Automatic Classification of Playing Techniques in Guitar Pro Songs

Junhao Wang

Music Technology Area

Department of Music Research

McGill University

Montréal, Québec, Canada

15 August 2022

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Masters of Arts

Abstract

Automatic guitar transcription has been an active research area for decades. Existing work in this area has mostly focused on estimating the onset, offset, and pitch of note events. Another important aspect of expressive guitar performance, the use of playing techniques, is less studied. The system presented in this thesis is designed to recognize five common playing techniques performed on the electric guitar: bend, vibrato, hammer-on, pull-off, and slide.

The system has three steps. For a given audio track, the monophony detector extracts all the monophonic segments, where most playing technique instances occur. Then, the note-event separator splits monophonic audio segments into note events. Finally, machine learning techniques are used to classify their playing techniques using various audio features such as pitch, spectral centroid, and mel-frequency cepstral coefficients.

The presented system is trained and evaluated on a novel dataset of 379 synthesized guitar solo recordings, mostly in the genre of rock and metal, generated using publicly available Guitar Pro files collected from tablature websites. As an extended guitar tablature format, a Guitar Pro file encodes information about every note event in a guitar track, such as the timestamp, pitch, fingering position, and playing techniques. Using the Guitar Pro software, a realistic guitar audio track can be generated from the Guitar Pro file. Its corresponding annotation can also be obtained by programmatically decoding the Guitar Pro file. The end-to-end testing experiments showed that the presented system can effectively recognize the five playing techniques in synthesized guitar solos, with per-class f1 scores ranging from 71.7% to 89.2%.

Résumé

La transcription automatique de la guitare est un domaine de recherche actif depuis des décennies. Les travaux existants dans ce domaine se sont principalement concentrés sur l'estimation du début, du décalage et de la hauteur des événements de note. Un autre aspect important de la performance expressive à la guitare, l'utilisation de techniques de jeu, est moins étudié. Le système présenté dans cette thèse est conçu pour reconnaître cinq techniques de jeu courantes exécutées sur la guitare électrique : le bend, le vibrato, le hammer-on, le pull-off et le slide.

Le système comporte trois étapes. Pour une piste audio donnée, le détecteur de monophonie extrait tous les segments monophoniques, où se produisent la plupart des instances de techniques de jeu. Ensuite, le séparateur d'événements de note divise les segments audios monophoniques en événements de note. Enfin, des techniques d'apprentissage automatique sont utilisées pour classer leurs techniques de jeu à l'aide de diverses caractéristiques audio telles que la hauteur, le centroïde spectral et les coefficients cepstraux de fréquence mel.

Le système présenté est formé et évalué à partir d'un nouvel ensemble de données de 379 enregistrements synthétisés de solos de guitare, principalement dans le genre du rock et du métal, générés à l'aide de fichiers Guitar Pro accessibles au public et recueillis à partir de sites Web de tablature. En tant que format étendu de tablature de guitare, un fichier Guitar Pro encode des informations sur chaque événement de note, telles que l'horodatage, la hauteur, la position des doigts et les techniques de jeu, dans une piste de guitare. À l'aide du logiciel Guitar Pro, une piste audio de guitare réaliste peut être générée à partir du fichier Guitar Pro. Son annotation correspondante peut également être obtenue en décodant par programmation le fichier Guitar Pro. Les expériences de test de bout en bout ont montré que le système présenté peut reconnaître efficacement les

cinq techniques de jeu dans les solos de guitare synthétisés, avec des scores f1 par classe allant de 71,7 % à 89,2 %.

Acknowledgements

Major thanks are owed to my supervisor, Professor Ichiro Fujinaga, who introduced me to the world of music information retrieval (MIR) and generously shared knowledge and expertise during my time at McGill. Thank you for your warm welcome and helping me settle into Montréal. Thank you for your advising on machine learning and MIR topics. Thank you for your generous funding and the opportunities to develop software projects for the lab. This thesis could not have been possible without your continued guidance and support.

I am also grateful to all my friends and colleagues at the Distributed Digital Music Archives and Libraries (DDMAL) Lab for their moral support and inspirations. Especially, thank you to Néstor Nápoles López and Timothy de Reuse for editing my thesis proposal and sharing great ideas and conversations throughout my studies. Thank you to Sevag Hanssian and Geneviève Gates-Panneton for translating my thesis abstract into French.

Finally, I would like to thank my parents for supporting my decision to pursue the degree abroad. Thank you for your unconditional love and encouragement in all my endeavours.

Contents

List of Figures

List of Tables

List of Acronyms

AMT	automatic music transcription.
CFC	chord and fingering configuration.
CNN	convolutional neural network.
DCT	discrete cosine transform.
F0	fundamental frequency.
FFT	fast Fourier transform.
GMM	Gaussian mixture model.
GP	Guitar Pro.
GPT	guitar playing techniques.
GUI	graphical user interface.
HMM	hidden Markov model.
k-NN	k-nearest neighbors.
LHA	latent harmonic allocation.

MFCC mel-frequency cepstral coefficient.

MIDI musical instrument digital interface.

MIR music information retrieval.

MIREX Music Information Retrieval Evaluation eXchange.

NDT note duration threshold.

NMF non-negative matrix factorization.

PCA principal component analysis.

PDT pitch difference threshold.

RBF radial basis function.

RSE Real Sound Engine.

STFT short-time Fourier transform.

SVM support vector machine.

TENT technique-embedded note tracking.

Chapter 1

Introduction

With previous work as early as Moorer (1977), automatic guitar transcription has been an area of active research for decades (Barbancho, Klapuri, et al. 2012; Wiggins and Kim 2019). In Klapuri (2006), the objective of automatic music transcription (AMT) is defined as to discover the "recipe" of a music signal. With modern computational methods, there has been a significant amount of automatic guitar transcription research, most of which focuses on extracting basic score-related parameters (i.e., pitch, onset time, and offset time). However, these parameters are only part of the "recipe". Another important aspect of expressive guitar performance, the use of playing techniques such as bend, vibrato, and slide, is less studied.

In a music performance, the performer often shapes the music with their personal expressions by imposing various playing techniques. These playing techniques, sometimes referred to as instrumental gestures (Traube, Depalle, and Wanderley 2003) or guitar articulations (Reboursière et al. 2012), contribute to expressivity via subtle variation of parameters such as pitch, dynamics, and timbre.

Although the score-related parameters returned by most transcription systems are adequate in most cases, being able to recognize various playing techniques would be a valuable supplement to the transcription system. Automatic recognition of the playing techniques can be considered a step towards a more complete music transcription system, which not only estimates the pitches played, but also unveils the specific techniques needed to reproduce the exact sound.

This introductory chapter first introduces the guitar playing techniques studied in this

thesis. Then, a brief introduction to the Guitar Pro file format is given. Finally, I present an overview of the project and the organization of this thesis.

1.1 Introduction to Guitar Playing Techniques

In this thesis, I focus on five common playing techniques performed on the electric guitar: *bend*, *vibrato*, *hammer-on*, *pull-off*, and *slide*. These techniques are typically applied by the guitarist as an expressive gesture after the string is plucked. Conceptually, these techniques can be divided into two categories based on their characteristics.

Bend and vibrato make up the first category, as they typically occur within an individual note event and modulate the fundamental frequency in a gradual and continuous manner. Bend is performed by stretching the string with the fretting hand. The fretting hand applies a force perpendicular to the fretboard, pushing the string away from its equilibrium position and continuously increasing the string tension. This results in a gradual increase in the pitch. Similarly, vibrato is performed by repeatedly bending the string up and down with the fretting hand in a short period of time. It modulates the string tension and in turn causes fluctuations in the pitch.

Hammer-on, pull-off, and slide form the other category, as they typically occur in the transition between two consecutive note events and introduce a sudden frequency change. Hammer-on refers to pressing the vibrating string down on a higher fret with another finger of the fretting hand while the previous note is still sounding. As the string, driven by the second finger, lands on the higher fret, the vibrating length of the string decreases instantly, causing a sudden rise in the pitch. Pull-off, the opposite process of hammer-on, is typically performed by pressing the non-vibrating part of the string down on a lower fret with another finger of the fretting hand, and then pulling the fretting finger off the string, while the previous note is still sounding. As the string leaves the higher fret, the vibrating length of the string increases instantly, causing a sudden drop in the pitch. Slide refers to sliding the fretting finger along the string to a higher or lower fret while the previous note is still sounding. The fretting finger can slide across one or more frets, producing a discrete and incremental change of pitch.

Although these playing techniques are fundamentally different, when played on a guitar, they can sound similar to each other. This is especially true for bend, hammer-

on, pull-off, and slide, which all produce a "smooth" variation of pitch. However, an experienced guitar player can often distinguish between these playing techniques by ear. In this thesis, I explore whether the machine, when trained on a large enough set of data, can also distinguish between these playing techniques, using audio features.

1.2 Introduction to the Guitar Pro File

The task of playing technique classification aims to identify playing techniques in music audio signals. To attempt this task using supervised machine learning, it is necessary to obtain a ground-truth dataset of guitar audio recordings with playing technique annotations. Manually annotating guitar recordings with respect to the playing techniques requires a significant amount of time and domain knowledge. This motivates the compilation of a synthetic dataset using publicly available Guitar Pro[1] files.

As an extended guitar tablature format, a Guitar Pro file encodes information about every note event in a guitar track, such as the timestamp, pitch, fingering position, and playing techniques. Using the Guitar Pro software, a realistic guitar audio track can be generated from the Guitar Pro file. Its corresponding ground-truth annotation can also be obtained by programmatically decoding the Guitar Pro file. The ground-truth generation process will be detailed in section 3.1.

1.3 Project Overview

Although the guitar playing techniques are mostly universal to all genres. In this thesis, I focus on the genre of rock and metal. The reasons are twofold: First, these two genres are the most common in publicly available Guitar Pro files.[2] Second, the electric guitar part in these two genres tends to include large sections of guitar solo, which are typically expressive and make heavy use of playing techniques.

The guitar part in a typical rock or metal song can be divided into two parts: the

1. https://www.guitarpro.com/

2. As of June 29, 2022, rock and metal are the top two genres on ultimate-guitar.com in terms of the number of available tablatures. Rock has 85,754 Guitar Pro tablatures and metal has 66,123 Guitar Pro tablatures.

rhythm guitar and the lead guitar. The rhythm guitar typically plays polyphonic chord sequences, emphasizing the chord progression of the song. The lead guitar, on the other hand, plays mostly monophonic sections including the guitar solos and riffs. It is the lead guitar that typically makes heavy use of the various playing techniques to add emotions and expressivity to the song. An example is shown in figure 1.1, which demonstrates a two-track guitar tablature sampled from the song *Back in Black* by *AC/DC*. The upper track is playing the monophonic solo (the lead guitar part) with many instances of bend (denoted by arrowed curves) and slide (denoted by slashes), while the lower track is playing the polyphonic accompaniment (the rhythm guitar part). Therefore, for the purpose of extracting the playing techniques, it is desirable to separate the monophonic part from the polyphonic part and focus on analyzing the former. The scope of this thesis is thus limited to classifying playing techniques in monophonic guitar solo recordings. This motivates the first component in the proposed playing technique classification system: the monophony detector.

As presented in section 1.1, the playing techniques studied in this thesis either occur within a single note event or span across two consecutive note events. Therefore, it is essential to locate the note events in the audio signal under analysis. This helps locate the regions of interest (i.e., the signal regions that could potentially correspond to a playing technique). This motivates the second component of the system: the note-event separator.

Finally, the regions of interest need to be classified as one of the playing technique classes. This motivates the last component of the system: the playing technique classifier.

The workflow is summarized as follows: 1. Collect the Guitar Pro files; 2. Generate annotations and synthesized audio; 3. Extract monophonic audio segments; 4. Identify the regions of interest; 5. Classify the regions of interest into playing technique classes.

1.4 Thesis Organization

This thesis spans a total of five chapters. Chapter 1 gives a brief introduction to the guitar playing techniques, the Guitar Pro file, and the playing technique classification system presented in this thesis. Chapter 2 first reviews previous works in the field of instrumental gesture acquisition. Then, it presents some background on audio features and machine learning classifiers. Chapter 3 describes the ground-truth generation strategy and the

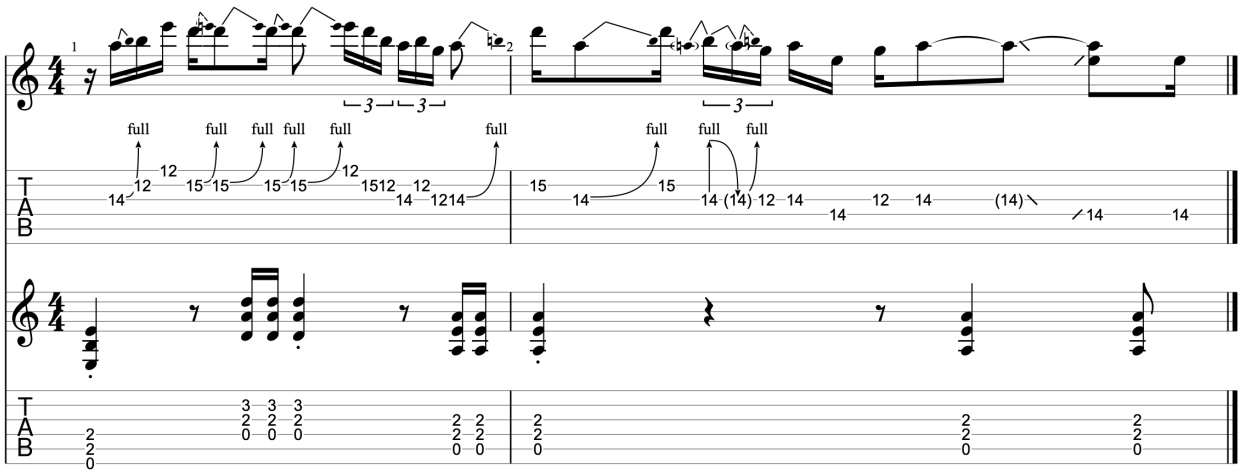

Figure 1.1: A segment of a two-track Guitar Pro file corresponding to the guitar solo section of the song *Back in Black*. The figure shows a total of four staves (from top to bottom): Standard music notation for the lead guitar track; Guitar tablature for the lead guitar track; Standard music notation for the rhythm guitar track; Guitar tablature for the rhythm guitar track. The arrowed curves on the tablature indicate instances of the *bend* technique. The two slashes by the side of the numbers on the tablature indicate instances of the *slide* technique.

design of each component in the playing technique classification system presented in this thesis. Chapter 4 presents the experiments conducted for developing and evaluating the playing technique classification system. This includes experiments for each individual component as well as an end-to-end integration test for evaluating the performance of the system as a whole. Chapter 5 summarizes the contributions and concludes this work. It also discusses directions for future work.

Chapter 2

Literature Review

This thesis focuses on recognizing guitar playing techniques in audio signals. Such study is situated in the domain of instrumental gesture acquisition. This chapter reviews relevant works in the research area of instrumental gesture acquisition, including guitar fingering estimation, gestural parameter estimation, and playing technique classification. Additionally, it gives a brief introduction to audio features and machine learning classifiers, which are key elements of the methodology used in this thesis.

2.1 Instrumental Gesture Acquisition

Instrumental gesture acquisition is the task of estimating the performer's physical interaction with the instrument for producing or modifying the musical sound. According to the typology established in Cadoz (1988), the instrumental gestures can be divided into three categories: exciter gestures, modification gestures, and selection gestures. The guitar is particularly interesting in this context because its gestures often span across multiple categories. Plucking the string is both a exciter gesture and a modification gesture, because the plucking action not only produces the vibration, but also affects the produced timbre. The guitarist's fingering on the fretboard is both a modification gesture and a selection gesture, because most pitch can be played at multiple fingering positions on the fretboard, and different fingering positions would produce different timbres (Traube, Depalle, and Wanderley 2003).

Over the years, the guitar has been studied extensively for its instrumental gestures.

For guitars in general, the fretting hand fingering estimation has been a popular task in automatic transcriptions (Paleari et al. 2008; Barbancho, Klapuri, et al. 2012; Yazawa et al. 2013). Besides fingering estimation, early works in this area mostly focused on analyzing the plucking action of the classical guitar and estimating the parameters that describe the interaction between the string and the plucking hand (Orio 1999; Traube and Smith 2000, 2001). The electric guitar, however, is mostly studied for the expression techniques performed by the fretting hand (Kehling et al. 2014; Su, Yu, and Yang 2014; Chen, Su, and Yang 2015; Su et al. 2019). This might be due to the fact that the electric guitar is usually played with a plastic pick and is often followed by a chain of audio effects that diminish the subtle timbre nuances produced by different plucking techniques. The same set of expression techniques has been studied in bass guitar recordings. Moreover, the bass guitar has also been studied for its typical plucking technique known as *slap* (Abeßer, Lukashevich, and Schuller 2010).

In this section, I review previous works on instrumental gesture acquisition. Specifically, I present them in three segments: fingering estimation, gestural parameter estimation, and playing technique classification. Gestural parameter estimation aims to estimate the value of a continuous output variable representing a certain gestural parameter such as the plucking point position or bowing angle. This output variable is a real-value, such as an integer or floating point value. Playing technique classification aims to distinguish between different predefined gesture classes such as bend, vibrato, and slide. Its output variable is a discrete categorical value representing a certain gesture class. In this thesis, the term ”playing techniques” refers to the predefined gesture classes. Fingering estimation is put into an individual segment because the output of fingering estimation can either be a numerical variable or a categorical variable, depending on the specific instrument under study. For the guitar, since the strings and frets form a discrete, grid-like structure, the estimated fingering is represented by a string number and a fret number, and they are categorical in nature.

2.1.1 Automatic transcription and fingering estimation

Automatic music transcription (AMT) is the task of automatically converting music audio signals to music notations. Ideally, a complete AMT system should cover a series

of subtasks, including pitch estimation, onset and offset detection, instrument recognition, beat and rhythm tracking, interpretation of expressive timing and dynamics, and score typesetting (Benetos et al. 2019). Developing such a comprehensive AMT system is a difficult task. Most existing AMT systems thus focus on a subset or certain aspects of these tasks. To date, most AMT research focuses on producing the basic descriptive note-level parameters (i.e., pitch, onset time, and offset time) as the output. For a comprehensive review on the AMT topic, please refer to Klapuri (2006), Benetos et al. (2012, 2013), and Benetos et al. (2019).

Although AMT does not seem immediately relevant to the gestural acquisition task attempted in this thesis, the two research areas find their overlap on the topic of automatic guitar transcription. In fact, guitar fingering estimation and automatic transcription are often approached together in existing guitar transcription systems. One reason for this overlap is the string ambiguity of the guitar. That is, different fingerings (i.e., string-fret combinations) can produce the same pitch. Since these fingerings often produce subtle but important timbre nuances, guitar transcription systems try to discover not only what notes are played, but also how they are played. Despite the fact that fingering position on the guitar is considered an instrumental gesture, most existing work have been approaching the fingering estimation in an automatic transcription context. Therefore, in this section, I present existing automatic guitar transcription systems with a focus on fingering estimation.

Video-based guitar transcription

Before presenting audio analysis-based transcription systems, I first review several video-based systems. These systems, although not transcribing musical audio, represent significant research efforts in leveraging computer vision methods for guitar fingering estimation. Burns and Wanderley (2006) used a camera mounted on the guitar to capture a top-down view of the fretting hand. In their proposed system, the finger positions are tracked via the circular Hough transform. The system assumes that fingertips generally have a quasi-circular shape and can thus be tracked by tracing circles with a certain radius in the image. By matching the finger positions with a precomputed string-fret grid, the system detects and recognizes the left-hand gesture in realtime.

Kerdvibulvech and Saito (2007b) proposed using stereo cameras to capture the guitar

performance in a three-dimensional space. This makes it easier to distinguish whether the finger is pressing the string or not. Moreover, they argued that the assumption about the semicircular shape of the fingertips does not hold from some camera angles and that the color contrast between the fingertip and the rest of the finger is not strong enough. To achieve more robust finger tracking, Kerdvibulvech and Saito (2007b) attached colored markers to the player's fingertips. Although this approach proved effective, the colored markers are sometimes obtrusive, making it unnatural for real-world guitar playing (Kerdvibulvech and Saito 2007a). In their later works, Kerdvibulvech and Saito (2007a, 2008) removed the colored markers and switched to using artificial neural networks and the semicircle template model for locating the fingertips.

More recently, Goldstein and Moses (2018) also used a camera mounted on the guitar to tackle the polyphonic transcription problem. However, instead of tracking the fingers, they used the camera to directly capture the string vibrations. It was proposed that the string vibrations can be recovered from the silent video by analyzing the intensity change at the string pixels. A novel string detection algorithm was developed to locate the string in the video as well as to obtain the string pixel signals. For each string, the string pixel signal is segmented in time using its spectrogram. The pitch is then computed for each segment. Knowing the pitch, the fret position can be calculated for each string.

Audio-based guitar transcription

Compared with their video-based counterparts, audio-based guitar transcription systems have the advantage of being non-obtrusive. They operate by analyzing the sound signal only, so they do not rely on any video equipment or extra accessories attached to the guitar.

Barbancho, Klapuri, et al. (2012) proposed a guitar transcription algorithm that identifies the chords as well as the corresponding fingerings on the fretboard. The system starts with a multiple fundamental frequency estimator, which measures the strength of each candidate frequency in an audio frame. The chord and fingering configuration (CFC) is then estimated by two trained internal models: the acoustic model and the musicological model. The acoustic model describes the probability of observing a certain fundamental frequency combination given a certain CFC. The musicological model describes the probability of switching between different CFCs.

These two internal models are formulated as a hidden Markov model (HMM) using CFCs as its hidden states. The output of this system is selected from a preconstructed corpus of CFCs.

Barbancho, Tardon, et al. (2012) estimated the pitches and string-fret combinations by analyzing the inharmonicity relations between the fundamentals and the partials. Inharmonicity refers to the difference between the observed frequency partials and the ideal harmonic series. It depends on the radius, tension, and length of the string (Barbancho, Tardon, et al. 2012). For single notes, the candidate pitches are estimated using the magnitude spectrum. Then the inharmonicity coefficients are computed for all possible string-fret combinations of all candidate pitches. The best candidate is found by using a voting scheme based on matching the partials. For chords with less than four notes (i.e., open chords), the chord transcription is based on estimating the notes and iteratively removing estimated fundamental frequencies and partials from the spectrum. Since the performance of this algorithm degrades as the chord complexity grows, for chords with more than four notes (i.e., barred chords), a different procedure based on chord templates is used. Four finger configuration templates are used as chord templates to address two major and two minor chord positions. For each chord played, all possible string-fret combinations are considered, and the template producing the largest number of identified partials is the transcription result.

Burlet and Fujinaga (2013) developed a web-based guitar transcription framework named *Robotaba*, which takes an audio file as input and generates a digital guitar tablature as output. This framework consists of three modules: the polyphonic transcription module, the guitar tablature arrangement module, and the guitar tablature engraving module. The polyphonic transcription module is based on the general-purpose polyphonic transcription algorithm proposed in Zhou and Reiss (2008), which estimates the onset and pitch for each note present in the polyphonic signal. The guitar tablature arrangement module features a novel algorithm named *A-star-guitar*, which arranges the note tracking results into a meaningful guitar tablature. A-star-guitar extended the A* path-finding algorithm (Hart, Nilsson, and Raphael 1968) for application in the specific transcription task. The search for an optimal sequence of string-fret combinations is modeled by a path-finding problem through a weighted directed graph. Specifically, each candidate string-fret combination, determined by the

estimated pitch and user-specified guitar configurations, is represented by a vertex. Vertices corresponding to a pair of adjacent notes or chords are connected by an edge, whose weight is determined by the biomechanical difficulty of transitioning between the two fingerings. As a result, the generated tablature arrangement is the optimal sequence of string-fret combinations considering the biomechanical difficulty and user-specified configurations such as tuning and capo position. It is worth noting that, the optimal tablature arrangement found may not be the same as the arrangement used when recording the original audio file. Most notes in the guitar's pitch range can be played at multiple positions on the fretboard. The same note played at different positions have the same pitch but different timbres. Therefore, to produce a desired timber, the performer may choose not to use the biomechanically optimal arrangement as assumed by the system.

Weighted directed graph was also used in several other audio-based guitar transcription systems for estimating the optimal fingering configurations. Yazawa et al. (2013) pointed out that traditional multipitch estimation algorithms such as latent harmonic allocation (LHA) (Yoshii and Goto 2012) often estimate unrealistic pitch combinations which can hardly be played due to the physical constraints of the guitar and human hand. To address this issue, Yazawa et al. (2013) proposed a transcription system where all fingering candidates pass through a validation process based on the physical constraints. A fingering configuration is deemed valid if the finger count is less than or equal to four and the finger spread is less than or equal to four frets. To determine the optimal fingering configurations, all valid fingering configurations are enumerated, and three constraints are added to suppress the unrealistic results estimated by LHA. First, the pitch combination produced by the optimal fingering configuration must maximize the likelihood estimated by LHA. Second, configuration changes may happen only at onset times. Third, the same fingering configuration must last for a certain duration. These three constraints are formulated into a weighted directed graph. The vertices are candidate fingering configurations at each time frame. An edge exists between two vertices only when the last two constraints are satisfied. The weight assigned to each edge is the sum of the likelihood corresponding to the associated vertices. Using dynamic programming, the longest path found in this graph corresponds to the optimal fingering sequence. This transcription system works for both

chords and single notes and does not require any training data.

The system developed in Yazawa et al. (2013) was further extended in Yazawa, Itoyama, and Okuno (2014), where the player's proficiency is taken into account. It was pointed out that in guitar practicing, experts care more about reproducing the piece as accurately as possible, while beginners may struggle to learn a piece and lose motivation because its difficulty level exceeds their proficiency. To solve this problem, the player's proficiency is quantified as a user parameter and added to the transcription system. The proficiency parameter controls the trade-off between the accuracy of reproducing the acoustic signal and the difficulty of performing the corresponding fingering. In Yazawa et al. (2013), the weight of the edges in the acyclic graph are defined only by acoustical reproducibility. While in Yazawa, Itoyama, and Okuno (2014), the weights are determined by a weighted sum (controlled by the proficiency parameter) of acoustical reproduction accuracy and fingering difficulty.

More recently, *TabCNN* was proposed in Wiggins and Kim (2019). This system is based on a deep convolutional neural network (CNN) structure. The CNN model learns a direct mapping from audio to tablature and makes use of the physical constraints and timbre nuances between different strings simultaneously. The raw audio data are transformed into image representations using constant-Q transform before entering the CNN model. The string-fret combinations are predicted on the frame level as the output.

Multimodal guitar transcription

There have also been multimodal guitar transcription systems, which draw information from both the video and the audio of the guitar performance to attempt accurate transcriptions. Paleari et al. (2008) proposed a system that combines computer vision and audio analysis, aimed to tackle the string ambiguity of guitar transcription. Given the video of the guitar performance, the fretboard is detected by analyzing the first frame. After obtaining the corner point coordinates of the fretboard, each fret on the fretboard is estimated and tracked using a series of computer vision algorithms. A string-fret grid is constructed by separating the fretboard region. The hand position is detected by counting the number of skin-colored pixels in the string-fret grid. In the audio domain, the pitch, onset, and duration of each note are extracted and written into a Musical Instrument Digital Interface (MIDI) file. Combining the hand position and the

pitch, the string-fret combination is estimated for each note.

Similarly, Hrybyk and Kim (2010) presented a multimodal approach for guitar chord transcription. In the audio domain, the chord scale (e.g., C major) is estimated using Specmurt Analysis (Saito et al. 2008), which suppresses the overtones in the log-frequency spectrum and reveals the fundamental frequencies in the chord. The specific chord voicing (i.e., open, barred, or inverted) is determined via video analysis. Since the camera is not mounted on the guitar, the fretboard could face the camera in many different orientations and angles. Homography rectification is thus applied to the image to adjust the observation angle as needed. The fretboard images are decomposed into eigen-chords using principal component analysis (PCA). Each input video frame is projected into the chord-space and the closest centroid determines its voicing.

Use of augmented guitars

Hexaphonic guitar has also been used to simplify guitar transcription and fingering estimation. One major challenge in guitar transcription is the polyphonic nature of the instrument. When multiple strings are plucked simultaneously, their audio signals are mixed and captured by microphones (for acoustic guitars) or pick-ups (for electric guitars). The mixture of multiple notes makes it difficult to separate the fundamental frequencies from their harmonics. This challenge can be sidestepped by using a special type of electric guitar named hexaphonic guitar. A standard guitar can be transformed into a hexaphonic guitar by installing the hexaphonic pick-ups. Unlike regular pick-ups, which mix the signals from all vibrating strings, hexaphonic pick-ups capture the signal from each string separately. In other words, hexaphonic guitars output six signals, corresponding to the six strings, instead of one mixed signal. This allows the signal from each string to be analyzed individually, changing the polyphonic transcription problem into a monophonic one.

O'Grady and Rickard (2009) built the necessary equipment for transforming a regular electric guitar into a hexaphonic one and proposed a hexaphonic guitar transcription algorithm based on non-negative matrix factorization (NMF). Similarly, Angulo, Giraldo, and Ramirez (2016) modified a regular acoustic guitar into a hexaphonic guitar using piezoelectric sensors and went on to transcribe the musical events in the guitar performance into visual forms instead of musical scores. With hexaphonic guitars, the

string ambiguity issue in guitar transcription is avoided, as each string is captured separately. It also makes it easier to go one step further and derive the string-fret combinations. Once the string is known, only one fret on that string is capable of producing the given pitch.

In summary, on the topic of automatic guitar transcription and fingering estimation, most works focus on audio-based approaches. Video-based and multimodal approaches, accurate as they can be, have serious limitations due to their dependence and strict requirements on the video. Hexaphonic guitars simplify the transcription problem but the required special equipment makes them less practical for general guitar players to use.

2.1.2 Gestural parameter estimation

Apart from fingering estimation, some existing instrumental gesture acquisition research aimed to estimate low-level gestural parameters represented by a continuous output variable. Previous audio-based works in this category mainly focused on two instruments: the violin and the guitar.

Perez-Carrillo and Wanderley (2012, 2015) and Perez-Carrillo (2016) focused on estimating violin bowing gestures from audio signals obtained using a vibration transducer at the violin bridge. The proposed systems are based on machine learning models, which learn a mapping from various audio features to gestural parameters captured by sensors. Specifically, Perez-Carrillo and Wanderley (2012) used random forests and multilayer perceptrons trained on a set of spectral features to predict the played string, bowing force, bowing velocity, bowing point position, and bow tilt. Perez-Carrillo and Wanderley (2015), improving upon their previous system, used a series of HMMs with observations parameterized as multivariate Gaussian mixtures. The features include the fundamental frequency, aperiodicity coefficient, delta of aperiodicity, delta of mean energy, and five energy coefficients representing the energy distribution over the frequency bands. The models learn from these features to predict the played string, bowing force, bowing velocity, and bowing point position. Perez-Carrillo (2016) proposed a CNN-based system that predicts the played string, bowing velocity, bowing force, and the bowing point position. Auditory energygrams

are extracted from the audio signals as features, and time series of the bowing gesture parameters measured with sensors are used as prediction targets. The CNN model is trained to predict a value for each gestural parameter from each frame.

The earliest attempts to retrieve guitar gestural information from audio recordings focused on estimating gestural parameters such as the plucking position (Traube and Smith 2000, 2001; Traube, Depalle, and Wanderley 2003; Traube and Depalle 2003), finger-string inclination (Orio 1999), plucking finger tension (Orio 1999), and their relationships with the produced timbre.

Orio (1999) studied the timbre differences produced by different plucking techniques on a classical nylon-string guitar. All sound samples analyzed were of the same pitch and loudness. The plucking position, finger-string inclination, hand-string inclination, and degree of finger relaxation were varied. Through time-frequency analysis, Orio (1999) found that the degree of inharmonicity is significantly relevant to the plucking position along the string. Harmonic amplitudes were found to be affected by all plucking variations, and the attack time of the fundamental varied with different hand-string inclination. Additionally, Orio (1999) extracted the irregularity and center of gravity of the spectrum. The center of gravity of the spectrum was found to be related to all plucking variations, while the irregularity of the spectrum was found to be relevant only to plucking positions.

Traube and Smith (2000, 2001) focused on estimating the fingering point and plucking point positions along the string using signal processing techniques. The plucking point is estimated by comparing the magnitude spectrum of the recorded note to the magnitude spectra corresponding to various plucking point positions computed for an ideal string. The estimated plucking point is the one that minimizes the difference between the observed spectrum and the ideal spectrum. The plucking point information, along with the tuning and detected pitch, is then used to estimate the fingering point. This plucking point estimation task was further explored in Traube and Depalle (2003), where an autocorrelation-based method is used to give an initial approximation. The comb filter delay for the physical model is then adjusted iteratively to fit the observed spectral envelop. This method was also presented in Traube, Depalle, and Wanderley (2003), where the plucking point estimation task was put in the context of instrumental gestures and their relationships with verbal timbre descriptors and timbral features.

More recently, Scherrer (2013) developed a system for automatically extracting the angle of release from monophonic classical guitar solo recordings. The angle of release refers to the angle between the string vibration and the guitar top plate after the plucking finger leaves the string. To extract this gestural parameter, the string vibration is modeled in both the horizontal axis and the vertical axis, using two digital waveguide models. The physical models are then incorporated with a signal model to analyze the velocity signal measured at the bridge of the guitar. Through acoustical analysis, an equation is established for the relationship between the angle of release and the frequency, damping, amplitudes, and phases of the velocity signal measured at the bridge. These signal parameters are retrieved via a series of signal processing techniques. This estimation system was evaluated on a set of synthetic guitar plucks produced with various angles of release and plucking point positions.

2.1.3 Playing technique classification

In playing technique classification, the objective is to predict a class label that represents a certain playing technique (i.e., a predefined gesture class). This task typically requires a large number of sound samples of the instrument, each represented by a set of audio features designed to capture the characteristics of the playing techniques. The classification is performed either by rule-based heuristics or machine learning classifiers trained to separate these samples in the feature space.

Piano playing technique classification

The classification-based gesture acquisition has been successfully applied to various instruments. In piano music, the pedaling techniques have been studied extensively in Liang, Fazekas, and Sandler (2018a, 2018b, 2019). Specifically, they focused on the sustain pedal, which prolongs the sound by lifting the dampers off the strings. By controlling the timing and depth of pedal press, the performer can subtly alter the timbre and duration of the notes. Liang, Fazekas, and Sandler (2018b) proposed a system for detecting the legato pedaling technique. The system starts with an NMF-based transcription performed on the piano recording, which converts the audio signal into a list of note events with estimated pitch, onset time, and offset time. Using the

transcription results, the partial frequencies of each note are estimated based on the fundamental frequencies and inharmonicity coefficients. The residual, where the effect of sympathetic resonance resides, is obtained by subtracting the sinusoidal components (i.e., fundamentals and partials) from the original signal. As the legato pedaling onset can only occur between the onsets of two successive note events, the residual signal is segmented at note onset points. The resulting segments are considered candidates for the onset of a legato pedaling instance. Finally, legato pedaling detection is modeled as a segment-level binary classification task. A logistic regression binary classifier is trained on segment-level features such as the maximum root-mean-square energy of partials corresponding to the unstruck strings as determined by the preceding notes. The same strategy was followed by Liang, Fazekas, and Sandler (2019), who used CNN classifiers to distinguish between the presence or absence of the pedaling technique on each audio frame. The authors generated a dataset of MIDI-synthesized piano recordings, where each piece was synthesized twice, once with sustain pedal technique and once without sustain pedal technique. The pedaled version and unpedaled version audio signals generated from the same MIDI file are used as a training pair, with the mel-spectrograms as the audio feature. One CNN classifier is trained on pedal onset frames and another CNN classifier is trained on the pedaled frames. The two classifiers jointly predict whether an audio frame is played with the pedaling technique.

Snare drum playing technique classification

Besides piano, the snare drum has also been studied for playing technique classification. Tindale et al. (2004) developed a classification system that distinguishes between seven playing techniques: *rimshot, brush stroke, center, near-center, halfway, near-edge*, and *edge*. These techniques are performed by hitting the drum with different types of drumsticks or at different positions along the radius of the snare drum, producing subtle differences in the resulting timbre. The system first extracts various temporal and spectral features from the recorded drum audio and then uses them to train an artificial neural network model for classification. This system was further improved in Tindale et al. (2005), which explored four types of classifiers: ZeroR, Gaussian mixture model (GMM), k-nearest neighbors (k-NN), and artificial neural network. Besides the snare drum, the system was also experimented for classifying four types of tabla strokes: *Na* stroke, *Ta* stroke, *Tu*

stroke, and *Ga* stroke.

A different set of snare drum playing techniques were studied by Wu and Lerch (2016). As opposed to the timbral nuances produced by single drum strokes, they focused on classifying four rudimentary techniques: *strike, buzz roll, flam,* and *drag*. The difference between these techniques lies in the presence, velocity and timing of grace notes preceding or succeeding the main stroke. Instead of classifying the single-source drum strokes as done in Tindale et al. (2004) and Tindale et al. (2005), Wu and Lerch (2016) aimed to detect the snare drum rudiments from polyphonic mixtures. The proposed system uses a support vector machine (SVM) classifier trained on the activation function and timbral features extracted from the training samples. The system proved effective in the evaluation performed on a dataset of real-world polyphonic mixtures.

Violin playing technique classification

As one of the most popular stringed instruments, the violin is known for its musical expressiveness. Barbancho et al. (2009) developed a violin transcription system that not only returns the pitch of the note events but also detects various playing techniques employed by the performer. The violin playing techniques covered by this system include *détaché*, *pizzicato*, *tremolo*, *spiccato*, and *flageolett-töne*. The authors employed a set of temporal and spectral parameters to characterize the playing techniques. Based on the parameterization, the detection system was developed and then evaluated on a dataset of real-world violin recordings featuring different violins played by different musicians with different recording qualities. The results suggest that the playing technique characteristics tend to vary depending on the specific performer and instrument, and it is difficult to define generalized conditions to predict the playing techniques from audio recordings. Su, Lin, and Yang (2014) expanded the scope of playing technique detection from the violin to four bowed string instruments: violin, viola, cello, and contrabass. Nine types of playing techniques are studied: *flageolet, normal, non-vibrato, pizzicato, sordino, spiccato, sul ponticello, sul tasto*, and *tremolo*. A number of temporal, spectral, cepstral, and phase-derived features were extracted from single note recordings and used to train SVM classifiers. The results suggest that sparse-coded magnitude and phase-derived spectral features are more effective than conventional timbre features in

playing technique classification.

Guitar Playing technique classification

Abeßer, Lukashevich, and Schuller (2010) is considered the pioneer in bass guitar playing technique classification. The authors focused on classifying ten playing techniques from bass guitar audio recordings. The playing techniques were divided into plucking styles and expression styles, which were then classified in two separate classification tasks. Plucking styles include *fingerstyle, picked, muted, slap-pluck,* and *slap-thumb*, which are generally executed by the plucking hand of the performer. Expression styles include *normal, vibrato, bending, harmonics,* and *dead note*, which are generally executed by the fretting hand of the performer. Abeßer, Lukashevich, and Schuller (2010) pointed out that plucking styles mainly affect the attack period of the note event. The short attack transient observed in bass guitar recordings motivated the authors to use the modified covariance method for estimating the power spectral density. The envelope function is then calculated by tracking the first twenty partials within the estimated power spectral density. From the envelope functions, a set of audio features are extracted, which characterizes the sound properties related to each playing technique, such as spectral crest factor for measuring the percussiveness and spectral centroid for measuring the brightness. All time-dependent features are aggregated over the attack and decay period of the note. A series of machine learning classifier models, including SVM, GMM, naive Bayes, and k-NN, were trained and evaluated on a dataset consisting of isolated single note recordings. The results suggest that models trained on audio features designed specifically for this task outperformed the baseline models trained on general-purpose mel-frequency cepstral coefficient (MFCC) features in both plucking style classification and expression style classification.

The topic proposed in Abeßer, Lukashevich, and Schuller (2010) was further explored in Abeßer, Dittmar, and Schuller (2011), which attempted the same play technique detection problem on the bass guitar with a specific focus on the expression styles that modulate the fundamental frequency (F0), namely *slide, vibrato*, and *bending*. In the proposed system, the F0 curve corresponding to each single note recording is first segmented temporally into monotonical F0 segments. The number of segments and the frequency modulation range are used as part of the features. The other features include

the autocorrelation of the F0 curve, the first non-zero local maximum of the autocorrelation function, which indicates the modulation frequency of the F0 curve, and the segment-level modulation range and modulation frequency, representing the temporal progression of the features. These segment-level features are aggregated over the duration of each note using statistic descriptors. On the note level, the F0 difference between the beginning and the end of the note is calculated as a feature. The combination of the above features is used to train SVM classifiers that distinguish between the playing technique classes on two different levels. First, an SVM classifier is trained to distinguish between the three playing technique classes and a *normal* class where no playing techniques are performed. Then, another SVM classifier is trained to perform more precise classification, where each playing technique class is further divided into two subclasses. The classifier aims to distinguish between *slide up*, *slide down*, *fast vibrato*, *slow vibrato*, *semi-tone bending*, *quarter-tone bending*, and the *normal* class. The models were trained and evaluated on an extension of the dataset compiled in Abeßer, Lukashevich, and Schuller (2010). The evaluation results suggest that the extracted task-specific features can effectively capture the playing technique characteristics on the subclass level. .

Both being plucked string instruments, the guitar shares many common playing techniques with the bass guitar. Özaslan et al. (2010) focused on analyzing two types of left-hand articulations performed on the nylon-string guitar: *legato* and *appoggiatura*. The proposed system starts with a plucking detection module, which performs onset detection within the guitar audio signal. The onset detection threshold is empirically set to detect right-hand attacks but ignore the left-hand attacks. The audio segment between two successive onsets is then analyzed individually by a pitch detection module. If a change of pitch is detected within the segment, the segment is considered a candidate for articulations. The appoggiaturas are distinguished heuristically from legatos using the duration of the signal segment before the change of pitch. A similar strategy was employed in Özaslan and Arcos (2010). Instead of legato and appoggiatura, this paper focused on detecting *legato* and *glissando*. In the last stage, a classifier is used in place of the simplistic heuristics used in Özaslan et al. (2010). Specifically, the classifier is trained on three audio features: amplitude, aperiodicity, and pitch. This system was tested on both nylon strings and metallic strings and achieved satisfactory results in classifying

legato and glissando.

Reboursière et al. (2012) proposed a rule-based playing technique detection system for the classical guitar. The playing techniques covered by this system include *hammer-on*, *pull-off*, *slide*, *bend*, *harmonic*, and *palm muting*. Based on whether the attack is performed by the left hand (fretting hand) or the right hand (plucking hand), the playing techniques are divided into two categories. Left-hand techniques include bend, slide, hammer-on, and pull-off. Right-hand techniques include normal, muted, and harmonic. The authors pointed out that, in the signal envelope of plucked notes, a trough can often be observed before the attack because the plucking finger temporarily stops the existing vibration when it touches the string. When a note is activated by a left-hand technique, the legato attack does not stop the vibration and there is thus no significant change in the amplitude. This property is utilized by the proposed system for distinguishing notes activated using left-hand techniques and those activated using right-hand techniques. To distinguish between various left-hand techniques, the proposed system uses a series of parameters related to the shape of the pitch curve, including the pitch time derivative, the interval of the note transition, and the maximal pitch slope. As for the right-hand techniques, the energy envelope slope at the attack is calculated for detecting muted notes based on the observation that muted notes exhibit logarithmic increase in the slope over time while normal notes stay relatively flat. Harmonics are detected with a combination of time-domain and spectral-domain parameters extracted from the attack period. Time-domain parameters are the attack duration (i.e., for how long the attack waveform remains positive) and the ratio between maximum amplitude and minimum amplitude during the attack period. The spectral-domain parameter is the difference in dB between the amplitude of the first harmonic and the amplitude of the subharmonic. Empirical thresholds were found for each parameter, and the detection rules operate by thresholding the parameters associated with each playing technique. For evaluating this rule-based playing technique detection system, Reboursière et al. (2012) constructed a dataset of audio samples recorded using a hexaphonic classical guitar. Onset detection and pitch detection are performed prior to applying the detection rules. The detection rules involve imposing empirical thresholds to the parameters discussed above. The results suggest that the system can successfully detect the playing techniques present in the guitar recordings.

For detecting playing techniques from electric guitar recordings, Kehling et al. (2014) proposed a comprehensive transcription system, which estimates both score-related parameters (i.e., pitch, onset, and offset) and instrument-specific parameters (i.e., string-fret combination and playing techniques). Following the previous work of Abeßer, Lukashevich, and Schuller (2010) and Abeßer, Dittmar, and Schuller (2011), the system proposed in Kehling et al. (2014) distinguishes between three plucking styles: *finger style*, *picked*, and *muted* and five expression styles: *bending*, *slide*, *vibrato*, *harmonics*, and *dead note* from isolated clean guitar signals. The system starts with onset detection and multipitch detection for estimating the score-related parameters. Based on the pitch estimation results, a partial tracking process is employed, which not only enables offset detection but also allows for correction of pitch estimation results. Each pair of onset and offset marks the region for a single note event and the input signal is cut into single-note segments accordingly. Frame-level audio features are computed and aggregated over the course of each note using statistic descriptors. A multiclass SVM classifier is then trained on these features to classify single-note segments into different playing technique classes. The classifier output and transcription results are postprocessed by plausibility filters designed based on guitar-specific domain knowledge. The plausibility filters eliminate improbable playing techniques usage determined by the classifier. This system was evaluated on the IDMT Guitar dataset, a guitar audio dataset annotated with respect to both score-related parameters and instrument-specific parameters. The results suggest that the system proposed in Kehling et al. (2014) can estimate both the score-related parameters and instrument-specific parameters at a high accuracy. In the guitar playing technique detection area, Kehling et al. (2014) is the first to attempt this task on both monophonic and polyphonic guitar recordings. Although expression styles used in chord playing are relatively limited, all plucking styles may be employed to play chords. Detecting chord plucking styles is a valuable addition to the playing technique detection task.

Different types of audio features and their effectiveness in classifying guitar playing techniques were studied in Su, Yu, and Yang (2014). In this comparative study, the authors extracted audio features from various representations including the magnitude spectrum, cepstrum, and phase derivatives. The three categories of features were processed by sparse coding, a feature learning technique that uses a precomputed

dictionary learned from training data to encode prominent information of audio representations. To evaluate the effectiveness of these features, the authors constructed the Guitar Playing Techniques (GPT) dataset, which consists of single-note electric guitar recordings with the seven types of playing techniques (*normal*, *muting*, *vibrato*, *hammer-on*, *pull-off*, *sliding*, and *bending*) studied in their paper. It is worth noting that the GPT dataset consists of not only clean-tone recordings, but also the same audio clips rendered in different guitar tones. The guitar tones differ in their levels of guitar effects such as distortion, reverb, and chorus. Introducing guitar effects improves the dataset quality because the audio clips now sound more similar to those found in real-world guitar performances. In their experiments, Su, Yu, and Yang (2014) trained SVM classifiers on different sets of features extracted from the GPT dataset and compared their performance with each other and with baseline models trained on features used in Abeßer, Lukashevich, and Schuller (2010) and general-purpose MFCC features. The authors found that sparse-coded features generally perform better than their non-sparse-coded counterparts, especially for low-level features. Moreover, features extracted from the cepstra and phase derivatives proved to be particularly helpful for distinguishing between similar expression styles such as bending, hammer-on, and pull-off.

While previous works mostly focused on detecting playing techniques from single-note recordings, Chen, Su, and Yang (2015) aimed to perform this task on entire tracks of guitar solo recordings. Specifically, they focused on detecting *bend*, *vibrato*, *slide*, *hammer-on*, and *pull-off*. The authors pointed out that the task of recognizing these frequency-modulating playing techniques can be modeled as a pattern recognition task on the melody contour. Therefore, the system developed in Chen, Su, and Yang (2015) starts with extracting melody contour from the entire track of guitar solo recording. It then selects candidate regions from the melody contour using algorithms designed specifically for each type of playing technique. Although the candidate selection algorithms can separate the playing technique candidate regions from the non-playing technique regions, they cannot successfully distinguish between regions corresponding to the five playing techniques. To classify between the playing technique classes, three sets of audio features are extracted from the candidate regions and SVM classifiers are trained to produce the final output. The audio features span across timbral features,

pitch-related features, and MFCC features. Five binary SVM classifiers, one for each playing technique class, are trained on isolated electric guitar solo recordings. The system was evaluated on isolated guitar tracks and guitar tracks with accompaniment. The results suggest that candidate selection followed by SVM classifiers can effectively detect the playing techniques from isolated guitar solo tracks. Exceptionally, when detecting the bending technique from accompanied guitar tracks, using only the candidate selection algorithms would yield better results than using the whole system. The authors attributed this to the fact that the SVM classifier was trained on unaccompanied guitar tracks and did not generalize well to the guitar tracks with accompaniment.

Extending the system proposed in Chen, Su, and Yang (2015), Su et al. (2019) developed an automatic guitar transcription system named *technique-embedded note tracking* (TENT). This system is capable of detecting playing techniques and transcribing note events in monophonic guitar solo recordings. Moreover, TENT uses detected playing techniques to inform the transcription process, which improved the transcription accuracy on realistic guitar performances as proved by the experiments (Su et al. 2019). The playing techniques covered by TENT include *bend*, *release*, *vibrato*, *hammer-on*, *pull-off*, and *slide*, which all modulate the F0. To recognize these techniques in the audio signal, TENT first estimates the F0 contour and then segments it into sub-melodies at the points where the F0 difference between two adjacent frames is higher than an empirical threshold. After this step, TENT divides the target playing techniques into two categories, which are addressed separately by two components in TENT. The first category includes normal instances of bend, release, slide, and vibrato. They generally do not introduce sudden frequency changes and can thus be captured within a sub-melody. TENT recognizes these techniques by inspecting the slope of each sub–melody. The second category consists of playing techniques that cannot be easily addressed on the sub-melody level. This includes hammer-on and pull-off, which often introduce a sudden change in F0. Short instances of bend, release, and slide also belong to this category as they tend to be overlooked due to their short temporal duration or subtle frequency change (Su et al. 2019). TENT employs CNN classifiers to recognize these techniques based on timbral features extracted from the audio signal. For an initial estimate, TENT naively transcribes each sub-melody to estimate the onset, offset, and

pitch of each note event. Then, it attempts to fix the transcription errors by rectifying the estimated note attributes using the playing techniques detected within or surrounding the note event. TENT was evaluated in a transcription task performed on a dataset of monophonic guitar solo phrases, and it outperformed an existing note tracking method that does not consider the playing techniques.

As shown by the previous works presented in this section, the qualities of the playing techniques vary a lot from one instrument to another, due to their specific physical and acoustic properties. The piano pedaling techniques color the timbre by producing short overlapping harmonic structures. The snare drum playing techniques produce different timbres by exciting the instrument using different materials or at different positions. A larger variety of playing techniques have been explored on the guitar and bass guitar, including the excitation styles, which produce subtle timbre nuances when exciting the string, and the expression styles, most of which modulate the frequency of sounding notes after the excitation of such notes. Despite the fundamental differences between these instruments, classifying their playing techniques generally involves extracting relevant audio features from candidate regions and then separating them in the feature space using either machine learning classifiers or empirical thresholds.

In this thesis, I focus on classifying some of the common playing techniques performed on the electric guitar. Following the steps of Chen, Su, and Yang (2015) and Su et al. (2019), the system presented in this thesis aims to classify expression techniques found in guitar solo recordings.

2.2 Audio Features and Machine Learning

As presented in the previous section, existing works on playing technique classification generally follow the same workflow. They start with preparing a set of candidates to be classified. Then, the candidates are represented by a set of audio features extracted from the audio signals. Finally, the candidates are separated in the feature space by decision boundaries learned from the data. This section presents some background information on audio feature extraction and machine learning techniques used in this thesis.

2.2.1 Audio feature extraction

Audio feature extraction has been an essential step in content-based music information retrieval (MIR). The objective of audio feature extraction is to convert the raw audio waveform into meaningful descriptors of various aspects of the sound. The audio features can be categorized in various dimensions. For example, based on their level of abstraction, audio features can be divided into low-level features, mid-level features, and high-level features. Typically, low-level features (e.g., spectral envelop, zero-crossing rate) are computed directly from the raw waveform, reflecting simple statistical properties of the signal. Mid-level features (e.g., pitch, onset) are often computed by combining low-level features and represent certain musical properties. High-level features (e.g., key, genre) describe music in terms of human perception and are thus the most musically meaningful features (Knees and Schedl 2016).

As presented in section 2.1.3, classifying between different playing techniques often involves low- to mid-level features. Here I introduce the extraction process for some common features used in playing technique classification. For a comprehensive discussion on audio feature extraction, please refer to Peeters (2004) and Knees and Schedl (2016).

Framing, windowing, and Fourier transform

Given a digital audio signal, the first step is to concatenate chunks of consecutive samples into frames. To produce meaningful audio features, the frame needs to be long enough to be perceivable by the human ear. Typical frame lengths are between 256 and 8,192 samples (Knees and Schedl 2016).

For time-domain features, the resulting frames can be used directly. For frequency-domain features, Fourier transform is used to convert each frame to a magnitude spectrum that represents the magnitudes of different frequency components. Before computing Fourier transform, a windowing function (e.g., Hann function) is often multiplied to the frame to avoid spectral artifacts. To compensate for the information loss caused by windowing, consecutive frames often have a certain amount of overlap. The difference between the frame size and the overlap is referred to as the hop size. Fourier transform is typically performed on the windowed frame to convert it

from the time domain to the frequency domain. What is commonly used for audio feature extraction is a variant of Fourier transform named fast Fourier transform (FFT). The output of FFT is a series of complex numbers, where the real part and imaginary part respectively indicate the strength of cosine and sine waves comprising the signal. Another popular variant of the Fourier transform is the discrete cosine transform (DCT). DCT uses only the real numbers rather than complex numbers, thus representing the signal with cosine waves only. To model the temporal evolution of the spectrum, short-time Fourier transform (STFT) is used. STFT computes the FFTs over consecutive frames and concatenates the resulting spectra. The output of STFT is referred to as the spectrogram.

Common audio features

Here I introduce some common audio features used in playing technique classification. These features are chosen because of their feasibility in playing technique classification demonstrated in previous works (Kehling et al. 2014; Chen, Su, and Yang 2015; Su et al. 2019).

Zero-crossing rate: Computed in the time domain, zero-crossing rate measures how many times the time-domain waveform crosses the horizontal axis within one frame. It roughly reflects the fundamental frequency for simple periodic signals as high-frequency signals typically has a higher zero-crossing rate. It has also been applied to distinguish between percussive sounds and noise (Gouyon, Pachet, and Delerue 2000).

Spectral centroid: As a frequency-domain feature, spectral centroid is defined as the center of gravity of the magnitude spectrum. It has been associated with the perceived brightness of the sound. Signals with a higher spectral centroid typically have more energy in high-frequency components and thus sound brighter. Spectral centroid has been applied to distinguish between different strings and different plucking techniques on the electric guitar (Kehling et al. 2014).

Spectral spread: Spectral spread is also referred to as the bandwidth of the magnitude spectrum. It represents the spread of the spectrum around the spectral centroid. It is obtained by viewing the magnitude spectrum as a distribution and calculating its variance.

Spectral flux: Spectral flux is defined as the difference in power spectra between two

consecutive frames. It measures the change in spectral content between frames. It has been commonly used for onset detection (Dixon 2006).

Spectral flatness: Spectral flatness measures how flat the spectrum is, which indicates the degree of similarity between the sound and white noise. White noise has a flat spectrum while tone-like sound typically has sharp spectral peaks (Dubnov 2004).

Spectral rolloff: Spectral rolloff is the frequency point below which a large portion of signal energy is contained. It is correlated to the cutting frequency between harmonic and noise (Peeters 2004).

Mel-frequency cepstral coefficients: Mel-frequency cepstral coefficients (MFCCs) are the first few coefficients that make up the mel-cepstrum. To obtain the MFCCs, a Fourier transform is first taken on the windowed frame. The frequency bins of the resulting spectrum are then mapped to the mel scale. A DCT is then performed on the logarithm of this intermediate representation (i.e., mel-spectrum), leading to the mel-cepstrum. Finally, the first few coefficients of the mel-cepstrum are taken as the MFCC feature. MFCCs are widely used in speech recognition (Rabiner and Juang 1993) and first applied to music modelling by Logan (2000). In playing technique classification, the MFCCs have been used as a timbral feature in Chen, Su, and Yang (2015).

All features listed here are computed on the frame level and thus represent only one frame. However, it is common practice to aggregate these instantaneous descriptors over a certain period of time (e.g., note event, measure, piece) using statistical values such as maximum, minimum, or median (Knees and Schedl 2016). When analyzing audio signals that evolve over time, it is also useful to record the difference in certain features between consecutive frames. The first- and second-order time derivative of a certain feature are referred to as the delta feature and delta-delta feature, respectively. For example, delta-MFCCs and delta-delta MFCCs are among the most used delta features. They measure the temporal modulations of MFCCs, which proved important in audio classification (McKinney and Breebaart 2003).

2.2.2 Machine learning classifiers

The abundance of data and computing power has facilitated the development of complex algorithms that learn from a large set of input data samples (i.e., training set) and make

predictions on new samples (i.e., test set).

Supervised machine learning is mainly used for solving two types of tasks: regression and classification. In both tasks, the model aims to learn a mapping from the input features to the output target. In regression tasks, the output target is a continuous numerical variable, while in classification tasks, the output target is a discrete categorical variable.

The playing technique classification task attempted in this thesis is a typical multiclass classification task. In a classification task, each training sample, typically represented by a set of features, has a label that represents the class that it belongs to. The features are typically a set of numerical or categorical attributes that are either hand-crafted using domain knowledge or automatically chosen by the classifier model. The label is typically a categorical value indicating the class from which the sample is taken. The objective of the classification task is, given the features extracted from the test samples, to predict their class labels.

Classifiers are the models used in classification tasks. In general, machine learning classifiers try to approximate a mapping function from the features to the class labels. In the training stage, the classifiers establish a decision boundary in the feature space that is calculated from a large amount of training data. Based on the decision boundary, new data samples to be classified can be assigned a label based on its position relative to the decision boundary in the feature space.

There exist various types of classifiers, each based on its own assumptions and thus having its own way of calculating the decision boundary. Common classifiers include logistic regression, naive Bayes, k-nearest neighbors (k-NN), decision tree, random forest, Gaussian mixture model (GMM), support vector machine (SVM), and artificial neural network. Here I introduce SVM as it is the main classifier used in this thesis and other playing technique classification systems (Abeßer, Lukashevich, and Schuller 2010; Kehling et al. 2014; Su, Yu, and Yang 2014; Chen, Su, and Yang 2015; Wu and Lerch 2016; Liang, Fazekas, and Sandler 2018a). For a comprehensive review on machine learning classifiers, please refer to, for example, Alpaydin (2014).

SVM is a type of supervised machine learning model. Mostly used as a classifier, the objective of SVM is to find an optimal hyperplane that separates the data points belonging to two classes in a feature space. For instance, given a linearly separable set of data points

in a 2-dimensional feature space, there may be an infinite number of lines that separate the two classes perfectly. SVM tries to obtain the optimal separation, while other linear classification algorithms such as logistic regression would end up with an arbitrary line among all possible solutions.

SVM finds the optimal hyperplane by maximizing its distance to the closest data points (i.e., margin) of both classes. The coordinates of these data points in the feature space are the support vectors. Support vectors influence the position and orientation of the optimal hyperplane, which is thus "supported" by the support vectors. If any of the support vectors are removed, the SVM will end up finding a different hyperplane. Maximizing the margin distance reinforces the separation so that future data points can be assigned, based on their coordinates in the feature space, to one of the two classes with more confidence.

For data points that are non-linearly separable, SVM can also perform classification efficiently using kernel functions (Pradhan 2012). Kernel functions takes low-dimensional input features and transform them into a higher-dimensional feature space, in which the data points become linearly separable. The optimal hyperplane found in the higher-dimensional feature space, when transformed back to the original feature space, usually gives a non-linear decision boundary.

Basic SVM is typically only applicable to binary classification tasks, where the data points are associated with only two classes. For multiclass classification, there are various approaches to break a multiclass classification problem into several binary classification problems, so that SVM can still be applied. One commonly used approach is the one-versus-all method, which trains an SVM binary classifier for each class, and each classifier distinguishes one class from the rest.

As presented in section 2.1.3, SVMs have been successfully applied in playing technique classification research for the bass guitar (Abeßer, Lukashevich, and Schuller 2010), electric guitar (Kehling et al. 2014; Su, Yu, and Yang 2014; Chen, Su, and Yang 2015), snare drum (Wu and Lerch 2016), and piano (Liang, Fazekas, and Sandler 2018a). Beyond playing technique classification, SVMs have also been used for automatic transcription and automatic music tagging tasks. Poliner and Ellis (2006, 2007) performed classification-based piano transcription, where SVMs trained on spectral features were used to classify frame-level note instances (i.e., the presence or absence of

each note in a given frame). 87 one-versus-all linear SVM classifiers were trained to detect the 87 notes (corresponding to MIDI note numbers 21 to 107). It was shown that more advanced kernels such as radial basis function (RBF) only gave limited performance gain while causing significant increase in computational complexity. SVM has also been used in semantic music retrieval. In Barrington et al. (2008), two acoustic features (MFCCs and chromagram) and two social context features (social tags and web-mined tags) were used to predict semantic tags for music pieces. One-versus-all SVM classifiers were trained for each semantic tag using the four individual feature sets and their combination. It was shown that the SVMs trained using the combined feature set yielded superior results on most of the tags.

Following the steps of Chen, Su, and Yang (2015) and Su et al. (2019), this thesis aims to classify expression techniques found in guitar solo recordings. The proposed playing technique classification system is based on SVM classifiers trained on various audio features as presented previously. Additionally, this thesis presents an automated workflow for generating synthesized guitar recordings and annotations from publicly available Guitar Pro (GP) files. The objective is to explore the usability of the generated ground-truth data in developing the playing technique classification system and compare its performance with those achieved by previous work in this area.

Chapter 3

Methodology

The proposed playing technique classification system consists of three components: the monophony detector, the note-event separator, and the playing technique classifier. For training and testing this system, a ground-truth dataset was compiled using annotations and audio signals generated from Guitar Pro (GP) files. This chapter presents the process of ground-truth generation and the design of each component of the proposed system. Unless otherwise noted, all audio features were extracted using *Librosa* (McFee et al. 2022). The classifier models and machine learning routines were implemented using *Scikit-learn* (Pedregosa et al. 2011).

3.1 Ground-truth Generation

The task of playing technique classification calls for a dataset of guitar audio recordings with their corresponding playing technique annotations. A dataset suitable for this task should satisfy the following two requirements: First, the audio recordings must include a large number of playing technique instances. Second, the corresponding annotations must precisely record the type and timing of every playing technique instance present in the recordings.

As previous automatic guitar transcription research has mostly focused on estimating the basic note-event parameters and string-fret combinations, most existing guitar datasets, such as GuitarSet (Xi et al. 2018), do not include annotations about playing techniques. Therefore, despite their feasibility in training and testing guitar transcription

models, they are not directly useful for the playing technique classification task.

At the time of writing, there exist two guitar datasets dedicated for playing technique classification research. The IDMT guitar dataset (Kehling et al. 2014) consists of single-track electric guitar audio recordings from live performances, covering a series of plucking techniques and expression techniques. However, this dataset focuses on classifying playing techniques in single-note recordings, where each audio file contains only one note event. Expression techniques that typically occur during the transition between note events (e.g., hammer-on), are thus not covered by the dataset. Similarly, the Guitar Playing Techniques (GPT) dataset (Su, Yu, and Yang 2014) contains guitar solo recordings taken from live performances with annotations covering all five types of playing techniques explored in this thesis. However, the number of instances in each playing technique class is rather small. In the GPT dataset, each recorded note is rendered in seven different guitar tones (e.g., clean, distortion, reverb). Bend, the most frequent playing technique, has 1,281 notes (i.e., 183 distinct notes in each tone). Pull-off, the least frequent playing technique, has 525 notes (i.e., 75 distinct notes in each tone).

In this thesis, I leverage publicly available GP files to generate ground-truth data for the playing technique classification task. The goal is to generate audio signals and their corresponding annotations with minimal human intervention. This section introduces the GP file format and the ground-truth generation process.

3.1.1 Guitar Pro background

Guitar Pro is a proprietary song-writing software developed by Arobas Music. Its core functionality is tablature-based score editing for fretted stringed instruments. As its name suggests, Guitar Pro specializes in tablature editing and playback for the guitar. Beyond the score editing basics, it allows guitar playing technique notations. Another key functionality of Guitar Pro is the realistic-sounding audio playback powered by its Real Sound Engine (RSE), a sound bank comprising a wide variety of guitar tones and effects. The multitrack tablature can be edited in the Guitar Pro software and then played back or exported as an audio file, where each track is rendered in audio by the RSE with customized settings. The score content and synthesizer settings are encoded in the software's proprietary file format, often referred to as the Guitar Pro (GP) file. The

GP file, as the most popular guitar tablature format, is widely accepted by the guitar player community, and there are vast quantities of GP files hosted online. This makes the GP files an ideal data source for the playing technique detection task attempted in this thesis. The synthetic audio files provide the signals for analysis, while the score information stored in the GP files serves as the corresponding ground-truth annotations.

3.1.2 Data collection

There are a few guitarist community websites that distribute user-contributed GP files for free. The most popular ones include Ultimate Guitar,[1] Songsterr,[2] and Guitarprotabs.[3] These websites allow contributors to upload GP files transcribed from popular songs and other users to view, rate, and download them. At the time of writing, Ultimate Guitar hosts 202,618[4] GP files, from which I manually collected the all-time top 100 GP files (by rating) and the all-time top 100 GP files (by hits), a total of 147 unique files, for this research. Since most GP files are manually transcribed by the users, they are prone to errors and inaccuracies. It is expected that, among the large quantity of GP files hosted on the website, the highest-rated and the most-visited files are of higher quality. The songs covered by this set of files are mostly in the genre of rock and metal. The tones and effects used on the guitar tracks range from clean acoustic guitar tone to electric guitar tone with distortion and other effects. More details about the GP files used in this thesis can be found in table A.1 and table A.2 in the appendix.

3.1.3 Generating annotations

As a comprehensive digital tablature format, a GP file contains all information about the tablature content and thus all information needed to generate the ground-truth annotations for its synthetic audio. To extract this information in an automated manner, I developed a parser based on *PyGuitarPro*,[5] a python library for reading, writing, and

1. https://www.ultimate-guitar.com
2. https://www.songsterr.com
3. https://guitarprotabs.org
4. Accessed 2021-12-14, 17:36 EST
5. https://pyguitarpro.readthedocs.io/en/stable

manipulating GP files. In this section, I discuss the process of parsing GP files and generating annotations.

GP files created by different versions of the Guitar Pro software have slightly different file structures and different filename extensions. At the time of writing, PyGuitarPro supports GP3, GP4, and GP5 files, which cover most files collected for this dataset. The few GP files under unsupported versions were first converted to the GP5 version before being processed.

The GP file parsing workflow is as follows: Each GP file is read into memory as a *song* object, which consists of one or multiple *tracks*, including guitar tracks and other instrumental tracks. Each *track* contains multiple *measures*, depending on the length of the song. Each *measure* contains a certain number of *beats*. The term *beat* in the Guitar Pro file differs from the general musical concept of beat. For example, for a song with a *4/4* time signature, musically, there are four beats in a bar, no matter how many notes are played. For the excerpt shown in figure 3.1, there are a total of six notes in the first bar. The first quarter note is on the first beat. The next two eighth notes belong to the second beat. The next quarter note takes up the third beat, and the last two eighth notes fill up the fourth beat. However, in a GP file, this same bar has six *beats*, where each distinct onset in time is considered one *beat*. Similarly, the second bar has four musical beats but six GP *beats*. In the rest of this thesis, the term *beat* refers to the GP *beat* and not the general musical beat. Each non-empty *beat* in the GP file can have one *note* (i.e., a single note) or multiple *notes* (i.e., a chord). The attributes needed for the annotations are found on the *beat* level and the *note* level, as shown in figure 3.2.

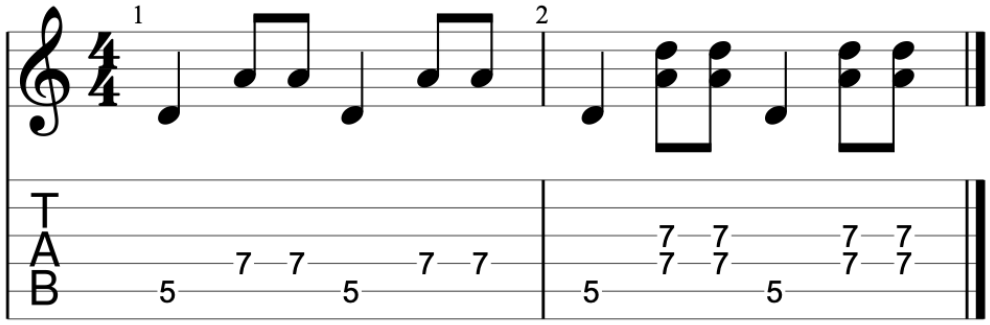

Figure 3.1: A two-bar excerpt of Guitar Pro file.

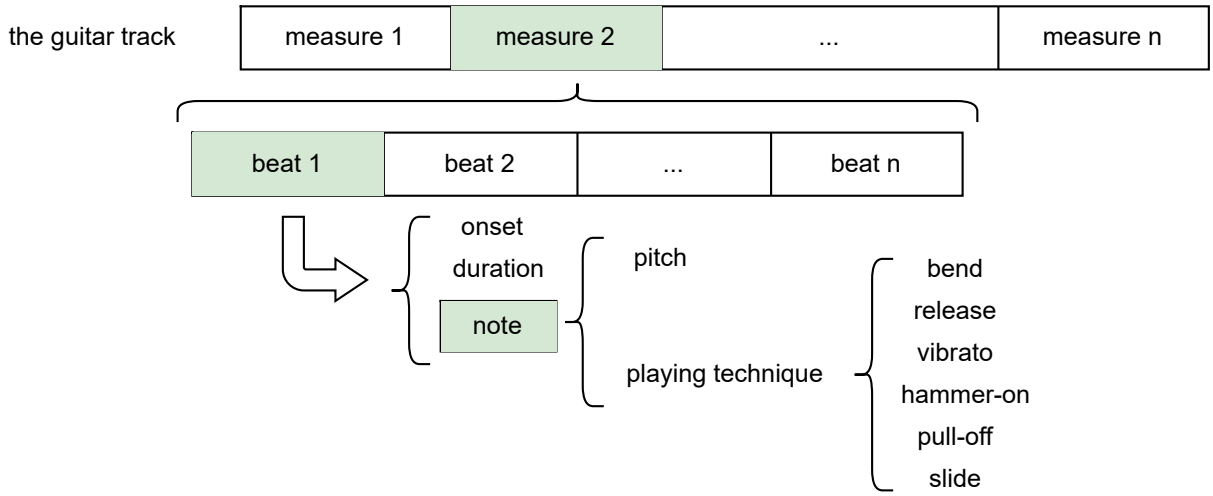

Figure 3.2: Extracting note-level information from a Guitar Pro file.

Before generating annotations from the GP files, a few preprocessing steps are needed. Most GP files collected are multitrack GP files, which typically consist of one or multiple guitar tracks, a bass track, a percussion track, and other instruments used in the song. As the aim is to recognize playing techniques in the guitar solo recordings, on the track level, the guitar tracks are separated into single-track GP files while the other tracks are discarded. Within every guitar track, the repetitions, alternate endings, and tempo changes are disabled in order to simplify the computation of the onset and offset time of each note.

After the preprocessing stage, the 147 multitrack GP files have been converted to 383 single-track GP files. The increase in the number of files is due to the multitrack nature of the GP files. The 147 multitrack GP files have a total of 383 guitar tracks, each of which is converted to a single-track GP file, thus the 383 single-track GP files. For each single-track GP file, an annotation file is created, which contains a list of note-level annotations. The process of extracting note attributes from a GP file is illustrated in figure 3.2.

For each note event in the annotations, the pitch is recorded as a MIDI note number. The onset and duration are converted from relative time to absolute time measured in seconds. A Boolean value is assigned to each playing technique. For playing techniques that occur within an individual note event (i.e., bend and vibrato), the value indicates

class	number of instances	percentage (%)
normal	235,676	93.20
bend	3,873	1.53
vibrato	3,038	1.20
hammer-on	3,564	1.41
pull-off	4,735	1.87
slide	1,986	0.79
total	252,872	100.00

Table 3.1: Playing technique class distribution in the ground-truth dataset.

whether the technique is present in the current note event. For playing techniques that occur during the transition between note events (i.e., hammer-on, pull-off, and slide), the value indicates whether the technique is present in the transition event following the current note event. Here, a transition event is defined as a short audio segment surrounding the transitional instant between two consecutive note events, where the first note event ends and the next begins.

The distribution of the playing technique classes is shown in table 3.1. The *normal* class consists of both note events and transition events where no special expression technique is applied.

3.1.4 Generating audio signals

Now that the annotations have been generated, this section discusses the audio generation process of the dataset. The Guitar Pro software includes the RSE sound bank, which serves as an audio synthesis tool supporting the audio playback of the tablature. RSE supports many types of instruments, but in this thesis, it is only used as a customizable guitar synthesizer. The synthesizer settings, such as the guitar tone and effects used, are encoded in the GP files.

The RSE is a built-in part of the Guitar Pro installation package. At the time of writing, the Guitar Pro software does not support converting GP files to audio in batches. This motivated a search for alternative solutions. I first experimented with MuseScore,[6] a popular open-source music notation software. Although it did support

6. https://musescore.org/en

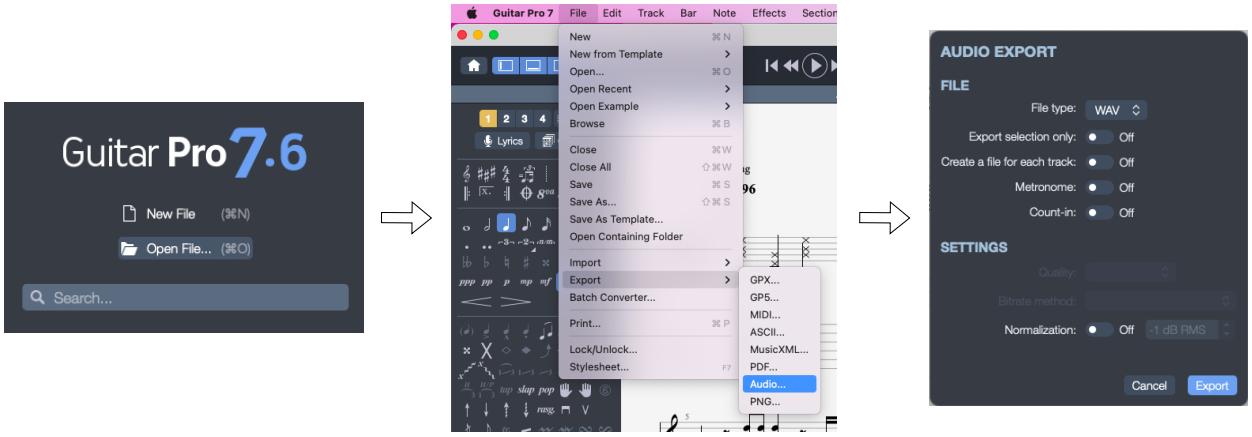

Figure 3.3: Exporting a Guitar Pro file to audio.

reading the GP format, playing techniques other than bend and vibrato were not interpreted correctly when converted to audio. TuxGuitar[7] is another open-source alternative to Guitar Pro. However, the playing techniques rendered by TuxGuitar sound much less realistic than their Guitar Pro counterparts.

To make the best effort to generate high-quality synthetic recordings, I chose to perform the conversion one file at a time, using the graphical interface of the Guitar Pro software. In preparing data for this thesis, the repetitive conversion process was automated by a Python script that controls mouse movement and clicks in a pre-determined pattern. The script is written with *PyAutoGUI*,[8] a Python library that supports automated interaction with graphical user interface (GUI) applications.

Figure 3.3 shows a series of screenshots, demonstrating the audio generation process using the Guitar Pro software. First, the GP file is opened using the "Open File" button. The opened GP file would then be displayed as an editable tablature in the Guitar Pro window. Next, the GP file is exported to audio via the File – Export – Audio menu. After clicking on the export button in the audio export window and then selecting the path, the current GP file would be exported to an audio file in WAV format. This process is repeated for every preprocessed single-track GP file in the dataset, with the auto clicker script controlling the mouse based on pre-measured screen coordinates.

7. http://www.tuxguitar.com.ar
8. https://github.com/asweigart/pyautogui

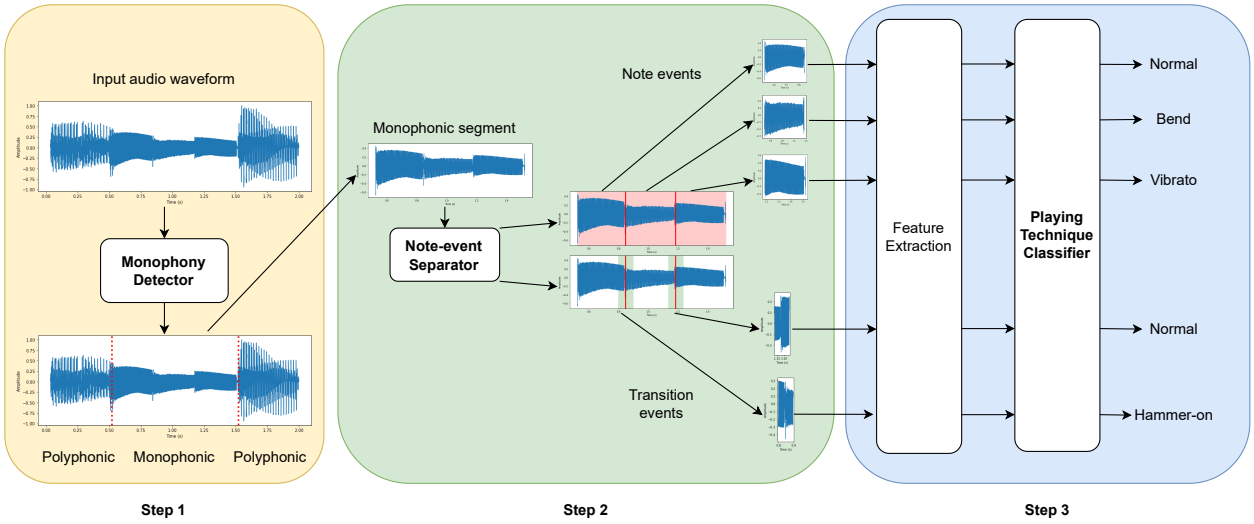

Figure 3.4: Diagram of the proposed playing technique classification system.

With the audio file and annotations generated from the same GP file, every monophonic note event present in the audio signal has its pitch, onset, duration, and associated playing techniques recorded in the corresponding annotations. This lays a solid foundation for the training and testing of the proposed playing technique classification system.

3.2 Playing Technique Classification

A summary of the proposed playing technique classification system is illustrated in figure 3.4. The input guitar audio signal is first processed by the monophony detector, which extracts the monophonic segments from the signal, where most playing technique instances occur. Then, the note-event separator estimates the onset and offset of each note event present in the monophonic segment and splits it into note events and transition events. Finally, the playing technique classifier recognizes the playing techniques. In this section, I describe in detail each component of this system.

3.2.1 Preprocessing: monophony detector

As discussed in 1.3, most playing technique instances occur in the monophonic segments of the guitar audio signal. Therefore, it is reasonable to focus on the monophonic segments. The job of the monophony detector is to locate and then separate the monophonic audio segments from polyphonic ones. Its key component is a frame-level binary classifier, classifying an audio frame as either polyphonic or monophonic. Based on the classifier's output, adjacent monophonic frames are concatenated into monophonic segments, which are then separated from the input audio signal for further analysis. This allows the system to operate on input audio without imposing explicit monophony requirements.

The classifier is trained in a supervised fashion, where each audio frame is labeled as either polyphonic or monophonic. The audio frames are generated from the guitar audio signals synthesized from the preprocessed single-track GP files. Those generated from the polyphonic segments are labeled as polyphonic frames while those generated from the monophonic segments are labeled as monophonic frames. Each frame is represented by a feature vector.

Two sets of audio features are extracted for the monophony detector: the mel spectrum and mel-frequency cepstral coefficients (MFCCs). These features are used to train a support vector machine (SVM) classifier, which distinguishes between polyphonic and monophonic frames. For any input audio signal, the trained monophony detector will serve as a preprocessing step, returning the timestamps of the beginnings and ends of monophonic segments. The monophonic segments are then cut out according to these timestamps and serve as the subjects of the downstream note-event separation and playing technique classification tasks.

3.2.2 Preprocessing: note-event separator

After obtaining the monophonic audio segments using the monophony detector, the next step is to locate the note events in these segments. Since the playing techniques covered in this thesis typically occur during a note event or the transition between note events, the signal regions corresponding to these events are the regions of interest. As the second component of the proposed playing technique classification system, the

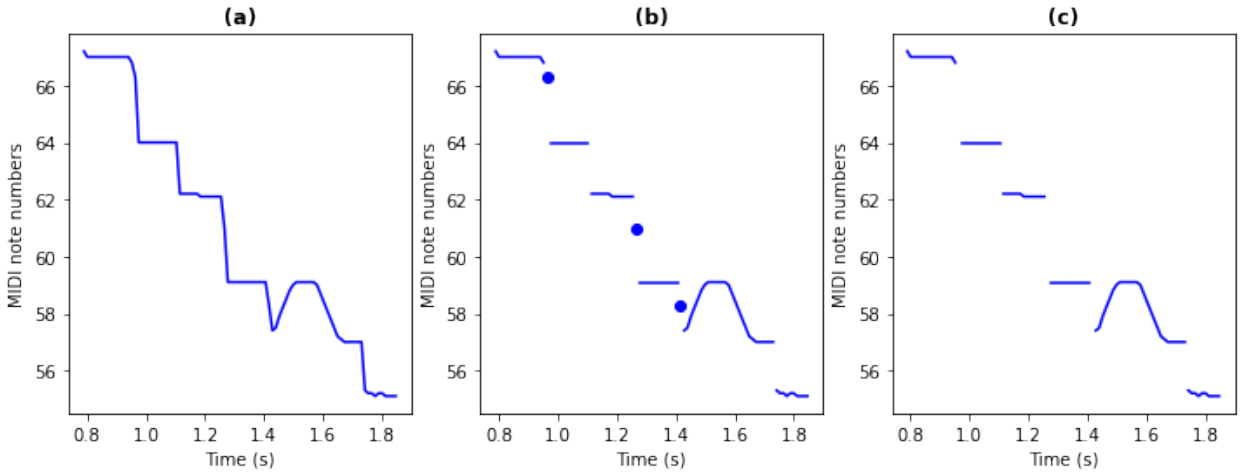

Figure 3.5: An example showing F0 segmentation-based note-event separation in action. (a) The raw F0 curve with the frequency axis mapped to MIDI note numbers. (b) The F0 curve segmented at points where the pitch difference between two consecutive frames exceeds the pitch difference threshold. Each segment represents a note event. The dots are spurious note events that last for only one frame. (c) The output note events, represented by F0 segments, with spurious note events removed.

note-event separator aims to estimate the timing information of the note events in the monophonic guitar audio signal. This enables the system to locate and analyze the individual note events and transition events (i.e., the regions of interest). This section introduces the two note-event separation strategies implemented for this thesis: the F0 segmentation-based strategy and the onset detection-based strategy.

Note-event separation via F0 segmentation

Inspired by Chen, Su, and Yang (2015), I implemented a note-event separator that directly operates on the F0 curve, referred to as the F0 segmentation-based strategy. Similar methods have also been employed in Adams, Bartsch, and Wakefield (2006) and Kong and Yu (2017). To illustrate this note-event separation strategy, a running example is shown in figure 3.5.

For an input monophonic audio segment, the pYIN algorithm (Mauch and Dixon

2014) is first applied to extract the F0 curve. A raw F0 curve returned by pYIN is shown in figure 3.5(a). Next, the continuous F0 curve is cut into note events. Two adjacent frames are considered the same note event if their pitch difference is smaller than an empirical threshold set around one semitone, because an instant pitch change of more than one semitone typically indicates the transition from one note to another.

As shown in figure 3.5(b), the continuous F0 curve has been cut into discrete note events. It is assumed that a valid note event should have a minimum duration of 46 ms, which corresponds to the duration of a 32nd note at 160 bpm. Note events shorter than this threshold, denoted by round dots in figure 3.5(b), are considered spurious and discarded. Six discrete note events are eventually obtained, as shown in figure 3.5(c).

In order to recognize the playing techniques that occur during the transition between note events, a transition event is defined as a short audio segment surrounding the cutting point between two adjacent note events. While a note event can have arbitrary length, a transition event is set to have a fixed length of five frames, with the transition frame in the center. Using a frame size of 2,048 and a hop size of 1,024 samples, the five-frame transition event has a duration of 139 ms. This design is consistent with Chen, Su, and Yang (2015), which also used a fixed-length segment centered at the candidate frame to characterize the transition event. Figure 3.6 shows the location of note events and transition events in the original waveform. The note events are highlighted in red, while the transition events are highlighted in green.

Note-event separation via onset detection

The F0 segmentation-based strategy presented above has the advantage of simplicity. The note-event separation relies on the F0 estimation algorithm alone. However, in some cases, it may not recognize the note onsets correctly. For example, if multiple notes of the same pitch are played consecutively with little rest in between, the F0 segmentation-based strategy is likely to mistakenly merge them into one longer note, because the pitch difference between the frames would never exceed the threshold. As an alternative, it might be more reliable to employ an onset detection algorithm to cut the monophonic audio signal into individual note events (Brossier, Bello, and Plumbley 2004). This is referred to as the onset detection-based strategy.

In implementing the onset detection-based strategy, the onset detection algorithm

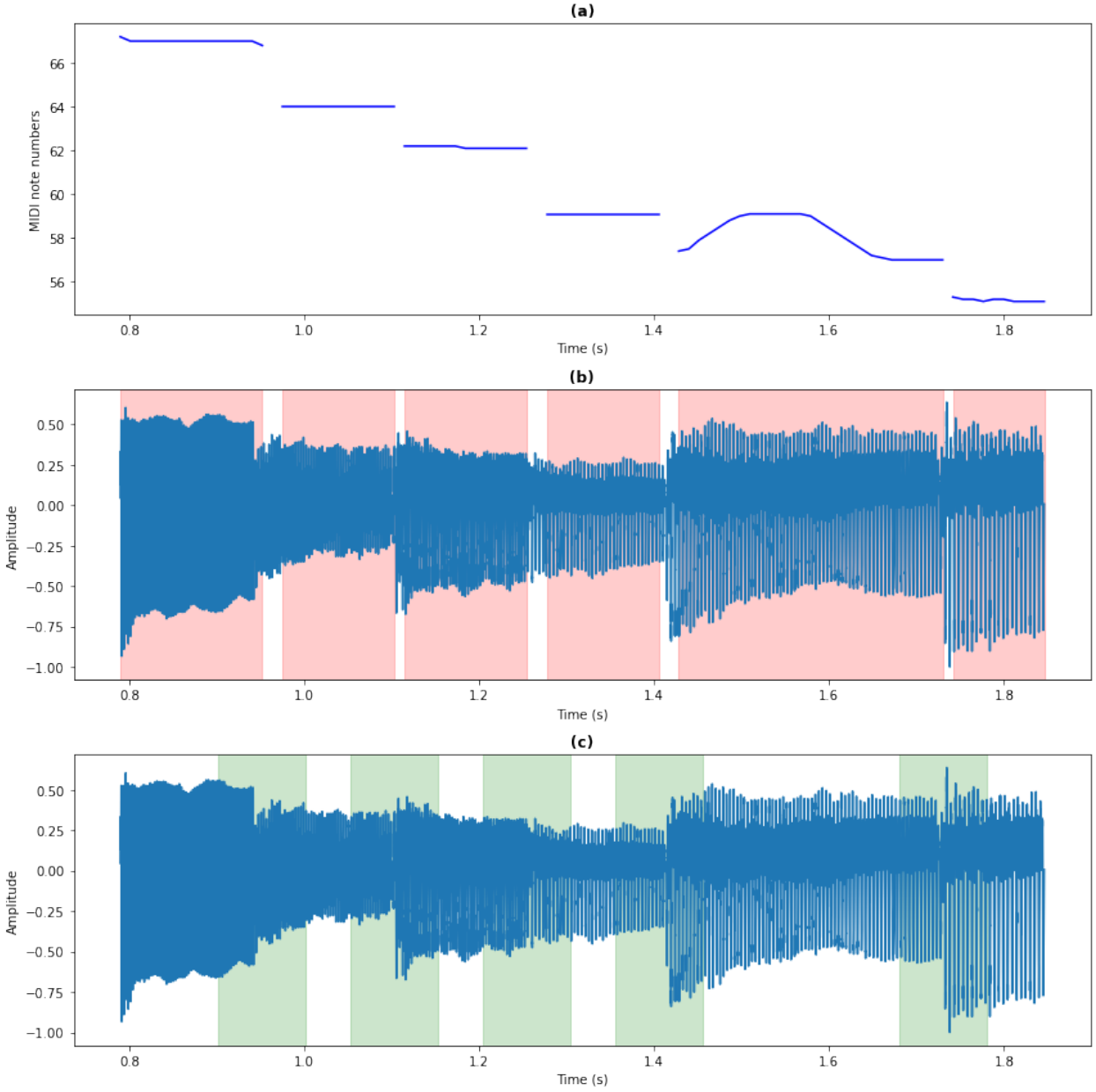

Figure 3.6: Note events and transition events localized with F0 segmentation. (a) The output note events represented by F0 segments. (b) The signal waveform with regions corresponding to note events highlighted in red. (c) The signal waveform with regions corresponding to transition events highlighted in green.

based on spectral flux envelop is used (McFee et al. 2022). It returns the onset time of each note event detected in the signal. As the analyzed signal (i.e., the monophonic segment returned by the monophony detector) is strictly monophonic with no silence, it is assumed that the onset of the current note event marks the offset of the previous note event. The example signal used in figure 3.6 is shown again in figure 3.7, where the detected onsets are marked by vertical red lines. Note events are found between the detected onsets, highlighted in red. The transition events are defined as five-frame segments centered at the note-event offsets, highlighted in green.

These two note-event separation strategies will be tested and compared in terms of their ability to identify the onset and offset of note events in the monophonic audio signals synthesized from the single-track GP files. The test process will be detailed in section 4.2.2.

For the input monophonic signal obtained using the monophony detector, the note-event separator will produce the signal regions corresponding to note events and transition events. Audio features are then extracted from these regions of interest and finally sent to the playing technique classifier.

3.2.3 Playing technique classifier

As discussed in the previous sections, the monophony detector takes as input the guitar audio signal and produces its monophonic segments. The monophonic segments are then processed by the note-event separator, which identifies the regions of interest. The final step is to recognize their playing techniques, using the playing technique classifier.

As presented in 3.1, the note events and transition events have been assigned labels according to their associated playing techniques. A note event or transition event can be one of the six classes: *normal*, *bend*, *vibrato*, *hammer-on*, *pull-off*, and *slide*. Note that these labels are mutually exclusive. That is, any note event or transition event can only belong to one of the playing technique classes.

In the feature extraction stage, each region of interest is represented by the following three sets of audio features:

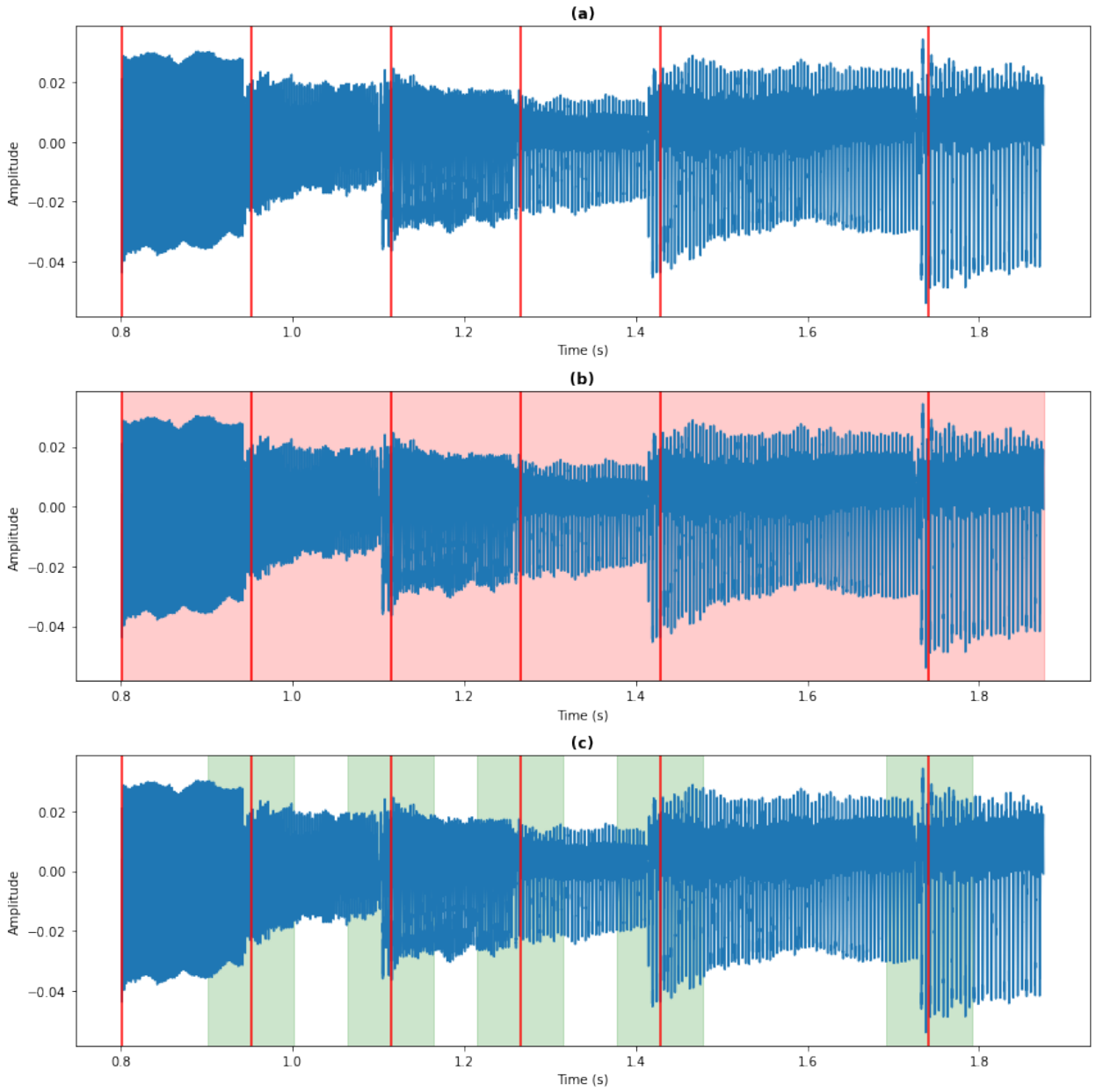

Figure 3.7: Note events and transition events localized with onset detection. (a) The signal waveform with detected onset represented by red vertical lines. (b) The signal waveform with regions corresponding to note events highlighted in red. (c) The signal waveform with regions corresponding to transition events highlighted in green.

MFCC: The MFCC features include the first 20 MFCCs[9] of each frame and their first- and second-order deltas, producing a 60-dimensional feature vector for each frame.

Timbre: The timbral features include the spectral centroid, spectral bandwidth, spectral flatness, spectral rolloff , spectral flux, zero-crossing rate and their first- and second-order deltas, resulting in an 18-dimensional feature vector for each frame.

Pitch: The pitch features include the estimated F0 time sequence and its first- and second-order deltas, resulting in a 3-dimensional feature vector for each frame.

All three sets of features are concatenated and then aggregated over the duration of each region of interest, by taking the six statistics: mean, standard deviation, maximum, minimum, skewness, and kurtosis. Each region of interest is eventually represented by a 486-dimensional feature vector. An RBF-kerneled SVM classifier is trained to distinguish between the six playing technique classes.

9. In speech and general sound analysis, it is common to use the first 8–13 MFCCs (Peeters 2004). However, music analysis typically requires higher-order coefficients (Mitrović, Zeppelzauer, and Breiteneder 2010). Here, 20 is chosen empirically after comparing the models trained separately on 8, 13, 20, and 40 MFCCs.

Chapter 4

Experiments

This chapter first gives an overview of the dataset used in the experiments. Then, the experiments are presented in two segments. Section 4.2 presents the experiments conducted for developing each individual component, and Section 4.3 presents the end-to-end integration test performed on the entire system. In each section, I first describe the experiment setup and then discuss the results. The source code for the whole workflow is published online[1] for reproducibility.

4.1 Dataset Overview

As presented in section 3.1, the ground-truth dataset used in this thesis was constructed using publicly available Guitar Pro (GP) files. Specifically, a total of 147 GP files were collected, most of which are multitrack GP files. A multitrack GP file contains one or more guitar tracks. After preprocessing, the 147 multitrack GP files became 383 single-track GP files, which were in turn converted to 383 annotation files and 383 audio files. This file conversion process is shown in figure 4.1.

To make sure that the system does not overfit to the set of data that it is trained on, I first split the ground-truth dataset into two subsets: the development set and the test set. The development set contained 100 multitrack GP files, and the test set contained the remaining 47 multitrack GP files. This split is illustrated in figure 4.2.

1. https://github.com/jwang44/GuitarPro-Stuff/tree/main/playing%20technique%20detection

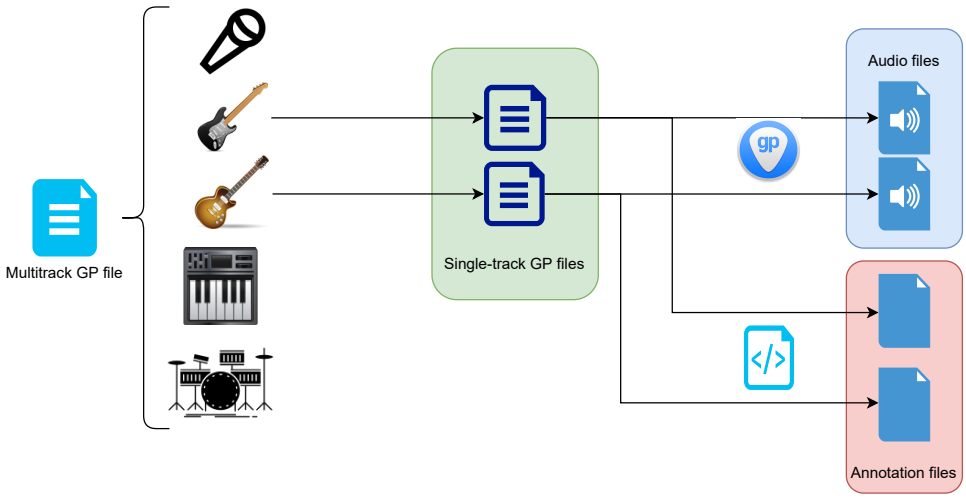

Figure 4.1: Converting multitrack GP files to audio files and annotation files. The guitar tracks are extracted from the multitrack GP file and saved as single-track GP files. Using the Guitar Pro software, the single-track GP files are converted to audio files. Using the PyGuitarPro library, the annotation files are constructed using information extracted from the single-track GP files.

The development set was used to train the machine learning models (i.e., the monophony detector and playing technique classifier) and to find appropriate parameter values for the note-event separator. Once the models and parameters were finalized, the entire system was tested end-to-end on the test set. Because the test set was never exposed to the system during training, the results obtained on the test set would reflect how well the system would generalize to previously unseen data.

4.2 Experiments on Individual Components

In this section, I present the experiment setup and results of each individual component, namely the monophony detector (4.2.1), the note-event separator (4.2.2), and the playing technique classifier (4.2.3). Using the development set, the monophony detector and playing technique classifier were independently trained and tested via nested cross validation. For the note-event separator, the onset detection-based strategy and F0

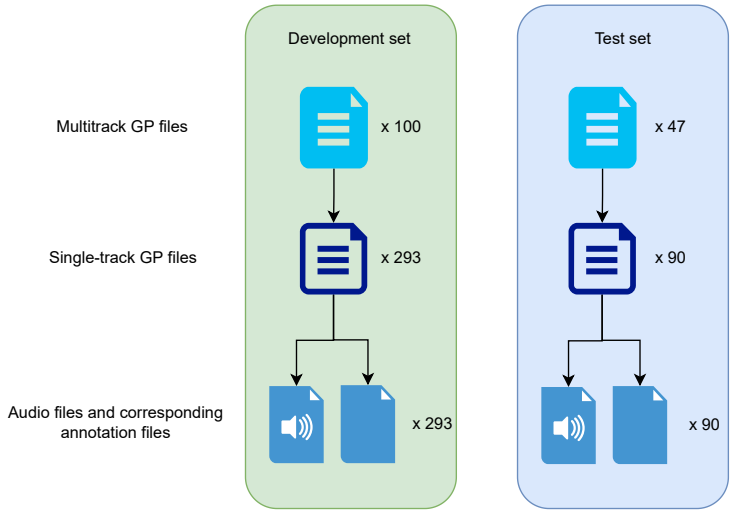

Figure 4.2: Overview of the development set and the test set. The development set consisted of 100 multitrack GP files, which were eventually converted to 293 audio files and 293 annotation files. The test set contained 47 multitrack GP files, which were eventually converted to 90 audio files and 90 annotation files.

segmentation-based strategy were compared in terms of their performance achieved on the development set. These experiments were performed to justify the design of each component and verify their effectiveness.

4.2.1 Monophony detector

Experiment setup

The core of the monophony detector is a binary classifier. Each input audio frame, represented by a feature vector, would be classified as either monophonic or polyphonic.

For training and testing the monophony detector, the 293 audio files in the development set were split into a total of 2,296,608 equal-length frames, using a frame size of 1,024 at a sampling rate of 44,100 Hz. The frame size was chosen empirically by running a grid search over different values. For complete grid search results, please see table 4.1.

Each frame was labeled as either monophonic or polyphonic according to the

n_mfcc	frame_size	clf	mean_accu (%)	var_accu ($*10^{-3}$)
8	1024	LR	81.5	1.09
		LSVM	81.1	1.10
		SVM	88.3	0.73
	2048	LR	83.8	0.52
		LSVM	83.7	0.46
		SVM	89.6	0.25
	4096	LR	79.9	0.35
		LSVM	79.8	0.30
		SVM	88.1	0.23
13	1024	LR	82.0	0.85
		LSVM	81.5	0.77
		SVM	93.6	0.26
	2048	LR	84.5	0.58
		LSVM	84.2	0.49
		SVM	93.6	0.12
	4096	LR	81.8	0.74
		LSVM	81.5	0.64
		SVM	93.0	0.06
20	1024	LR	82.9	0.91
		LSVM	82.5	0.85
		SVM	96.0	0.04
	2048	LR	84.9	0.62
		LSVM	84.6	0.58
		SVM	95.8	0.04
	4096	LR	82.9	0.81
		LSVM	82.5	0.65
		SVM	95.3	0.03
40	1024	LR	83.4	0.79
		LSVM	82.9	0.78
		SVM	**97.0**	**0.02**
	2048	LR	84.7	0.58
		LSVM	84.3	0.53
		SVM	96.8	0.03
	4096	LR	83.4	0.73
		LSVM	82.6	0.60
		SVM	96.8	0.05

Table 4.1: Grid search results for the monophony detector. The searched parameters are the number of MFCCs (n_mfcc), the frame size (frame_size), and the classifier type (clf). LR: logistic regression, LSVM: support vector machine with linear kernel. SVM: support vector machine with radial basis function kernel. The mean accuracy (mean_accu) and variance (var_accu) obtained in the five-fold cross validation are shown.

class	number of frames	percentage(%)
monophonic	1,170,782	50.98
polyphonic	1,125,826	49.02
total	2,296,608	100.00

Table 4.2: Class distribution of the audio frames in the development set. The audio files in the development set were split into equal-length frames labeled as either monophonic or polyphonic. These frames were later used to train and test the binary classifier.

annotations generated from the GP files. If a frame belonged to a single note, it was labeled monophonic. If it belonged to a chord, it was labeled polyphonic. Silent frames were discarded. The class distribution of the classified frames is shown in table 4.2.

In the experiments, three types of classifiers were used: logistic regression, support vector machine (SVM) with linear kernel, and SVM with radial basis function (RBF) kernel. Both logistic regression and linear SVM are linear models, indicating a linear decision boundary in the feature space between the classes while the SVM with an RBF kernel is a nonlinear model, indicating a more complicated, nonlinear decision boundary. Two sets of features were extracted from the frames: mel spectrum and MFCCs. Before entering the classifier, all feature vectors were standardized by removing the mean and scaling to unit variance.

The different features and classifiers were evaluated using 5*2-fold nested cross validation (Cawley and Talbot 2010) within the development set. The inner loop performed a two-fold cross validation with grid search for hyperparameter optimization. The mean and variance of the test results from the five folds in the outer loop is reported. The development set was split into folds in a randomized and stratified manner.

As for the evaluation metrics, accuracy, precision, recall, and f1 score are popular metrics in classification tasks. In situations where the classes are highly imbalanced, i.e., some classes have significantly more samples than the others, precision, recall, and f1 score are preferred because accuracy cannot truly reflect the model performance. In this dataset, as shown in table 4.2, the numbers of polyphonic and monophonic frames differ only by a small amount. Therefore, accuracy was used as the evaluation metrics.

	logistic regression	linear SVM	RBF SVM
Mel spectrum	70.5 (0.91)	71.5 (1.09)	90.3 (0.26)
MFCCs	84.9 (0.62)	84.6 (0.58)	**97.0 (0.02)**

Table 4.3: Mean accuracies (%) achieved by the monophony detector using different classifier types and features. These accuracies were obtained from the five-fold cross validation. Variances ($*10^{-3}$) are shown in parentheses. The highest mean accuracy is marked in bold.

Experiment results

A total of six models were trained in the experiment. Each model used one of the three classifier types and one of the two feature sets. The mean and variance of cross validation accuracy achieved by each model is shown in table 4.3. The RBF-kernel SVM trained on the MFCCs achieved the best average accuracy at 97.0%.

Discussion

Comparing the performance of different classifiers, it is not surprising that logistic regression and linear SVM yielded similar performance on either set of features, considering they are both linear classifiers. The SVM with an RBF kernel achieved significantly higher accuracies than the linear classifiers, suggesting that in the feature space established by the mel spectrum and MFCCs, the polyphonic and monophonic frames were better separated by a non-linear hyperplane as the decision boundary.

The mel spectra and MFCCs are both audio representations inspired by human perception. Although the results suggest that both features are useful in distinguishing polyphonic frames from monophonic ones, MFCCs are more suitable for this task as switching from mel spectrum to MFCC features boosted the performance of all three types of classifiers. As mentioned in section 2.2.1, MFCCs are the coefficients that represent the amplitude of the mel-frequency cepstrum, which is obtained by taking a discrete cosine transform (DCT) on the mel spectrum. Similar to how the spectrum captures the periodic structure in the signal waveform, the mel frequency cepstrum reflects information about the periodic structure in the mel spectrum, which corresponds to the harmonic structure of the time-domain signal. Using the first few coefficients as

the audio feature, MFCCs fundamentally preserve the most informative macro spectral structure (i.e., spectral envelop) and discard the noisy micro structure. This may explain why the MFCCs led to better performance than the mel spectrum.

4.2.2 Note-event separator

Experiment setup

As the second preprocessing step, the objective of the note-event separator is to split each monophonic audio segment into individual note events. For each monophonic audio segment, the note-event separator returns a list of time intervals, with each interval marking the onset and offset time of an estimated note event. Here I compare two different strategies for note-event separation, namely the F0 segmentation-based strategy and the onset detection-based strategy.

For the F0 segmentation-based strategy, the pYIN algorithm (Mauch and Dixon 2014) estimated pitches from C2 (approximately 65 Hz) to G6 (approximately 1,568 Hz). This range covers the pitch range of most guitars. It used a frame size of 2,048 and a hop size of 1,024 samples. Two parameters were involved in segmenting the F0 contour estimated by pYIN: the pitch distance threshold (PDT) and the note duration threshold (NDT). The pitch difference threshold determines the cutting points along the F0 curve, and the note duration threshold helps remove spurious notes which often have a very short duration. Considering the physical constraints of the guitar and the smoothing effect of the overlapping windows, the pitch difference threshold should be set well below one semitone, as switching from one note to another on the guitar normally requires a change of fret, and the pitch difference between two adjacent frets is one semitone. Here I also explored different values for the note duration threshold. Among all tested values, the maximum was 75 ms, which corresponds to the duration of a 16th note at 200 bpm. It was assumed that most note events would last longer than that. For the onset detection-based strategy, the spectral flux onset strength envelop was computed on the monophonic audio segment, using a frame size of 1,024 and hop size of 512. In both strategies, the note duration threshold (NDT) parameter was applied for removing spurious notes. The frame size and hop size used for both strategies were chosen empirically using a grid search over various frame size and hop size values. The

complete results obtained from the grid search can be found in table B.1 and table B.2 in the appendix.

In testing the note-event separator, the estimated notes and reference notes (i.e., the ground-truth note events) were matched based on the onset and offset time. The pitch was not taken into account. There are two reasons behind this: First, most pitch-related evaluations assume that a note event has a single and constant pitch value, which is no longer the case considering that the playing techniques modulate the pitch. Second, the job of the note-event separator as a preprocessing step is merely to separate the monophonic audio segment into note events for the playing technique classifiers to operate on. Estimating onset and offset time is sufficient for this purpose.

An estimated note event was deemed correct if its onset fell within a 50-ms tolerance range of the ground-truth onset. It was also required to have its offset within a tolerance range of the ground-truth offset, which was defined as 50 ms or 20% of the reference note's duration, whichever is larger. The evaluation metrics used were precision, recall, and f1 score. Precision is the ratio between the number of correctly estimated note events and the total number of estimated note events. Recall is the ratio between the number of correctly estimated note events and the total number of ground-truth note events. F1 score is the harmonic mean of precision and recall. The evaluation method and metrics applied here are adopted from the note tracking task in Music Information Retrieval Evaluation eXchange (MIREX),[2] as implemented in the *mir_eval* library.[3]

Experiment results

The note-event separator was tested on all monophonic audio segments extracted from the development set. The results obtained using the two strategies are put in comparison in table 4.4. The best f1 score achieved by each strategy is marked in bold.

Discussion

It is observed that, for locating note events in the audio signal, the onset detection-based strategy achieved better performance. Across various parameter settings, F0

2. https://www.music-ir.org/mirex/wiki/2020:Multiple_Fundamental_Frequency_Estimation_%26_Tracking
3. https://craffel.github.io/mir_eval/#module-mir_eval.transcription

strategy	parameters	precision	recall	f1 score
F0	PDT 0.1; NDT 25	40.1	50.5	44.7
	PDT 0.1; NDT 50	51.6	55.7	53.5
	PDT 0.1; NDT 75	53.4	51.0	52.2
	PDT 0.2; NDT 25	69.4	63.2	66.2
	PDT 0.2; NDT 50	78.0	64.9	**70.9**
	PDT 0.2; NDT 75	78.1	60.6	68.3
	PDT 0.4; NDT 25	74.5	60.4	66.7
	PDT 0.4; NDT 50	77.9	60.5	68.1
	PDT 0.4; NDT 75	77.4	56.2	65.2
	PDT 0.8; NDT 25	65.1	41.2	50.5
	PDT 0.8; NDT 50	65.9	40.7	50.3
	PDT 0.8; NDT 75	64.6	37.9	47.8
Onset	NDT 25	91.1	95.4	**93.2**
	NDT 50	92.1	94.2	93.1
	NDT 75	92.6	92.1	92.4

Table 4.4: Note-event separation performance achieved by the two strategies on the development set. PDT: pitch difference threshold. NDT: note duration threshold. The best f1 score achieved by each strategy is marked in bold.

segmentation-based strategy generally had a low recall rate, indicating a high number of false negatives (i.e., missed notes). For the parameters in the F0-based strategy, the most appropriate value for PDT was 0.2 semitones, and the most appropriate value for NDT was 50 ms.

For the onset detection-based strategy, as the note duration threshold got lower, the precision dropped while the recall rose. This behavior is expected because lower duration threshold would lead to more notes being returned. Among the returned notes, most were estimated correctly and can be successfully matched to a reference note. The other notes, however, were resulted from onset detection errors, and these erroneous notes tend to be short in duration. A high duration threshold tends to eliminate these spurious notes, resulting in fewer false-positives and potentially more false-negatives (i.e., high precision, low recall). A low duration threshold tends to preserve the short notes that are truly part of the recording and to overlook spurious notes with a suspiciously short duration, leading to fewer false negatives and potentially more false positives (i.e., low

precision, high recall). An optimal balance was achieved when using a 25-ms threshold, which achieved the highest f1 score on the development set.

Although the onset detection-based strategy achieved satisfactory performance, the result can potentially still be improved by combining onset detection and F0 segmentation. Additionally, Adams, Bartsch, and Wakefield (2006) proposed a series of note segmentation methods, ranging from applying filter functions on the raw pitch contour to employing HMMs to estimate both pitch and temporal boundaries of the notes. These strategies can also potentially be incorporated to improve the performance of the note-event separator. This further exploration on note-event separation is left for future work.

4.2.3 Playing technique classifier

Experiment setup

Two strategies were used for designing the playing technique classifier. In the first strategy, two classifiers, namely the note-event classifier and the transition-event classifier, were trained separately for classifying the note events and transition events. In the second strategy, one unified classifier was trained to distinguish between all classes, without first differentiating between note events and transition events. For training the classifiers, signal segments corresponding to note events and transition events were extracted from the development set.

The note-event classifier, trained on note-event audio segments, distinguishes between bend, vibrato, and normal note event. The transition-event classifier, trained on transition-event audio segments, distinguishes between hammer-on, pull-off, slide, and normal transition event. The normal note event class consists of note events played without any special expression technique. The normal transition event class consists of transition events where no special expression technique is applied. The distribution of the playing technique classes for note events and transition events in the development set is shown in table 4.5 and table 4.6, respectively.

The unified classifier, trained on both note-event audio segments and transition-event audio segments, distinguishes between all playing technique classes (i.e., normal, bend, vibrato, hammer-on, pull-off, and slide). For the unified classifier, the normal class is a

	number of instances	percentage (%)
normal note event	102,070	94.96
bend	3,044	2.83
vibrato	2,370	2.21
total	107,484	100.00

Table 4.5: Playing technique class distribution of note events in the development set.

	number of instances	percentage (%)
normal transition event	86,316	90.82
hammer-on	2,900	3.05
pull-off	4,016	4.23
slide	1,808	1.90
total	95,040	100.00

Table 4.6: Playing technique class distribution of transition events in the development set.

combination of normal note events and normal transition events.

The classifiers used in the experiments are multiclass SVMs with RBF kernels. To alleviate the effect of class imbalance, the SVMs were trained in the "balanced" mode, where the class weights were set inversely proportional to the class frequencies observed in the training data. For feature extraction, a frame size of 1,024 and a hop size of 512 were used for the Fourier transform, at a sampling rate of 44,100 Hz. These values were chosen to stay consistent with the parameter settings that produced the best performance in the previous steps: monophony detector (4.2.1) and note-event separator (4.2.2). The note events and transition events were represented by three sets of features: Timber (T), Pitch (P), and MFCCs (M), as presented in section 3.2.3. Before entering the classifier, all feature vectors were standardized by removing the mean and scaling to unit variance.

The models were evaluated using 5*2-fold nested cross validation within the development set. The development set was split into folds in a randomized and stratified manner. The inner loop used a two-fold cross validation with grid search to optimize the hyperparameters (i.e., C and gamma for the SVM). The average test result and the variance from the five folds in the outer loop are reported. As the number of samples in each class was highly imbalanced, per-class f1 score was used as the

	normal note event	bend	vibrato	macro average
T	98.5 (0.02)	87.4 (8.58)	82.0 (23.72)	89.3
P	98.5 (0.02)	89.6 (7.00)	84.0 (7.52)	90.7
M	99.5 (0.06)	87.8 (12.68)	82.7 (28.68)	90.0
TP	99.5 (0.02)	89.8 (6.44)	**84.8 (24.86)**	91.4
TM	**99.6 (0.02)**	89.9 (4.98)	84.7 (14.32)	91.4
PM	**99.6 (0.02)**	89.8 (6.16)	84.5 (15.02)	91.3
TPM	**99.6 (0.02)**	**91.4 (4.18)**	**84.8 (6.30)**	**91.9**

Table 4.7: Per-class f1 scores (%) of the note-event classifier trained on various feature sets. These scores were obtained from the five-fold cross validation. For each entry, the average score over the five folds is shown in percentages, with the variance ($*10^{-5}$) in parentheses. The highest per-class f1 score for each technique is marked in bold.

evaluation metrics.

Experiment results

To verify the effectiveness of each feature set and their combinations, a model was trained and tested on each of the seven feature combinations (T, P, M, TP, TM, PM, TPM). The average per-class f1 scores and variances obtained from nested cross validation are shown in table 4.7 and table 4.8.

The unified classifier was trained on the combination of all features (TPM) to distinguish between all playing technique classes. Table 4.9 compares the per-class f1 scores on the five playing technique classes (excluding normal) achieved by the unified classifier with the combined results of the two separate classifiers.

Discussion

It is observed in table 4.7 that the pitch features seem to be the best-performing individual feature set for classifying note events. Moreover, combining multiple feature sets generally led to better results than using a single feature set. For example, the models trained on the combination of the pitch and MFCC features (PM) could better recognize bend and vibrato than the models trained on the pitch (P) or MFCC features (M) alone. The best results in note-event classification were achieved by combining all

	normal transition event	hammer-on	pull-off	slide	macro average
T	99.7 (0.02)	92.7 (0.90)	98.2 (0.50)	91.4 (11.6)	95.5
P	99.7 (0.02)	95.9 (0.58)	**98.3 (1.74)**	91.4 (5.94)	96.3
M	**99.8 (0.02)**	96.9 (2.18)	98.2 (1.10)	91.0 (8.36)	96.5
TP	**99.8 (0.02)**	97.6 (0.86)	**98.3 (0.98)**	91.2 (18.5)	96.7
TM	**99.8 (0.02)**	97.8 (0.70)	**98.3 (0.78)**	**91.6 (4.26)**	96.9
PM	**99.8 (0.02)**	97.8 (0.42)	98.2 (0.94)	91.4 (6.02)	96.8
TPM	**99.8 (0.02)**	**98.2 (1.58)**	**98.3 (1.26)**	**91.6 (3.32)**	**97.0**

Table 4.8: Per-class f1 scores (%) of the transition-event classifier trained on various feature sets. These scores were obtained from the five-fold cross validation. For each entry, the average score over the five folds are shown in percentages, with the variance ($*10^{-5}$) in parentheses. The highest per-class f1 score for each technique is marked in bold.

	unified classifier	separate classifiers
bend	89.7 (5.50)	91.4 (4.18)
vibrato	82.5 (1.58)	84.8 (6.30)
hammer-on	97.7 (1.38)	98.2 (1.58)
pull-off	98.1 (1.98)	98.3 (1.26)
slide	88.9 (7.78)	91.6 (3.32)
macro average	91.4	92.9

Table 4.9: Per-class f1 scores (%) achieved by the unified classifier and those achieved by the two separate classifiers. These scores were obtained from the five-fold cross validation. For each entry, the average score over the five folds are shown in percentages, with the variance ($*10^{-5}$) in parentheses.

three feature sets (TPM), which achieved performance superior to or comparable to its counterparts in recognizing every class.

Similar observations can be made in table 4.8, where better results were generally obtained by combining multiple feature sets. Exceptionally, combining the timbre and pitch features (TP) resulted in a drop in the score than using the timber features (T) or pitch features (P) alone for recognizing slide. The MFCC features (M) proved to be the best-performing individual feature set for classifying transition events, while it was still the combination of all feature sets that achieved the best overall performance in transition-event classification.

Results presented in table 4.7 and table 4.8 suggest that, given pre-segmented note events and transition events, the playing technique classifiers can effectively recognize all five playing techniques covered in this thesis. Furthermore, the classifiers seem to acquire complementary knowledge from the three sets of audio features, as combining all feature sets helped improve the overall performance.

The note-event classifier, transition-event classifier, and unified classifier all achieved satisfactory results. Trained on the same set of features, the unified classifier needs to distinguish between more classes than either one of the separate classifiers. However, as shown in table 4.9, the per-class f1 scores achieved by the unified classifier are only slightly lower than those achieved by the separate classifiers. Therefore, for the simplicity of using one classifier for all classes, the unified classifier was used as the playing technique classifier in the system. More importantly, using one unified classifier to distinguish between all playing technique classes is consistent with previous works on this task (Kehling et al. 2014; Su, Yu, and Yang 2014; Chen, Su, and Yang 2015; Su et al. 2019), making it more fair to compare the presented system to the ones proposed by others.

4.3 End-to-end Integration Test

As presented in the previous section, each individual component of the system was tuned and evaluated on the development set. In order to verify how they would perform when working together, an end-to-end integration test was conducted. In this integration test, the components were connected into one workflow, and a designated set of test data was

	number of instances	percentage (%)
normal	47,290	93.93
bend	829	1.65
vibrato	668	1.32
hammer-on	664	1.32
pull-off	719	1.43
slide	178	0.35
total	50,348	100.00

Table 4.10: Playing technique class distribution of the test set. The count for each playing technique is the number of note events or transition events with the corresponding label. The normal class is the combination of normal note events and normal transition events.

used to test the performance of the entire system. The integration test results should reflect the true performance of the playing technique classification system.

4.3.1 Experiment setup

As presented in section 4.1, the ground-truth dataset collected for this thesis consists of 147 multitrack GP files. Among these files, 100 files (i.e., the development set) were used for parameter optimization and the other 47 files (i.e., the test set) were reserved for the end-to-end integration test. Because the test set was never exposed to the parameter optimization process, the results obtained in the end-to-end integration test should reflect how well the system would generalize to previously unseen data. For an illustration of the data split, please refer to figure 4.2. For an overview of the proposed system, please refer to figure 3.4.

The playing technique class distribution in the test set is shown in table 4.10. Since the unified classifier was chosen as the playing technique classifier, I no longer differentiate between normal note events and normal transition events. They are combined into one normal class.

With each component using the best parameters found in the individual experiments, the playing technique classification system takes as input the audio signals synthesized from the single-track GP files in the test set. The audio signals are in turn processed by the monophony detector, the note-event separator, and the playing technique classifier.

	precision	recall	f1 score
bend	91.2	87.2	89.2
vibrato	88.2	76.1	81.7
hammer-on	96.4	76.0	85.0
pull-off	92.7	77.4	84.3
slide	88.3	60.4	71.7

Table 4.11: Per-class metrics (%) obtained from the end-to-end integration test.

The final output is a list of timestamps for estimated note events and transition events. Each entry is accompanied by an estimated class label representing its associated playing technique.

For bend and vibrato, an estimated note event is considered correct if it temporally overlaps with a ground-truth note event of the same class. Specifically, they are considered to overlap if one of the following two conditions is satisfied: 1. The onset of the estimated note event falls within a 50-ms tolerance window of the onset of the ground-truth note event. 2. The offset of the estimated note event falls within a tolerance window of 50 ms or 20% of the reference note's duration, whichever is larger, of the offset of the ground-truth note event.

For hammer-on, pull-off, and slide, an estimated transition event is considered correct if its center falls within a 50-ms tolerance window of the center of a ground-truth transition event of the same class. In both cases, every ground-truth instance is matched against at most one estimated instance. Per-class precision, recall, and f1 score are calculated as evaluation metrics. For instance, the precision for the class bend is the ratio between the number of correctly recognized bend instances and the total number of estimated bend instances. The recall is the ratio between the number of correctly recognized bend instances and the total number of ground-truth bend instances. The f1 score is the harmonic mean of the precision and recall.

4.3.2 Results and discussion

The per-class results of the end-to-end integration test are presented in table 4.11.

As expected, the per-class f1 scores dropped slightly when switching from individual component tests to the end-to-end integration test. This is explained by the fact that

during individual component tests, the input to each component was taken directly from the ground-truth instead of the upstream component. In other words, individual components did not need to handle erroneous data produced by their upstream components. Meanwhile, in the end-to-end test, errors made by the monophony detector would propagate to the downstream note-event separator, whose errors would in turn affect the playing technique classifier. Therefore, it should be noted that the metrics obtained from the individual component tests are overly optimistic. The result of the end-to-end test is a more realistic estimate of the system performance.

In all classes, the recall is considerably lower than the precision, indicating that the classification system is more prone to false negatives than false positives. In other words, many playing technique instances were missed, but for those recognized by the system, the recognition was quite accurate. A potential explanation for this result is that the playing technique classifier tends to predict the majority class label (i.e., normal) when the input feature vector does not fit into the feature space occupied by the other playing technique classes. This is likely to happen when the monophony detector and note-event separator make mistakes, which would result in feature vectors that look nothing like the "clean-cut" note events and transition events that the classifier has seen during the training stage. Nonetheless, the results of the end-to-end integration test suggest that the proposed system can effectively recognize the five playing techniques, with bend being the easiest to recognize and slide being the most difficult.

4.3.3 Comparison to previous works

Comparing the proposed system to those developed in previous works, the per-class f1 scores achieved by different systems are shown in table 4.12. For the system presented in this thesis, the per-class f1 scores obtained from the end-to-end integration test are displayed. For the other systems, the displayed scores are those reported in their respective papers. The results suggest that the system presented in this thesis can better recognize hammer-on, pull-off, and slide than previous systems, whereas it did not achieve a higher performance in recognizing bend and vibrato.

However, it is important to note that the scores of different systems were achieved under different conditions. The systems proposed in Su, Yu, and Yang (2014) operates on

pre-segmented single-note audio recordings, which gives it the advantage of not having errors in the note-event separation stage. In contrast, the system presented in this thesis, as well as the systems proposed in Kehling et al. (2014), Chen, Su, and Yang (2015), and Su et al. (2019), operates on entire tracks of guitar solo recordings. Potential errors from note-event separation would have a negative effect on the end result.

Another difference to consider lies in the data. The five systems compared all used their own datasets for their respective experiments. The datasets vary in many aspects such as size, genre, guitar tone, and recording conditions. Especially, for the system presented in this thesis, the audio signals used for training and testing were synthesized using the Guitar Pro software. Although the synthesized guitar sounds somewhat realistic, it lacks natural asynchronies and variations in tempo and dynamics that are often present in recordings taken from a real guitar. This potentially makes it easier to achieve a good performance on synthesized guitar recordings than on real guitar recordings.

Despite the differences between the studies listed, the f1 score of slide is lower than the other classes in most of the studies (i.e., all except Su, Yu, and Yang (2014)). One potential explanation is that slide can either increase or decrease the pitch, while its counterparts hammer-on and pull-off can only change the pitch monotonically. This makes it more difficult to recognize slide from other classes. Besides, slide has the least amount of training samples among all classes in the datasets used in Kehling et al. (2014), Su et al. (2019), and this thesis. The lack of training samples tends to make it more difficult for the classifier to learn the characteristics of the target class.

	bend	vibrato	hammer-on	pull-off	slide
Kehling et al. (2014)	71.3	66.7	82.4	-	50.9
Su, Yu, and Yang (2014)	**89.4**	**86.9**	55.2	52.0	65.0
Chen, Su, and Yang (2015)	87.7	84.0	66.3	74.4	57.7
Su et al. (2019)	76.7	-	68.1	73.0	38.8
proposed system	89.2	81.7	**85.0**	**84.3**	**71.7**

Table 4.12: Per-class f1 scores (%) achieved by previous systems and the system presented in this thesis. The highest score for each technique is marked in bold.

Chapter 5

Conclusion

This chapter summarizes the work presented in this thesis, which focused on the automatic classification of five guitar playing techniques. Specifically, chapter 1 introduced the guitar playing techniques and gave an overview of the presented system. Chapter 2 presented relevant works on instrumental gesture acquisition and the background about audio features and machine learning classifiers. A detailed description of the ground-truth generation workflow and the playing technique classification system was given in chapter 3. Individual component tests and an end-to-end integration test were performed to evaluate the system. The experiment procedures and results were discussed in chapter 4.

In this chapter, I first summarize the contributions of this thesis and then discuss directions for future work.

5.1 Summary of Contributions

In this thesis, I have presented a playing technique classification system that operates in three steps. For a given guitar audio signal, the monophony detector extracts the monophonic segments, where most playing technique instances occur. This step operates by performing a binary classification, distinguishing between monophonic and polyphonic frames. Then, the note-event separator estimates the onset and offset timestamps for every note event present in a given monophonic audio segment. Two strategies have been tested for this step: the F0 segmentation-based strategy and onset

detection-based strategy. The regions of interest, found by the note-event separator, are represented by a set of audio features and finally classified by the playing technique classifier. The playing technique classifier classifies each region of interest as one of the six playing technique classes: normal, bend, vibrato, hammer-on, pull-off, and slide.

For constructing the dataset used in this thesis, a ground-truth generation workflow has been developed. The highest-rated and most-visited Guitar Pro (GP) files were collected from the Ultimate Guitar website. These raw, multitrack GP files were first preprocessed into single-track GP files. The Guitar Pro software was used to generate audio signals from the single-track GP files. The PyGuitarPro library was used to extract information from the single-track GP files and construct annotations. The guitar playing technique classification task is still in its early stage, for which the amount of annotated data remains limited. This thesis is the first to leverage GP files for generating a large amount of public training data for the playing technique classification task.

To test the presented playing technique classification system, experiments were conducted for each individual component as well as the entire system.

For the monophony detector, it was found that the SVM classifier trained on MFCCs gave the best performance, with an f1 score of 97.0% in distinguishing between monophonic and polyphonic audio frames.

For the note-event separator, the onset detection-based strategy proved to be more effective than the F0 segmentation-based strategy, with an f1 score of 93.2% over 70.9% in tracking note events.

For the playing technique classifier, it was found that combining multiple feature sets (i.e., pitch features, timbral features, and MFCCs) generally led to better performance than using a single feature set. The classifier seemed to acquire complementary knowledge from the three sets of audio features. Trained on the combination of all three feature sets, the SVM classifier achieved per-class f1 scores ranging from 82.5% to 97.7% in the six-way classification, suggesting that the classifier can effectively distinguish between the six playing technique classes.

To verify the effectiveness of the entire system, the three components were put together and an end-to-end integration test was performed. The test samples were in turn processed by the three components and the result was compared with the ground-truth annotations. The per-class f1 scores obtained in the end-to-end integration

test ranged from 71.7% to 89.2%, which proved that the presented three-step system can effectively locate and identify the five playing techniques in guitar audio signals synthesized from GP files.

5.2 Future Work

On the topic of playing technique classification, this thesis limited its scope to guitar recordings synthesized from GP files. The system, trained solely on synthesized recordings, may end up relying on Guitar Pro sound artefacts instead of true characteristics of the playing techniques, which are applicable to a wider range of guitar sound. To overcome this limitation, a promising direction for further exploration is to train on both real recordings taken from live performances and synthesized recordings and then test the model on real recordings. As real recordings are more expensive to obtain and usually require manual annotation, the amount of ground-truth data is often limited. Augmenting training data with synthesized recordings might facilitate the training of playing technique classifiers and lead to better performances than training on real recordings alone.

Additionally, there are more guitar playing techniques beyond the five common ones studied in this thesis. The GP file format supports a wide variety of other playing techniques such as palm muting, artificial harmonics, and interactions with the tremolo bar. These techniques, although less common, also contribute to the expressivity of the guitar performance. Using GP files as a source of ground-truth data, a more comprehensive playing technique classification system can potentially be developed to recognize these playing techniques as well.

One significant contribution of this thesis is the automated workflow for ground-truth generation. With GP files, it is possible to generate synthesized guitar signals and corresponding annotations with minimal manual work. Considering the abundance of publicly available GP files (Sarmento et al. 2021), it is a promising source of data for audio-based MIR tasks. It is hoped that the wealth of publicly available GP files and the ground-truth generation workflow developed in this thesis will motivate and facilitate future research that leverages this particular symbolic music format.

Bibliography

Abeßer, Jakob, Christian Dittmar, and Gerald Schuller. 2011. "Automatic Recognition and Parametrization of Frequency Modulation Techniques in Bass Guitar Recordings." In *Proceedings of the AES International Conference on Semantic Audio*, 1–8. Ilmenau, Germany.

Abeßer, Jakob, Hanna M. Lukashevich, and Gerald Schuller. 2010. "Feature-Based Extraction of Plucking and Expression Styles of the Electric Bass Guitar." In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2290–2293. Dallas, TX, USA.

Adams, Norman H., Mark A. Bartsch, and Gregory H. Wakefield. 2006. "Note Segmentation and Quantization for Music Information Retrieval." *IEEE Transactions on Audio, Speech, and Language Processing* 14 (1): 131–141.

Alpaydin, Ethem. 2014. *Introduction to Machine Learning*. Cambridge, MA, USA: The MIT Press.

Angulo, Iñigo, Sergio Giraldo, and Rafael Ramirez. 2016. "Hexaphonic Guitar Transcription and Visualization." In *Proceedings of the International Conference on Technologies for Music Notation and Representation*, 187–192. Cambridge, UK.

Barbancho, Ana M., Anssi Klapuri, Lorenzo J. Tardon, and Isabel Barbancho. 2012. "Automatic Transcription of Guitar Chords and Fingering from Audio." *IEEE Transactions on Audio, Speech, and Language Processing* 20 (3): 915–921.

Barbancho, Isabel, Cristina de la Bandera, Ana M. Barbancho, and Lorenzo J. Tardon. 2009. "Transcription and Expressiveness Detection System for Violin Music." In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 189–192. Taipei, Taiwan.

Barbancho, Isabel, Lorenzo J. Tardon, Simone Sammartino, and Ana M. Barbancho. 2012. "Inharmonicity-Based Method for the Automatic Generation of Guitar Tablature." *IEEE Transactions on Audio, Speech, and Language Processing* 20 (6): 1857–1868.

Barrington, Luke, Mehrdad Yazdani, Douglas Turnbull, and Gert R. G. Lanckriet. 2008. "Combining Feature Kernels for Semantic Music Retrieval." In *Proceedings of the International Conference on Music Information Retrieval*, 614–619. Philadelphia, PA, USA.

Benetos, Emmanouil, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. 2019. "Automatic Music Transcription: An Overview." *IEEE Signal Processing Magazine* 36 (1): 20–30.

Benetos, Emmanouil, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. 2012. "Automatic Music Transcription: Breaking the Glass Ceiling." In *Proceedings of the International Society for Music Information Retrieval Conference*, 379–384. Porto, Portugal.

———. 2013. "Automatic Music Transcription: Challenges and Future Directions." *Journal of Intelligent Information Systems* 41 (3): 407–434.

Brossier, Paul, Juan Pablo Bello, and Mark D. Plumbley. 2004. "Real-time Temporal Segmentation of Note Objects in Music Signals." In *Proceedings of the International Computer Music Conference*, 244–247. Miami, FL, USA.

Burlet, Gregory, and Ichiro Fujinaga. 2013. "Robotaba Guitar Tablature Transcription Framework." In *Proceedings of the International Society for Music Information Retrieval Conference*, 517–522. Curitiba, Brazil.

Burns, Anne-Marie, and Marcelo M. Wanderley. 2006. "Visual Methods for the Retrieval of Guitarist Fingering." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 196–199. Paris, France.

Cadoz, Claude. 1988. "Instrumental Gesture and Musical Composition." In *Proceedings of the International Computer Music Conference*, 1–12. Cologne, Germany.

Cawley, Gavin, and Nicola Talbot. 2010. "On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation." *Journal of Machine Learning Research* 11 (70): 2079–2107.

Chen, Yuan-Ping, Li Su, and Yi-Hsuan Yang. 2015. "Electric Guitar Playing Technique Detection in Real-World Recording Based on F0 Sequence Pattern Recognition." In *Proceedings of the International Society for Music Information Retrieval Conference*, 708–714. Málaga, Spain.

Dixon, Simon. 2006. "Onset Detection Revisited." In *Proceedings of the International Conference on Digital Audio Effects*, 133–137. Montréal, QC, Canada.

Dubnov, Shlomo. 2004. "Generalization of Spectral Flatness Measure for Non-Gaussian Linear Processes." *IEEE Signal Processing Letters* 11 (8): 698–701.

Goldstein, Shir, and Yael Moses. 2018. "Guitar Music Transcription from Silent Video." In *Proceedings of the British Machine Vision Conference*, 309–321. Newcastle, UK.

Gouyon, Fabien, Francois Pachet, and Olivier Delerue. 2000. "On the Use of Zero-Crossing Rate for an Application of Classification of Percussive Sounds." In *Proceedings of the COST G-6 Conference on Digital Audio Effects*, 72–77. Verona, Italy.

Hart, Peter, Nils Nilsson, and Bertram Raphael. 1968. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." *IEEE Transactions on Systems Science and Cybernetics* 4 (2): 100–107.

Hrybyk, Alex, and Youngmoo E. Kim. 2010. "Combined Audio and Video Analysis for Guitar Chord Identification." In *Proceedings of the International Society for Music Information Retrieval Conference*, 159–164. Utrecht, The Netherlands.

Kehling, Christian, Jakob Abeßer, Christian Dittmar, and Gerald Schuller. 2014. "Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score- and Instrument-Related Parameters." In *Proceedings of the International Conference on Digital Audio Effects*, 219–226. Erlangen, Germany.

Kerdvibulvech, Chutisant, and Hideo Saito. 2007a. "Vision-Based Detection of Guitar Players' Fingertips Without Markers." In *Proceedings of the International Conference on Computer Graphics, Imaging and Visualisation*, 419–428. Bangkok, Thailand.

———. 2007b. "Vision-Based Guitarist Fingering Tracking Using a Bayesian Classifier and Particle Filters." In *Advances in Image and Video Technology*, 625–638.

———. 2008. "Markerless Guitarist Fingertip Detection Using a Bayesian Classifier and a Template Matching for Supporting Guitarists." In *Proceedings of the ACM/IEEE Virtual Reality International Conference*, 201–208. Laval, France.

Klapuri, Anssi. 2006. "Introduction to Music Transcription." In *Signal Processing Methods for Music Transcription*, edited by Anssi Klapuri and Manuel Davy, 3–20. Boston, MA, USA: Springer.

Knees, Peter, and Markus Schedl. 2016. "Basic Methods of Audio Signal Processing." In *Music Similarity and Retrieval*, edited by Peter Knees and Markus Schedl, 33–50. Berlin, Germany: Springer.

Kong, Chenchen, and Yibiao Yu. 2017. "Musical Note Segmentation Based on the Double-threshold Endpoint Detection and Fundamental Frequency Curve Fluctuation Measure." In *Proceedings of the International Conference on Systems and Informatics*, 1109–1113. Hangzhou, China.

Liang, Beici, György Fazekas, and Mark B. Sandler. 2018a. "Measurement, Recognition, and Visualization of Piano Pedaling Gestures and Techniques." *Journal of the Audio Engineering Society* 66 (6): 448–456.

———. 2018b. "Piano Legato-Pedal Onset Detection Based on a Sympathetic Resonance Measure." In *Proceedings of the European Signal Processing Conference*, 2484–2488. Roma, Italy.

———. 2019. "Piano Sustain-pedal Detection Using Convolutional Neural Networks." In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 241–245. Brighton, UK.

Logan, Beth. 2000. "Mel Frequency Cepstral Coefficients for Music Modeling." In *Proceedings of the International Symposium on Music Information Retrieval,* 1–13. Plymouth, MA, USA.

Mauch, Matthias, and Simon Dixon. 2014. "PYIN: A Fundamental Frequency Estimator Using Probabilistic Threshold Distributions." In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing,* 659–663. Florence, Italy.

McFee, Brian, Alexandros Metsai, Matt McVicar, Stefan Balke, Carl Thomé, Colin Raffel, Frank Zalkow, et al. 2022. *librosa/librosa: 0.9.1.* https://doi.org/10.5281/zenodo.6097378.

McKinney, Martin F., and Jeroen Breebaart. 2003. "Features for Audio and Music Classification." In *Proceedings of the International Conference on Music Information Retrieval,* 27–34. Baltimore, MD, USA.

Mitrović, Dalibor, Matthias Zeppelzauer, and Christian Breiteneder. 2010. "Features for Content-Based Audio Retrieval." In *Advances in Computers: Improving the Web,* edited by Marvin V. Zelkowitz, 78:71–150. San Diego, CA, USA: Elsevier.

Moorer, James A. 1977. "On the Transcription of Musical Sound by Computer." *Computer Music Journal* 1 (4): 32–38.

O'Grady, Paul D., and Scott T. Rickard. 2009. "Automatic Hexaphonic Guitar Transcription Using Non-Negative Constraints." In *Proceedings of the IET Irish Signals and Systems Conference,* 1–6. Dublin, Ireland.

Orio, Nicola. 1999. "The Timbre Space of the Classical Guitar and its Relationship with the Plucking Techniques." In *Proceedings of the International Computer Music Conference,* 391–394. Beijing, China.

Özaslan, Tan Hakan, and Josep Lluís Arcos. 2010. "Legato and Glissando Identification in Classical Guitar." In *Proceedings of the Sound and Music Computing Conference,* 457–463. Barcelona, Spain.

Özaslan, Tan Hakan, Enric Guaus, Eric Palacios, and Josep Lluís Arcos. 2010. "Attack Based Articulation Analysis of Nylon String Guitar." In *Proceedings of the International Symposium on Computer Music Modeling and Retrieval*, 219–241. Málaga, Spain.

Paleari, Marco, Benoit Huet, Antony Schutz, and Dirk T. M. Slock. 2008. "A Multimodal Approach to Music Transcription." In *Proceedings of the International Conference on Image Processing*, 93–96. San Diego, CA, USA.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12:2825–2830.

Peeters, Geoffroy. 2004. "A Large Set of Audio Features for Sound Description (Similarity and Classification) in the CUIDADO Project." *CUIDADO IST Project Report* 54:1–25.

Perez-Carrillo, Alfonso. 2016. "Statistical Models for the Indirect Acquisition of Violin Bowing Controls from Audio Analysis." In *Proceedings of Meetings on Acoustics*, 1–9. Honolulu, HI, USA.

Perez-Carrillo, Alfonso, and Marcelo M. Wanderley. 2012. "Learning and Extraction of Violin Instrumental Controls from Audio Signal." In *Proceedings of the International Workshop on Music Information Retrieval with User-Centered and Multimodal Strategies*, 25–30. Nara, Japan.

———. 2015. "Indirect Acquisition of Violin Instrumental Controls from Audio Signal with Hidden Markov Models." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23 (5): 932–940.

Poliner, Graham E., and Daniel Ellis. 2006. "A Discriminative Model for Polyphonic Piano Transcription." *EURASIP Journal on Advances in Signal Processing* 2007 (1): 1–9.

———. 2007. "Improving Generalization for Classification-Based Polyphonic Piano Transcription." In *Proceedings of the IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 86–89. New Paltz, NY, USA.

Pradhan, Ashis. 2012. "Support Vector Machine: A Survey." *International Journal of Emerging Technology and Advanced Engineering* 2 (8): 82–85.

Rabiner, Lawrence R., and B. H. (Biing-Hwang) Juang. 1993. "Signal Processing and Analysis Methods for Speech Recognition." In *Fundamentals of Speech Recognition*, edited by Lawrence R. Rabiner and B. H. (Biing-Hwang) Juang, 69–73. Englewood Cliffs, NJ, USA: PTR Prentice Hall.

Reboursière, Loïc, Otso Lähdeoja, Thomas Drugman, Stéphane Dupont, Cécile Picard-Limpens, and Nicolas Riche. 2012. "Left and Right-Hand Guitar Playing Techniques Detection." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 205–208. Ann Arbor, MI, USA.

Saito, Shoichiro, Hirokazu Kameoka, Keigo Takahashi, Takuya Nishimoto, and Shigeki Sagayama. 2008. "Specmurt Analysis of Polyphonic Music Signals." *IEEE Transactions on Speech and Audio Processing* 16 (3): 639–650.

Sarmento, Pedro, Adarsh Kumar, C. J. Carr, Zack Zukowski, Mathieu Barthet, and Yi-Hsuan Yang. 2021. "DadaGP: A Dataset of Tokenized GuitarPro Songs for Sequence Models." In *Proceedings of the International Society for Music Information Retrieval Conference*, 610–617. Online.

Scherrer, Bertrand. 2013. "Physically-Informed Indirect Acquisition of Instrumental Gestures on the Classical Guitar." Doctoral dissertation, McGill University.

Su, Li, Hsin-Ming Lin, and Yi-Hsuan Yang. 2014. "Sparse Modeling of Magnitude and Phase-Derived Spectra for Playing Technique Classification." *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 22 (12): 2122–2132.

Su, Li, Li-Fan Yu, and Yi-Hsuan Yang. 2014. "Sparse Cepstral, Phase Codes for Guitar Playing Technique Classification." In *Proceedings of the International Society for Music Information Retrieval Conference*, 9–14. Taipei, Taiwan.

Su, Ting-Wei, Yuan-Ping Chen, Li Su, and Yi-Hsuan Yang. 2019. "TENT: Technique-Embedded Note Tracking for Real-World Guitar Solo Recordings." *Transactions of the International Society for Music Information Retrieval* 2 (1): 15–28.

Tindale, Adam R., Ajay Kapur, W. Andrew Schloss, and George Tzanetakis. 2005. "Indirect Acquisition of Percussion Gestures Using Timbre Recognition." In *Proceedings of the Conference on Interdisciplinary Musicology*, 9–16. Montréal, QC, Canada.

Tindale, Adam R., Ajay Kapur, George Tzanetakis, and Ichiro Fujinaga. 2004. "Retrieval of Percussion Gestures Using Timbre Classification Techniques." In *Proceedings of the International Conference on Music Information Retrieval*, 541–545. Barcelona, Spain.

Traube, Caroline, and Philippe Depalle. 2003. "Deriving the Plucking Point Location Along a Guitar String from a Least-Square Estimation of a Comb Filter Delay." In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, 2001–2004. Montréal, QC, Canada.

Traube, Caroline, Philippe Depalle, and Marcelo M. Wanderley. 2003. "Indirect Acquisition of Instrumental Gesture Based on Signal, Physical and Perceptual Information." In *Proceedings of the International Conference on New Interfaces for Musical Expression*, 42–48. Montréal, QC, Canada.

Traube, Caroline, and Julius O. Smith. 2000. "Estimating the Plucking Point on a Guitar String." In *Proceedings of the COST G-6 Conference on Digital Audio Effects*, 153–158. Verona, Italy.

———. 2001. "Extracting the Fingering and the Plucking Points on a Guitar String from a Recording." In *Proceedings of the IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, 7–10. New Paltz, NY, USA.

Wiggins, Andrew, and Youngmoo Kim. 2019. "Guitar Tablature Estimation with a Convolutional Neural Network." In *Proceedings of the International Society for Music Information Retrieval Conference*, 284–291. Delft, The Netherlands.

Wu, Chih-Wei, and Alexander Lerch. 2016. "On Drum Playing Technique Detection in Polyphonic Mixtures." In *Proceedings of the International Society for Music Information Retrieval Conference*, 218–224. New York City, NY, USA.

Xi, Qingyang, Rachel M. Bittner, Johan Pauwels, Xuzhou Ye, and Juan Pablo Bello. 2018. "GuitarSet: A Dataset for Guitar Transcription." In *Proceedings of the International Society for Music Information Retrieval Conference*, 453–460. Paris, France.

Yazawa, Kazuki, Katsutoshi Itoyama, and Hiroshi G. Okuno. 2014. "Automatic Transcription of Guitar Tablature from Audio Signals in Accordance with Player's Proficiency." In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 3122–3126. Florence, Italy.

Yazawa, Kazuki, Daichi Sakaue, Kohei Nagira, Katsutoshi Itoyama, and Hiroshi G. Okuno. 2013. "Audio-based Guitar Tablature Transcription Using Multipitch Analysis and Playability Constraints." In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 196–200. Vancouver, BC, Canada.

Yoshii, Kazuyoshi, and Masataka Goto. 2012. "A Nonparametric Bayesian Multipitch Analyzer Based on Infinite Latent Harmonic Allocation." *IEEE Transactions on Audio, Speech, and Language Processing* 20 (3): 717–730.

Zhou, Ruohua, and Joshua D. Reiss. 2008. "A Real-Time Polyphonic Music Transcription System." In *Proceedings of the Music Information Retrieval Evaluation eXchange*, 4–7. Philadelphia, PA, USA.

Appendix A

Lists of Guitar Pro Files

A total of 147 Guitar Pro files were collected for generating the ground-truth dataset used in this thesis. Table A.1 provides the song title, artist name, and the original filename of the 100 GP files in the development set. Table A.2 provides the same information for the 47 GP files in the test set. These files were all collected from the Ultimate Guitar website.

Table A.1: The 100 raw GP files in the development set.

ID	Title	Artist	Filename
1	Back In Black	ACDC	ACDC - Back In Black (ver 4 by GuitarManiac09).gp5
2	Back In Black	ACDC	ACDC - Back In Black.gp5
3	Highway To Hell	ACDC	ACDC - Highway To Hell (ver 3).gp5
4	Rolling In The Deep	Adele	Adele - Rolling In The Deep.gp5
5	Dream On	Aerosmith	Aerosmith - Dream On (ver 3).gp5
6	Afterlife	Avenged Sevenfold	Avenged Sevenfold - Afterlife.gp5
7	Beast And The Harlot	Avenged Sevenfold	Avenged Sevenfold - Beast And The Harlot.gp5
8	Unholy Confessions	Avenged Sevenfold	Avenged Sevenfold - Unholy Confessions.gp4
9	Iron Man	Black Sabbath	Black Sabbath - Iron Man (ver 3 by jessew).gp4
10	Paranoid	Black Sabbath	Black Sabbath - Paranoid.gp5
11	Tears Dont Fall	Bullet For My Valentine	Bullet For My Valentine - Tears Dont Fall (ver 2 by dannyloughran).gp4
12	Waking The Demon	Bullet For My Valentine	Bullet For My Valentine - Waking The Demon.gp5
13	Johnny B Goode	Chuck Berry	Chuck Berry - Johnny B Goode (ver 6).gp5
14	Smoke On The Water	Deep Purple	Deep Purple - Smoke On The Water.gp4
15	Sultans Of Swing	Dire Straits	Dire Straits - Sultans Of Swing.gp5
16	Through The Fire And Flames	DragonForce	DragonForce - Through The Fire And Flames.gp5
17	Hotel California	Eagles	Eagles - Hotel California.gp3
18	Layla	Eric Clapton	Eric Clapton - Layla (ver 2).gp4
19	Tears In Heaven	Eric Clapton	Eric Clapton - Tears In Heaven.gp3
20	Cliffs Of Dover	Eric Johnson	Eric Johnson - Cliffs Of Dover (ver 2 by lem_ian).gp5
21	More Than Words	Extreme	Extreme - More Than Words (ver 3).gp5
22	Everlong	Foo Fighters	Foo Fighters - Everlong.gp4
23	The Pretender	Foo Fighters	Foo Fighters - The Pretender.gp4
24	Still Got The Blues	Gary Moore	Gary Moore - Still Got The Blues.gp3
25	21 Guns	Green Day	Green Day - 21 Guns (ver 4 by tangoso).gp5
26	American Idiot	Green Day	Green Day - American Idiot (ver 2).gp5

Continued on next page

Table A.1 – *Continued from previous page*

ID	Title	Artist	Filename
27	Wake Me Up When September Ends	Green Day	Green Day - Wake Me Up When September Ends (ver 3).gp5
28	Dont Cry	Guns N' Roses	Guns N' Roses - Dont Cry.gp3
29	Knockin On Heavens Door	Guns N' Roses	Guns N' Roses - Knockin On Heavens Door (ver 8).gp4
30	November Rain	Guns N' Roses	Guns N' Roses - November Rain (ver 2).gp5
31	Sweet Child O Mine	Guns N' Roses	Guns N' Roses - Sweet Child O Mine (ver 2).gp4
32	Welcome To The Jungle	Guns N' Roses	Guns N' Roses - Welcome To The Jungle (ver 3).gp4
33	Fear Of The Dark	Iron Maiden	Iron Maiden - Fear Of The Dark.gp4
34	The Trooper	Iron Maiden	Iron Maiden - The Trooper (ver 5).gp5
35	The Trooper	Iron Maiden	Iron Maiden - The Trooper.gp4
36	Im Yours	Jason Mraz	Jason Mraz - Im Yours (ver 3 by Maitinin).gp5
37	Canon Rock	JerryC	JerryC - Canon Rock.gp4
38	Are You Gonna Be My Girl	Jet	Jet - Are You Gonna Be My Girl.gp4
39	All Along The Watchtower	Jimi Hendrix	Jimi Hendrix - All Along The Watchtower (ver 2).gp3
40	Hey Joe	Jimi Hendrix	Jimi Hendrix - Hey Joe (ver 2).gp3
41	Little Wing	Jimi Hendrix	Jimi Hendrix - Little Wing.gp3
42	Dust In The Wind	Kansas	Kansas - Dust In The Wind.gp5
43	Laid To Rest	Lamb of God	Lamb of God - Laid To Rest.gp5
44	Black Dog	Led Zeppelin	Led Zeppelin - Black Dog.gp5
45	Stairway To Heaven	Led Zeppelin	Led Zeppelin - Stairway To Heaven.gp5
46	Free Bird	Lynyrd Skynyrd	Lynyrd Skynyrd - Free Bird.gp5
47	Sweet Home Alabama	Lynyrd Skynyrd	Lynyrd Skynyrd - Sweet Home Alabama.gp5
48	Sweet Dreams Are Made Of This	Marilyn Manson	Marilyn Manson - Sweet Dreams Are Made Of This (ver 2).gp5
49	Symphony Of Destruction	Megadeth	Megadeth - Symphony Of Destruction (ver 6 by Hanger.18).gp5
50	Battery	Metallica	Metallica - Battery (ver 2).gp3
51	Creeping Death	Metallica	Metallica - Creeping Death.gp5
52	Enter Sandman	Metallica	Metallica - Enter Sandman (ver 5).gp4
53	Enter Sandman	Metallica	Metallica - Enter Sandman.gp5
54	Fade To Black	Metallica	Metallica - Fade To Black (ver 4).gp5
55	For Whom The Bell Tolls	Metallica	Metallica - For Whom The Bell Tolls.gp3
56	Master Of Puppets	Metallica	Metallica - Master Of Puppets (ver 4 by DUDERMAN).gp5
57	Master Of Puppets	Metallica	Metallica - Master Of Puppets.gp3
58	Nothing Else Matters	Metallica	Metallica - Nothing Else Matters (ver 5).gp5
59	Nothing Else Matters	Metallica	Metallica - Nothing Else Matters (ver 6).gp3
60	Nothing Else Matters	Metallica	Metallica - Nothing Else Matters (ver 7).gp4
61	One	Metallica	Metallica - One (ver 2).gp5
62	Orion	Metallica	Metallica - Orion (ver 5).gp4
63	Seek And Destroy	Metallica	Metallica - Seek And Destroy (ver 2).gp5
64	The Day That Never Comes	Metallica	Metallica - The Day That Never Comes.gp4
65	Welcome Home Sanitarium	Metallica	Metallica - Welcome Home Sanitarium.gp4
66	Beat It	Michael Jackson	Michael Jackson - Beat It.gp4
67	Beat It	Michael Jackson	Michael Jackson - Beat It.gp5
68	Super Mario Brothers Theme	Misc Computer Games	Misc Computer Games - Super Mario Brothers Theme (ver 2 by nay-palm).gp5
69	Pirates Of The Caribbean	Misc Soundtrack	Misc Soundtrack - Pirates Of The Caribbean - Hes A Pirate (ver 5 by jariss).gp5
70	Pirates Of The Caribbean	Misc Soundtrack	Misc Soundtrack - Pirates Of The Caribbean - Hes A Pirate (ver 9 by ccb51310).gp5
71	Ace Of Spades	Mötorhead	Mötorhead - Ace Of Spades.gp5
72	Hysteria	Muse	Muse - Hysteria.gp4
73	Knights Of Cydonia	Muse	Muse - Knights Of Cydonia.gp5
74	Come As You Are	Nirvana	Nirvana - Come As You Are.gp3
75	Smells Like Teen Spirit	Nirvana	Nirvana - Smells Like Teen Spirit (ver 2).gp5
76	Wonderwall	Oasis	Oasis - Wonderwall.gp4
77	Crazy Train	Ozzy Osbourne	Ozzy Osbourne - Crazy Train.gp5
78	Cemetery Gates	Pantera	Pantera - Cemetery Gates (ver 4).gp5
79	Cowboys From Hell	Pantera	Pantera - Cowboys From Hell (ver 2).gp3
80	Another Brick In The Wall Part 2	Pink Floyd	Pink Floyd - Another Brick In The Wall Part 2.gp4
81	Comfortably Numb	Pink Floyd	Pink Floyd - Comfortably Numb.gp3
82	Wish You Were Here	Pink Floyd	Pink Floyd - Wish You Were Here (ver 5).gp5
83	Wish You Were Here	Pink Floyd	Pink Floyd - Wish You Were Here.gp4
84	Bohemian Rhapsody	Queen	Queen - Bohemian Rhapsody.gp5
85	Californication	Red Hot Chili Peppers	Red Hot Chili Peppers - Californication (ver 3).gp5
86	Cant Stop	Red Hot Chili Peppers	Red Hot Chili Peppers - Cant Stop (ver 5).gp4
87	Scar Tissue	Red Hot Chili Peppers	Red Hot Chili Peppers - Scar Tissue.gp4

Continued on next page

Table A.1 – *Continued from previous page*

ID	Title	Artist	Filename
88	Snow Hey Oh	Red Hot Chili Peppers	Red Hot Chili Peppers - Snow Hey Oh.gp5
89	Under The Bridge	Red Hot Chili Peppers	Red Hot Chili Peppers - Under The Bridge (ver 6 by dorissie).gp5
90	Under The Bridge	Red Hot Chili Peppers	Red Hot Chili Peppers - Under The Bridge.gp5
91	Europa	Santana	Santana - Europa.gp3
92	Rock You Like A Hurricane	Scorpions	Scorpions - Rock You Like A Hurricane (ver 2).gp5
93	Still Loving You	Scorpions	Scorpions - Still Loving You (ver 3).gp5
94	Raining Blood	Slayer	Slayer - Raining Blood (ver 3).gp4
95	Chop Suey	System Of A Down	System Of A Down - Chop Suey (ver 6 by Benzie101).gp5
96	Toxicity	System Of A Down	System Of A Down - Toxicity.gp4
97	Yesterday	The Beatles	The Beatles - Yesterday.gp5
98	Reptilia	The Strokes	The Strokes - Reptilia.gp5
99	Seven Nation Army	The White Stripes	The White Stripes - Seven Nation Army.gp3
100	Got To Give It Up	Thin Lizzy	Thin Lizzy - Got To Give It Up.gp5

Table A.2: The 47 raw GP files in the test set.

ID	Title	Artist	Filename
1	Hells Bells	ACDC	ACDC - Hells Bells.gp5
2	You Shook Me All Night Long	ACDC	ACDC - You Shook Me All Night Long.gp5
3	Faded	Alan Walker	Alan Walker - Faded.gpx
4	War Pigs	Black Sabbath	Black Sabbath - War Pigs.gp3
5	12 Bar Blues	Blues	Blues - 12 Bar Blues.gp5
6	Sunshine Of Your Love	Cream	Cream - Sunshine Of Your Love (ver 2 by joshscus).gp5
7	Hotel California	Eagles	Eagles - Hotel California.gp5
8	Layla	Eric Clapton	Eric Clapton - Layla (ver 3).gp3
9	Layla	Eric Clapton	Eric Clapton - Layla (ver 5 by roadiekill).gp5
10	Gymnopedie No 1	Erik Satie	Erik Satie - Gymnopedie No 1 (ver 2).gp3
11	Mad World	Gary Jules	Gary Jules - Mad World (ver 6 by Krystof).gp
12	Hallelujah	Jeff Buckley	Jeff Buckley - Hallelujah (ver 2).gpx
13	Hallelujah	Jeff Buckley	Jeff Buckley - Hallelujah (ver 7 by mandelstamdavid).gpx
14	Canon Rock	JerryC	JerryC - Canon Rock.gp4
15	Canon In D	Johann Pachelbel	Johann Pachelbel - Canon In D (ver 6 by Ezechiel).gp5
16	Hurt	Johnny Cash	Johnny Cash - Hurt (ver 5 by Lemmers).gp5
17	Breaking The Law	Judas Priest	Judas Priest - Breaking The Law (ver 8 by Manowarrior).gpx
18	Babe Im Gonna Leave You	Led Zeppelin	Led Zeppelin - Babe Im Gonna Leave You.gp4
19	Since Ive Been Loving You	Led Zeppelin	Led Zeppelin - Since Ive Been Loving You.gp5
20	Whole Lotta Love	Led Zeppelin	Led Zeppelin - Whole Lotta Love.gp4
21	Blues	Lessons	Lessons - Blues - 12 Killer Blues Licks.gp3
22	Moonlight Sonata	Ludwig van Beethoven	Ludwig van Beethoven - Moonlight Sonata - 1St Movement Op 27 No 2 (ver 3 by Gameguy327).gpx
23	Simple Man	Lynyrd Skynyrd	Lynyrd Skynyrd - Simple Man (ver 3).gp4
24	Classical Gas	Mason Williams	Mason Williams - Classical Gas (ver 4).gp3
25	Holy Wars The Punishment Due	Megadeth	Megadeth - Holy Wars The Punishment Due (ver 5 by emad).gp4
26	The Unforgiven	Metallica	Metallica - The Unforgiven (ver 12 by DUDERMAN).gp5
27	Twinkle Twinkle Little Star	Misc Children	Misc Children - Twinkle Twinkle Little Star.gp5
28	Silent Night	Misc Christmas	Misc Christmas - Silent Night (ver 4 by Paulicz).gp5
29	A Star Is Born	Misc Soundtrack	Misc Soundtrack - A Star Is Born - Shallow.gp
30	Harry Potter	Misc Soundtrack	Misc Soundtrack - Harry Potter - Hedwigs Theme.gp5
31	Inception	Misc Soundtrack	Misc Soundtrack - Inception - Time (ver 5).gpx
32	Game Of Thrones Theme	Misc Television	Misc Television - Game Of Thrones Theme (ver 8 by LewtElune).gp5
33	Happy Birthday	Misc Traditional	Misc Traditional - Happy Birthday (ver 3 by luhudroid).gp5
34	Spanish Romance	Misc Traditional	Misc Traditional - Spanish Romance (ver 3 by GuiTaR ChOppER).gp5
35	The Star-Spangled Banner	Misc Traditional	Misc Traditional - The Star-Spangled Banner (ver 4 by Euclid47).gp5
36	Comfortably Numb	Pink Floyd	Pink Floyd - Comfortably Numb.gp5
37	Is There Anybody Out There	Pink Floyd	Pink Floyd - Is There Anybody Out There (ver 3).gp3
38	Money	Pink Floyd	Pink Floyd - Money (ver 2 by patrick_guitar).gp4
39	Time	Pink Floyd	Pink Floyd - Time (ver 4 by Rock Glenn).gp5
40	Love Of My Life	Queen	Queen - Love Of My Life.gpx
41	Creep	Radiohead	Radiohead - Creep (ver 8 by Rock Glenn).gp5

Continued on next page

Table A.2 – *Continued from previous page*

ID	Title	Artist	Filename
42	Black Magic Woman	Santana	Santana - Black Magic Woman (ver 2).gp4
43	House Of The Rising Sun	The Animals	The Animals - House Of The Rising Sun (ver 7 by mandelstamdavid).gpx
44	Blackbird	The Beatles	The Beatles - Blackbird (ver 4 by James_McLeod).gpx
45	Blackbird	The Beatles	The Beatles - Blackbird.gp4
46	Seven Nation Army	The White Stripes	The White Stripes - Seven Nation Army (ver 12 by gerusbel).gp5
47	La Grange	ZZ Top	ZZ Top - La Grange (ver 2).gp5

Appendix B

Grid search results of the note-event separator

For developing the note-event separator, two strategies were implemented and tested: the F0 segmentation-based strategy and the onset detection-based strategy. The frame size and hop size used for both strategies were chosen empirically using a grid search over various frame size and hop size values. The complete results obtained from the grid search are presented in table B.1 and table B.2.

Table B.1: Grid search results of the note-event separator using the F0 segmentation-based strategy. PDT: pitch difference threshold. NDT: note duration threshold.

frame_size	hop_size	PDT	NDT	precision (%)	recall (%)	f1 (%)
1024	256	0.1	25	28.5	32.4	30.3
			50	38.0	36.1	37.0
			75	40.7	33.4	36.7
		0.2	25	60.0	38.2	46.6
			50	64.0	38.8	48.4
			75	63.8	37.5	47.3
		0.4	25	56.3	30.4	39.5
			50	57.4	30.5	39.8
			75	56.7	29.3	38.6
		0.8	25	41.3	16.7	23.8
			50	41.8	16.8	23.9

Continued on next page

Table B.1 – *Continued from previous page*

frame_size	hop_size	PDT	NDT	precision (%)	recall (%)	f1 (%)
			75	41.4	16.4	23.5
	512	0.1	25	32.2	34.7	33.4
			50	38.4	36.0	37.2
			75	39.5	32.4	35.6
		0.2	25	57.1	37.2	45.1
			50	61.0	37.9	46.7
			75	60.1	35.6	44.7
		0.4	25	57.0	32.9	41.7
			50	58.6	32.9	42.2
			75	57.1	30.6	39.9
		0.8	25	49.9	24.5	32.9
			50	50.1	24.4	32.8
			75	48.5	22.8	31.1
	768	0.1	25	30.5	35.4	32.8
			50	37.4	37.9	37.7
			75	40.1	32.7	36.0
		0.2	25	56.3	41.5	47.8
			50	64.3	43.7	52.1
			75	63.2	39.0	48.2
		0.4	25	61.3	40.3	48.7
			50	64.5	41.1	50.2
			75	63.0	37.0	46.6
		0.8	25	59.9	35.6	44.6
			50	61.1	35.7	45.1
			75	59.5	32.7	42.2
2048	512	0.1	25	37.5	47.8	42.0
			50	51.7	55.4	53.5
			75	54.1	51.8	52.9
		0.2	25	67.1	57.0	61.6
			50	77.8	60.7	68.2
			75	77.6	57.6	66.1
		0.4	25	70.3	51.2	59.2
			50	73.6	52.0	60.9
			75	73.4	49.2	59.0
		0.8	25	52.4	25.6	34.4
			50	52.7	25.4	34.2

Continued on next page

Table B.1 – *Continued from previous page*

frame_size	hop_size	PDT	NDT	precision (%)	recall (%)	f1 (%)
			75	51.9	24.2	33.0
	1024	0.1	25	40.1	50.5	44.7
			50	51.6	55.7	53.5
			75	53.4	51.0	52.2
		0.2	25	69.4	63.2	66.2
			50	78.0	64.9	70.9
			75	78.1	60.6	68.3
		0.4	25	74.5	60.4	66.7
			50	77.9	60.5	68.1
			75	77.4	56.2	65.2
		0.8	25	65.1	41.2	50.5
			50	65.9	40.7	50.3
			75	64.6	37.9	47.8
	1536	0.1	25	14.3	25.2	18.2
			50	41.5	46.1	43.6
			75	46.4	42.3	44.3
		0.2	25	29.1	38.2	33.0
			50	66.3	58.5	62.2
			75	69.9	54.4	61.2
		0.4	25	40.0	43.3	41.6
			50	74.3	60.5	66.7
			75	75.8	55.8	64.3
		0.8	25	56.1	42.0	48.0
			50	67.1	45.4	54.1
			75	65.8	40.7	50.3
4096	1024	0.1	25	13.2	20.9	16.2
			50	23.8	28.4	25.9
			75	27.4	28.6	28.0
		0.2	25	29.7	33.9	31.7
			50	57.0	51.3	54.0
			75	61.5	51.6	56.1
		0.4	25	42.6	35.2	38.6
			50	50.1	37.2	42.7
			75	53.8	37.2	44.0
		0.8	25	44.5	29.7	35.7
			50	48.1	30.6	37.4

Continued on next page

Table B.1 – *Continued from previous page*

frame_size	hop_size	PDT	NDT	precision (%)	recall (%)	f1 (%)
			75	50.5	30.7	38.2
	2048	0.1	25	9.8	20.0	13.2
			50	18.3	21.2	19.6
			75	18.3	21.2	19.6
		0.2	25	19.4	29.0	23.2
			50	34.4	33.6	34.0
			75	34.4	33.6	34.0
		0.4	25	28.4	29.4	28.9
			50	37.3	29.9	33.2
			75	37.3	29.9	33.2
		0.8	25	31.9	27.7	29.7
			50	37.5	28.1	32.1
			75	37.5	28.1	32.1
	3072	0.1	25	8.5	15.3	11.0
			50	8.5	15.3	11.0
			75	17.7	16.3	17.0
		0.2	25	14.2	20.9	16.9
			50	14.2	20.9	16.9
			75	25.4	22.1	23.6
		0.4	25	22.4	23.8	23.1
			50	22.4	23.8	23.1
			75	34.9	25.3	29.3
		0.8	25	25.4	23.8	24.6
			50	25.4	23.8	24.6
			75	35.1	24.8	29.1

Table B.2: Grid search results of the note-event separator using the onset detection-based strategy. NDT: note duration threshold.

frame_size	hop_size	NDT	precision (%)	recall (%)	f1 (%)
1024	256	25	86.4	95.2	90.6
		50	89.5	95.2	92.0
		75	92.3	93.2	92.7
	512	25	91.1	95.4	93.2
		50	92.1	94.2	93.1

Continued on next page

Table B.2 – *Continued from previous page*

frame_size	hop_size	NDT	precision (%)	recall (%)	f1 (%)
		75	92.6	92.1	92.4
	768	25	78.2	84.7	81.3
		50	80.9	84.6	82.7
		75	83.6	80.7	82.1
2048	512	25	85.9	95.1	90.3
		50	89.4	94.7	92.0
		75	92.1	91.2	91.7
	1024	25	74.8	79.0	76.8
		50	75.5	78.1	76.8
		75	77.1	73.1	75.0
	1536	25	30.6	32.5	31.5
		50	31.1	32.5	31.8
		75	29.6	27.6	28.6
4096	1024	25	59.8	75.4	66.7
		50	66.4	76.5	71.1
		75	70.5	76.5	73.4
	2048	25	47.8	54.7	51.0
		50	52.7	56.3	54.5
		75	52.8	56.3	54.5
	3072	25	11.5	11.1	11.3
		50	11.5	11.1	11.3
		75	11.5	11.1	11.3