording, such that columns of the matrix are samples of the magnitude spectrum over time. Columns of the non-negative matrix W are the *basis* or *latent feature* vectors that represent the magnitude spectra of each note template. Rows of the non-negative matrix H correspond to the temporal aspects of each note template and govern which note templates contribute to the entire audio recording at any given time. In effect, this method is a blanket solution to both the problem of pitch estimation and note temporal estimation. This first application of this method to the problem of polyphonic transcription was performed by Smaragdis and Brown [107]. Since this method was first proposed, several extensions that explore different learning methods and constraints have been developed [32, 87, 116]. On the MIREX polyphonic piano dataset, these algorithms received up to 62% transcription accuracy.

A minor modification to this approach was proposed by Lee et al. [59, 60] who modeled the matrix X in the same way as before, but H was changed from a matrix

of temporal note activities for the entire audio signal to a matrix of weights for the note templates (bases) for each audio analysis frame. In this way, a weighted sum of note templates seek to match the frequency spectrum of the input audio analysis frame. Temporal note estimation was performed afterwards as a postprocessing step and received inferior results in comparison to the original NMF method proposed by Smaragdis and Brown [107]. The modified approach proposed by Lee et al. [59, 60] was later extended to accommodate note templates of multiple instruments [4, 6].

Neural Networks

A specific adaptation of neural networks have been applied to the problem of pitch estimation [65], though this technique is less prevalent in the literature. Using multiple neural networks, Marolt [66, 67] developed the *SONIC* system to transcribe polyphonic piano recordings. The pitch estimation algorithm begins by applying a set of band-pass filters to the input audio recording. The output of each band-pass filter is routed to an adaptive oscillator, which aims to sync with the fundamental frequency and harmonics present within the signal. A neural network is trained for each of the 76 adaptive oscillators, which attempt to sync with the pitches $A1$–$C8$. The neural network has a binary output node indicating the presence or absence of a pitch in the current audio signal being analyzed. This transcription system was evaluated on six piano recordings and received a transcription accuracy of approximately 86.5%. The results indicate that this algorithm performs quite well, but it is difficult to compare with other algorithms given that it was evaluated on a non-standard dataset.

Support Vector Machines

The problem of pitch estimation can be formulated as a classic pattern recognition task where the input audio recording is partitioned into sequential analysis frames whose

features are input to a classifier that predicts the presence of a pitch. Presently, pitch estimation algorithms that formulate the problem in this way do not employ multi-label classification techniques to predict a binary vector of pitches, which indicates the presence or absence of each pitch in the audio segment being analyzed. Instead, several binary one-versus-all classifiers are trained for each pitch class to accommodate the prediction of multiple pitches sounding simultaneously. Following this technique, Poliner and Ellis [82, 83, 84] proposed the use of 87 one-versus-all binary support vector machine (SVM) classifiers to detect the presence of pitches in piano recordings. The premise of a binary SVM classifier is to locate a hyperplane that separates and maximizes the distance between training points with different class labels [27]. This algorithm was evaluated on the MIREX polyphonic piano transcription dataset and received relatively exceptional results, noting a transcription accuracy of 65%.

Deep Belief Networks

A deep belief network (DBN) is an algorithm that is recently trending in the field of machine learning and has very recently permeated the field of MIR [49, 48, 75]. A DBN is similar in structure to an artifical neural network but with a relatively large number of layers and substantially different training method. Input to the DBN is typically the STFT of an audio signal, though raw audio samples have been used as well [33]; output of the DBN is a pitch class label. A detailed review of the training and classification procedure of a DBN is provided in Section 2.4.

Though DBNs have recently been applied to other tasks in the field of MIR such as genre recognition [41, 61], music tagging [41], prediction of subsequent notes or chords in symbolic music notation [16], artist identification [62], music recommendation [78], and music emotion recognition [105], there exists only one paper applying DBNs to the task of polyphonic transcription. Nam et al. [75] proposed the use of a DBN to

generate latent audio features that are then input into a battery of binary one-versus-all SVMs that predict the presence of a single pitch in an audio segment, identical to the original algorithmic structure proposed by Poliner and Ellis [82]. Latent audio features derived by the DBN were constructed from the frequency-domain magnitude spectrum of audio analysis frames rather than time-domain samples. Using features derived by the DBN rather than the standard DFT features used by Poliner and Ellis [82], a 4.8% increase in pitch estimation accuracy was noted on the same dataset of piano recordings. The use of latent features derived by a DBN showed immediate gains in frame-level pitch estimation accuracy.

Although frame-level pitch estimates are essential for transcription, converting these estimates into note events with an onset time and duration is not a trivial task. In order to determine this missing information, note temporal estimation algorithms—also referred to as *note tracking* algorithms—are used.

2.2.2 Note Temporal Estimation

There are two predominantly used methods in the literature for estimating the onset time and duration of note events occurring in an audio recording: explicit onset and offset estimation algorithms and analysis frame-smoothing algorithms.

2.2.2.1 Onset and Offset Estimation

Onset estimation involves searching for the start time of note events in an audio recording. In the literature, onset estimation is predominantly solved by digital signal processing algorithms although neural networks have been applied to this task with some success [67, 104].

Digital signal processing algorithms for onset estimation search for *transients* in the audio recording. A transient refers to a portion of a signal where the amplitude or frequency spectrum rapidly change beyond its norm. To estimate where transients lie in the audio recording, it is important to first understand the evolution of a single note. The evolution of a note typically conforms to the following schema: *attack*, *decay*, *sustain*, and *release*. Attack occurs at the point of initial excitation of the instrument, such as the pluck of a string or the strike of a hammer, and continues until the note's maximal amplitude is achieved. Decay refers to the time after the attack where the amplitude is dampened to the level at which the note sounds for its duration, referred to as the sustain. Release refers to the section of the amplitude envelope where, in piano terms, the key is released and the amplitude drops from the sustain level to zero. In terms of the amplitude envelope of the audio signal, the transient typically occurs at the attack portion of the note evolution, displayed in Figure 2.3.

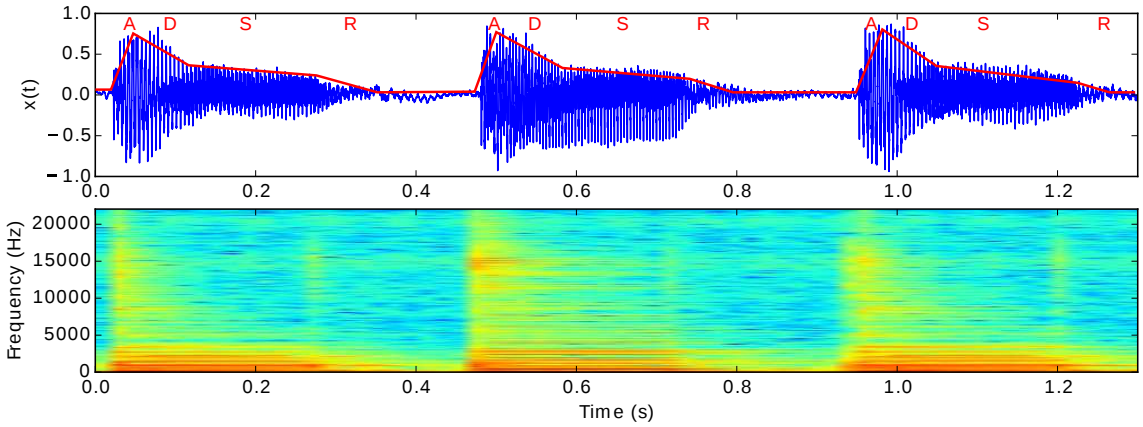


Figure 2.3: Three notes performed on a guitar with steel strings. The time domain of the audio signal is displayed above the frequency spectrogram. The letters A, D, S, and R annotate the attack, decay, sustain, and release portions, respectively, of each note.

Onset estimation algorithms can be decomposed into three steps [2]. The first optional step is preprocessing, which aims to amplify sections of the signal where

transients occur and to attenuate other portions of the audio signal. In the literature, this step often involves the translation of the audio signal from the time domain to the frequency domain for analysis [54, 102], exploiting the fact that transients are essentially noise, which have broad frequency spectra in comparison to the rest of the audio signal [95].

The second step of onset estimation algorithms is reduction, which aims to down-sample the preprocessed audio signal into a detection function that further emphasizes signal transients. Techniques for the creation of a detection function are numerous, including observation of the amplitude envelope of notes [103], observation of the first derivative of the time-domain of an audio signal [94], observation of the high-frequency content of an audio signal [69], and observation of *spectral flux* [34]. Spectral flux measures the difference in frequency content between sequential audio analysis frames to emphasize abrupt changes in spectral content, thus emphasizing transients. Regardless of the reduction technique, the result is a detection function with a significantly coarser time domain than the original audio signal. The x-axis of the detection function represents time and the y-axis represents the energy or probability of an onset occurring.

The final step of onset estimation algorithms is peak-picking, which aims to search for local maxima in the derived onset detection function. A naïve method of peak-picking is to define a sensitivity threshold such that maxima below the threshold are ignored and maxima above the threshold are considered. However, typically a subset of local maxima do not necessarily correspond with onsets and so the threshold parameter affects the number of false positives and false negatives [2]. Onset detection algorithms have since moved on to adaptive thresholding algorithms [95] that are capable of adjusting the onset threshold to accommodate changes in amplitude between different musical passages, for example.

After the onset time of a note has been estimated, an offset estimation algorithm predicts the duration of the note event. The offset time of a note event has less perceptual importance than the onset time [28] and therefore offset estimation algorithms receive less attention in the literature. Consequently, offset estimation algorithms are often simplistic in nature, predicting the offset of a note when subsequent audio analysis frames have lost trace of the pitch estimate [28, 120, 129].

As a more complex note duration model, left-to-right HMMs have been proposed to predict the evolution of notes along their amplitude envelope. To accomplish this, the internal states of the left-to-right HMM represent the attack, decay, sustain, and release portion of a note event. A silence state is also appended to represent the end of a note event. The state transitions are arranged such that once the note has progressed from the attack to decay state, for example, the Markov chain can never reach the attack state again (Figure 2.4). The self-transition probability of a state partially controls the length of time the Markov chain stays at a particular portion of the note evolution. Features of the audio signal are used to create an emission distribution for each state, which also has a hand in controlling the evolution of the note model. Several polyphonic transcription systems have employed this method [5, 9, 24, 97].

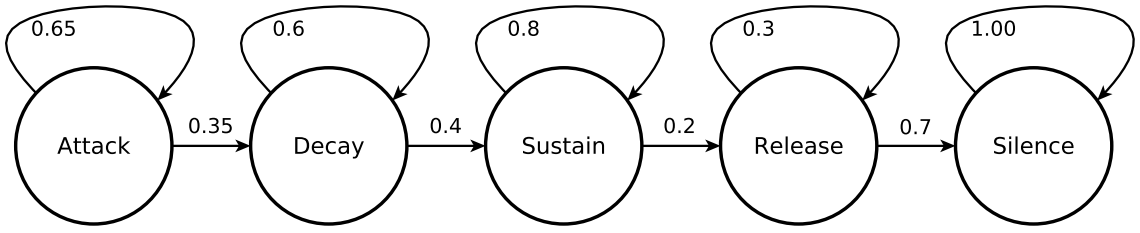


Figure 2.4: The underlying state structure of a hidden Markov model used to predict the duration of note events.

2.2.2.2 Analysis Frame Smoothing

An alternative to explicitly using onset and offset estimation algorithms to estimate the temporal attributes of note events in an audio recording is to use the technique of analysis frame smoothing. This technique is widely used in modern polyphonic transcription systems as a postprocessing step that "smooths" the pitch estimates made in sequential audio analysis frames. Poliner and Ellis [84] were the first to propose this postprocessing technique, in which an HMM is responsible for tracking the frame-level estimates of each possible pitch. This technique allows a pitch estimate to be contextualized amongst neighbouring pitch estimates rather than independently estimating pitch for each analysis frame.

Each HMM is composed of two states: note `ON` and note `OFF`. The state transition probabilities are trained using a ground-truth dataset of synthesized musical instrument digital interface (MIDI) files. MIDI is a binary file format composed of tracks consisting of note events, which each have an integer pitch from 0–127, a velocity value indicating the intensity of a note, and a tick number indicating when the note event occurs. The observations to each HMM are a sequence of binary values indicating whether the pitch estimation algorithm found the respective pitch in the audio analysis frame. Therefore, the time resolution of this frame-smoothing algorithm is identical to that of the frame-level pitch estimation algorithm that precedes it. The emission distribution is a Bernoulli distribution indicating the probability that the pitch estimation algorithm observed a pitch given the true note state of `ON` or `OFF`. These probabilities may be calculated by observing the true and false positives and negatives of the pitch estimation algorithm on a training dataset or by observing the probability of a pitch being active in an analysis frame as produced by a binary classifier [81]. The most likely state sequence of pitch `ON` and `OFF` values is found using the Viterbi algorithm.

2.2.3 Ensemble Methods

Automatic polyphonic transcription methods that coordinate information from multiple sources are referred to in this thesis as ensemble methods. An example of an ensemble method is blackboard transcription [1, 3, 68]. The blackboard problem-solving model originated from the field of artificial intelligence [76] and refers to a group of experts in different subdomains working together on a larger problem by writing on a blackboard. Each expert adds or modifies information on the blackboard when they have realized a significant contribution to the problem. For example, a blackboard polyphonic transcription system may have a separate expert for the estimation of fundamental frequencies, onsets, tempo, time signature, musical key, chords, and music notation. A scheduler is responsible for organizing the contributions of experts such that they do not overwrite each other. Although this concept seems promising, blackboard transcription systems [1, 3, 68] do not place among the top algorithms evaluated on the MIREX polyphonic piano dataset.

Specifically tailored towards polyphonic guitar transcription, several systems have been proposed that integrate alternate sources of information other than the raw audio waveform. These systems include a computer vision analysis of the guitar fretboard as a guitarist performs [20, 21, 53, 101], a multi-modal analysis of guitar fretboard video in conjunction with audio [46, 79, 85], as well as hyperinstruments [72], which involves the installation of additional sensors to the guitar to aid transcription [77, 93]. The main disadvantage of these systems in terms of the commercial application of transcription algorithms is that consumers would need to purchase, install, and configure additional hardware in order to perform transcriptions.

2.2.4 Summary

Regardless of the polyphonic transcription algorithm being used, the output is a list of note event estimates. The list of note event estimates may then be written as a music score in common Western music notation or tablature notation. In the case of tablature notation, more processing of the note event estimates is required by a guitar tablature arrangement algorithm.

2.3 Guitar Tablature Arrangement

Guitar tablature arrangement algorithms aim to generate tablature—a sequence of guitar string and fret combinations—from common Western music notation. This is a difficult task because, unlike the piano which has a one-to-one correspondence between a key and a pitch, the guitar can generate the same pitch in several different ways. For example, on a 24-fret guitar in standard tuning, there exists six different string and fret combinations to generate the pitch $E4$, as depicted in Figure 2.5. This property of the guitar adds more ambiguity to the transcription process [38].

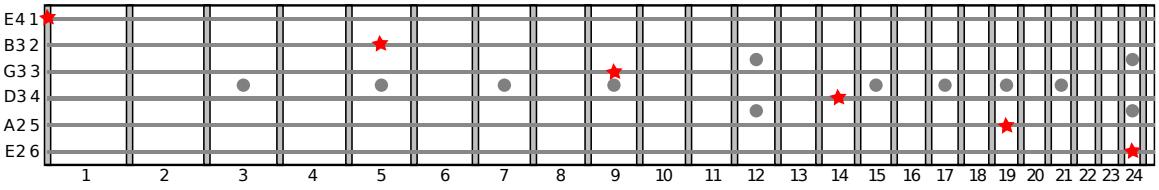


Figure 2.5: A 24-fret guitar fretboard indicating six different string and fret combinations, annotated as stars, that generate the pitch $E4$ when in standard tuning.

The task of guitar tablature arrangement can be formulated as a traditional search problem in which there are several string and fret combinations for each note in a musical passage and the goal is to find an arrangement that maximizes a metric measuring the quality of tablature. The quality of a tablature arrangement is an

arguably subjective measurement, since different musicians have varying styles and preferences as to how the music should be performed.

Leading towards an objective measurement of tablature quality, Sawayama et al. [99] propose that good tablature should be biomechanically easy to perform. Heijink and Meulenbroek [42] propose that tablature difficulty depends on the position of the fretting hand along the neck of the guitar as well as the amount of fret-hand repositioning and finger spanning needed to perform a sequence of notes or chords.[3] Confirming this hypothesis, a study of the fretting-hand movements of classical guitar players indicated that guitarists favour hand positions at the beginning of the fretboard nearest the tuning knobs and avoid arrangements that require excessive fretting-hand movement and finger spans [42].

Using these quantitative metrics of "good" tablature, a search algorithm can be used to traverse the domain of possible guitar string and fret combinations for each note and find an optimal tablature arrangemen. The problem of tablature arrangement is difficult because of the size of the search space, which can quickly become computationally intractable. For a sequence of n notes being performed on a 24-fret electric guitar in standard tuning, there is an upper bound of 6^n different tablature arrangements, assuming a maximum of six different string and fret combinations for each note. The search space grows even larger when considering polyphonic guitar music. A sequence of n chords each containing k notes that may be performed in up to six different ways results in an upper bound of 6^{kn} different tablature arrangements. To search this large space of solutions, several guitar tablature arrangement algorithms have been proposed, including graph algorithms, neural networks, and genetic algorithms.

[3]The *fretting hand* refers to the hand that depresses frets along the neck of the guitar. For right-handed guitarists, the left hand is used for fretting and the right hand is used for strumming.

2.3.1 Graph Algorithms

All possible string and fret candidates for a sequence of note events can be modeled as a directed acyclic weighted graph $G = (V, E)$, such that V is a set of vertices and E is a set of directed weighted edges. Each vertex represents the string and fret candidate(s) for a note or chord in a musical passage. In the case of a note, a vertex is formed for each string and fret candidate that generates the pitch of the note. In the case of a chord, a vertex is formed for each combination of string and fret candidates for each note. A sequence of two notes or chords is modeled as a directed bipartite graph, such that each vertex for the first note or chord (source) is connected to each vertex for the second note or chord (sink). The weight of each edge in the graph represents the "cost" of transitioning between fingering positions. The cost of traversal should reflect the biomechanical difficulty of performing the notes on the guitar. Figure 2.6 displays an example of a directed acyclic weighted graph of candidate string and fret combinations for a music score consisting of four notes: $G\sharp 4$, $A5$, $B5$, and $C5$.

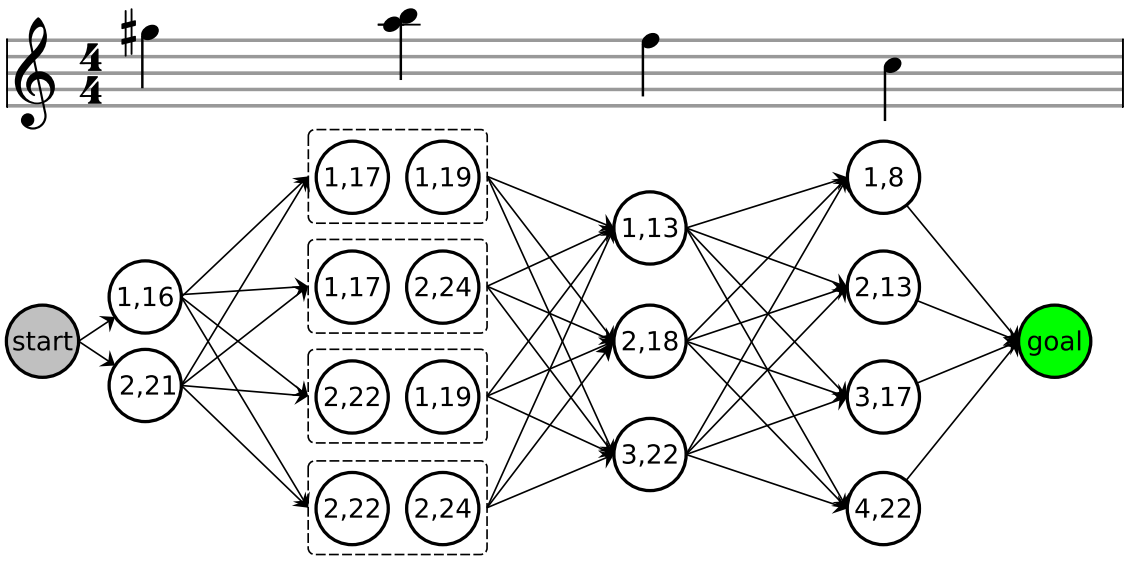


Figure 2.6: A directed acyclic graph of string and fret candidates for a note and chord followed by two more notes. Weights have been omitted for clarity.

Several graph-based tablature arrangement algorithms have been proposed in the literature. Sayegh [100] assigned weights to edges of the graph by increasing the cost of traversal when the transition requires a change in hand position or string. The Viterbi algorithm was used to search the lattice of string and fret candidates for each note. No formal evaluation was conducted and the arrangement algorithm only processed monophonic music scores.

Radicioni et al. [88] proposed further changes to edge weight assignments, considering both horizontal and vertical movement of the fretting hand along the neck of the guitar. Their proposed algorithm was later extended to handle polyphonic music scores by assessing the difficulty of biomechanically forming chords with the fretting hand [90]. Radicioni and Lombardo [89] proposed further constraints to prune the guitar tablature arrangement graph: a string may only sound one note at a time, fretting-hand fingers that are further down the fretboard should depress higher fret numbers, and the maximum spanning distance of fretting-hand fingers along the fretboard should be considered.

Burlet and Fujinaga [19] proposed a polyphonic tablature arrangement algorithm that considers the guitar tuning, *capo* position, and number of frets.[4] Edge weights were assigned by considering the fret-wise distance between consecutive notes or chords, the fret-wise finger span required to perform a chord, and a penalty for string and fret candidates that are high on the fretboard. The relative importance of each factor contributing to the edge weight were assigned by hand. The shortest path through the graph was found using Dijkstra's algorithm.

Radisavljevic and Driessen [91] used similar factors to calculate edge weights in the graph, but optimized the weighting of each factor by applying gradient descent to

[4] A *capo* is a device that is clipped to the guitar fretboard that raises the pitch of the instrument (Appendix A).

minimize the differences between the tablature output by the algorithm and published guitar tablature. The space of string and fret candidates was searched using dynamic programming, whereby the problem of arranging tablature for an entire music score is decomposed into smaller subproblems that are later combined to form the global solution. This algorithm, trained on excerpts of published classical guitar scores, was evaluated on the same dataset yielding tablature that was 97% equivalent to the published scores.

2.3.2 Neural Networks

Artificial neural networks have also been used to arrange guitar tablature. Tuohy and Potter [112] proposed a three-layer neural network that sequentially processes notes in a music score. The output layer of the neural network consists of 20 nodes indicating the probability that the note should be performed on a certain fret from 0–19, where the zeroth fret indicates the pluck of an open string. One drawback to the structure of this network is that it only accommodates 19-fret guitars. A significant detriment of this approach is that contextual information of the note is lost by processing each note individually, making tablature arrangement difficult.

In response to this issue, Tuohy and Potter added features to the network input that provide information about previous or subsequent notes as well as other notes in a chord. After processing a music score, the network output is postprocessed using a local search algorithm that determines if the tablature can be improved, according to biomechanical constraints of the fretting hand [111]. The network was trained using a dataset of human-arranged guitar tablature. Using the training dataset for testing, the output tablature was 94% identical to the ground-truth tablature.

2.3.3 Genetic Algorithms

Genetic algorithms have also been used to search for good tablature arrangements. For tasks such as guitar tablature arrangement that quickly become intractable by exhaustive search algorithms that do not utilize heuristics or domain constraints, genetic algorithms are well suited to provide adequate solutions [109]. A genetic algorithm is a stochastic optimization algorithm that refines the population of possible solutions through generations of breeding and mutation. A *gene* represents a note or chord on the guitar. A *chromosome* is a sequence of genes that represent a candidate tablature arrangement. A *population* is composed of many candidate tablature arrangements. A *fitness function* assesses each candidate tablature arrangement and assigns it a score based on its biomechanical ease of performance or other desirable stylistic traits. The fitness function typically affects the probability with which two individuals within the population mate. A compelling benefit to using a genetic algorithm for tablature arrangement is that the input population size dictates the number of different tablature arrangements generated. Therefore, multiple arrangements can be generated and ranked in terms of their respective fitness.

Using this schema, Rutherford [96] proposed a guitar tablature arrangement algorithm for monophonic musical passages, which was extended to polyphonic musical passages [18, 109, 110, 111]. Through an analysis of guitar tablature arrangements generated using a graph-search algorithm [19] versus a genetic algorithm [18] on the same dataset of 75 hand-arranged tablature excerpts, it was found that the genetic algorithm generated more difficult tablature than the graph-search algorithm, on average. Moreover, the difficulty of tablature generated by the genetic algorithm was more variable than tablature produced by the graph-search algorithm.

2.4 Deep Belief Networks

The intent of deep architectures for machine learning is to form a multi-layered and structured representation of sensory input with which a classifier or regressor can use to make informed predictions about its environment [115]. Although "deep architectures can in principle be more powerful than a shallow one" [11], empirical evidence has shown that artificial neural networks with multiple hidden layers often underperform in relation to their shallow counterparts [10, 11, 12]. Possible explanations for this phenomenon is that the gradient-based backpropagation algorithms used to train these networks start from a random point in the weight space and often get stuck in poor local minima [11], or the problem of "vanishing gradients" where backpropagation becomes less effective at passing the training signal to lower layers [13].

Recently, Hinton et al. [45] proposed a specific formulation of a multi-layered artificial neural network called a deep belief network (DBN), which addresses the training and performance issues arising when many hidden network layers are used. A preliminary unsupervised training algorithm aims to set the network weights to good initial values in a layer-by-layer fashion, followed by a more holistic supervised fine-tuning algorithm that considers the interaction of weights in different layers with respect to the desired network output [44].

2.4.1 Unsupervised Pretraining

In order to pretrain the network weights in an unsupervised fashion, it is necessary to think of the network as a generative model rather than a discriminative model. A generative model aims to form an internal model of a set of observable data vectors, described using latent variables; the latent variables then attempt to recreate

the observable data vectors with some degree of accuracy. On the other hand, a discriminative model aims to set the value of its latent variables, typically used for the task of classification or regression, without regard for recreating the input data vectors. A discriminative model does not explicitly care how the observed data was generated, but rather focuses on producing correct values of its latent variables.

Hinton et al. [45] proposed that a deep neural network be composed of several restricted Boltzmann machines (RBMs) stacked on top of each other, such that the network can be viewed as both a generative model and a discriminative model. An RBM is an undirected bipartite graph with m visible nodes and n hidden nodes, as depicted in Figure 2.7. Typically, the domain of the visible and hidden nodes are binary such that $\mathbf{v} \in \{0,1\}^m$ and $\mathbf{h} \in \{0,1\}^n$, respectively. A lucrative property of an RBM is that the visible and hidden units are conditionally independent:

$$P(\mathbf{v}|\mathbf{h}) = \prod_i P(v_i|\mathbf{h}) \quad \text{and} \quad P(\mathbf{h}|\mathbf{v}) = \prod_j P(h_j|\mathbf{v}), \tag{2.3}$$

such that

$$P(h_j = 1|\mathbf{v}) = \frac{1}{1 + e^{-W_j \mathbf{v}}} \quad \text{and} \quad P(v_i = 1|\mathbf{h}) = \frac{1}{1 + e^{-W_i^T \mathbf{h}}}, \tag{2.4}$$

where $W \in \mathbb{R}^{n \times m}$ is the matrix of weights between the visible and hidden nodes. For simplicity, Equation 2.4 does not include bias nodes for $\mathbf{v}$ and $\mathbf{h}$.

Each RBM in the DBN is trained sequentially from the bottom up, such that the hidden nodes of the previous RBM are input to the subsequent RBM as an observable data vector. The unsupervised training of a single RBM involves iteratively modifying the model weights according to a learning signal that measures how well the generative model reflects reality. More specifically, the objective of the generative model is to maximize the log likelihood of the training data vectors by calculating the gradient of

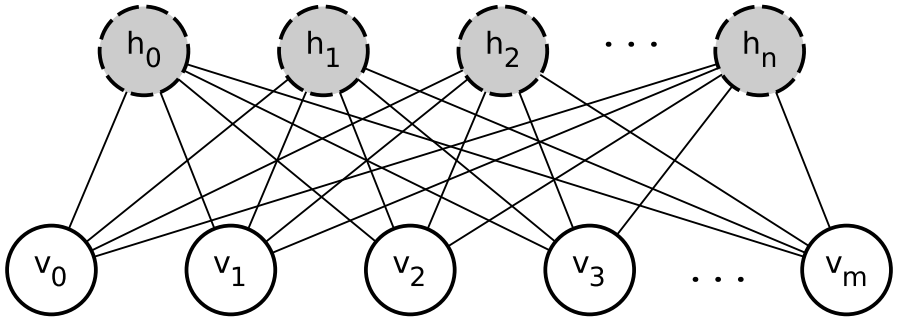


Figure 2.7: A restricted Boltzmann machine with m visible nodes and n hidden nodes. Weights on the undirected edges have been omitted for clarity.

this objective function with respect to each edge weight [45]:

$$\frac{\partial \ln P(\mathbf{v}^0|\Theta)}{\partial W_{ij}} = \mathbb{E}\left[v_i^0 h_j^0|\mathbf{v}^0\right] - \mathbb{E}\left[v_i^\infty h_j^\infty\right], \tag{2.5}$$

such that Θ denotes the current model parameters and superscripts of v and h indicate the number of times that the model has recreated an input data vector $\mathbf{v}^0$ through successive sampling from the Bernoulli distributions defined in Equation 2.4. In more detail, sampling is conducted iteratively such that

$$\mathbf{h}^k \sim P(\mathbf{h}^k|\mathbf{v}^k) \quad \text{and} \quad \mathbf{v}^{k+1} \sim P(\mathbf{v}^{k+1}|\mathbf{h}^k). \tag{2.6}$$

The learning signal presented in Equation 2.5 operates on the notion that nodes connected with large positive weights coerce each other to have the same binary state, while nodes connected with large negative weights force each other to have complementary binary states, as illustrated in Equation 2.4. Therefore, an informative learning signal involves calculating the discrepancy between $\mathbb{E}\left[v_i^0 h_j^0|\mathbf{v}^0\right]$—the expected value of the association between an input state v_i^0 and a connected hidden state h_j^0—and $\mathbb{E}\left[v_i^\infty h_j^\infty\right]$, which denotes the expected value of this association using samples

from the true posterior distribution. Sampling from the true posterior (equilibrium) distribution is possible by letting the generative model recreate the input data vector an infinite number of times.

Though it is desirable to sample from the true posterior distribution, it is practically infeasible to generate an infinite number of samples from the RBM. Therefore, the latter expectation in Equation 2.5 is often approximated by sampling k times: a process called k-step contrastive divergence [45]. Using this approximation, the gradient of the log likelihood with respect to a single edge weight becomes

$$\frac{\partial \ln P(\mathbf{v}^0|\Theta)}{\partial W_{ij}} = \mathbb{E}\left[v_i^0 h_j^0|\mathbf{v}^0\right] - \mathbb{E}\left[v_i^k h_j^k|\mathbf{v}^k\right] \tag{2.7}$$

$$= v_i^0 P(h_j^0 = 1|\mathbf{v}^0) - v_i^k P(h_j^k = 1|\mathbf{v}^k) \tag{2.8}$$

$$= \frac{v_i^0}{1 + e^{-W_j \mathbf{v}^0}} - \frac{v_i^k}{1 + e^{-W_j \mathbf{v}^k}}. \tag{2.9}$$

Gradient descent is performed in order to minimize the negative log likelihood, such that each edge weight is updated using the equation

$$W_{ij} = W_{ij} + \eta \frac{\partial \ln P(\mathbf{v}^0|\Theta)}{\partial W_{ij}}, \tag{2.10}$$

where $\eta \in \mathbb{R}^+$ is the learning rate. The training data is iterated over until the desired convergence criterion is met.

2.4.2 Supervised Finetuning

The unsupervised pretraining of the stacked RBMs is a relatively efficient method that sets good initial values for the network weights. However, the increased computational efficiency comes at a price: training one layer at a time explicitly ignores the interaction

of other layers. Moreover, in the case of a supervised learning task such as classification or regression, the ground-truth labels for each training data vector have not yet been considered. The supervised fine-tuning step of the DBN addresses these issues.

There are several schools of thought surrounding how to perform the supervised DBN fine-tuning step. The work in this thesis focuses on supervised fine-tuning to improve the discriminative quality of the network, although algorithms such as the "up-down" algorithm [45] have been proposed to improve the generative quality of the network. Another method of supervised fine-tuning is to add a layer of output nodes to the network for the purposes of (logistic) regression and to perform standard backpropagation as if the DBN was a multi-layered neural network [10]. Rather than creating features from scratch, this fine-tuning method is responsible for modifying the latent features in order to adjust the class boundaries [44]. Yet another method of creating a regressor or classifier from the pretrained network is to sample the latent variables from the last hidden layer in the network for each training data instance. The values of these latent variables are used as features to train any number of different regressors or classifiers, such as an SVM [75].

After fine-tuning the network, a feature vector can be fed forward through the network and a result realized at the output layer. In the context of pitch estimation, the feature vector represents the frequency content of an audio analysis frame and the output layer of the network is responsible for classifying the pitches that are present. In the case of polyphonic music, several pitches may occur in a single audio analysis frame, and so the output layer of the network should be able to assign multiple classes (pitches) to an input analysis frame. Multi-label classification techniques are one solution to such a problem.

2.5 Multi-label Classification

Multi-label learning algorithms assign a subset of labels from a finite corpus of labels $\mathcal{L}$ to each input data instance. Formally, multi-label learning can be defined as a function $f : \phi \mapsto \mathbf{y}$ that maps a feature vector $\phi \in \mathbb{R}^m$ to a binary vector $\mathbf{y} \in \{0,1\}^{|\mathcal{L}|}$ of length equal to the cardinality of the set of possible labels. If a label is associated with the input data instance, the corresponding bit in $\mathbf{y}$ is toggled. In the context of pitch estimation, the *label cardinality* $\sum_{i=1}^{|\mathcal{L}|} y_i$ is equivalent to the polyphony of the audio signal.

The most common multi-label learning technique is to learn a function $c : \phi \times y \mapsto \mathbb{R}$ that outputs a real-valued number representing the confidence that a label $y \in \mathcal{L}$ is associated with the input feature vector ϕ [126]. Ideally, this function should yield a high confidence value when a ground-truth label is input into the function. In practice, the confidence value of a label is often set as the probability of the label as output by a binary classifier. To predict which labels should be assigned to the input feature vector, a threshold $\alpha \in \mathbb{R}$ is defined; if $c(\phi, y) > \alpha$, then the label is relevant to the input. The sensitivity of the threshold affects the label cardinality and hence the number of false positives and negatives.

A constant threshold is one of several strategies to select which labels are relevant to an input data vector. Ioannou et al. [50] provide a detailed review and evaluation of other label prediction methods, which first rank the label confidence values for each data instance in descending order and then partition the label corpus into a set of relevant and irrelevant labels using the following methods: `RCut` outputs a constant number of labels having the highest confidence; `SCut` computes a different constant threshold for each label; `PCut` allows a certain number of testing data instances to belong to a specific label, which varies according to the prior probability of any training

instance being assigned that label; `MetaLabeler` trains a multi-class classifier that predicts the label cardinality for each data instance [108]; and `ThresholdPrediction` sets the confidence threshold such that it minimizes the number of misclassifications on the training dataset [35, 127]. Another thresholding strategy proposed by Largeron et al. [58] is `MCut`, which orders the labels by confidence value and locates the two labels whose confidence values differ by the greatest amount; the threshold is set to the average of these two confidence values.

Ioannou et al. [50] performed an evaluation of these multi-label learning techniques in the context of several different classification algorithms operating on five different datasets. The results indicate that the performance of any technique is largely dependent on the algorithm's parameters and the dataset being processed. Due to its simplicity, the authors note that the `OneThreshold` method is practical and performs relatively well in the context of multiple different datasets and classifiers.

However, the `OneThreshold` technique, as well as the majority of other surveyed techniques, do not allow for strict constraints on the maximum prediction label cardinality. This is an important constraint for the task of polyphonic pitch estimation. For example, a piano may only produce ten notes simultaneously, assuming a single musician is performing with two hands. Similarly, a guitar with six strings may only produce six notes simultaneously. Consequently, for the task of polyphonic guitar transcription the binary vector $\mathbf{y}$ of note pitch labels should conform to the following constraint:

$$\sum_{i=1}^{|\mathcal{L}|} \mathbf{y}_i \leq 6. \tag{2.11}$$

One multi-label learning technique that allows this constraint to be imposed is the `MetaLabeler` method [108], which trains a multi-class classifier to predict the label cardinality.

The preliminary work surveyed in this literature review lays the foundation for several avenues of expansion for the task of guitar note pitch estimation. First, multi-label learning can be used to generate a binary vector of pitch estimates for each analysis frame to circumvent the need to train a binary classifier for each pitch, which is the predominant method in the literature (Section 2.2). Second, DBNs have recently permeated the field of MIR and more experiments need to be conducted to determine their efficacy for the task of polyphonic transcription. Specifically, it is unclear whether the DFT of an audio signal is the optimal input to a DBN in terms of transcription accuracy; perhaps alternative audio features would be better suited to generate more discriminative latent features. Third, the application of DBNs to polyphonic piano transcription by Nam et al. [75] does not explore polyphony constraints that may be imposed by a specific instrument and outsources the latent features derived in higher network layers to a separate classifier instead of using the DBN itself for pitch classification. The work presented in this thesis will address these open problems within the task of pitch estimation.

Chapter 3

Deep Belief Network Tablature Transcription

The previous chapter surveyed a variety of algorithms for automatic polyphonic music transcription and introduced the notion of using deep belief networks and multi-label learning for the task of note pitch estimation in polyphonic guitar recordings. In the current literature, machine learning algorithms for note pitch estimation train a number of one-versus-all binary classifiers instead of a single multi-label classifier that predicts binary vectors of pitch estimates. Moreover, for the purposes of conforming to the constraints of an instrument, algorithms proposed in the literature offer no solutions to restrict the cardinality (polyphony) of the set of pitch estimates at any given time.

The novel pitch estimation algorithm presented in this chapter addresses these shortcomings. The proposed pitch estimation algorithm is followed by a state-of-the-art temporal estimation (note tracking) algorithm [84] that is prevalently used in the literature to estimate the onset and offset time of each predicted note event. Following this temporal estimation algorithm, a state-of-the-art onset estimation

algorithm [34] with a finer time resolution is used as a note quantizer, which adjusts the exact time of note onset estimates to be closer to their ground-truth counterparts. Following quantization, common Western music notation is generated from the note event estimates. Finally, the shortest-path, graph-based guitar tablature arrangement algorithm proposed by Burlet and Fujinaga [19] converts the common music notation to tablature notation. The workflow of this polyphonic guitar tablature transcription algorithm is illustrated in Figure 3.1. The algorithms described in this chapter were implemented in the Python programming language.

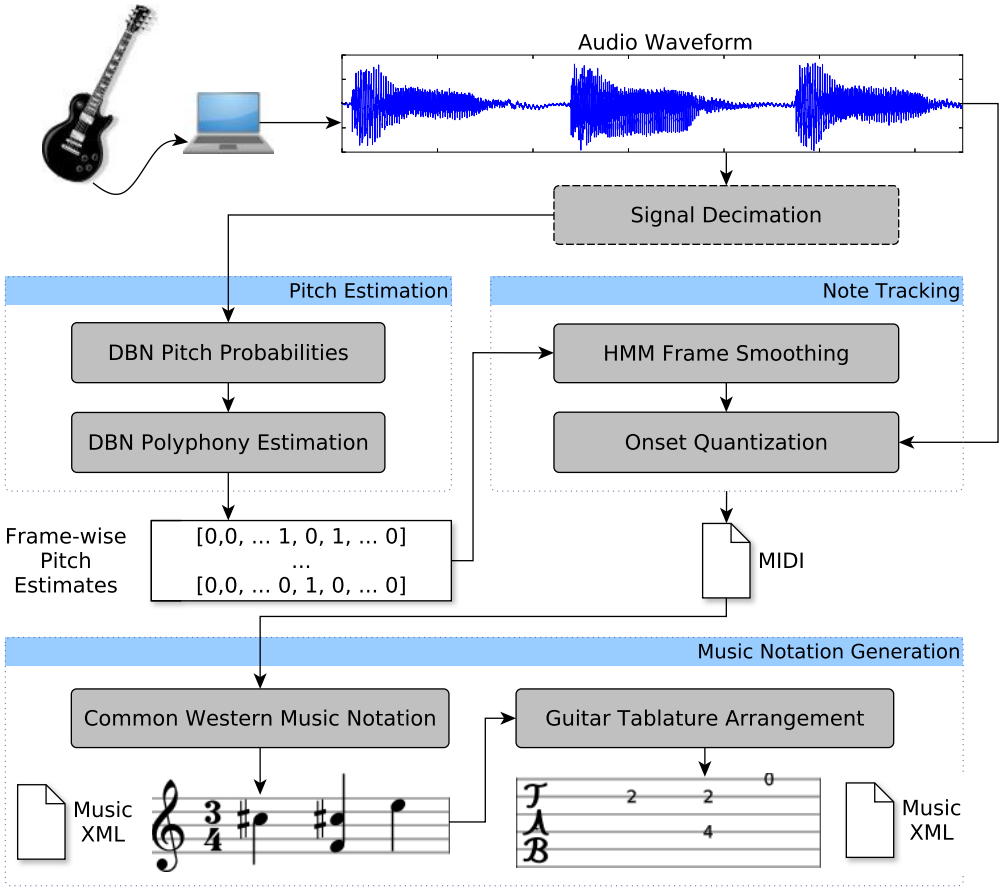


Figure 3.1: Workflow of the proposed polyphonic guitar tablature transcription algorithm, which converts a guitar recording to tablature notation.

3.1 Note Pitch Estimation

Note pitch estimation involves three steps: feature extraction, which extracts frequency-domain features from the input audio signal; calculation of pitch and polyphony probabilities using a DBN; and logic to cohere the pitch and polyphony probabilities to produce a list of pitch estimates.

3.1.1 Feature Extraction

Prior to feature extraction, the input audio signal is optionally preprocessed to lower the sampling rate using a signal-processing algorithm called *decimation*. The effect of lowering the sampling rate is twofold: the number of samples to be processed by the subsequent analysis algorithms is lowered and the frequency range of the new signal is $1/m$ that of the new sampling rate.

The feature extraction process begins by decomposing the input audio recording into sequential analysis frames using a window of $w \in \mathbb{N}^+$ samples that slides $h \in \mathbb{N}^+$ samples with each iteration. Each audio analysis frame is optionally multiplied by a Hamming window function to mitigate against *leakage*: a phenomenon that arises when calculating the DFT on not exactly periodic waveforms and consequently smears the signal energy over a wider frequency range than desired.

The desired features can then be extracted from each audio analysis frame. A common set of audio features for pitch estimation is the *power spectrum*. The power spectrum is derived from the DFT, which converts the time-domain audio samples to the frequency domain. When the DFT is calculated for each sliding analysis window, the process is referred to as the STFT. An example of STFT feature extraction from an audio waveform sampled at $f_s = 44100$ Hz is presented in Figure 3.2. The window

and hop size used in this example are purely for illustration and are usually much smaller in practice.

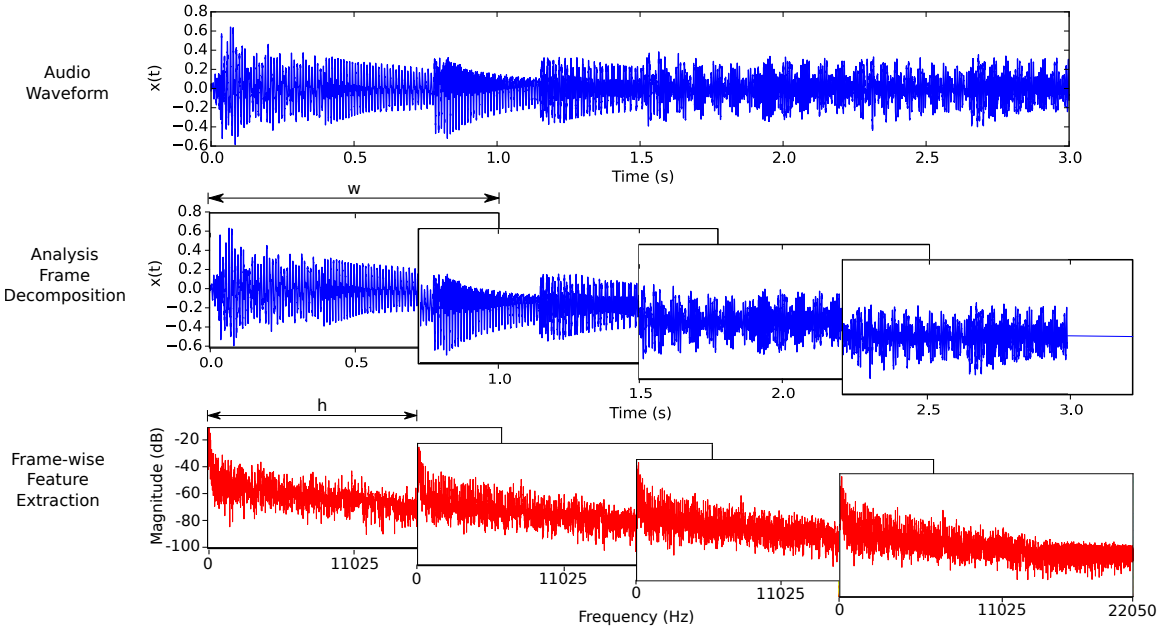


Figure 3.2: Analysis frame decomposition and feature extraction for an audio waveform. Window size is denoted by w and hop size is denoted by h.

The DFT of a digital audio signal $x \in \mathbb{R}^N$ is defined by the function

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/w}, \tag{3.1}$$

such that $x[n]$ is the signal amplitude at sample $n \in \{0, 1, \ldots, w-1\}$, j is the electrical engineering nomenclature for the imaginary number i, and $k \in \{0, 1, \ldots, w-1\}$ represents the k^{th} frequency sample, kf_s/w Hz, given an audio sampling rate of f_s Hz. Since the input audio signal is real-valued across the entire time domain, the range of the DFT is mirrored about the Nyquist frequency, $f_s/2$ Hz. Hence, for audio feature extraction it is only necessary to retain the values of $X(k)$ for $k \in \{0, 1, \ldots, (w-1)/2\}$. Finally, the DFT is transformed to its power spectrum, which is the squared magnitude of each frequency component.

The selection of parameters for the sliding window feature extraction method requires balancing the trade off between frequency resolution and temporal resolution. A larger window size w offers a finer frequency resolution when transformed by the DFT, at the expense of having a coarser time resolution for analysis. On the other hand, a smaller window size w offers a finer time resolution but a coarser frequency resolution. For example, given an audio sampling rate of $f_s = 11025$ Hz and a window size of $w = 2048$ samples, each analysis frame spans $w/f_s \approx 186$ ms and the frequency resolution of a single DFT bin is $f_s/w \approx 5$ Hz. Raising the window size to $w = 8192$ samples, each analysis frame spans $w/f_s \approx 743$ ms and the frequency resolution of a single DFT bin is $f_s/w \approx 1$ Hz.

Another frequently used set of features in the MIR community are Mel frequency cepstral coefficients (MFCCs) [63]. MFCCs provide an extension to STFT features by attempting to mimic how humans perceive sound as well as compressing the dimensionality of the feature set. To calculate the MFCCs, the periodogram estimate of the power spectrum is first calculated for each audio analysis frame using the formula

$$P(k) = \frac{|X(k)|^2}{w}. \tag{3.2}$$

Then, a set of band-pass filters, which only let a range of frequencies through, are arranged across the periodogram power spectral estimate (Equation 3.2) at logarithmically spaced frequencies. The filters are logarithmically spaced because humans have difficulty discerning between closely spaced frequencies, especially in the upper frequency range. Hence, spacing the filters in this way spreads out samples of the signal energy across the frequency domain to better replicate this property of human audition. The signal energy is aggregated within each filter band and the logarithm is once again taken, since humans also perceive loudness on a logarithmic scale. Finally,

the discrete cosine transform of the log filter energies is computed to compress the feature set by removing high-frequency changes in signal energy.

Regardless of the features being computed, each audio analysis frame's features represent a data point $\phi \in \mathbb{R}^m$ that may be used as input for training or testing the DBN pitch estimation algorithm described in the following section. Before being input to the DBN, the features are normalized such that $\phi \in [0,1]^m$. Given a set of audio recordings, the feature extraction step creates a dataset of $\Phi \in [0,1]^{n \times m}$ such that n is the number of audio analysis frames spanning the audio recordings and m is the size of the feature set. For DFT features, $m = \lfloor w/2 \rfloor$ and for MFCC features, m is equal to the number of computed coefficients.

In addition, given a set of MIDI files containing note annotations for the processed set of audio recordings, a binary matrix of pitch annotations $Y^{(pitch)} \in \{0,1\}^{n \times k}$ is computed, where k is the number of pitches being considered. The proposed polyphonic pitch estimation algorithm considers 51 pitches from $C2$–$D6$, which spans the lowest note capable of being produced by a guitar in *Drop C tuning* to the highest note capable of being produced by a 22-fret guitar in *standard tuning* [118]. The matrix of pitch labels is generated by computing the start and end time of each audio analysis frame and, from the MIDI file corresponding to the appropriate audio file, retrieving the pitches that occur during that time span. For each analysis frame, a one is entered into the matrix at the positions corresponding to the sounding pitches.

An additional binary matrix of frame polyphony labels $Y^{(poly)} \in \{0,1\}^{n \times p}$ is computed, where p is the maximum number of notes that may sound during an analysis frame. Though a standard guitar with six strings is only capable of producing six notes simultaneously, it may be that a chord transition occurs within an audio analysis

frame and so the maximum polyphony increases above this bound. The equation

$$p = \max_i \left(\left(Y^{(pitch)} 1 \right)_i \right) + 1, \tag{3.3}$$

such that $1^{k \times 1}$ is a vector of ones, computes the maximum polyphony that the pitch estimation model will consider. The addition of one to the maximum polyphony is to accommodate silence where no pitches sound in an analysis frame. In terms of model construction, this value is calculated using the training data partition only. Using this information, the polyphony label for the i^{th} audio analysis frame is a one-hot binary vector such that $Y_{ij}^{(poly)} = 1$ for $j = \left(Y^{(pitch)} 1 \right)_i$. In other words, a binary vector is formed for each analysis frame with a single one at the index of the polyphony level.

The aforementioned label matrices, $Y^{(pitch)} \in \{0,1\}^{n \times k}$ and $Y^{(poly)} \in \{0,1\}^{n \times p}$, are stacked horizontally into one matrix of feature labels, $Y \in \{0,1\}^{n \times (k+p)}$, to be used for training and testing the DBN pitch estimation algorithm.

The Python bindings for the *Marsyas* opensource framework for audio processing [113] were used to extract STFT or MFCC features from audio recordings using the sliding window approach described in this section.

3.1.2 Deep Belief Network Structure

The structure of the proposed deep belief network for polyphonic guitar transcription is illustrated in Figure 3.3. The input layer consumes normalized audio features $\phi \in [0,1]^m$ and therefore consists of m nodes. There can be any number of stochastic binary hidden layers, each consisting of any number of nodes. Since this is a deep network, it is assumed that the number of hidden layers is greater than one. The output layer of the network consists of $k + p$ nodes, where $k = 51$ output nodes are allocated for pitch estimation and correspond to the range of pitches from $C2$–$D6$,

and p (Equation 3.3) output nodes are allocated for polyphony estimation. Each hidden and output layer node activation is transformed using the nonlinear sigmoid activation function to induce a Bernoulli probability distribution.

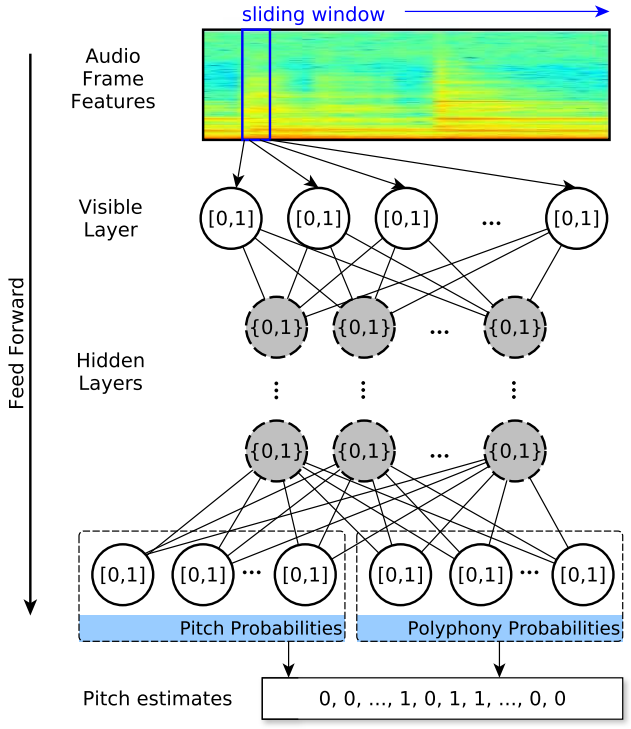


Figure 3.3: Structure of the deep belief network for polyphonic pitch estimation. The input layer accepts normalized features $\phi \in [0,1]^m$ and the output layer estimates a probability for each pitch and polyphony level. Weights are omitted for clarity.

Towards the end goal of computing frame-level pitch estimates, the feature vectors Φ are fed forward through the deep belief network with parameters Θ, resulting in a matrix of probabilities $\mathrm{P}(\hat{Y}|\Phi,\Theta) \in [0,1]^{k+p}$ that is then split into a matrix of pitch probabilities $\mathrm{P}(\hat{Y}^{(pitch)}|\Phi,\Theta)$ and polyphony probabilities $\mathrm{P}(\hat{Y}^{(poly)}|\Phi,\Theta)$. For notation clarification, Y represents the training and testing labels, $\hat{Y}$ represents the label predictions made by the classifier, and $\mathrm{P}(\hat{Y}|\Phi,\Theta)$ denotes the probability of each output node being activated during an audio analysis frame.

Pitch estimation is performed using a multi-label learning technique similar to the *MetaLabeler* system [108], which trains a multi-class classifier for label cardinality estimation. Instead of using the matrix of pitch probabilities as features for a separate polyphony classifier, increased recall was noted by training the polyphony classifier alongside the pitch classifier using the original audio features. The polyphony of the i^{th} analysis frame is estimated by selecting the polyphony class with the highest probability using the equation

$$\rho_i = \underset{j}{\operatorname{argmax}} \left(\mathrm{P}(\hat{Y}_{ij}^{(poly)} | \Phi_i, \Theta) \right). \tag{3.4}$$

Formally, the pitches sounding in the i^{th} analysis frame are estimated by selecting the indices of the ρ_i highest pitch probabilities (Equation 3.4) produced by the DBN. With these estimates, the corresponding vector of pitch probabilities is converted to a binary vector $\hat{Y}_i^{(pitch)} \in \{0,1\}^k$ by turning on bits that correspond to the ρ_i highest pitch probabilities. Performing this step on all audio analysis frames results in a binary matrix of pitch estimates $\hat{Y}^{(pitch)} \in \{0,1\}^{n \times k}$.

3.1.3 Deep Belief Network Training

The network training procedure involves stacking a sequence of RBMs and pretraining each in succession before conducting supervised fine-tuning on the network as a whole. This training procedure is outlined in detail in Section 2.4 of the previous chapter. The MIDI-annotated audio recordings are partitioned into two sets for training and testing the performance of the network.

For the supervised training fine-tuning step, the canonical error function to be minimized for a set of separate binary classifications is the cross-entropy error function

$$E(\Theta) = -\sum_{i=1}^{n}\sum_{j=1}^{k+p} Y_{ij} \ln \mathrm{P}(\hat{Y}_{ij}|\Phi_i, \Theta) + (1 - Y_{ij}) \ln(1 - \mathrm{P}(\hat{Y}_{ij}|\Phi_i, \Theta)), \tag{3.5}$$

such that $Y \in \{0,1\}^{n\times(k+p)}$ is the binary matrix of ground-truth labels for all training instances, $\mathrm{P}(\hat{Y}|\Phi,\Theta) \in [0,1]^{n\times(k+p)}$ is the matrix of label probabilities output by the DBN for all training instances, and Θ is the current model parameters. The aim of this objective function is to adjust the network weights to pull output node probabilities closer to one for ground-truth label bits that are on and to pull probabilities closer to zero for bits that are off.

Construction of the network, training, and testing were performed using the *Theanos* numerical computation library for Python [14]. Theanos allows many matrix computations to be performed in parallel on the graphics processing unit (GPU), significantly speeding up the computation time required for training and testing.

3.2 Note Tracking

After training, an entire audio recording may be decomposed into a sequence of analysis frame features and presented to the pitch estimation algorithm. The result is a matrix of pitch activation estimates $\hat{Y}^{(pitch)} \in \{0,1\}^{n\times k}$ for all analysis frames. Now it is necessary to postprocess the resulting pitch estimates to determine when note events begin and end.

The simplest temporal estimation method is to slice the pitch estimation matrix $\hat{Y}^{(pitch)}$ column-wise to isolate the activations of each pitch across the analysis frames spanning the audio recording. Iterating over each analysis frame, a new note event is

formed when the pitch estimate transitions from off to on (onset) and the note event ends when the pitch estimate transitions from on to off (offset).

A more sophisticated method of note temporal estimation is the HMM frame-smoothing algorithm proposed by Poliner and Ellis [84]. This algorithm is the predominantly used method in academic transcription systems. The algorithm allows a frame-level pitch estimate to be contextualized amongst its neighbours instead of solely trusting the independent pitch estimates made by a classification algorithm on individual audio analysis frames. The algorithm may extend the length of notes, shorten the length of notes, or correct analysis frames exhibiting false negatives.

The frame-smoothing algorithm operates by training an HMM for each pitch. In the context of the transcription algorithm proposed in this work, 51 HMMs are trained to postprocess the frame-level activation estimates for the pitches $C2$–$D6$. Each HMM is composed of two hidden states: `NOTE ON` and `NOTE OFF`, representing the true pitch activation of the analysis frame. The input observations to an HMM is a binary vector of activation estimates for the corresponding pitch over all analysis frames of an audio recording. Hence, the discrete alphabet size of each HMM is two. The output of the Viterbi algorithm, which searches for the optimal underlying state sequence of the HMM, is a revised binary vector of activation estimates for a single pitch.

The transition probabilities of each HMM govern the temporal dynamics of a note event, specifically the predisposition for a note event to arise, sustain, or cease. Using a set of ground-truth MIDI-annotated audio recordings, the HMM transition probabilities are trained by observing the frequency with which a pitch transitions between and within the `ON` and `OFF` states across analysis frames. Moreover, the initial state distribution of the HMM is trained in a similar fashion, by observing the frequency with which an audio recording begins with the pitch being processed.

The emission distribution of each HMM state is a Bernoulli distribution that models the certainty of each frame-level pitch estimate. By construction, the underlying HMM state at any analysis frame t is $q_t \in \{\texttt{ON}, \texttt{OFF}\}$ and the estimated pitch activation (observation) at any analysis frame t is $o_t \in \{\texttt{ON}, \texttt{OFF}\}$. If the underlying HMM state is ON then $P(o_t = \texttt{ON} | q_t = \texttt{ON})$ models the probability of a true positive and $P(o_t = \texttt{OFF} | q_t = \texttt{ON})$ models the probability of a false negative. If the underlying HMM state is OFF then $P(o_t = \texttt{ON} | q_t = \texttt{OFF})$ models the probability of a false positive and $P(o_t = \texttt{OFF} | q_t = \texttt{OFF})$ models the probability of a true negative.

These emission distributions can be estimated by calculating the percentage of true and false positives and negatives from the output of the pitch estimation algorithm relative to the ground-truth training labels. However, Poliner [81] discovered that a better approach is to use the probability of a pitch being active in an analysis frame, as computed by a machine learning classifier, as the emission distribution. In effect, this approach leverages the certainty with which the pitch estimation algorithm believes in its estimate. In its most basic form, the HMM emission distribution is stationary over time; however, this approach necessitates a non-stationary emission distribution that updates at each time step to reflect the beliefs of the pitch estimation algorithm.

Figure 3.4 illustrates the structure of each HMM as well as example input and output. To summarize, each HMM frame smoother requires as input a binary observation sequence of a single pitch across all analysis frames of an audio recording in tandem with the probabilities of that pitch's activation, as output by the DBN. The output of each HMM frame smoother is a revised sequence of frame-level pitch activation estimates. These results can be aggregated across all of the HMMs to produce a revised matrix of pitch activation estimates $\hat{Y}'^{(pitch)} \in \{0, 1\}^{n \times k}$.

Output of the transcription algorithm at each stage—from feature extraction to DBN pitch estimation to HMM frame smoothing—is displayed in Figure 3.5 for a

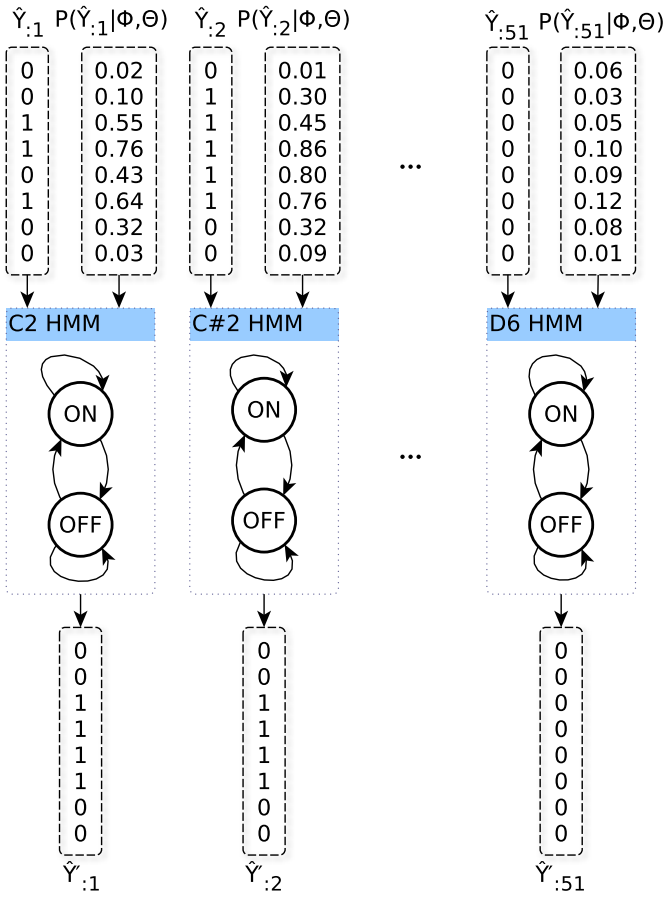


Figure 3.4: Structure of the 51 hidden Markov models used to smooth the frame-level estimates for the pitches $C2$–$D6$. Example input and output is provided. $\hat{Y}_{:i}$ denotes the binary pitch estimates of the i^{th} pitch across all analysis frames, while $P(\hat{Y}_{:i}|\Phi,\Theta)$ indicates the probability of these estimates.

four-second segment of a synthesized guitar recording. The pitch probabilities output by the DBN show that the classifier is quite certain about its estimates; there are few grey areas indicating indecision. Note that the short, spurious notes present after DBN pitch estimation are removed in the note tracking process.

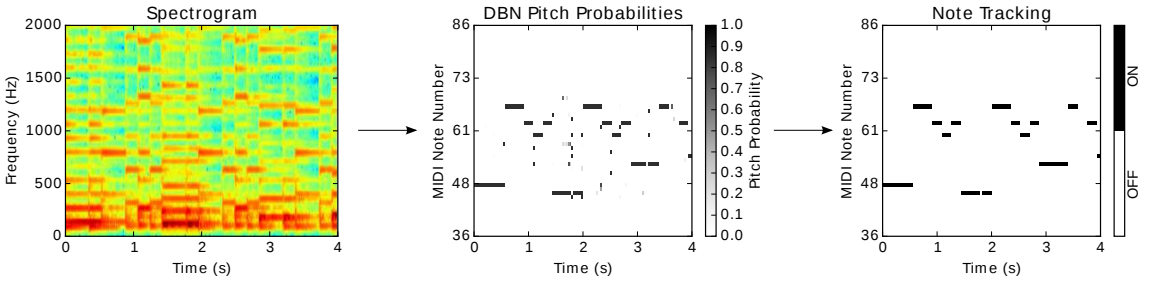


Figure 3.5: An overview of the transcription workflow on a four-second segment of a synthesized guitar recording.

The HMMs described in this section were implemented using a modified version of the *hmmlearn* Python library[1] that was recently orphaned from the well-known *scikit-learn* machine learning library.[2]

3.3 Onset Quantization

Revisiting the earlier discussion surrounding the impact of window size w on feature quality, it was noted that increasing w decreases time resolution but increases frequency resolution. On the contrary, decreasing w increases time resolution but decreases frequency resolution. In terms of note pitch estimation, a smaller window size lowers the chance that chord transitions occur within an analysis frame because it spans a smaller period of time, but yields more frames to be processed by the pitch estimation algorithm at a lower frequency resolution. Since frequency resolution is paramount to pitch estimation because it allows nearby frequencies to be differentiated from each other, a smaller window size is arguably undesirable. A larger window size may capture more chord transitions within an analysis frame, which raises the frame-level polyphony and makes pitch detection more difficult, but yields less analysis frames to be processed by the pitch detection algorithm and offers a finer frequency resolution.

[1] `https://github.com/hmmlearn/hmmlearn`
[2] `http://scikit-learn.org/stable`

The work in this thesis opts for a larger window size, and hence a finer frequency resolution at the expense of a coarser time resolution. Consider a window size of $w = 2048$ samples at an audio sampling rate of $f_s = 11025$ Hz. If the HMM frame smoothing algorithm claims a pitch arises within in an analysis frame, it could onset at any time within the $w/f_s \approx 186$ ms window. The algorithm could arbitrarily place onsets at the beginning, end, or middle of the analysis frame but that merely provides a superficial solution to the problem.

To mitigate this issue, a state-of-the-art onset detection algorithm described by Dixon [34] is run at a finer time resolution than the DBN pitch estimation algorithm to pinpoint the exact time of note event onsets. This onset detection algorithm uses spectral flux, which measures changes in spectral content between sequential audio analysis frames, to form a detection function. A constant threshold $\alpha \in [0, 1]$ is applied to the detection function to select local maxima that correspond to probable onsets in the audio signal. For more detail on the operation of onset estimation algorithms, refer to Section 2.2.2 of the previous chapter.

The list of time estimates computed by the onset detection algorithm is used in conjunction with the revised frame-level pitch estimates of the HMM smoothing algorithm, $\hat{Y}'^{(pitch)} \in \{0, 1\}^{n \times k}$, to form a MIDI file that documents the note events (pitch, onset time, and duration) occuring in the audio recording. If a pitch estimate transitions from off to on between consecutive analysis frames, a corresponding onset estimate is searched for that is within $2w/f_s$ seconds of the beginning of the analysis frame. If found, the onset time estimate is used as the time for the beginning of the note; otherwise, the time stamp of the beginning of the analysis frame is used. The note offset is calculated by following the pitch estimate across consecutive analysis frames until it transitions from on to off, at which point the time stamp of the end of this analysis frame is used.

The Marsyas audio processing framework [113] provides an implementation of the onset estimation algorithm described by Dixon [34], which is used to obtain onset estimates from audio recordings. The *python-midi* library is used to generate MIDI files from the note event estimates produced by the transcription algorithm proposed in this work.[3]

3.4 Common Western Music Notation Generation

The MIDI file output by the note onset quantizer contains the note event (pitch, onset, and duration) transcriptions of an audio recording. However, a MIDI file lacks certain information necessary to write sheet music in common Western music notation such as time signature, key signature, clef type, and the value (duration) of each note described in divisions of a whole note.

There are several robust opensource programs that derive this missing information from a MIDI file using logic and heuristics in order to generate common Western music notation that is digitally encoded in the *MusicXML* file format.[4] MusicXML is a standardized extensible markup language (XML) definition allowing digital symbolic music notation to be universally encoded and parsed by music applications. In this work, the command line tools shipped with the opensource application *MuseScore* are used to convert MIDI to common Western music notation encoded in the MusicXML file format.[5]

[3] `https://github.com/vishnubob/python-midi`
[4] `http://www.musicxml.com`
[5] `http://musescore.org`

3.5 Guitar Tablature Arrangement

The graph-based guitar tablature arrangement algorithm proposed by Burlet and Fujinaga [19] is used to append a guitar string and fret combination to each note event encoded in a MusicXML transcription file. The guitar tablature arrangement algorithm operates by using Dijkstra's algorithm to search for the shortest path through a directed weighted graph, in which the vertices represent candidate string and fret combinations for a note or chord.

The edge weights between nodes in the graph indicate the biomechanical difficulty of transitioning between fretting-hand positions. Three biomechanical complexity factors are aggregated to form each edge weight: the fret-wise distance required to transition between notes or chords, the fret-wise finger span required to perform chords, and a penalty of one if the fretting hand surpasses the seventh fret. The value of this penalty and fret threshold number were determined through subjective analysis of the resulting tablature arrangements. In the event that a note is followed by a chord, the fret-wise distance is calculated by the expression

$$\left| f - \frac{\max(g) - \min(g)}{2} \right|, \tag{3.6}$$

such that $f \in \mathbb{N}$ is the fret number used to perform the note and g is a vector of fret numbers used to perform each note in the chord.

The source code for this guitar tablature arrangement algorithm is opensource and available on GitHub.[6] A more detailed review of graph-search algorithms for guitar tablature arrangement can be found in Section 2.3 of the previous chapter.

[6] `https://github.com/gburlet/astar-guitar`

3.6 Summary of Contributions

The polyphonic guitar transcription algorithm described in this chapter consists of a novel pitch estimation algorithm that addresses three arguable shortcomings in modern pattern recognition approaches to pitch estimation: first, the task of estimating multiple pitches sounding simultaneously is often approached using multiple one-versus-all binary classifiers [82, 75] in lieu of estimating the presence of multiple pitches using a single classifier; second, there exists no standard method to impose constraints on the polyphony of pitch estimates at any given time; and third, the discriminative power of latent audio feature representations, as produced by deep neural networks and autoencoders, are often overlooked in favour of more traditional features such as the STFT. In response to these points, the pitch estimation algorithm described in this work uses a deep belief network (DBN) in conjunction with multi-label learning techniques to produce multiple pitch estimates for each audio analysis frame that conform to the polyphony constraints of the input instrument.

The pitch estimates output by the DBN are then input to several existing algorithms in the literature to produce common Western music notation. The music notation is then input into a previously developed guitar tablature arrangement algorithm [19] to generate tablature that may be saved, edited, or published online for the reference of other guitarists.

Chapter 4

Transcription Evaluation

The previous chapter presented the developed polyphonic guitar transcription algorithm consisting of a novel pitch estimation algorithm followed by existing algorithms in the literature for note temporal estimation and symbolic music notation generation. The algorithms following the developed pitch estimation algorithm are well established in the MIR community and formal evaluations of their output have already been conducted. For results of these evaluations, refer to the following conference proceedings: for note tracking [84]; for note onset estimation [34]; and for guitar tablature arrangement [19]. Therefore, the evaluation of the implemented polyphonic guitar transcription algorithm will focus on the primary contribution of this work: the proposed DBN note pitch estimation algorithm.

This chapter begins with an overview of the compiled ground-truth dataset of synthesized guitar recordings and accompanying MIDI note annotations used for training and testing the implemented transcription algorithm. Section 4.2 provides a description of the metrics used to evaluate the multi-label pitch estimates of the DBN as well as metrics used to evaluate the note event estimates of the entire polyphonic guitar transcription algorithm. Section 4.3 proposes several hypotheses

and experiments to evaluate the influence of specific algorithm parameter values on the accuracy of the note pitch estimates. Additionally, an experiment is outlined that compares the accuracy of note event transcriptions made by the proposed algorithm to a state-of-the-art, single-instrument polyphonic transcription digital signal processing algorithm [129].

4.1 Ground-truth Dataset

Ideally, the note pitch estimation algorithm proposed in this work should be trained and tested using recordings of acoustic or electric guitars that are subsequently hand-annotated with the note events being performed. In practice, however, it would be expensive to fund the compilation of such a dataset and there is a risk of annotation error. Unlike polyphonic piano transcription datasets that are often created using a mechanically controlled piano, such as a Yamaha Disklavier, to generate acoustic recordings that are time aligned with note events in a MIDI file, mechanized guitars are not widely available. Therefore, the most feasible course of action for compiling a polyphonic guitar transcription dataset is to synthesize a set of ground-truth note events using an acoustic model of a guitar.

Using the methodology proposed by Burlet [18, 19], a ground-truth dataset of 45 synthesized acoustic guitar recordings paired with MIDI note-event annotations was compiled. The dataset was created by harvesting the wealth of crowd-sourced guitar tablature transcriptions uploaded in the *Guitar Pro* file format to `www.ultimate-guitar.com`. Guitar Pro is a popular desktop application that supports symbolic music notation engraving, editing, and synthesis.[1] The application allows guitarists to input a tablature transcription of their favourite musical work,

[1] `http://www.guitar-pro.com`

which is then saved as a digitally encoded binary file that can be shared with other guitarists, converted to a MIDI file, or synthesized as an audio file. The encoded note events can be synthesized using a variety of guitar models, which imitate the acoustic properties of different guitar brands with different string types.

The songs in the ground-truth dataset were selected by using the advanced search function on `www.ultimate-guitar.com` with the keyword "acoustic" and filtering the results to only display Guitar Pro files—as opposed to transcriptions in plain text format—that have been rated by the community as five out of five stars. Guitar Pro song transcriptions were selected based on popularity, which was inferred using the number of users who rated the transcription. In the end, 45 Guitar Pro transcriptions were selected for the dataset. A detailed list of guitar tracks in the ground-truth dataset can be found in Appendix B.

Before synthesis, several modifications were made to each Guitar Pro file to ensure a standardized dataset. Guitar Pro does not offer any command line tools for batch processing and the proprietary file format prevents the creation of a script that can parse, modify, and write a revised version of the encoded symbolic music score. Therefore, the graphical user interface of the Guitar Pro desktop application must be used to individually process each song transcription file. The following modifications were made to each Guitar Pro file:

i. Guitar Pro files encoding a music score often consist of multiple tracks representing the transcriptions of multiple instruments other than guitar, such as bass, drums, piano, or vocals. All superfluous tracks other than guitar are removed. If multiple guitar tracks are present, such as a rhythm and lead track, the tracks are blended into one to include some rhythm and some lead riffs.[2] Rhythm

[2] A *riff* is a sequence of notes or chords that are performed on a guitar (Appendix A).

guitar tracks often consist of high-polyphony chords while lead guitar tracks often consist of low-polyphony chords, individual notes, and solos.

ii. Bars of rests, indicating a period of silence, are removed from the beginning and end of the remaining guitar track.

iii. Repeated bars are modified so that they are only synthesized once.

iv. The following note ornamentations are removed: dead notes, palm muting, note let ring, harmonics, pitch bends, hammer on, hammer off, slides, tremolo, and vibrato. See Appendix A for a description of these guitar performance techniques. Many of these ornamentations affect the spectrogram of notes or chords, potentially affecting the performance of the pitch estimation algorithm. Although these note ornamentations often occur in real guitar recordings, they make the problem of automatic music transcription even more difficult. Transcription algorithms should be developed for notes without ornamentations before considering more difficult extensions of the problem.

v. The gain of the guitar track is set to -2.6 decibels to avoid clipping on notes that are to be performed loudly.

vi. The guitar model for note synthesis is set to a Martin & Co. acoustic guitar with steel strings and no *capo*. The function of a guitar capo is explained in Appendix A.

Each preprocessed Guitar Pro file, encoding a guitar transcription on a single track, is synthesized and exported as a WAV file using the Guitar Pro desktop application. Additionally, the Guitar Pro file is exported as a MIDI file, which contains the ground-truth note event annotations for the synthesized guitar track. The resulting

ground-truth dataset consists of 45 audio files and 45 MIDI files.[3] The distribution of note pitches in the dataset is displayed in Figure 4.1.

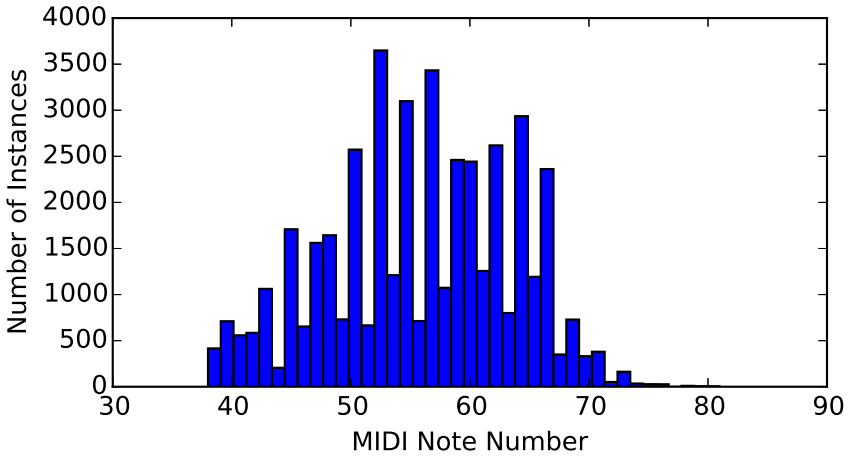


Figure 4.1: Distribution of note pitches in the ground-truth dataset.

4.2 Evaluation Metrics

Several multi-label learning metrics are used to evaluate the DBN note pitch estimation algorithm proposed in this work. Moreover, several information retrieval metrics are used to evaluate the polyphonic transcription algorithm proposed in this work. The purpose of these metrics is to determine the quality of frame-level pitch estimates and note event estimates produced by the algorithms described in the previous chapter. To reiterate, the note temporal estimation algorithm [84], note onset estimation algorithm [34], and guitar tablature arrangement algorithm [19] used in this work are not explicitly evaluated because formal evaluations of their output have already been conducted and presented to the MIR community.

[3]The MIDI files in the ground-truth dataset are publicly available at `https://archive.org/details/DeepLearningIsolatedGuitarTranscriptions`.

4.2.1 Note Pitch Estimation Metrics

Given the pitch estimates output by the DBN pitch estimation algorithm for n audio analysis frames, $\hat{Y}^{(pitch)} \in \{0,1\}^{n \times k}$, and the corresponding ground-truth pitch label matrix for the corresponding audio analysis frames, $Y^{(pitch)} \in \{0,1\}^{n \times k}$, the following metrics can be computed:

- Precision:

$$p = \frac{1(\hat{Y}^{(pitch)} \,\&\, Y^{(pitch)})1}{1\hat{Y}^{(pitch)}1}, \tag{4.1}$$

  such that the logical operator & denotes the element-wise `AND` of two binary matrices and 1 indicates a vector of ones. In other words, this equation calculates the number of correct pitch estimates divided by the number of pitches the algorithm predicts are present across the audio analysis frames.

- Recall:

$$r = \frac{1(\hat{Y}^{(pitch)} \,\&\, Y^{(pitch)})1}{1Y^{(pitch)}1}, \tag{4.2}$$

  such that the logical operator & denotes the element-wise `AND` of two binary matrices and 1 indicates a vector of ones. In other words, this equation calculates the number of correct pitch estimates divided by the number of ground-truth pitches that are active across the audio analysis frames.

- f-measure:

$$f = \frac{2pr}{p + r}, \tag{4.3}$$

  such that p and r is the precision and recall calculated using Equation 4.1 and Equation 4.2, respectively. The f-measure calculated in Equation 4.3 is the balanced f-score, which is the harmonic mean of precision and recall. In other words, precision and recall are weighted evenly.

- Polyphony recall:

$$r_{\text{poly}} = \frac{\sum_{i=1}^{n} \mathbb{1}\{(\hat{Y}^{(pitch)}1)_i = (Y^{(pitch)}1)_i\}}{n}, \tag{4.4}$$

such that $\mathbb{1}\{\cdot\}$ is an indicator function that returns 1 if the predicate is true, and n is the number of audio analysis frames being evaluated. In other words, this equation calculates the number of correct polyphony estimates across all audio analysis frames divided by the number of analysis frames.

- One error: given the matrix of pitch probabilities $P(\hat{Y}^{(pitch)}|\Phi, \Theta) \in [0,1]^{n \times k}$ output by the DBN with model parameters Θ when processing the input audio analysis frame features Φ, the predominant pitch of the i^{th} audio analysis frame is calculated using the equation

$$j = \underset{j}{\operatorname{argmax}} \Big[P(\hat{Y}_{ij}^{(pitch)}|\Phi_i, \Theta) \Big], \tag{4.5}$$

which can then be used to calculate the one error:

$$one\ err = \frac{\sum_{i=1}^{n} \mathbb{1}\{Y_{ij}^{(pitch)} \neq 1\}}{n}, \tag{4.6}$$

such that $\mathbb{1}\{\cdot\}$ is an indicator function that maps to 1 if the predicate is true. The one error calculates the fraction of analysis frames in which the top-ranked label is not present in the ground-truth label set. In the context of pitch estimation, this metric provides insight into the number of audio analysis frames where the predominant pitch—often referred to as the melody—is estimated incorrectly.

- Hamming loss:

$$hamming\ loss = \frac{1(\hat{Y}^{(pitch)} \oplus Y^{(pitch)})1}{nk}, \tag{4.7}$$

such that n is the number of audio analysis frames, k is the cardinality of the label set for each analysis frame, and the boolean operator $\oplus$ denotes the element-wise `XOR` of two binary matrices. The hamming loss provides insight into the number of false positive and false negative pitch estimates across the audio analysis frames.

4.2.2 Polyphonic Transcription Metrics

Several information retrieval metrics are also used to evaluate the note event estimates produced by the polyphonic transcription algorithm described in the previous chapter, which consists of a note pitch estimation algorithm followed by a note temporal estimation algorithm. Given an input audio recording, the polyphonic transcription algorithm outputs a set of note event estimates in the form of a MIDI file. A corresponding ground-truth MIDI file contains the set of true note events for the audio recording. Each note event contains three pieces of information: pitch, onset time, and offset time.

The MIREX, an annual evaluation of MIR algorithms, has a *multiple fundamental frequency estimation and note tracking* category in which polyphonic transcription algorithms are evaluated. The MIREX metrics used to evaluate polyphonic transcription algorithms are:

- Precision:

$$p = \frac{|\hat{N} \cap N|}{|\hat{N}|}, \tag{4.8}$$

such that $\hat{N}$ is the set of estimated note events and N is the set of ground-truth note events.

- Recall:

$$r = \frac{|\hat{N} \cap N|}{|N|}, \tag{4.9}$$

  such that $\hat{N}$ is the set of estimated note events and N is the set of ground-truth note events.

- f-measure:

$$f = \frac{2pr}{p + r}, \tag{4.10}$$

  such that p and r are calculated using Equation 4.8 and Equation 4.9, respectively.

The criteria for a note event being correct, as compared to a ground-truth note event, are as follows:

- The pitch name and octave number of the note event estimate and ground-truth note event must be equivalent.

- The note event estimate's onset time is within ±250ms of the ground-truth note event's onset time.

- Only one ground-truth note event can be associated with each note event estimate.

The offset time of a note event is not considered in the evaluation process because offset times exhibit less perceptual importance than note onset times [28].

Each of these evaluation metrics can also be calculated under the condition that octave errors are ignored. Octave errors occur when the algorithm predicts the correct pitch name but incorrectly predicts the octave number. Octave errors are prevalent in digital signal processing fundamental frequency estimation algorithms because high-energy harmonics can be misconstrued as a fundamental frequency, resulting in an incorrect estimate of the octave number [64]. Reporting the evaluation metrics

described in this section under the condition that octave errors are ignored will reveal whether machine learning transcription algorithms also succumb to a high number of octave errors.

4.3 Hypotheses and Experiments

The described polyphonic transcription algorithm is controlled by many parameters. Moreover, the underlying DBN pitch estimation algorithm requires a considerable amount of computing time to sufficiently train. Though iterating over combinations of parameters to find an optimal configuration is desirable, this method is infeasible due to the amount of time required for training.

Therefore, the hypotheses and experiments in this section are intended to hone in on good parameter configurations, while discovering how the implemented transcription algorithm operates under certain conditions. Detailed in this section, five experiments are proposed to evaluate the DBN note pitch estimation algorithm and one experiment is proposed to evaluate the polyphonic transcription algorithm as a whole.

There are several DBN training parameters that remain constant across each experiment. The number of epochs (training iterations) for pretraining is set to 400; the number of epochs for network fine-tuning is set to 30000. The convergence threshold, which ceases training if the value of the objective function between epochs does not fluctuate more than the threshold, is set to $1E-18$ for both pretraining and fine-tuning. The learning rate for both pretraining and fine-tuning is set to 0.05. To allow large datasets to be processed, the training data is partitioned into batches consisting of 1000 training instances each. For pretraining, 1-step contrastive divergence is used. These parameters were selected on the basis of preliminary tests.

4.3.1 Note Pitch Estimation Evaluation

The following five hypotheses and experiments are concerned with the DBN note pitch estimation algorithm whose workflow is depicted in Figure 4.2. The experiments evaluate the frame-level pitch estimates of the DBN. For each experiment, the ground-truth dataset (Section 4.1) is partitioned into a training and testing dataset such that 80% of the guitar tracks are allocated for training and 20% of the guitar tracks are allocated for testing. Features and labels are then extracted from audio analysis frames spanning each synthesized audio signal in both the training and testing set. The training instances are randomly shuffled before training to avoid overfitting the model to a single guitar track.

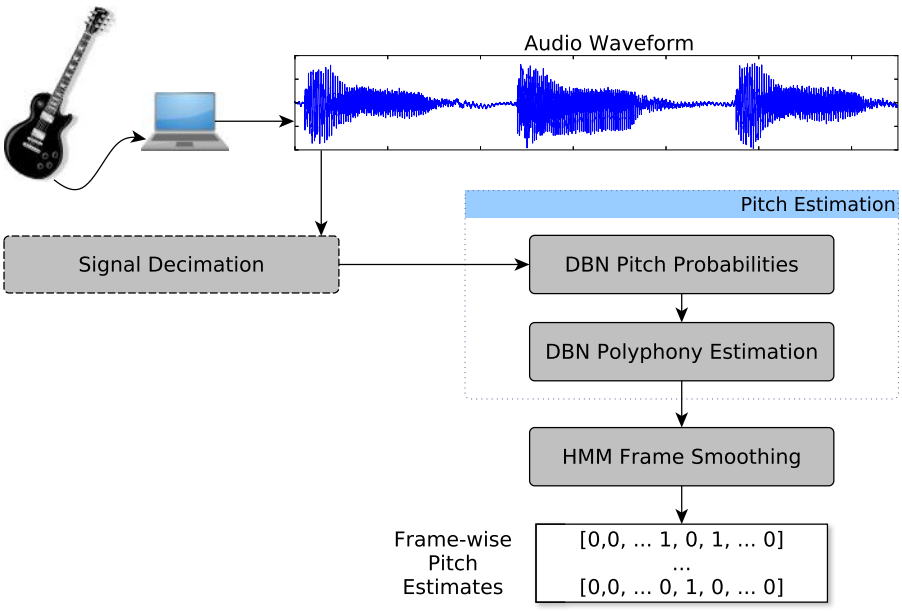


Figure 4.2: Pitch estimation workflow that is evaluated in Experiments 1–5. Note that this workflow only includes the pitch estimation algorithm and frame-smoothing algorithm.

Since the HMM frame-smoothing algorithm also affects the quality of pitch estimates, the frame-level pitch estimates after frame smoothing are also evaluated.

The transition matrices of the HMM frame-smoothing algorithm are computed by observing the frequency with which a pitch transitions between and within the `ON` and `OFF` states across analysis frames in the training dataset.

4.3.1.1 Audio Sampling Rate

Hypothesis: Increasing the audio sampling rate above 11025 Hz will not improve pitch estimation f-measure.

Rationale: Looking at the spectrogram of the synthesized guitar track for the song "Paranoid Android" composed by Radiohead (Figure 4.3), though there is some high-frequency signal energy, the majority lies within the frequency range 0 Hz to 5100 Hz. This is within the Nyquist frequency for the audio sampling rate 11025 Hz. Therefore, the rationale is that the majority of discriminative data for pitch estimation lies within this frequency range.

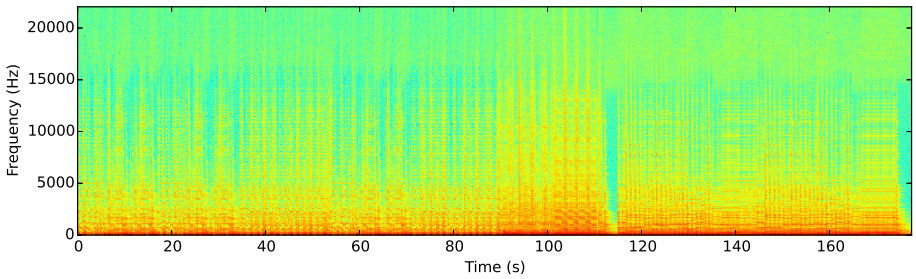


Figure 4.3: Spectrogram for the song "Paranoid Android" by Radiohead.

Experiment 1: Table 4.1 outlines the pitch estimation algorithm variables for Experiment 1. This experiment sets the audio sampling rate f_s as the independent variable. Values of the controlled variables were selected based on preliminary tests. The f-measure (Equation 4.3) over all analysis frames is the dependent variable.

Additionally, the other evaluation metrics described in Section 4.2.1 are also reported. If the f-measure is highest for f_s = 11025 Hz then the hypothesis is confirmed.

Table 4.1: Independent, controlled, and dependent variables for Experiment 1.

Variable	Value
Audio sampling rate, f_s (Hz)†	11025 | 22050 | 44100
Window size, w (samples)	1024 | 2048 | 4096
Hop size, h (samples)	75% of window size
Number of hidden layers	3
Number of nodes per layer	ℓ_1: 250, ℓ_2: 250, ℓ_3: 1000
Features	DFT power spectrum
f-measure‡	

† Denotes the independent variable.

‡ Denotes the dependent variable.

Note that the window size changes with the sampling rate: when f_s = 11025 Hz then w = 1024 samples but when f_s = 44100 Hz then w = 4096 samples. Although the window size changes with the dependent variable, it is still considered a controlled variable because it implicitly controls for frequency resolution. For example, at f_s = 11025 Hz and w = 1024 samples the frequency resolution of the DFT is $\Delta f = f_s/w \approx 10.7$ Hz. Multiplying both the audio sampling rate and the window size by a factor of two results in the same frequency resolution. Frequency resolution is an important variable to control because it impacts whether two nearby frequencies can be distinguished and therefore influences pitch estimation.

4.3.1.2 Audio Analysis Window Size

Hypothesis: Increasing the audio analysis window size will improve pitch estimation f-measure up to a point in which further increases to the window size will decrease pitch estimation f-measure.

Rationale: This hypothesis stems from the balance between STFT frequency resolution and time resolution. Increasing window size results in greater frequency resolution but poorer time resolution, while decreasing window size results in a coarser frequency resolution but finer time resolution. Taking this into consideration, the rationale behind this hypothesis is that there exists a middle value for the window size that optimizes frame-level pitch estimation f-measure.

Experiment 2: Table 4.2 outlines the pitch estimation algorithm variables for Experiment 2. This experiment sets the window size, and hence, the frequency resolution of the DFT, as the independent variable. Values of the controlled variables were selected based on preliminary tests. The f-measure (Equation 4.3) over all analysis frames is the dependent variable. Other evaluation metrics described in Section 4.2.1 are also reported. If the f-measure increases as window size increases, and then f-measure begins to decrease with further increases to w, then the hypothesis is confirmed.

Table 4.2: Independent, controlled, and dependent variables for Experiment 2.

Variable	Value
Audio sampling rate, f_s (Hz)	22050
Window size, w (samples)†	1024, 2048, 4096
Hop size, h (samples)	75% of window size
Number of hidden layers	3
Number of nodes per layer	ℓ_1: 350, ℓ_2: 350, ℓ_3: 1100
Features	DFT power spectrum
f-measure‡	

† Denotes the independent variable.

‡ Denotes the dependent variable.

4.3.1.3 Network Structure

Hypothesis: Including an "associative memory" [45] before the output layer of the DBN will increase pitch estimation f-measure relative to other common network structures.

Rationale: In the application of a DBN to the task of MNIST handwritten digit recognition [31], Hinton et al. [45] discovered that having a large number of nodes in the final hidden layer results in relatively good classification accuracy. Moreover, this layer—called the "associative memory"—plays an important role when using the DBN as a generative model, which lies outside the scope of this work. It is reasonable to assume that the benefits of using an "associative memory" layer will transfer from computer vision to computer audition.

Experiment 3: Table 4.3 describes the pitch estimation algorithm variables for Experiment 3. This experiment sets the number of nodes in each layer as the independent variable. Three network structures are evaluated: a network with an increasing number of nodes in each hidden layer, a network with a decreasing number of nodes in each layer, and a network with an associative memory layer. Values of the controlled variables were selected based on preliminary tests. The f-measure (Equation 4.3) over all analysis frames is the dependent variable. Again, other evaluation metrics described in Section 4.2.1 are also reported. The hypothesis is confirmed if the network with an associative memory layer yields the highest f-measure.

4.3.1.4 Number of Network Hidden Layers

Hypothesis: Increasing the number of hidden layers in the DBN will increase pitch estimation f-measure.

Table 4.3: Independent, controlled, and dependent variables for Experiment 3.

Variable	Value
Audio sampling rate, f_s (Hz)	22050
Window size, w (samples)	2048
Hop size, h (samples)	75% of window size
Number of hidden layers	3
Number of nodes per layer†	ℓ_1: 600, ℓ_2: 400, ℓ_3: 200
	ℓ_1: 200, ℓ_2: 400, ℓ_3: 600
	ℓ_1: 300, ℓ_2: 300, ℓ_3: 1200 (*associative memory*)
Features	DFT power spectrum
f-measure‡	

† Denotes the independent variable.

‡ Denotes the dependent variable.

Rationale: Hinton et al. [45] also noted that increasing the number of network layers is guaranteed to improve a lower bound on the log likelihood of the training data. In other words, the worst-case performance of the DBN is theoretically guaranteed to improve as hidden layers are added. Furthermore, taking a step above their shallow counterparts, deep networks provide a closer approximation to the expanse of neurons in the human brain. From a biological perspective, a sensible assumption is that as more layers of neurons are added to the DBN, the model further replicates the auditory perception power of the human brain and therefore, the f-measure of the pitch estimation algorithm should increase.

Experiment 4: Table 4.4 describes the pitch estimation algorithm variables for Experiment 4. This experiment sets the number of hidden layers as the independent variable, while keeping the number of nodes in each layer constant. Values of the controlled variables were selected based on preliminary tests. The f-measure (Equation 4.3) over all analysis frames is the dependent variable, as well as the other

evaluation metrics described in Section 4.2.1. The hypothesis is confirmed if the f-measure increases as the number of hidden layers increases.

Table 4.4: Independent, controlled, and dependent variables for Experiment 4.

Variable	Value
Audio sampling rate, f_s (Hz)	22050
Window size, w (samples)	2048
Hop size, h (samples)	75% of window size
Number of hidden layers†	2, 3, 4
Number of nodes per layer	300
Features	DFT power spectrum
f-measure‡	

† Denotes the independent variable.

‡ Denotes the dependent variable.

4.3.1.5 Audio Features

Hypothesis: Inputting MFCC features to the DBN pitch estimation algorithm will result in a higher f-measure than STFT features.

Rationale: Though the viability of DFT power spectrum features for pitch estimation have been confirmed by Nam et al. [75], the viability of MFCC features for DBN pitch estimation have not yet been investigated. Inputting MFCC features to a DBN have yielded exceptional results in other audio problem domains such as speech recognition [29, 43, 98], which labels phonemes present in acoustic features of speech. Thus, it is reasonable to assume that MFCC features will also perform well for the task of pitch estimation.

Experiment 5: Table 4.5 describes the pitch estimation algorithm parameters for Experiment 5. This experiment sets the input feature set as the independent variable. Several different numbers of MFCC coefficients are evaluated. Values of the controlled

variables were selected based on preliminary tests. The f-measure (Equation 4.3) over all analysis frames is the dependent variable, along with the other evaluation metrics presented in Section 4.2.1. The hypothesis is confirmed if any of the MFCC feature sets yield a higher f-measure than inputting STFT features to the DBN pitch estimation algorithm.

Table 4.5: Independent, controlled, and dependent variables for Experiment 5.

Variable	Value
Audio sampling rate, f_s (Hz)	22050
Window size, w (samples)	2048
Hop size, h	75% of window size
Number of hidden layers	3
Number of nodes per layer	ℓ_1: 350, ℓ_2: 350, ℓ_3: 1100
Features†	DFT power spectrum
	1024 MFCCs
	512 MFCCs
	256 MFCCs
f-measure‡	

† Denotes the independent variable.

‡ Denotes the dependent variable.

4.3.2 Polyphonic Transcription Evaluation

The following experiment is concerned with evaluating the note event estimates computed by the polyphonic transcription algorithm whose workflow is depicted in Figure 4.4. Note that this workflow includes the pitch estimation algorithm, temporal estimation algorithms, and writing to MIDI, whereas the workflow for Experiments 1–5 only consisted of the pitch estimation algorithm and frame-smoothing algorithm. The transition matrices of the HMM frame-smoothing algorithm are computed by observing the frequency with which a pitch transitions between and within the `ON` and `OFF` states across analysis frames in the training dataset.

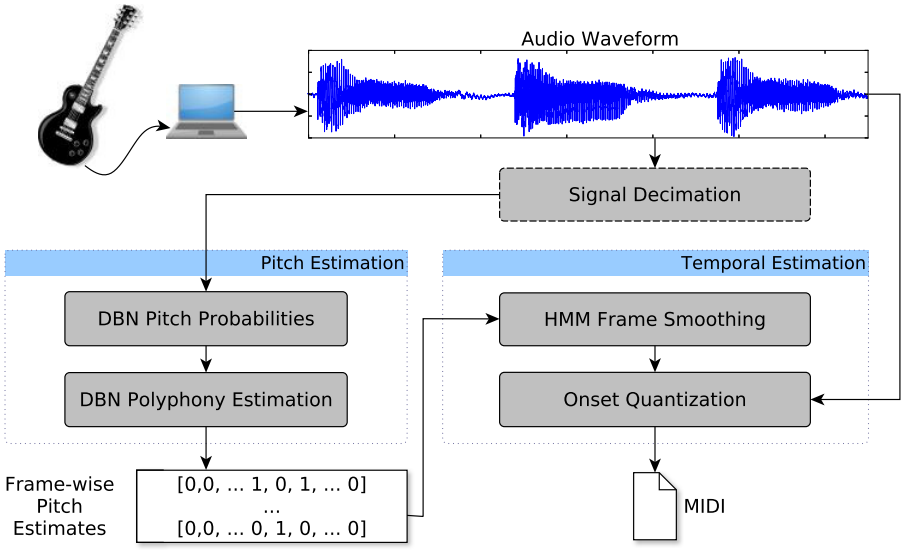


Figure 4.4: Polyphonic transcription workflow that is evaluated in Experiment 6.

Hypothesis: The f-measure of note event estimates made by the polyphonic transcription algorithm proposed in this work will outperform that of the current state-of-the-art, single-instrument polyphonic transcription algorithm [129].

Rationale: The rationale for this hypothesis stems from the recent advances in polyphonic transcription made possible by machine learning algorithms [75, 84]. Machine learning algorithms are trending in the field of MIR because they often outperform their digital signal processing counterparts [7].

Experiment 6: In this experiment, the lessons learned from previous experiments are considered when setting the parameters of the DBN. These parameters are revealed in the following chapter after a critical analysis of the results of the first five experiments. The note event transcriptions made by the algorithm proposed in this work are compared to the current state-of-the-art, single instrument polyphonic transcription algorithm [129]. The note-event f-measure (Equation 4.10) is used to

quantitatively evaluate the output of each transcription algorithm. If the algorithm proposed in this thesis has a higher f-measure, then the hypothesis is confirmed.

The Zhou and Reiss polyphonic transcription algorithm [129] processes audio signals at a sampling rate of f_s = 44100 Hz and makes use of the RTFI analysis technique [128] to estimate fundamental frequencies. A window size of w = 441 samples and a hop size of h = 441 samples is set by the authors for optimal transcription performance [129]. For a detailed description of the implementation details of this algorithm, refer to [129, 130]. To evaluate this algorithm, Python bindings were written for the opensource C++ code distributed by the authors.[4] The resulting Python application is opensource and available on GitHub.[5]

[4] `http://www.vamp-plugins.org/plugin-doc/qm-vamp-plugins.html\#qm-transcription`
[5] `http://github.com/gburlet/zhoutranscription`

Chapter 5

Results and Discussion

The previous chapter introduced the dataset used for training and testing the polyphonic guitar transcription algorithm proposed in this work, along with metrics and experiments to evaluate the accuracy of different components under various conditions. This chapter presents the results of these experiments and provides a critical analysis of the results. This is followed by a discussion of the positive and negative aspects of the algorithm, its applicability to other instruments, and its commercial viability.

The experiments performed in this chapter were run on a machine with an Intel® Core™ i7 3.07 GHz quad core CPU, 24 GB of RAM, and an Nvidia GeForce GTX 970 GPU with 1664 CUDA cores.

5.1 Note Pitch Estimation Results

5.1.1 Audio Sampling Rate

Hypothesis:

Increasing the audio sampling rate above 11025 Hz will not improve pitch estimation f-measure (Equation 4.3).

Results and Discussion:

Table 5.1 presents the results of Experiment 1. Strictly speaking, the hypothesis is *refuted* because the frame-level pitch estimation f-measure is the lowest when f_s = 11025 Hz. However, the results show that as the sampling rate is increased, there is little increase in pitch estimation f-measure. Performing a Tukey-Kramer honest significance test on the f-measures of songs in the test dataset for each DBN shows no significant differences between the models. This finding is also reflected in the other frame-level pitch estimation metrics: as the audio sampling rate increases above 11025 Hz, the one error, hamming loss, and polyphony recall show marginal or no improvements at the expense of significant increases to network training time. At 11025 Hz, the network took approximately 4 hours to train, while at 44100 Hz (standard compact disc sampling rate) the network took approximately 8 hours to train. In summary, the results suggest that 11025 Hz is an appropriate sampling rate for polyphonic pitch estimation and if training time is not an issue, a higher sampling rate of 22050 Hz should be used.

This result is somewhat surprising given the psychoacoustical properties of human audition. Humans are capable of hearing frequencies from 20 Hz – 20000 Hz, and so

Table 5.1: Results of Experiment 1.

	Sampling Rate (Hz)	Precision	Recall	f-measure	One Error	Hamming Loss	Polyphony Recall	Training Time (M:S)	Classification Time (S)
Before HMM	11025	0.642	0.594	0.617	0.196	0.043	0.437	232:28	0.22
	22050	0.655	0.601	**0.627**	0.219	0.042	0.455	379:57	0.24
	44100	0.648	0.594	0.620	0.212	0.042	0.438	480:13	0.27
After HMM	11025	0.743	0.605	0.667	–	0.035	–	–	–
	22050	0.754	0.615	**0.678**	–	0.034	–	–	–
	44100	0.748	0.593	0.662	–	0.035	–	–	–

an audio sampling rate of 40000 Hz is necessary to properly encode this frequency range, according to the Nyquist-Shannon Sampling Theorem. Since trained musicians are capable of performing this task with some degree of accuracy [40], it makes sense to model an algorithm that mimics the psychoacoustic processes of humans. However, the experimental results reveal that only a subset of this frequency range (0 Hz – 5512.5 Hz) is necessary for guitar pitch discrimination. Consequently, it may be true that the neural networks in our brain that are responsible for attributing pitch to incoming acoustical signals only consider these frequency ranges as well.

Apart from the results pertinent to the hypothesis, another noteworthy result is the one error. The one error reveals that the predominant pitch is incorrectly estimated in $\approx 20\%$ of analysis frames, which could be an indication of lack of training data. For example, there is one song in the testing dataset that has several pitches for which there is little to no corresponding training data. Therefore, improvements to this metric could be accomplished with the compilation of more isolated guitar tracks to be used as training data. Note that the one error is not reported after HMM frame smoothing because the note tracking algorithm does not affect this evaluation metric.

With respect to polyphony estimation, the results reveal that the ≈ 45% polyphony recall likely hinders the frame-level f-measure of the pitch estimation algorithm. The maximum frame-level polyphony with a window size of 2048 samples and a sampling rate of 22050 Hz is 13. Therefore, the polyphony classifier must choose between 13 different class labels to estimate the polyphony of an analysis frame. Investigating further, when using the ground-truth polyphony for each frame, an f-measure of 0.68 is noted before HMM smoothing when using a sampling rate of 22050 Hz. The 5% increase in f-measure reveals that the polyphony estimates are close to their ground-truth value and that further improvements to pitch estimation f-measure are to be accomplished with modifications to the DBN to influence the probabilities of pitch estimates. Note that the polyphony recall is not reported after HMM frame smoothing because the note tracking algorithm does not affect this evaluation metric.

5.1.2 Audio Analysis Window Size

Hypothesis:

Increasing the audio analysis window size will improve pitch estimation f-measure up to a point in which further increases to the window size will decrease pitch estimation f-measure (Equation 4.3).

Results and Discussion:

Table 5.2 presents the results of Experiment 2. The results reveal an inverse relationship between window size and frame-level pitch estimation f-measure: as the window size increases, the f-measure decreases. Drastic increases in polyphony recall are also seen as the window size decreases, which is likely the cause of the increased f-measure. The effect of smaller windows—at the expense of a coarser frequency resolution—is that less

note or chord transitions are likely to occur in an audio analysis frame. Consequently, the maximum frame-level polyphony (Equation 3.3) will be lower, resulting in less classification labels for the polyphony classifier to choose from, and hence, less chance for errors.

Table 5.2: Results of Experiment 2.

	Window Size (samples)	Precision	Recall	f-measure	One Error	Hamming Loss	Polyphony Recall	Training Time (M:S)	Classification Time (S)
Before HMM	1024	0.671	0.615	**0.642**	0.198	0.038	0.520	748:33	0.34
	2048	0.651	0.595	0.623	0.214	0.042	0.455	417:06	0.24
	4096	0.635	0.576	0.604	0.193	0.048	0.370	257:50	0.20
After HMM	1024	0.761	0.631	**0.690**	–	0.031	–	–	–
	2048	0.740	0.596	0.660	–	0.036	–	–	–
	4096	0.745	0.592	0.660	–	0.039	–	–	–

However, there is a cost to pay for lowering the window size. A smaller window size means that more analysis frames are needed to span the input audio recordings, and thus, more training and testing instances are presented to the pitch estimation algorithm. At a window size of 1024 samples, the network training time is roughly 12.5 hours. Moreover, with a window size of 1024 samples at a sampling rate of 22050 Hz, the frequency resolution of each bin of the DFT is ≈ 22 Hz which is becoming quite coarse. With these parameters the DFT is incapable of individually measuring the energy of the fundamental frequencies of pitches that are two semitones apart in the lower frequency range of the guitar. For example, the fundamental frequencies of the pitches $C2$ and $D2$ are mapped to the same frequency bin of the DFT, making pitch discrimination more difficult.

More investigation is needed to confirm or refute the hypothesis. The pitch estimation algorithm is trained and tested with a smaller window size of 512 samples. Given the results (Table 5.2), an even smaller window size seems encouraging, but at this value the frequency resolution of the DFT will drop to ≈ 43 Hz, which is becoming very coarse in the lower pitch range of the guitar. The result before HMM smoothing is an f-measure of 0.630, a one error of 0.271, a hamming loss of 0.038, a polyphony recall of 0.556, and a classification time of 0.51 seconds. Setting the window size to 512 samples finally results in degraded pitch estimations due to the coarseness of the DFT frequency resolution. Adding insult to injury, the network training time took approximately 22 hours. From this result, the hypothesis is *confirmed*; a window size of 1024 samples is the point at which further increases to window size result in decreased pitch estimation f-measure.

A Friedman statistical test is run on the f-measures of songs in the test dataset for each DBN trained in this experiment in order to determine if the performance of the models are significantly different from each other. With a p-value of 0.108, at $\alpha = 0.05$ we conclude that, although a window size of 1024 samples yielded the best results in the experiment, it does not significantly outperform the other models. A Tukey-Kramer honest significance test on the f-measures of songs for each model also corroborates this finding.

5.1.3 Network Structure

Hypothesis:

Including an "associative memory" [45] before the output layer of the DBN will increase pitch estimation f-measure (Equation 4.3) relative to other common network structures.

Results and Discussion:

Table 5.3 presents the results of Experiment 3. The network structures investigated in this experiment each have three layers $\{\ell_1, \ell_2, \ell_3\}$ with variable numbers of nodes:

i. ℓ_1: 600 nodes, ℓ_2: 400 nodes, ℓ_3: 200 nodes

ii. ℓ_1: 200 nodes, ℓ_2: 400 nodes, ℓ_3: 600 nodes

iii. ℓ_1: 300 nodes, ℓ_2: 300 nodes, ℓ_3: 1200 nodes (*associative memory*)

Table 5.3: Results of Experiment 3.

	Network Structure	Precision	Recall	f-measure	One Error	Hamming Loss	Polyphony Recall	Training Time (M:S)	Classification Time (S)
Before HMM	i.	0.650	0.583	0.615	0.207	0.043	0.416	555:32	0.26
	ii.	0.640	0.586	0.612	0.219	0.043	0.417	493:45	0.29
	iii.	0.643	0.589	0.614	0.215	0.043	0.414	574:14	0.31
After HMM	i.	0.742	0.594	0.660	–	0.036	–	–	–
	ii.	0.732	0.607	0.664	–	0.036	–	–	–
	iii.	0.734	0.602	0.661	–	0.036	–	–	–

Strictly speaking, the hypothesis is *refuted* because the frame-level pitch estimation f-measure is not the highest for the associative memory network structure. Practically speaking, the result of this experiment reveals that network structure has little impact on the f-measure of the frame-level pitch estimates; the f-measure deviates by mere tenths of a percent for each network structure evaluated. This is confirmed by performing a Friedman statistical test (p-value: 0.459) and a Tukey-Kramer honest significance test on the f-measures of songs in the test dataset for each DBN, which shows no significant differences between the models. Hence, the associative memory

network structure is completely viable, as well as network structures with increasing or decreasing numbers of nodes.

Given that the network structure has little impact on the accuracy of pitch estimates, what insights does this provide regarding the model? The results suggest that regardless of the network structure being evaluated, the final hidden layer of derived latent features are equally discriminative for pitch estimation. However, the first network structure ends in 200 nodes, the second ends in 600 nodes, and the third ends in 1200 nodes. From this we can gather that the feature set can be significantly compressed, potentially down to 200 nodes, without seeing diminishing f-measure. More formally, there exists a transformation of the original feature set into a lower dimensional subspace such that the transformed feature set retains the ability to discriminate between pitch classes.

5.1.4 Number of Network Hidden Layers

Hypothesis:

Increasing the number of hidden layers in the DBN will increase pitch estimation f-measure (Equation 4.3).

Results and Discussion:

The hypothesis speculated that increasing the number of hidden layers, and consequently the number of model parameters, would increase frame-level pitch estimation f-measure. The rationale for this hypothesis was motivated by the mechanics of the human brain, which consists of hundreds of thousands of connected neurons that are responsible for auditory perception and attributing pitch to audio signals. Considering this biological model, it is reasonable to assume that increasing the number of hidden

layers in the deep network will yield increasingly better results; however, the results presented in Table 5.4 provide evidence supporting the contrary.

Table 5.4: Results of Experiment 4.

	Number of Layers	Precision	Recall	f-measure	One Error	Hamming Loss	Polyphony Recall	Training Time (M:S)	Classification Time (S)
Before HMM	2	0.675	0.601	**0.636**	0.192	0.040	0.463	329:24	0.19
	3	0.650	0.600	0.623	0.200	0.042	0.452	422:52	0.22
	4	0.643	0.591	0.616	0.211	0.043	0.433	512:33	0.25
After HMM	2	0.760	0.604	**0.673**	–	0.034	–	–	–
	3	0.739	0.610	0.669	–	0.035	–	–	–
	4	0.728	0.602	0.659	–	0.036	–	–	–

The results invalidate the hypothesis and suggest that a more complex model does not correlate positively with model performance. Rather, the results show that the number of hidden layers is negatively correlated with pitch estimation f-measure. As the number of hidden network layers is increased, the precision and recall of the frame-level note pitch estimates decrease. However, the decrease in f-measure is quite minimal: roughly -1% f-measure for each additional layer. Confirming how minimal these changes are, a Tukey-Kramer honest significance test on the f-measure of songs in the test dataset for each DBN trained in this experiment shows no significant differences between the models. Though the f-measures of each model are not significantly different, the trend of decreasing f-measure as the number of network layers increases is still apparent.

There are several potential causes of this result. First, increasing the complexity of the model could have resulted in overfitting the network to the training data. Second, the issue of "vanishing gradients" [13] could be occurring in the network

fine-tuning training procedure, whereby the training signal passed to lower layers gets lost in the depth of the network. Yet another potential cause of this result is that the pretraining procedure may have found insufficient initial edge weights for networks with increasing numbers of hidden layers. On a more general note, the machine learning research community has yet to discover a robust method of training deep neural networks. Pretraining involves 1-step contrastive divergence, which provides a mediocre approximation of the gradient of the log likelihood objective function (Equation 2.7) used to determine a good direction of change for the model parameters. Therefore, more robust approximations of this gradient would result in better initial parameters for the deep network.

5.1.5 Audio Features

Hypothesis:

Inputting MFCC features to the DBN pitch estimation algorithm will result in a higher f-measure (Equation 4.3) than the power spectrum of DFT features.

Results and Discussion:

Table 5.5 presents the results of Experiment 5. The hypothesis is undoubtedly *refuted* because the pitch estimation f-measure when using the power spectrum of STFT features is significantly higher than the f-measure when using MFCC features, regardless of how many MFCCs are considered. When training the networks that consume MFCC features, each fine-tuning process stopped prematurely (i.e., did not reach 30000 epochs) because the gradient descent optimization had reached a local minima in the objective function. This differs from training networks that consume

the power spectrum of DFT features, where the fine-tuning process halts at the set maximum of 30000 epochs.

Confirming these results, a Friedman statistical test on the f-measures of songs in the test dataset for each DBN trained in this experiment shows that, at $\alpha = 0.05$, the models do differ in a significant way (p-value: 0.0005). Investigating further, a Tukey-Kramer honest significance test unsurprisingly shows that there is a significant difference between the DBN model trained on DFT power spectrum features and each of the DBN models trained on MFCC features.

Table 5.5: Results of Experiment 5.

	Features	Precision	Recall	f-measure	One Error	Hamming Loss	Polyphony Recall	Training Time (M:S)	Classification Time (S)
Before HMM	DFT	0.640	0.60	**0.617**	0.210	0.043	0.437	319:14	0.19
	MFCC:1024	0.206	0.079	0.114	0.794	0.062	0.317	226:07	0.23
	MFCC:512	0.210	0.081	0.117	0.790	0.062	0.316	195:42	0.19
	MFCC:256	0.209	0.080	0.116	0.791	0.0626	0.315	145:55	0.19
After HMM	DFT	0.745	0.612	**0.672**	–	0.035	–	–	–
	MFCC:1024	0.275	0.128	0.175	–	0.209	–	–	–
	MFCC:512	0.281	0.131	0.178	–	0.203	–	–	–
	MFCC:256	0.276	0.128	0.175	–	0.204	–	–	–

Neural networks are often described as "black boxes" because it is not trivial to attribute an output network activation to a specific component of the model or a set of model parameters. Therefore, it is difficult to troubleshoot why one network fails to find a good set of parameters while other networks settle on weights that yield good pitch estimates. Ultimately, the latent features composing the penultimate layer of the deep neural network are responsible for the efficacy of pitch estimates.

The generated latent features should exhibit enough discriminative information to differentiate between pitch classes. The results suggest that the network training procedure is unable to find weights that are able to transform the input MFCC features into a set of latent features that are adequate for pitch differentiation.

The significant difference between STFT features and MFCC features is that the measurements of signal energy at specific frequencies are logarithmically spaced in the case of MFCCs. Furthermore, the MFCC measurements of the signal energy at specific frequencies are also in the logarithmic domain in an attempt to mimic how humans perceive loudness. A reduction in pitch estimation f-measure could be a result of either of these attributes of the feature set. For instance, a finer resolution of the high-frequency information of the audio signal could be integral to forming discriminative features for pitch estimation. Moreover, a large measurement of signal energy at a specific frequency is mapped to a much smaller value in the logarithmic domain. Therefore, it may be difficult for the network to attribute peaks in the frequency domain of the audio signal to harmonics of a fundamental frequency, and consequently a pitch, if the peaks become smaller and blend into the other frequency content of the signal.

5.1.6 Summary of Results

In summary, the previous five experiments provided several insights into how the model operates and which properties of the feature set are important for pitch estimation. First, a *sampling rate of 22050 Hz*, which measures the signal energy at frequencies up to 11025 Hz, is desirable but requires a longer amount of time for model training because there are more audio samples to process. If model training time is an issue, a sampling rate of 11025 Hz offers slightly less discriminative information for pitch

estimation but offers significantly faster training times. Second, a *window size of 1024 samples* offers the best trade off between frequency resolution and time resolution. Lowering the window size decreases frequency resolution and starts to hinder the f-measure of pitch estimates. Increasing the window size above 1024 samples decreases the time resolution and hinders polyphony recall, and thus, pitch recall. Third, the structure of the network has little effect on pitch estimates produced. A more important discovery is that the *dimensionality of the latent feature set can be reduced* without hindering the f-measure of pitch estimates. Fourth, and perhaps counterintuitively, *increasing the number of hidden layers in the network actually hinders the f-measure of pitch estimates.* Finally, modifying the power spectrum of the DFT in an attempt to mimic how humans perceive pitch and loudness (MFCC audio features) produces inferior pitch estimates; *the power spectrum of the DFT yields far superior results.*

5.2 Polyphonic Transcription Results

Table 5.6 describes the polyphonic transcription algorithm parameters for Experiment 6. One new parameter is the *onset threshold*, which governs the sensitivity of selecting note onset candidates for the onset detection algorithm; a value of 0.3 denotes that onset candidates with more than 30% confidence are selected. This onset threshold was selected by running the onset detection algorithm on guitar recordings in the training dataset. The onsets output by the algorithm were sonified as short bursts of white noise—an audio signal with a uniform distribution of frequencies—which were superimposed over each input audio signal. The onset threshold was iteratively adjusted until a good balance was found between false positives and false negatives.

After pretraining the DBN with the parameters outlined in Table 5.6, fine-tuning is performed on the network. Figure 5.1 displays the value of the negative log likelihood

Table 5.6: Independent, controlled, and dependent variables for Experiment 6.

Variable	Value
Audio sampling rate, f_s	22050 Hz
Window size, w	1024 samples
Hop size, h	768 samples
Number of hidden layers	2
Number of nodes per layer	ℓ_1: 400, ℓ_2: 300
Features	DFT power spectrum
Onset threshold	0.3
Algorithm†	Burlet, Zhou and Reiss [129]
f-measure‡	

† Denotes the independent variable.

‡ Denotes the dependent variable.

function of the fine-tuning step over each training epoch. Analyzing this plot uncovers a couple of interesting conclusions. First, the negative log likelihood is stagnant over the first 3500 traning epochs before finding an appropriate gradient with which to modify the network weights to begin minimizing the objective function. The structure of this plot reflects a plateau in the landscape of the objective function that then begins descending towards a local minima. After this point there begins a steep decline in the negative log likelihood which begins to level off around 20000 epochs. Subsequent epochs offer comparitively little improvement in the negative log likelihood.

Before presenting the note event metrics for the polyphonic transcription algorithm as a whole, the frame-level pitch estimation metrics are first reported in Table 5.7 for the DBN pitch estimation algorithm using the parameters described in Table 5.6. With a frame-level f-measure of 0.70 after HMM frame smoothing, the results are slightly improved in comparison to those seen in Experiments 1–5, where the highest reported frame-level f-measure was 0.69.

Now that we have measurements of the frame-level f-measure for each of the experiments, another interesting facet of analysis is plotting f-measure versus model

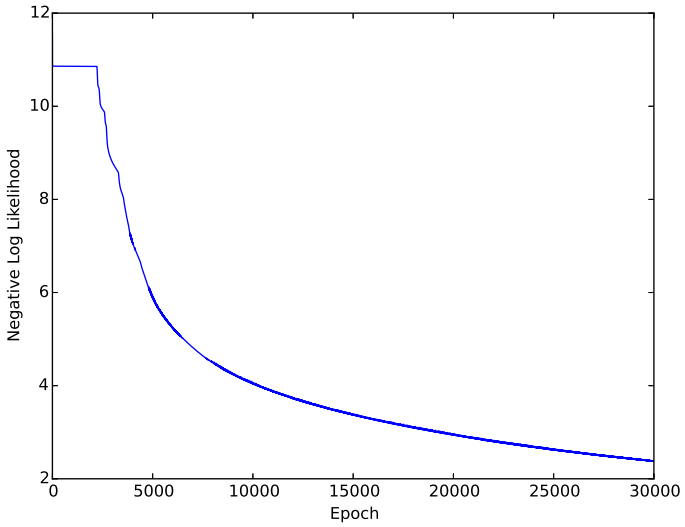


Figure 5.1: Plot of the negative log likelihood for the fine-tuning step of the final network trained for Experiment 6.

Table 5.7: Frame-level pitch estimation metrics of the DBN transcription algorithm using the final set of parameters.

	Precision	Recall	f-measure	One Error	Hamming Loss	Polyphony Recall	Training Time (M:S)	Classification Time (S)
Before HMM	0.683	0.630	**0.655**	0.184	0.036	0.519	312:22	0.27
After HMM	0.742	0.661	**0.70**	–	0.031	–	–	–

training time (Figure 5.2). The graph also displays the *Pareto frontier*, which delineates the boundary between optimal and non-optimal allocation of resources. It is desirable to minimize model training time while maximizing model performance (f-measure). At one end of the Pareto frontier is the model that requires the least amount of

training time, while the other end is the model that results in the highest f-measure. The points that lie on the frontier are the models with which improvements to one metric, such as training time or f-measure, can not be made with hindering the other metric. The models to the right of the frontier are *dominated* by the other models and are therefore non-optimal. The model outlined in Experiment 6 (Table 5.6) is located on the frontier.

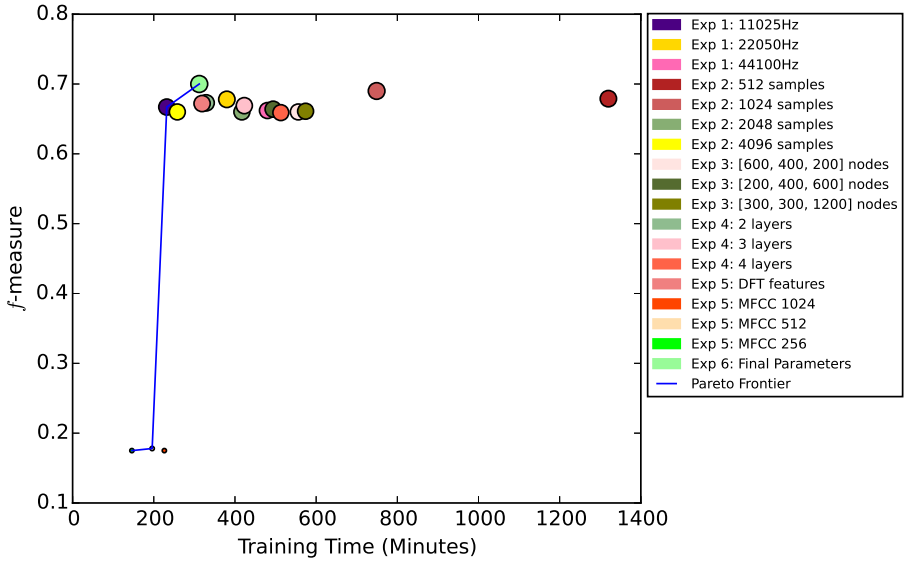


Figure 5.2: Pareto frontier for the six experiments, which seeks to minimize model training time while maximizing frame-level f-measure after HMM frame smoothing.

After frame-level pitch estimation, the polyphonic transcription algorithm continues with its note tracking procedure and writes the estimated note events as a MIDI file. This MIDI file is compared to the ground-truth MIDI file for the corresponding guitar recording and the precision (Equation 4.8), recall (Equation 4.9), and f-measure (Equation 4.10) of the estimated note events are reported. The results are presented in Table 5.8.

The result of this evaluation is an f-measure of 0.629 when considering note octave errors and an f-measure of 0.657 when disregarding note octave errors. A less than

Table 5.8: Precision, recall, and f-measure evaluation of note events transcribed using the DBN transcription algorithm compared to the Zhou and Reiss [130] algorithm.

	DBN Transcription			
	Precision	Recall	f-measure	Runtime (s)
Octave Errors	0.808	0.548	**0.629**	51.36
No Octave Errors	0.844	0.572	0.657	–
	Zhou and Reiss [130]			
	Precision	Recall	f-measure	Runtime (s)
Octave Errors	0.679	0.438	**0.503**	296.10
No Octave Errors	0.755	0.488	0.561	–

3% increase in f-measure when disregarding octave errors provides evidence that the transcription algorithm does not often mislabel the octave number of note events, which is often a problem with digital signal processing transcription algorithms [64]. Note that the frame-level pitch estimation f-measure of 0.70 (Table 5.7) does not translate to an equivalently high f-measure for note events because onset time is considered as well as pitch.

Another interesting property of the transcription algorithm is its conservativeness: the precision of the note events transcribed by the algorithm is 0.808 while the recall is 0.548, meaning that the algorithm favours false negatives over false positives. In other words, the transcription algorithm includes a note event in the final transcription only if it is quite certain of the note's correctness, even if this hinders the recall of the algorithm. Another cause of the high precision and low recall is that when several guitar strums occur quickly in succession, the implemented transcription algorithm often transcribes only the first chord and prescribes it a long duration. This is likely a result of the temporally "coarse" window size of 1024 samples or a product of the HMM frame-smoothing algorithm, which may extend the length of notes causing them to "bleed" into each other. A remedy for this issue is to lower the window

size to increase temporal resolution; however, this has an undesirable side-effect of lowering the frequency resolution of the DFT which is undesirable. A subjective, aural analysis of the guitar transcriptions reflects these results: the predominant pitches and temporal structures of notes occurring in the input guitar recordings are more or less maintained.[1]

Additionally, the guitar recordings in the testing dataset were transcribed by the single-instrument polyphonic transcription algorithm developed by Zhou and Reiss [130]. This algorithm was evaluated in the 2008 MIREX and attained a note event f-measure of 0.76 on a dataset of 30 synthesized and real piano recordings [129], which far surpassed the accuracy of other evaluated transcription algorithms.[2]

Evaluated on the dataset of synthesized guitar recordings compiled in this thesis, the Zhou and Reiss algorithm [130] yielded more errant note event transcriptions relative to the transcriptions generated by the algorithm described in this thesis (Table 5.8). Specifically, the transcription algorithm presented in this work resulted in a 12% increase in f-measure, or 25% relative increase in f-measure, compared to the Zhou and Reiss transcription algorithm when evaluated on the same dataset. With a precision of 0.679 and a recall of 0.438 when considering octave errors, the Zhou and Reiss transcription algorithm also exhibits a significantly higher precision than recall; in this way, it is similar to the transcription algorithm described in this work.

When disregarding octave errors, the f-measure of the Zhou and Reiss transcription algorithm increases by approximately 6%. Therefore, this state-of-the-art digital signal processing transcription algorithm makes two times the number of note octave errors as the transcription algorithm described in this thesis. This result suggests that the

[1] MIDI transcriptions generated by this algorithm are available at `http://archive.org/details/DeepLearningIsolatedGuitarTranscriptions`

[2] `http://www.music-ir.org/mirex/wiki/2008:Multiple\_Fundamental\_Frequency\_Estimation\_\&\_Tracking\_Results\#Piano\_Subset\_Results\_Based\_on\_Onset\_Only`

latent audio features derived by the DBN are well-suited for not only pitch name discrimination, but also note octave discrimination.

5.3 Discussion

Although the polyphonic transcription algorithm described in this thesis significantly outperforms the current state-of-the-art single instrument polyphonic transcription algorithm, there are several detriments that may outweigh the benefits of the algorithm depending on its intended use.

First, there are several benefits of the transcription algorithm:

- As previously mentioned, the accuracy of automatically generated transcriptions surpasses the current state of the art.
- In comparison to other digital signal processing transcription algorithms, the developed polyphonic transcription algorithm exhibits less octave errors.
- The developed polyphonic transcription algorithm can generate transcriptions for full-length guitar recordings in the order of seconds, rather than minutes or hours.
- Given the speed of transcription, the proposed polyphonic transcription algorithm could be adapted for real-time transcription applications, where live performances of guitar are automatically transcribed. This could be accomplished by buffering the input guitar signal into analysis frames as it is performed.
- The trained network weights can be saved to disk such that future transcriptions do not require retraining the model.

- The size of the model is relatively small (less than 12MB) and so the network weights can fit on a portable device or microcontroller.
- Feeding forward audio features through the DBN is a computationally inexpensive task and could operate on a portable device or microcontroller.
- The developed polyphonic transcription algorithm could easily be adapted to accommodate the transcription of other instruments. All that is required is a set of audio files that have accompanying MIDI annotations for supervised training.

On the other hand, there are several detriments of the transcription algorithm:

- The amount of time required to properly train the model is substantial and varies depending on several parameters such as the audio sampling rate, window size, hop size, and network structure. In this thesis, the longest amount of time required to train a model was 22 hours.
- To make training time reasonable, the computations should be outsourced to a GPU that is capable of performing many calculations in parallel. Using a GPU with less CUDA cores, or just a CPU, significantly increases the amount of time required to train the model.
- After training ceases, either by reaching the set number of training epochs or when the objective function stops fluctuating, it is not guaranteed that the resulting network weights are optimal because the training algorithm may have settled at a local minima of the objective function.
- As a consequence of the amount of time required to train the pitch estimation algorithm, it is difficult to search for good combinations of parameters such as audio sampling rate, window size, hop size, and network structure.

- The underlying DBN pitch estimation algorithm is essentially a *black box*. After training, it is difficult to ascertain how the model reaches a solution. This issue is exacerbated as the depth of the network increases.

- The accuracy of polyphony estimates produced by the algorithm leaves room for improvement.

- It is possible to overfit the model to the training dataset. When running the fine-tuning process for another 30000 epochs, the f-measure of the transcription algorithm began to deteriorate due to overfitting. To mitigate against overfitting, the learning rate could be dampened as the number of training epochs increase. Another solution involves the creation of a validation dataset, such that the fine-tuning process stops when the f-measure of the algorithm begins to decrease on the guitar recordings in the validation dataset. The method used in this thesis is *early stopping*, where the fine-tuning process is limited to a certain number of epochs instead of allowing the training procedure to reach a local minima.

Chapter 6

Threats to Validity

The previous chapter described the results of several experiments to evaluate the implemented polyphonic guitar transcription algorithm. Although the algorithm described in this work performs significantly better than the current state-of-the-art, single-instrument polyphonic transcription algorithm [130], some threats to validity exist.

First, guitar synthesis models are used in an attempt to replicate the recordings of a real acoustic guitar. However, the frequency content of synthesized recordings is often non-realistic due to overly simplistic instrument synthesis models. As well, instrument synthesis models often lack variability in the sound of individual notes or chords that are found in real guitar recordings. Therefore, it remains to be seen whether the trained model is capable of transcribing real acoustic guitar recordings with the same degree of accuracy. This threat to validity could be addressed by hiring guitarists and expert musicians to double-key annotate a ground-truth dataset of real guitar recordings. However, the amount of resources required to record and annotate such a dataset is currently infeasible.

Another threat to validity is the sampling methodology used to compile the ground-truth dataset used in this thesis. By gathering acoustic guitar songs on `www.ultimate-guitar.com` that are most popular in that online community leaves an opportunity for sampling bias. It could be that the acoustic guitar songs that are highly rated and most often viewed by individuals in that community have similar pitch distributions, tempos, and note or chord transitions. Therefore, training and testing the developed polyphonic transcription algorithm on these songs alone could lead to optimistic results that do not generalize to the population of guitar songs outside of the sample.

The final threat to validity involves the reproducibility of results. The gradient descent approach to optimizing the highly non-convex objective function of the DBN may result in different network weights each time the model is trained. These different solutions reflect the multitude of local minima present in the objective function. Therefore, slightly different results may occur when retraining a specific network, making experiments difficult to reproduce exactly.

Chapter 7

Conclusion

When applied to the problem of polyphonic guitar transcription, deep belief networks outperform existing single-instrument transcription algorithms. Moreover, the developed transcription algorithm is fast: the transcription of a full-length guitar recording occurs in the order of seconds and is therefore suitable for real-time guitar transcription. As well, the algorithm is adaptable for the transcription of other instruments, such as the bass guitar or piano, as long as the pitch range of the instrument is provided and MIDI annotated audio recordings are available for training.

The polyphonic transcription algorithm described in this thesis is capable of forming discriminative, latent audio features that are suitable for quickly transcribing guitar recordings. The algorithm workflow consists of audio signal preprocessing, feature extraction, a novel pitch estimation algorithm that uses multi-label learning techniques to enforce polyphony constraints, frame smoothing, and onset quantization. The generated note event transcriptions are digitally encoded as a MIDI file, that is processed further to create a *MusicXML* file that encodes the corresponding guitar tablature notation.

An evaluation of the frame-level pitch estimates generated by the deep belief network on a dataset of synthesized guitar recordings resulted in an f-measure of 0.70 after frame smoothing. An evaluation of the note events output by the entire transcription algorithm resulted in an f-measure of 0.629, which is 12% higher than the f-measure reported by a state-of-the-art, single-instrument transcription algorithm [130] on the same dataset.

The results of this work encourage the use of deep architectures such as belief networks or autoencoders to form alternative representations of industry-standard audio features for the purposes of instrument transcription. Moreover, this work demonstrates the effectiveness of multi-label learning for pitch estimation, specifically when an upper bound on polyphony exists.

7.1 Future Work

There are several directions of future work to improve the accuracy of transcriptions. First, there are substantial variations in the distribution of pitches across songs, and so the compilation of more training data is expected to improve the accuracy of frame-level pitch estimates made by the DBN. Second, alternate methods could be explored to raise the accuracy of frame-level polyphony estimates, such as training a separate classifier for predicting polyphony on potentially different audio features. Third, an alternate frame-smoothing algorithm that jointly considers the probabilities of other pitch estimates across analysis frames could further increase pitch estimation f-measure relative to the HMM method [82], which smooths the estimates of one pitch across the audio analysis frames. Finally, it would be beneficial to investigate whether the latent audio features derived for transcribing one instrument are transferable to the transcription of other instruments.

In the end, the big picture is a self-sufficient guitar tablature transcription algorithm that is capable of feeding itself data to improve its transcriptions. There are many guitarists that share manual tablature transcriptions online that would personally benefit from having an automated system capable of generating transcriptions that are almost correct and can subsequently be corrected manually. There is incentive to manually correct the output transcriptions because this method is significantly faster than performing a transcription from scratch, depending on the quality of the automated transcription and the difficulty of the song. The result is a crowdsourcing model that is capable of producing large ground-truth datasets for polyphonic transcription that can then be used to further improve the polyphonic transcription algorithm. Not only would this improve the accuracy of the developed polyphonic transcription algorithm, but it would also provide a centralized repository of ground-truth guitar transcriptions for MIR researchers to train and test future state-of-the-art transcription algorithms.

Appendix A

Guitar Terminology

This appendix defines guitar-specific terminology introduced in this thesis. The following definitions include physical attributes of the guitar, guitar accessories, note ornamentations, and other words that are commonly used in the guitar community.

Frets

Metal dividers embedded in the fretboard on the neck of the guitar. Pressing a string over a specific fret changes the length of the string that is permitted to oscillate, and thus, changing the resulting pitch.

Tuning Knobs

Also called *tuning pegs* or *tuning keys*, these knobs are located on the head of the guitar at the tip of the neck. Each guitar string is wound around a tuning knob, which can be turned to increase or decrease the string tension.

Capo

A device that is clipped onto the fretboard and raises the pitches of the open strings of the guitar.

Riff

A commonly used word that refers to a sequence of notes or chords performed on a guitar. The sequence is often repeated throughout the song.

Dead Note

A type of note ornamentation that instructs the guitarist to touch a string with the fretting hand above a fret in order to create a more percussive sound with little to no discernable pitch.

Palm Muting

A type of note ornamentation that instructs the guitarist to rest the palm of their strumming hand against the strings, resulting in a dampened sound.

Let Ring

A type of note ornamentation that instructs the guitarist to not use their palm to stop the sounding of a note, but rather let it ring indefinitely.

Harmonics

A type of note ornamentation that instructs the guitarist to lightly touch a string above a fret to create a high-pitched note that is difficult or impossible to create using frets.

Bend

A type of note ornamentation that instructs the guitarist to drag a pressed string vertically along the fretboard, which continuously raises the sounding pitch.

Hammer On, Pull Off

A type of note ornamentation that instructs the guitarist to quickly press or depress a fret without exciting the string with a pluck.

Slide

A type of note ornamentation that instructs the guitarist to gradually move their finger between two fret positions along a string, resulting in a gradual increase or decrease in pitch.

Tremolo

A type of note ornamentation that instructs the guitarist to very rapidly and continuously pluck a string, such that there are many note attacks in a vert short period of time.

Vibrato

A type of note ornamentation that instructs the guitarist to quickly twist their wrist while pressing a fret. Consequently, the string bends slightly and periodically changes the pitch.

Appendix B

Dataset Details

This appendix provides a detailed list (Table B.1) of the songs in the ground-truth dataset used for training and testing the polyphonic transcription algorithm described in this thesis. The average polyphony for each track is calculated by dividing the number of note events by the number of chords plus the number of individual notes. Therefore, an average polyphony of one describes a music score consisting of several individual notes, while an average polyphony of six describes a music score consisting of several six-note chords.

The dataset is publicly distributed as 45 musical instrument digital interface (MIDI) files describing the note events occurring in the guitar tracks.[1] The synthesized audio is not distributed with the ground-truth annotations due to copyright issues. Should the audio be desired, the guitar tracks may be synthesized using the Guitar Pro desktop application or using standard MIDI synthesis.[2]

The intended use of the dataset is primarily to further research in automatic guitar transcription. The dataset may be used to train and test machine learning transcription algorithms or to evaluate digital signal processing transcription algorithms.

[1] `https://archive.org/details/DeepLearningIsolatedGuitarTranscriptions`.
[2] `http://www.guitar-pro.com`

Table B.1: Ground-truth polyphonic guitar transcription dataset consisting of 45 popular songs performed on an acoustic guitar.

Artist	Title	Length (m:s)	Tempo (BPM)	Notes	Average Polyphony
3 Doors Down	Here Without You	2:05	145	822	1.67
3 Doors Down	Kryptonite	2:38	100	860	1.75
Alice and Chains	Down in a Hole	2:16	95	681	2.06
Alice in Chains	Nutshell	3:08	58	1172	2.78
Beatles	Blackbird	1:56	93	604	1.32
Beatles	Here Comes the Sun	2:11	132	790	1.89
Beatles	Hey Jude	3:21	80	1538	3.08
Beatles	Norwegian Wood	1:08	95	1146	3.86
Beatles	While My Guitar Gently Weeps	2:15	112	1556	4.25
Beatles	Yesterday	1:42	80	370	1.50
Billy Talent	Pins and Needles	2:36	75	946	1.95
Black Keys	Little Black Submarines	2:29	77	1306	2.93
Chris Cornell	Black Hole Sun	1:57	112	527	1.81
Creed	One Last Breath	2:49	120	879	1.82
Death Cab for Cutie	I Will Follow You Into the Dark	1:50	156	679	1.33
Eric Clapton	Layla	1:32	120	692	2.60
Eric Clapton	Tears in Heaven	2:43	78	601	1.61
Eric Clapton	Wonderful Tonight	2:46	85	545	1.23
Fleetwood Mac	Landslide	2:38	74	710	1.14
Green Day	Good Riddance Time of Your Life	1:34	150	1512	4.82
Incubus	Drive	1:52	90	688	2.34
James Taylor	Fire and Rain	1:52	77	927	2.59
Kansas	Dust in the Wind	2:02	96	787	1.17
Kinks	Lola	2:06	76	1945	4.04
Led Zeppelin	Babe I'm Gonna Leave You	2:48	134	1813	2.77
Led Zeppelin	Over the Hills and Far Way	2:07	90	1686	3.27
Led Zeppelin	Stairway to Heaven	2:52	75	947	2.52
Maroon 5	She Will Be Loved	1:37	106	950	4.66
Metallica	Fade To Black	1:57	112	693	1.80
Metallica	Nothing Else Matters	3:24	69	748	1.59
Muse	Starlight	2:01	120	902	2.03
Neil Young	Old Man	1:56	138	1106	2.24
Nickelback	How You Remind Me	2:24	83	970	2.68
Nirvana	Come as You Are	1:51	120	562	1.63
Oasis	Wonderwall	2:25	88	1420	2.59
Pink Floyd	Brain Damage	3:08	131	798	1.44
Plain White Tees	Hey There Delilah	1:38	100	456	1.49
Radiohead	Creep	2:15	92	702	2.22
Radiohead	Karma Police	2:37	76	1207	2.34

Continued on next page

Table B.1 – Continued from previous page

Artist	Title	Length (m:s)	Tempo (BPM)	Notes	Average Polyphony
Radiohead	Paranoid Android	2:57	85	1750	2.73
Red Hot Chili Peppers	Californication	2:55	96	1447	3.31
Red Hot Chili Peppers	Under the Bridge	3:03	69	1470	3.06
Simon and Garfunkel	The Boxer	1:47	170	585	1.06
The Fray	How to Save a Life	2:12	118	1302	3.03
U2	With or Without You	2:14	114	639	1.24
Averages		2:18	101	987	2.34
Totals		103:56	–	44436	–

Bibliography

[1] J. P. Bello. *Towards the automated analysis of simple polyphonic music: A knowledge based approach*. PhD thesis, Department of Electronic Engineering, Queen Mary University of London, 2003.

[2] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1047, 2005.

[3] J. P. Bello and M. Sandler. Blackboard system and top-down processing for the transcription of simple polyphonic music. In *Proceedings of the COST G-6 Conference on Digital Audio Effects*, Verona, Italy, 2000.

[4] E. Benetos and S. Dixon. Multiple-instrument polyphonic music transcription using a convolutive probabilistic model. In *Proceedings of the Sound and Music Computing Conference*, pages 19–24, Padova, Italy, 2011.

[5] E. Benetos and S. Dixon. Polyphonic music transcription using note onset and offset detection. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 37–40, Prague, Czech Republic, 2011.

[6] E. Benetos and S. Dixon. Multiple-F0 estimation and note tracking for MIREX 2012 using a shift-invariant latent variable model. In *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2012/BD3.pdf`, 2012.

[7] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri. Automatic music transcription: Challenges and future directions. *Journal of Intelligent Information Systems*, 41(3):407–434, 2013.

[8] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchoff, and A. Klapuri. Automatic music transcription: Breaking the glass ceiling. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 1002–1007, Porto, Portugal, 2012.

[9] E. Benetos and T. Weyde. Explicit duration hidden Markov models for multiple-instrument polyphonic music transcription. In *Proceedings of the International*

Conference on Music Information Retrieval, pages 269–274, Curitiba, Brazil, 2013.

[10] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009.

[11] Y. Bengio, F. Bastien, A. Bergeron, N. Boulanger-Lewandowski, T. Breuel, Y. Chherawala, M. Cisse, M. Côté, D. Erhan, J. Eustache, X. Glorot, X. Muller, S. P. Lebeuf, R. Pascanu, S. Rifai, F. Savard, and G. Sicard. Deep learners benefit more from out-of-distribution examples. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 164–172, Fort Lauderdale, FL, 2011.

[12] Y. Bengio, P. Lamblin, D.Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. (19):153–160, 2006.

[13] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.

[14] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference*, pages 3–10, Austin, TX, 2010.

[15] J. A. Bilmes. What HMMs can do. *IEICE Transactions on Information and Systems*, 89(3):869–891, 2006.

[16] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the International Conference on Machine Learning*, pages 1159–1166, Edinburgh, Scotland, 2012.

[17] A. S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound*. MIT Press, Cambridge, MA, 1990.

[18] G. Burlet. Automatic guitar tablature transcription online. Master's thesis, McGill University, 2013.

[19] G. Burlet and I. Fujinaga. Robotaba guitar tablature transcription framework. In *Proceedings of the International Society for Music Information Retrieval*, pages 421–426, Curitiba, Brazil, 2013.

[20] A. M. Burns. Computer vision methods for guitarist left-hand fingering recognition. Master's thesis, McGill University, 2007.

[21] A. M. Burns and M. M. Wanderley. Visual methods for the retrieval of guitarist fingering. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 196–199, Paris, France, 2006.

[22] F.J. Canadas-Quesada, F. Rodriguez-Serrano, P Vera-Candeas, and N. Ruiz Reyes. Improving multiple-f0 estimation by onset detection for polyphonic music transcription. In *IEEE International Workshop on Multimedia Signal Processing*, pages 7–12, Saint Malo, France, 2010.

[23] Celemony: What is Melodyne? `http://www.celemony.com/en/melodyne`. Last Accessed: July 27, 2015.

[24] W. Chang, A. W. Y. Su, C. Yeh, A. Roebel, and X. Rodet. Multiple-F0 tracking based on a high-order HMM model. In *Proceedings of the International Conference on Digital Audio Effects*, Espoo, Finland, 2008.

[25] E. C. Cherry. Some experiments on the recognition of speech, with one and with two ears. *Journal of the Acoustical Society of America*, 25(5):975–979, 1953.

[26] A. Cheveigné and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. *Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[27] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[28] G. Costantini, R. Perfetti, and M. Todisco. Event based transcription system for polyphonic piano music. *Signal Processing*, 89(9):1798–1811, 2009.

[29] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

[30] M. Davy and S. J. Godsill. Bayesian harmonic models for musical signal analysis. In *Proceedings of the Valencia International meeting Bayesian Statistics VII*, Tenerife, Spain, 2002. Oxford University Press.

[31] L. Deng. The MNIST database of handwritten digit images for machine learning research. *Signal Processing Magazine, IEEE*, 29(6):141–142, 2012.

[32] A. Dessein, A. Cont, and G. Lemaitre. Real-time polyphonic music transcription with non-negative matrix factorization and beta-divergence. In *Proceedings of the International Society for Music Information Retrieval Conference*, Utrecht, Netherlands, 2010.

[33] S. Dieleman and B. Schrauwen. End-to-end learning for music audio. In *Proceedings of the International Conference on Acoustic, Speech, and Signal Processing*, pages 6964–6968, Florence, Italy, 2014.

[34] S. Dixon. Onset detection revisited. In *Proceedings of the International Conference on Digital Audio Effects*, pages 133–137, Montréal, QC, 2006.

[35] A. Elisseeff and J. Weston. A kernel method for mutli-labelled classification. (14):681–687, 2002.

[36] V. Emiya, R. Badeau, and B. David. Multipitch estimation of inharmonic sounds in colored noise. In *Proceedings of the International Conference of Digital Audio Effects*, pages 93–98, Bordeaux, France, 2007.

[37] V. Emiya, R. Badeau, and B. David. Automatic transcription of piano music based on HMM tracking of jointly-estimated pitches. In *Proceedings of the European Signal Processing Conference*, Lausanne, Switzerland, 2008.

[38] X. Fiss and A. Kwasinski. Automatic real-time electric guitar audio transcription. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 373–376, Prague, Czech Republic, 2011.

[39] N. F. Fletcher and T. D. Rossing. *The physics of musical instruments,* 2nd ed. Springer, New York, 1998.

[40] S. W. Hainsworth and M. D. Macleod. The automated music transcription problem. Technical report, Department of Engineering, University of Cambridge, 2003.

[41] P. Hamel and D. Eck. Learning features from music audio with deep belief networks. In *Proceedings of the International Society for Music Information Retrieval*, pages 339–344, Utrecht, Netherlands, 2010.

[42] H. Heijink and R. G. J. Meulenbroek. On the complexity of classical guitar playing: Functional adaptations to task constraints. *Journal of Motor Behavior*, 34(4):339–351, 2002.

[43] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[44] G. E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10):428–434, 2007.

[45] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[46] Alex Hrybyk and Youngmoo Kim. Combined audio and video analysis for guitar chord identification. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 159–164, Utrecht, Netherlands, 2010.

[47] X. Huang, A. Acero, and H. Hon. *Spoken Language Processing: A guide to theory, algorithm, and system development*. Prentice Hall, Upper Saddle River, NJ, 2001.

[48] E. Humphrey, J. Bello, and Y. LeCun. Moving beyond feature design: Deep architectures and automatic feature learning in music informatics. In *Proceedings of the International Society for Music Information Retrieval*, pages 403–408, Porto, Portugal, 2012.

[49] E. Humphrey, J. Bello, and Y. LeCun. Feature learning and deep architectures: New directions for music informatics. *Journal of Intelligent Systems*, 41(3):461–481, 2013.

[50] M. Ioannou, G. Sakkas, G. Tsoumakas, and L. Vlahavas. Obtaining bipartitions from score vectors for multi-label classification. In *IEEE International Conference on Tools with Artificial Intelligence*, volume 1, pages 409–416, 2010.

[51] H. Kameoka, T. Nishimoto, and S. Sagayama. A multipitch analyzer based on harmonic temporal structured clustering. *IEEE Transaction on Audio, Speech, and Language Processing*, 15(3):982–994, 2007.

[52] K. Kashino and H. Tanaka. A sound source separation system with the ability of automatic tone modeling. In *Proceedings of the International Computer Music Conference*, pages 248–255, Tokyo, Japan, 1993.

[53] C. Kerdvibulvech and H. Saito. Guitarist fingertip tracking by integrating a Bayesian classifier into particle filters. *Advances in Human-Computer Interaction*, 2008:1–10, 2008.

[54] A. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 115–118, Phoenix, AZ, 1999.

[55] A. Klapuri. Automatic music transcription as we know it today. *Journal of New Music Research*, 33(3):269–282, 2004.

[56] A. Klapuri. A perceptually motivated multiple-F0 estimation method. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 291–294, New Paltz, NY, 2005.

[57] A. Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 216–221, Victoria, BC, 2006.

[58] C. Largeron, C. Moulin, and M. Gry. Mcut: A thresholding strategy for multi-label classification. *Lecture Notes in Computer Science*, 7619:172–183, 2012.

[59] C. Lee, Y. Yang, and H. Chen. Automatic transcription of piano music by sparse representation of magnitude spectra. In *Proceedings of the IEEE International Conference on Multimedia and Expo*, pages 1–6, 2011.

[60] C. Lee, Y. Yang, K. Lin, and H. Chen. Multiple fundamental frequency estimation of piano signals via sparse representation of Fourier coefficients. In *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2010/AR1.pdf`, 2010.

[61] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the International Conference on Machine Learning*, pages 609–616, Montréal, QC, 2009.

[62] H. Lee, P. Pham, Y. Largman, and A. Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. (22):1096–1104, 2009.

[63] B. Logan. Mel frequency cepstral coefficients for music modeling. In *Proceedings of the International Symposium on Music Information Retrieval*, pages 1–11, Plymouth, MA, 2000.

[64] R. C. Maher and J. W. Beauchamp. Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *Journal of the Acoustical Society of America*, 95(4):2254–2263, 1994.

[65] M. Marolt. Adaptive oscillator networks for partial tracking and piano music transcription. In *Proceedings of the International Computer Music Conference*, Berlin, Germany, 2000.

[66] M. Marolt. SONIC: Transcription of polyphonic piano music with neural networks. In *Proceedings of the Workshop on Current Research Directions in Computer Music*, pages 217–224, Barcelona, Spain, 2001.

[67] M. Marolt. A connectionist approach to automatic transcription of polyphonic piano music. *IEEE Transactions on Multimedia*, 6(3):439–449, 2004.

[68] K. D. Martin. A blackboard system for automatic transcription of simple polyphonic music. Technical Report 385, Massachusetts Institute of Technology, 1996.

[69] P. Masri. *Computer modeling of sound for transformation and synthesis of musical signal*. PhD thesis, University of Bristol, Bristol, UK, 1996.

[70] R. Meddis and M. J. Hewitt. Virtual pitch and phase sensitivity of a computer model of the auditory periphery. i: Pitch identification. *Journal of the Acoustical Society of America*, 89(6):2866–2882, 1991.

[71] R. Meddis and L. O'Mard. A unitary model of pitch perception. *Journal of the Acoustical Society of America*, 102(3):1811–1820, 1997.

[72] Eduardo R. Miranda and Marcelo M. Wanderley. *New Digital Musical Instruments: Control and interaction beyond the keyboard*, volume 21 of *The Computer Music and Digital Audio Series*, chapter 2: Gestural Controllers. A-R Editions, Middleton, WI, 2006.

[73] J. A. Moorer. *On the segmentation and analysis of continuous musical sound by digital computer*. PhD thesis, Department of Music, Stanford University, Stanford, CA, 1975.

[74] M. Nakano, K. Egashira, N. Ono, and S. Sagayama. Hamonic temporal structured clustering with unsupervised model learning for multipitch estimation. In *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2009/NEOS2.pdf`, 2009.

[75] J. Nam, J. Ngiam, H. Lee, and M. Slaney. A classification-based polyphonic piano transcription approach using learned feature representations. In *Proceedings of the International Society for Music Information Retrieval*, pages 175–180, Miami, FL, 2011.

[76] A. Newell. Some problems of the basic organization in problem-solving programs. In *Proceedings of the Conference on self-organizing systems*, pages 393–423, Chicago, IL, 1962.

[77] Paul D. O'Grady and Scott T. Rickard. Automatic hexaphonic guitar transcription using non-negative constraints. In *Proceedings of the IET Irish Signals and Systems Conference*, pages 1–6, Dublin, Ireland, 2009.

[78] A. Oord, S. Dieleman, and B. Schrauwen. Deep content-based music recommendation. (26):2643–2651, 2013.

[79] Marco Paleari, Benoit Huet, Antony Schutz, and Dirk Slock. A multimodal approach to music transcription. In *Proceedings of the IEEE International Conference on Image Processing*, pages 93–96, San Diego, CA, 2008.

[80] M. D. Plumbley, S. A. Abdallah, J. P. Bello, M. E. Davies, G. Monti, and M. B. Sandler. Automatic music transcription and audio source separation. *Cybernetics and Systems*, 33(6):603–627, 2002.

[81] G. E. Poliner. *Classification-based music transcription*. PhD thesis, Columbia University, New York, NY, 2008.

[82] G. E. Poliner and D. P. W. Ellis. Improving generalization for polyphonic piano transcription. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 86–89, New Paltz, NY, 2007.

[83] Graham E. Poliner and Daniel P.W. Ellis. A classification approach to melody transcription. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 161–166, London, UK, 2005.

[84] Graham E. Poliner and Daniel P.W. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, pages 1–9, 2006.

[85] Garry Quested, Roger Boyle, and Kia Ng. Polyphonic note tracking using multimodal retrieval of musical events. In *Proceedings of the International Computer Music Conference*, Belfast, Ireland, 2008.

[86] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[87] S. A. Raczyński and S. Sagayama. Multiple frequency estimation for piano recordings with concatenated regularized harmonic NMF. In the *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2009/RS.pdf`, 2009.

[88] D. Radicioni, L. Anselma, and V. Lombardo. A segmentation-based prototype to compute string instruments fingering. In *Proceedings of the Conference on Interdisciplinary Musicology*, Graz, Austria, 2004.

[89] D. P. Radicioni and V. Lombardo. Computational modeling of chord fingering for string instruments. In *Proceedings of the International Conference of the Cognitive Science Society*, pages 1791–1796, Stresa, Italy, 2005.

[90] D. P. Radicioni and V. Lombardo. Guitar fingering for music performance. In *Proceedings of the International Computer Music Conference*, pages 527–530, Barcelona, Spain, 2005.

[91] A. Radisavljevic and P. Driessen. Path difference learning for guitar fingering problem. In *Proceedings of the International Computer Music Conference*, Miami, FL, 2004.

[92] C. Raphael. Automatic transcription of piano music. In *Proceedings of the International Society for Music Information Retrieval Conference*, pages 1–5, Paris, France, 2002.

[93] L. Reboursière and S. Dupont. EGT: Enriched guitar transcription. In *Intelligent Technologies for Interactive Entertainment*, volume 124 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 163–168. Springer International Publishing, 2013.

[94] J. Richard. An implementation of multi-band onset detection. In *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/evaluation/mirex-results/articles/onset/ricard.pdf`, 2005.

[95] X. Rodet and F. Jaillet. Detection and modeling of fast attack transients. In *Proceedings of the International Computer Music Conference*, pages 30–33, Havana, Cuba, 2001.

[96] N. Rutherford. Fingar, a genetic algorithm approach to producing playable guitar tablature with fingering instructions. Undergraduate thesis, University of Sheffield, 2009.

[97] M. Ryynänen and A. Klapuri. Polyphonic music transcription using note event modeling. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 319–322, New Paltz, NY, 2005.

[98] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 6655–6659, Vancouver, BC, 2013.

[99] K. Sawayama, N. Emura, M. Miura, and M. Yanagida. A system yielding the optimal chord-form sequence on the guitar. In *Proceedings of the International Conference on Music Perception and Cognition*, pages 1545–1550, Bologna, Italy, 2006.

[100] S. I. Sayegh. Fingering for string instruments with the optimum path paradigm. *Computer Music Journal*, 13(3):76–84, 1989.

[101] J. Scarr and R. Green. Retrieval of guitarist fingering information using computer vision. In *Proceedings of the International Conference of Image and Vision Computing New Zealand*, pages 1–7, Queenstown, New Zealand, 2010.

[102] E. D. Scheirer. Tempo and beat analysis of acoustic music signals. *Acoustic Society of America*, 103(1):558–601, 1998.

[103] A. W. Schloss. *On the automatic transcription of percussive music—From acoustic signal to high-level analysis*. PhD thesis, Stanford University, Stanford, CA, 1985.

[104] J. Schlüter and S. Böck. Improved musical onset detection with convolutional neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 6979–6983, Florence, Italy, 2014.

[105] E. M. Schmidt and Y. E. Kim. Learning rhythm and melody features with deep belief networks. In *Proceedings of the International Society for Music Information Retrieval*, pages 21–26, Curitiba, Brazil, 2013.

[106] M. Slaney and R. F. Lyon. A perceptual pitch detector. In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 357–360, Albuquerque, NM, 1990.

[107] P. Smaragdis and J. C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 177–180, New Paltz, NY, 2003.

[108] L. Tang, S. Rajan, and V. K. Narayanan. Large scale multi-label classification via metalabeler. In *Proceedings of the International Conference on World Wide Web*, pages 211–220, New York, NY, 2009.

[109] D. R. Tuohy and W. D. Potter. A genetic algorithm for the automatic generation of playable guitar tablature. In *Proceedings of the International Computer Music Conference*, pages 499–502, Barcelona, Spain, 2005.

[110] D. R. Tuohy and W. D. Potter. GA-based music arranging for guitar. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1065–1070, Vancouver, BC, 2006.

[111] D. R. Tuohy and W. D. Potter. *Generating Guitar Tablature with LHF Notation Via DGA and ANN*, volume 4031 of *Lecture Notes in Computer Science*, pages 244–253. Springer Berlin/Heidelberg, 2006.

[112] Daniel R. Tuohy and W. D. Potter. An evolved neural network/HC hybrid for tablature creation in GA-based guitar arranging. In *Proceedings of the International Computer Music Conference*, pages 576–579, New Orleans, LA, 2006.

[113] G. Tzanetakis and P. Cook. MARSYAS: A framework for audio analysis. *Organised Sound*, 4(3):169–175, 2000.

[114] Ubisoft. Rocksmith. `http://rocksmith.ubi.com/rocksmith/en-ca/home`. Last Accessed: July 27, 2015.

[115] P. E. Utgoff and D. J. Stracuzzi. Many-layered learning. *Neural Computation*, 14:2497–2539, 2002.

[116] E. Vincent, N. Bertin, and R. Badeau. Two nonnegative matrix factorization methods for polyphonic pitch transcription. In *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2007/F0_vincent.pdf`, 2007.

[117] Y. Wang and Y. Zhang. Non-negative matrix factorization: A comprehensive review. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1336–1353, 2012.

[118] D. Weissman. *Guitar tunings: A comprehensive guide*. Routledge, Taylor & Francis, New York, NY, 1st edition, 2006.

[119] C. Yeh. *Multiple fundamental frequency estimation of polyphonic recordings*. PhD thesis, Université Paris VI, 2008.

[120] C. Yeh and A. Roebel. Multiple-F0 estimation for MIREX 2011. In *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2011/YR1.pdf`, 2011.

[121] C. Yeh, A. Roebel, and X. Rodet. Multiple fundamental frequency estimation of polyphonic music signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 145–148, Philadelphia, PA, 2005.

[122] C. Yeh, A. Roebel, and X. Rodet. Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1116–1126, 2010.

[123] Guitarbots. `https://guitarbots.com`. Last Accessed: July 27, 2015.

[124] Wildchords. `http://company.yousician.com/products`. Last Accessed: July 27, 2015.

[125] Yousician. `http://get.yousician.com`. Last Accessed: July 27, 2015.

[126] M. Zhang and Z. Zhou. A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, 26(8):1819–1837, 2014.

[127] M. Zhang and Z. Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1338–1351, 206.

[128] R. Zhou and M. Mattavelli. A new time-frequency representation for music signal analysis: resonator time-frequency image. In *Proceedings of the International Symposium on Signal Processing and its Applications*, pages 1–4, Sharjah, United Arab Emirates, 2007.

[129] R. Zhou and J. D. Reiss. A real-time polyphonic music transcription system. In the *Music Information Retrieval Evaluation eXchange*, `http://www.music-ir.org/mirex/abstracts/2008/F0_zhou.pdf`, 2008.

[130] R. Zhou, J. D. Reiss, M. Mattavelli, and G. Zoia. A computationally efficient method for polyphonic pitch estimation. *EURASIP Journal on Advances in Signal Processing*, 2009(729494):1–11, 2009.