Automatic Tablature Estimation with Convolutional Neural Networks: Approaches and Limitations

Thomas M. Maaiveld

Supervisors:
Albert Meroño, PhD
Jonathan Driedger, PhD
Delia Fano Yela, PhD

A thesis presented for the degree of
Master of Science

Department of Computer Science, Faculty of Sciences
Vrije Universiteit Amsterdam
Netherlands
20th of April, 2021

Automatic Tablature Estimation with Convolutional Neural Networks: Approaches and Limitations

Thomas M. Maaiveld

Vrije Universiteit Amsterdam

Abstract. This thesis investigates new strategies to transcribe guitar music to tablature using a Convolutional Neural Network (CNN). Automatic Music Transcription (AMT) has been a long-standing challenge in Music Information Retrieval (MIR). Tablature is a type of notation that provides the guitarist with specific information on how to produce notes on the guitar. Transcribing audio to this form of notation comes with additional challenges, as the same note may be produced on different strings of a guitar. Using an existing CNN implementation for tablature transcription, four adaptations to this CNN were implemented and evaluated using the existing approach as a baseline: The use of spectral CNN kernels, which may learn characteristic frequency patterns relevant to AMT; a fully convolutional network architecture, which uses a reduced number of parameters to perform tablature estimation; an adaptation that includes chord-based domain information usually not available in a transcription task, to investigate whether CNNs can benefit from this additional information when performing tablature estimation; and lastly, a transposition-based data augmentation strategy to address challenges concerning data set size and availability in tablature transcription. Based on initial results, an additional experiment into the effect of data set size was also performed. Upon evaluation, only the spectral kernel adaptation proved to be a significant improvement over a standard image recognition CNN architecture, and showed promise for improving pitch recognition, although tablature accuracy was not significantly improved. The implications and conclusions of the other approaches are discussed. Among these, the need for larger data sets for tablature estimation and model limitations for polyphonic transcription are established. The code used to implement the adaptations is available at `https://github.com/tmaaiveld/tab-cnn/`.

Keywords: Music Information Retrieval · Automatic Music Transcription · Tablature · Convolutional Neural Network

Acknowledgements

First and foremost, I would like to extend my heartfelt thanks to my thesis supervisors, Jonathan, Delia and Albert. Early on in the project, they enthusiastically agreed to work on a project and suggested I look at CNNs for tablature transcription, helping me bridge the gap between my musical and computer science interests. They guided my initial forays into the MIR literature and the problem of machine transcription, and the ideas they suggested during our brainstorming sessions became the basis of the investigations I carried out. In the later phases of writing my thesis, the scrupulous feedback they provided on my drafts not only vastly improved the quality of the produced result, but also helped me develop my skill in describing the rationale and process behind the research. The fact that they were able to perform these roles so well during the upheaval of the pandemic constitutes an even more impressive feat. I also want to thank Chordify for their excellent cooperation, and for allowing me to participate in their organization at various levels. Although the pandemic unfortunately thwarted any possibility of working at their office, I learned a lot from seeing how they get things done at a distance.

In a more broad sense, I am indebted to the MIR community as a whole (in particular to the ISMIR conference and MIREX) for stimulating and facilitating innovation in this field, which provided me with a wealth of literature to dive into in my attempt to deliver a modest contribution. My first online conference experience at ISMIR 2020 was a blast. In particular, I am indebted to Andy Wiggins and Youngmoo Kim for having developed the model used as a starting point for this thesis. Andy was even kind enough to write me back when I had some questions about how their research was carried out. A thank you is also in order to my friend and fellow musician Matúš Tejiščák at Chordify, who helped by getting me in touch with his company and brainstormed ideas with me at an early stage of the project. Last but not least, I thank my parents for their moral and financial support during my long studies, and for helping me continue to develop my passions into my young adulthood.

Table of Contents

Chapter 1

Introduction

Automatic Music Transcription (AMT) is a fundamental problem in Music Information Retrieval (MIR) [12,97]. The aim of AMT is to generate a symbolic notation from a musical audio signal. In particular, *polyphonic* transcription has become an important topic of research in AMT, which is a challenging task that involves identifying multiple concurrent pitches produced by one or more instruments [5,13]. Additionally, the variety of timbres, recording settings, musician characteristics and notation schemes involved in different tasks within AMT complicate the creation of data sets and the search for techniques that generalize well between applications [102,11]. Due to the challenges involved in AMT, highly skilled human annotators currently outperform automated methods [52,31]. However, manual transcription is a time-consuming task that requires several years of experience or training to learn. Improvements to current AMT methods would enrich a broad range of interactions between people and music, including music education, music creation, music production, music search and musicology [11]. For example, automatic score generation could enable amateur musicians to notate their own creations with greater ease, or more easily obtain instructions to play their favourite songs on their instrument.

In music, *transcription* refers to the practice of creating a notation of a performed or recorded piece of music. Given an audio recording, the aim of transcription is to generate a *score* notation of the performed music. A score usually contains pitch and timing information about a piece, and provides a musician or group with instructions on how to perform it. Examples of score notations include sheet music[1] and lead sheets[2]. However, these notations can be ambiguous for guitar music, as the same note may be played on different positions on the fretboard of the guitar. Therefore, the most popular form of score notation for guitar is *tablature* notation. Tablature not only provides a guitarist with pitch and timing information, but also with information on how to produce these on a guitar by indicating the string and fret to be played. While guitar music may be notated in traditional score music form, many guitarists prefer a notation that indicates the fret positions to use when playing a piece of music [64]. This removes the ambiguity of choosing a string to play a note. Figure 1 gives an example of the sheet music and tablature notation of the guitar arrangement of the popular tune Blackbird (McCartney, 1968). The bottom staff lines indicate strings and the numbers indicate the frets to be played.

[1] `https://imslp.org/`

[2] `https://jazzleadsheets.com/`

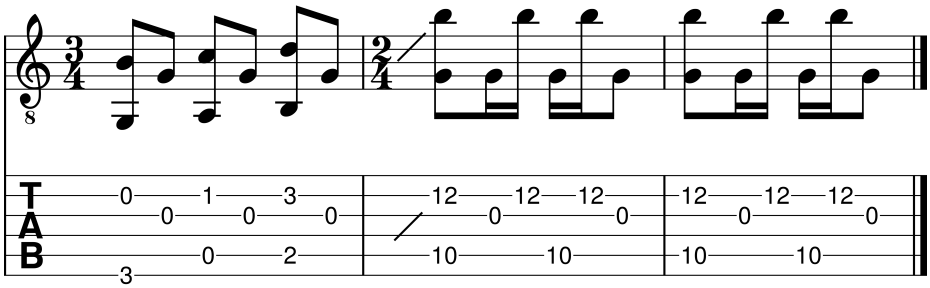


Fig. 1: Sheet music and tablature of first three bars of the guitar backing of Blackbird (McCartney, 1968). The lines of the bottom staff indicate the guitar string to be played, while the numbers denote which fret to press.

An AMT pipeline may be imagined as consisting of 3 elements: time-frequency transformation, pitch estimation and score generation. The time-frequency transformation step involves applying a transformation to an audio recording that decodes the time and frequency information it contains. The choice of technique for this step has been relatively consistent in existing approaches, with a preference towards using *spectrograms* as a time-frequency representation. Spectrograms are 2-dimensional images that contain a mapping of the activity of frequencies in a signal over time, making pitch and timing information easier to decode than in a raw signal. Therefore, they are a particularly suitable representation for many of the approaches applied in AMT [71]. However, there are several factors that make the musical information in spectrograms difficult to decode. The spectral pattern of a note performed on one instrument may look different to the same note on another instrument. This not only applies to different kind of instruments, but also to individual instruments of the same kind. The same note on the same instrument may even look different if performed differently. This problem is exacerbated in polyphonic transcription, where different instruments, pitches and voices may overlap on the spectrogram. Furthermore, different pitches in a polyphonic signal tend to be separated by rational intervals, which causes their spectral profiles to become mixed, and makes them difficult to disentangle. Automated methods are as of yet unable to reproduce the accuracy with which the human ear can decode this information.

A wide variety of techniques have been proposed to perform the pitch estimation step. Many successful AMT approaches utilize either spectrogram factorization techniques or discriminative approaches to perform pitch estimation [11,18,97]. Spectrogram factorization is based on the principle of describing the input spectrogram as a weighted combination of basis spectra [35,33,97]. By decomposing a spectrogram into a dictionary of spectral templates and temporal activations of these templates, the spectrogram can be transformed into a rep-

resentation that is easier for a machine to decode. In contrast, discriminative approaches aim to directly classify pitches, or even the score itself, from features extracted from the audio [97]. In recent years, the predictive performance of discriminative approaches has eclipsed that of spectrogram factorization techniques. While the performance of spectrogram factorization seems to have reached a plateau, the application of machine learning to train discriminative models has led to significant improvements in polyphonic transcription tasks such as pitch estimation and the tracking of note onsets [36,9,12,11].

For score generation, the method used is usually dependent on the desired format or type of annotation, and may be combined with the pitch estimation step depending on the applied technique. The transcription of guitar music to tablature is considered an especially challenging task within AMT, as it requires the system to recognize on which string a note was played. Estimating the *string-fret combinations* played in a piece is difficult, as the timbral difference between strings is subtle and difficult to interpret on a spectrogram, especially in a polyphonic signal [69,117,49]. While the correct string-fret combination may often be inferred from musical context or common guitar practice, such abstract domain knowledge is difficult to model.

A recent and particularly successful approach to guitar tablature transcription is that of Wiggins & Kim [114]. This approach uses a Convolutional Neural Network (CNN) as a discriminative model to directly classify features to string-fret annotations on the guitar. Previous applications of CNNs in AMT include piano transcription [97] and chord annotation [46], and have shown their potential for improving on existing transcription methods. The application of CNNs to tablature estimation involves a novel approach to this task in which the likelihood of each fretting is estimated for each guitar string individually. In order to train the model, the authors used a data set annotated using a special type of guitar pickup called a hexaphonic pickup. This pickup allows the annotation process to be partially automated [115]. The evaluation of the model yielded promising results, significantly improving predictions compared to the initial results presented by the authors of the data set and other state-of-the-art approaches on other data sets [114,26]. CNNs are *black-box models*, meaning the decisions they make are not simple to interpret. This warrants additional investigation of the model, as understanding the means by which CNNs process audio data may yield new insights for their application to AMT.

In their analysis of the model's performance, the authors noted three main errors: false positives, false negatives and cases where a note was assigned to the wrong string [114]. However, the examples given are limited to observations from the same data set used to train and evaluate the model. In order to take a further look at the capabilities of the system and evaluate the performance on recordings not included in the data set, we conducted initial tests using two recordings of performances by the author: one of a short test piece of three chords and one of the piece written in Fig. 1. The model was trained in accordance with [114],

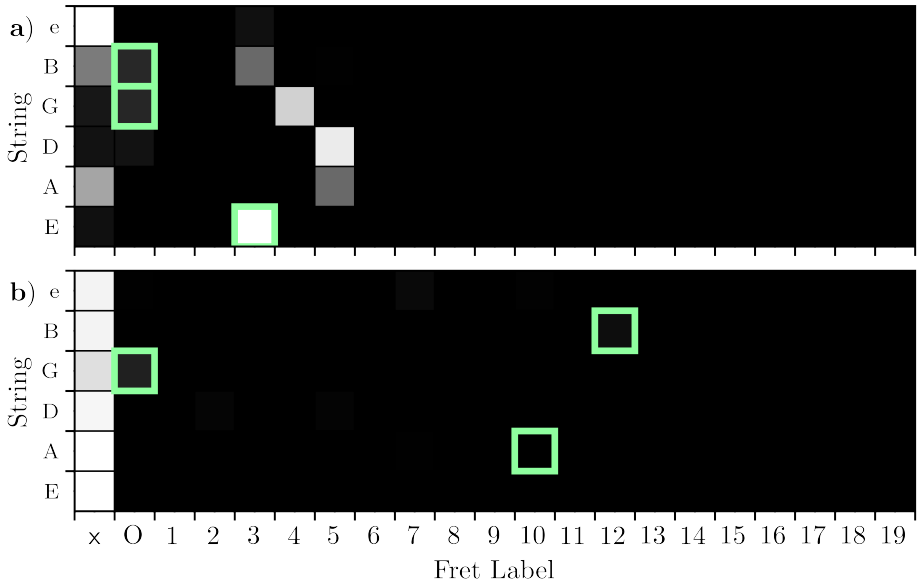


Fig. 2: Two example predictions of the Wiggins & Kim model, plotted as a matrix of string-fret combinations. The cell luminance indicates the estimated probability by string of each fretting. The leftmost column indicates an inactive string. The true frettings are marked in green.

and the results were visualized in a video and uploaded to the web[3,4]. Figure 2 shows the estimated annotations produced by the model for two time-steps of the second piece, where the predicted probabilities are plotted as a matrix of strings and frets resembling the fretboard of the guitar. The probabilities sum to 1 for each row of the matrix. The string-fret combinations played during these time-steps are indicated in green.

The generated annotations contained many of the errors identified by Wiggins & Kim. However, the excerpts plotted in Fig. 2 show some particularly interesting effects. To discuss these, we use a letter and number combination separated by a stroke to denote string-fret combinations in the matrix (e.g. 'A|5' indicates the 5$^{\text{th}}$ fret of the A string). Figure 2a shows the prediction for a section of the recording which corresponds to the first three notes in Fig. 1. Here, the model predicts a standard 'bar chord' guitar fretting (E|3, A|5, D|5, G|4, B|3). This is interesting because the voicing used here in Blackbird is uncommon compared to the standard bar chord voicing. The pitches of A|5 and D|5 are correct, but placed on the wrong string (the true labels are G|0 and

[3] `https://www.youtube.com/watch?v=2OExkioJuKw`
[4] `https://www.youtube.com/watch?v=iq15elk9bwQ`

B|0). This indicates that the model is biased towards 'standard' combinations that are heavily represented in the data used to train the model.

The string-fret matrix plotted in Fig. 2b corresponds to the first few notes played in the second bar of Fig. 1. The uncommon combination of A|10, G|0 and B|12 in Blackbird is missed by the model, and it is also unable to predict an alternative fretting. Notably, the model is unable to predict the notes played high on the fretboard of the guitar. This is interesting because CNNs for audio have *shift-invariant* properties, meaning they are able to recognize the same pattern across different parts of the pitch spectrum. Thus, if the model is able to recognize a note played on A|0, it should also be able to recognize a note played on A|10. This example indicates that the model is limited in this capacity. These observations warrant investigation of whether adapting the model to address these weaknesses could improve its performance in tablature transcription.

This thesis contains an investigation of state-of-the-art methods for tablature transcription, and describes an evaluation of several proposed approaches to improving them. By modifying the model proposed by Wiggins & Kim and using it as a baseline for a set of experiments, the potential of each modification is evaluated. An additional aim of the experiments is to explore how CNN tablature estimation models function. Given the nature of black-box models, additional investigation of the baseline and adapted models is warranted, yielding valuable insights about the performance of the model for future AMT implementations. Thus, the aim of this thesis is to yield insights into the functioning of black-box discriminative models for AMT, and to propose and evaluate improvements based on literature and observation that may improve their shortcomings.

We conducted five experiments with the model proposed by Wiggins & Kim. In the first experiment, we apply a pitch shift data augmentation strategy to evaluate its potential for predicting pitches and string-fret combinations, and to investigate to what degree imbalances in the training data may bias the model. We also a test a series of adaptions to the model architecture. The second experiment investigates the potential of using musically motivated filters in CNNs for tablature estimation, inspired by the literature [82]. The third experiment evaluates an architecture with only convolutional layers to test if a design with fewer parameters can maintain a similar performance to the baseline. In the fourth experiment, the scores used by the musicians that recorded the training data are given to the model to see how much it can benefit from additional information about the harmonic context. The only aim of this approach is to investigate this model's functioning in comparison with the other models and baseline, as chord annotations are usually assumed to be unavailable in the setting of a transcription task. The fifth and final experiment tests the influence of training set size on the performance of the model, and was carried out based on observations in the first four experiments. The evaluations of the results are carried out using the metrics proposed by Wiggins & Kim, which evaluate the model's performance on pitch estimation as well as on prediction of string-fret combinations.

The structure of this thesis is as follows. Section 2 gives a thorough introduction to the field of AMT and the history of mathematical models for automatic

transcription. Section 3 discusses the theoretical background applied in CNNs for automatic tablature estimation. Section 4 gives a mathematical definition of the applied models, and describes the hypotheses and motivations for the adaptations made to the model. In Section 5, the setup and results of the experiments are discussed, as well as the metrics used for evaluation. Section 6 discusses the implications of the findings for AMT research. The code used to implement the adaptations is available on GitHub[5].

[5] `https://github.com/tmaaiveld/tab-cnn/`

Chapter 2

Related Work

This section provides an overview of the related work in Automatic Music Transcription. Section 2.1 gives an overview of the process of AMT, as well as its applications and related fields. Section 2.2 describes the development of the state-of-the-art for AMT and discusses the successes and limitations of proposed methods. In Section 2.3, we discuss the state-of-the-art for guitar transcription and tablature estimation.

2.1 Automatic Music Transcription

The aim of Automatic Music Transcription (AMT) is to design computational algorithms that convert acoustic music signals into music notation [52,3,11]. It is considered a fundamental problem in music signal processing and information retrieval [52]. AMT involves several tasks, including pitch estimation, note onset and offset detection, instrument recognition, beat and rhythm tracking, interpretation of expressive timing and dynamics, and score typesetting. Such tasks are considered challenging due to the nature of musical audio signals, which may originate from several sources that produce one or more concurrent sound events which are assumed to be correlated over time and frequency [11]. While most monophonic music signals may be accurately transcribed by automated means [12,89], transcription of polyphonic audio (audio containing multiple concurrent sounds) remains an open problem. Furthermore, the wide variety of instruments, recording settings, notation schemes and task types ensure that every application of AMT presents unique challenges. Therefore, AMT solutions may not generalize well across different problems in the field.

Common practical applications of AMT systems include transcribing music to traditional score notation [97,112,18,38,31,50,33], extracting melodies from a signal with various sound sources [55,100,8,39,45,63], extracting chords from audio [46,54,22,30,65,61,53] and generating sheet music from pitch representations [62,69]. These tasks are difficult and time-consuming for human annotators to perform, require significant expertise and are prone to error [102]. A successful AMT system would facilitate the interaction between people and music on many levels, including music education, composition, production, categorization, search and analysis [11]. For this reason, AMT has attracted commercial interest as well, with companies such as Chordify[6] applying AMT algorithms to provide transcription services to customers.

[6] `https://www.chordify.net/`

AMT is situated within the field of Music Information Retrieval (MIR), and is closely related to other fields in signal processing and sequence analysis. It is related to Automatic Speech Recognition (ASR), since both fields involve converting acoustic signals to symbolic sequences [11]. Natural Language Processing (NLP) may also be considered a related field, since the structured nature of music is similar to that of language and may be similarly modeled as a sequence analysis problem [75,21,76,94]. AMT is also related to image processing and computer vision due to the frequent use of 2-dimensional time-frequency representations for musical audio signals [11,3]. Advancements in these fields have often led to breakthroughs in AMT. Conversely, innovations in AMT could yield benefits for improving the state-of-the-art for these related fields.

2.1.1 The AMT Process

A typical AMT pipeline may be imagined as comprising of three main steps. The intermediate data representations for these steps are visualized in Fig. 4. The first step is to apply a *time-frequency transformation* to process an audio signal into a representation that highlights the activity of frequencies over time, as this is a key aspect of transcribing music. This step usually involves taking an audio waveform (Fig. 4a) as input and computing the activity of different parts of the frequency spectrum over time (Fig. 4b). The second step is to estimate the pitches played on the instrument(s) using the representation computed in the previous step. An example pitch representation is shown in Fig. 4c. This step, called *pitch estimation*, may be considered the key challenge in automating the AMT process [13]. A crucial part of this step is the detection of (multiple) *fundamental frequencies*. The fundamental frequency is the lowest frequency of a distinct sound wave produced on the instrument, while *partials* are the upper components of that wave and appear in integer multiples of the fundamental frequency. Detecting these and distinguishing them from the fundamental is important for enabling a system to deduce what the performer played [9]. The third and final step is to arrange the transcribed pitches into the desired musical notation scheme, such as modern musical notation (Fig. 4d). Due to the wide variety of musical notation schemes and task goals in transcription, this step may differ significantly between tasks be omitted entirely.

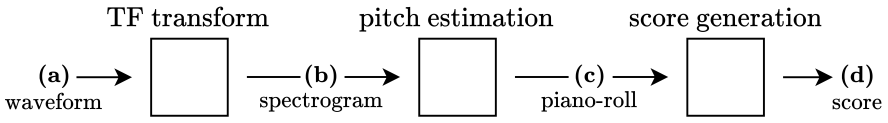


Fig. 3: Steps of the AMT process: Time-frequency (TF) transformation, pitch estimation and score generation. Letters **a**, **b**, **c** and **d** correspond to the intermediate representations shown in Fig. 4.

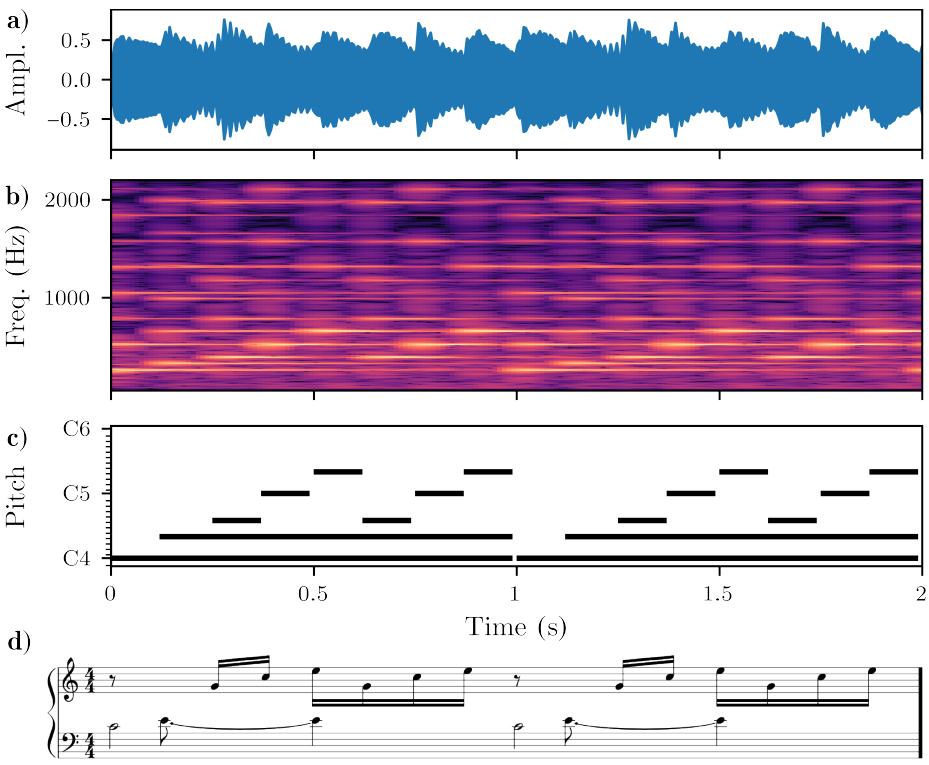


Fig. 4: Representations used in a typical AMT system, as shown in [11]. a) waveform representation (amplitude). b) Time-frequency spectrogram. c) Piano-roll representation. d) Modern musical notation of Prelude No. 1 in C major (J.S. Bach, BWV 846).

2.2 Progression of the State-of-the-Art

To give an overview of important innovations in AMT, we start by briefly discussing the first attempts at automating transcription of music, as well as the first significant successes in AMT. The most important methods of the last decade are discussed in greater detail, as these are relevant to the current state-of-the-art.

2.2.1 Early Approaches

The earliest forays into AMT used a relatively straightforward approach to perform monophonic pitch estimation. Piszczalski & Galler [80] computed a time-frequency representation from the source signal and performed fundamental frequency estimation by finding the strongest frequency activations in for each time section of the input. While this approach works for recordings of monophonic

music with a clear fundamental, it cannot be used to transcribe polyphonic music due to the fact that only one maximum frequency can be detected in each time section. Transcription of polyphonic signals was first explored by Moorer [70], who applied a periodicity detector to detect candidate fundamentals, after which notes were identified through a search of partials. This approach was limited to transcribing duets of single-note lines on two guitars, and was computationally inefficient. The practical applications of these systems were severely limited, as only monophonic voices with a harmonically fixed spectral profile could be transcribed. Real-world transcription applications often require not only the capacity to transcribe polyphonic signals, but also to transcribe instruments with varying harmonic profiles, such as the human voice.

Although the early successes achieved in AMT were modest, they did prove that computational analysis on time-frequency spectrograms could be used to transcribe monophonic signals, and could even decipher polyphonic signals to a limited extent. These straightforward approaches to AMT were only improved upon in the 2000s through the development of three main families of approaches to AMT. In *feature-based* methods, notes are detected using audio features derived from the input time-frequency representation rather than using a specific model [13]. The features are computed for a set of pitch candidates derived from the spectral peaks within a time section of the audio, and used to select the most likely candidate [78,118]. The performance achieved by these methods dominated the state-of-the-art until around 2012, but have since yielded no further improvements. The plateau is likely a consequence of the simplicity of these approaches, lacking a complex model that can capture the more abstract properties and relationships in music. *Statistical model-based* methods apply statistical models to model spectral peaks and estimate the likelihood of each fundamental frequency. Various models have been proposed as estimators, including Gaussian Mixture Models [42,48] and Bayesian approaches [38,119,85]. However, these modeling techniques produce very complex and computationally intensive models, and can realistically only handle small snippets of audio. Given the decreasing interest in feature- and statistical model-based AMT techniques, a thorough discussion of their advantages and disadvantages is outside the scope of this thesis. The third method, *spectrogram factorization*, will be discussed in the next section.

2.2.2 Spectrogram Factorization

Spectrogram factorization techniques are among the most important and widely applied methods in AMT. The aim of these techniques is to describe the input spectrogram as a weighted combination of basis spectra. The most prominent of these for AMT is Non-negative Matrix Factorization (NMF) [98,12]. NMF is a family of unsupervised iterative algorithms in multivariate analysis and was originally proposed in 1999 [60] and later applied to AMT [98]. NMF algorithms factorize a non-negative matrix $\mathrm{V} \in \mathbb{R}_{\geq 0}^{M \times N}$ with M rows and N columns into a product of two non-negative matrices $\mathrm{D} \in \mathbb{R}_{\geq 0}^{M \times R}$ and $\mathrm{A} \in \mathbb{R}_{\geq 0}^{R \times N}$. R is a parameter that sets the *rank* of the approximation, which determines the number

of components used to construct V from the product of D and A. The objective of NMF is to find D and A such that V $\approx$ DA by iteratively performing the optimization in Eq. (2.1):

$$\underset{\mathrm{D}\geq 0,\, \mathrm{A}\geq 0}{\operatorname{argmin}} \; ||\mathrm{DA} - \mathrm{V}||_{\xi} \tag{2.1}$$

where $||\cdot||_{\xi}$ represents a norm that measures the discrepancy between V and DA. In the context of AMT, D is often referred to as a *dictionary* and A as an *activation* matrix, while V is a magnitude spectrogram of the audio [11] with M frequency bins and N time-steps. The columns of D encode spectral information of the source signal and are referred to as *spectral templates*. These represent the expected spectral energy distribution for specific notes or tones. The corresponding rows in A indicate the temporal envelope of the templates, and can be used to derive the onsets and offsets of specific notes.

In its most basic form, NMF relies on multiplicative updates to iteratively optimize D and A in turn using a cost function [60]. This is referred to as *unsupervised* NMF. However, in the context of AMT, this straightforward approach to optimization suffers several shortfalls [74,11]. The first shortfall is a consequence of the fact that the partials of simultaneous musical elements tend to overlap in polyphonic signals. This effect is especially pronounced in the commonly occurring case where simultaneous notes are separated by musically consonant intervals, which makes it difficult to disentangle how much energy belongs to each note [11]. Furthermore, unsupervised NMF does not extract notes, but rather unique sound events [98]. Thus, if the spectral templates in D are learned in concurrence with the activations A, events with multiple concurrent notes may be learned as entirely new components instead of as combinations of separate components relating to each note [98,74]. The second problem with unsupervised NMF is caused by the property of *inharmonicity*, whereby the positions of partials relative to a fundamental tend to deviate from exact integer multiples due to the physical properties of the instrument. This comes in addition to the fact that the harmonic profile of the notes may vary considerably depending on pitch, style of play, dynamics and recording conditions [11]. Unsupervised NMF relies on a fixed harmonic note profile to learn the spectral templates from the training data [98].

Several extensions to the basic NMF model have been proposed to address its shortfalls. One such extension is the application of *sparse coding*, in which parameter R is set to a high value (where $R > M$) to enforce sparsity in the activation matrix A [1,2,74,73]. *Supervised* NMF approaches focus on dictionary design, such as by pre-computing a spectral template for each playable note on an instrument [33,111,84] or by enforcing harmonic constraints on the dictionary matrix [110,84,111]. Another successful spectrogram factorization approach is to fit a latent component probabilistic model to the input spectrograms [99,43,10]. Applying these extended matrix factorization techniques to AMT brought about significant successes in AMT tasks. Vincent *et al.* [9] achieved an average F-measure of 52.7% and placed second in the 2007 Music Information Retrieval

Evaluation eXchange (MIREX)[7] note onset detection challenge [109]. A probabilistic latent component model with a pre-trained dictionary achieved an F-measure of 60.1% on the same task in 2012 [10], and an F-measure of 66.8% on the piano-only onset detection challenge. Similar performances on different data sets using spectrogram factorization were reported in [111] and [33].

Although the results for extended NMF approaches were a promising improvement on previous methods, they also highlight the fact that accurate transcription remains a difficult challenge. The issues concerning the limitations of spectral templates in identifying and disentangling notes remain an important limitation of NMF. Evaluation scores are highest on piano data sets, which have a relatively stable harmonic profile compared to fretted instruments or wind instruments. Transcribing instruments with varying harmonic profiles, such as wind instruments or the human voice, is much more difficult. To allow NMF to recognize more harmonic profiles, one may increase the number of templates used, controlled by parameter R. Specific notes are then modeled as a linear combination of these templates. Therefore, an increased number of templates means a broader range of sounds that can be modeled. However, this leads to a trade-off where the number of invalid spectra that can be represented increases much more quickly compared to the number of valid spectra [11]. Lastly, NMF requires a post-processing step to generate a representation like the one in Fig. 4c. These issues led to an eventual stagnation in progress for NMF methods and AMT in general [12].

2.2.3 Discriminative Models

Like many approaches in AMT, *discriminative models* first gained prominence in related fields before being applied to transcription [58,96,44]. Discriminative approaches aim to directly classify features extracted from audio to the output pitches [97]. Early applications of discriminative models in AMT achieved modest results, and were only barely competitive with the state-of-the-art at the time [81,19]. However, the introduction of *neural networks* achieved significant results in related fields such as speech recognition and computer vision, which prompted their application in AMT. Neural networks are a type of discriminative model that consist of multiple layers of units between the input and the output [17]. Each unit computes a linear combination of the units in the previous layer and applies a nonlinear function to its output before passing it on to the next layer. Neural networks are flexible in the sense that the input and the output layer can be structured in various ways to fit the data format required for a specific application. They are also able to learn to extract information from low-level feature representations, requiring relatively little pre-processing [56,97].

The first applications of neural networks in AMT brought about rapid and significant progress. In an early approach, Böck and Schedl [18] were able to

[7] `https://music-ir.org/mirex/` The MIREX challenge provides a useful benchmark for comparing AMT techniques using standardized evaluation procedures and metrics.

classify piano onsets with a neural network and were able to outperform earlier discriminative methods. Sigtia *et al.* [97] applied neural networks to separately model acoustic properties and musical language. Elowsson & Friberg [37] achieved an F-measure of 82.1% on the MIREX multi-instrument onset detection task using a neural network (80.1% on piano-only onset detection). Similar results have since been achieved with many different neural network-based approaches [3,50,112,105,31]. Neural networks have also achieved successes in other MIR tasks such as melody extraction [8,63,100], genre classification [34,123] and chord recognition [53,62]. They even proved capable of directly transcribing audio to score representations like the one shown in Fig. 4d [28,89].

The application of neural networks in AMT has some important implications for the AMT process. While approaches such as NMF aim to generate a reconstruction of the input data in their components [98,35,97], the aim of discriminative approaches is to train a classifier that can approximate a function that separates the input data in order to assign a label to each instance. While NMF models produce a representation of the input spectrogram that is simpler to decode, discriminative models receive a similar representation and learn to label the input directly. The model parameters are learned by providing many labeled training examples, which the model uses to iteratively adjust its parameters. Hence, this method relies on the availability of sizeable labeled training sets, which can be challenging in AMT [97,11]. Nonetheless, neural networks offer some clear benefits as well. Firstly, they are much more flexible in both the input representation they receive and the output representation they learn to produce. In terms of input, neural networks are able to process various low-level features in the early layers of the network [97]. While the result of spectrogram factorization requires additional processing to produce a representation such as the one shown in Fig. 4, neural networks can be programmed to directly output such a representation [97,11]. Furthermore, since neural networks are able to capture the variability in their input through training on large data sets, they require less hand-crafted parameter selection and instrument-specificity than NMFs [97].

2.3 Tablature Transcription

Within the domain of AMT, the transcription of guitar tablature presents a unique challenge [26,114]. Tablature provides a guitarist with explicit instructions of not only the pitch and timing of the notes of a piece, but also on how to produce them on the instrument. Figure 1 in the Introduction gives an example of tablature notation. Since the same note may be produced on different strings of the guitar, tablature removes the ambiguity of how to play a particular piece by annotating the specific *string-fret combination* used to play each note. Thus, to transcribe guitar music to tablature, each note must be annotated with both the fret number and the string used to produce it. Identifying the correct string presents an additional challenge for automated transcription of tablature, since the timbral difference between strings is small and difficult to detect on a

spectrogram. Furthermore, the domain knowledge used by human transcribers to infer the correct string-fret annotation is abstract and difficult to model.

Compared to general pitch estimation tasks in AMT, relatively little attention has been paid to the problem of transcribing tablature. Some approaches use a probability-based method to model finger configurations for chords, maximizing the likelihood over possible chord states to enable guitar chord transcription [7,6]. A different approach to tablature transcription is to combine conventional pitch estimation methods with playability constraints that determine whether a predicted fingering is practical to play [117,116]. Various methods have also been proposed to generate tablature that conforms to playability constraints from a piano-roll representation (Fig. 4c), such as graph search [87,86,25], genetic algorithms [108,91,88] and neural networks [26,69]. However, it is worth noting that these approaches do not actually transcribe tablature. Rather, they automate the arrangement of a feasibly playable tablature, either by starting from a pitch annotation or by using the output of a pitch estimation method like those described in Section 2.2.

Until recently, no systematic investigations or evaluations of automatic transcription from raw audio to guitar tablature had been attempted, aside from very limited attempts at predicting the string played on a guitar [49,67]. However, in a recently proposed method, Andy Wiggins & Youngmoo Kim use a *Convolutional Neural Network* (CNN) to automate the transcription of guitar music to tablature [114]. CNNs were introduced in computer vision as a type of neural network that is capable of efficiently handling higher-dimensional data such as images [58,57]. This enables them to process (sections of) time-frequency spectrograms directly as input to learn complicated dependencies from large amounts of training examples. The proposed method directly estimates the string-fret combination played on each string of the guitar from a time-frequency representation of the audio. The model is trained through supervised learning over a large data set of tablature-annotated guitar improvisations by six guitarists. This novel data set was recorded and annotated using a hexaphonic pickup, which largely automates this time-consuming process [115].

The approach proposed by Wiggins & Kim presents a number of important advantages over previous approaches. Firstly, the use of GuitarSet offers a larger data set with fine-grained ground-truth annotations, which is important for training neural networks [97]. This approach to making large, high-quality data sets overcomes an important obstacle to progress in AMT research [12,115]. Furthermore, while the model makes a separate prediction for the annotation of each string, the model learns to estimate these in unison during training, and thus can learn complex dependencies between string frettings in its predictions, such as chord shapes and intervals. Additionally, the fact that the model can take temporal context into account by using time-frequency spectrograms as features makes it a powerful tool for transcribing guitar tablature. The ability of a CNN to learn these temporal and string-related dependencies, coupled with the power of neural networks as discriminative models for AMT, have greatly improved on the state-of-the-art in guitar transcription.

Despite its significant successes and advantages, the system also presents certain limitations that warrant investigation. The approach is relatively novel and applies a black-box modeling technique, meaning a model that does not make its decisions explicit to the researcher. This warrants an empirical investigation into the limitations of CNNs for tablature transcription, and the means by which it estimates tablature. The experiments discussed in the Introduction, of which examples were given in Fig. 2, showed that the system struggles to recognize note combinations it has not encountered before. This could be a consequence of the procedure used to generate GuitarSet, which used improvised guitar performances over standard chord progressions and has been noted to contain certain imbalances [27]. Lastly, the long training time of the model makes extensive hyperparameter search infeasible, which has been noted to be an important aspect for neural network performance in AMT [50]. These aspects of the model form a basis for the adaptations and experiments described in this thesis.

Chapter 3

Background

This section summarizes important background knowledge relevant to the approaches described in this thesis. Section 3.1 introduces the mathematical background for the time-frequency representations used as input for the model. Section 3.2 describes the concepts relevant to the procedure applied during training. Section 3.3 discusses the theory applied in Neural Networks and gives a mathematical specification of the layer types used in the approach described by Wiggins & Kim.

3.1 Time-Frequency Transformation of Discrete Signals

When analyzing an audio signal, measuring the activity of periodic frequencies over time gives important information about the notes played to produce the recorded sound. This process, known as *frequency decomposition*, is usually part of the first of the three standard AMT steps described in Section 2.1. Figure 4b in Section 2.1 showed an example of a *time-frequency representation* of an audio signal derived by frequency decomposition. The method used to obtain such a representation is explained in this section.

The most commonly used time-frequency representations in AMT are the short-time Fourier transform (STFT) and constant-Q transform (CQT) [9]. The discussion of the Fourier transform in Section 3.1.1 serves as an introduction to frequency decomposition for audio data. The constant-Q transform is derived from the Fourier transform and is of particular importance, because it is used to construct the input representation to the model used in this thesis. For a complete overview of STFT and CQT, the reader may refer to [71] and [23].

3.1.1 Fourier Analysis

Fourier analysis is a method to obtain a frequency-based decomposition of a periodic waveform by decomposing it into a sum of basic trigonometric functions. In Fourier analysis, the input signal is compared to sinusoid waves with various frequencies [71]. We may define the similarity as the integral of the product wave of two functions $f(t)$ and $g(t)$:

$$\int_{t\in\mathbb{R}} f(t)\cdot g(t)\,\mathrm{d}t \tag{3.1}$$

The integral measures the area enclosed by the product wave $f(t)\cdot g(t)$ and the horizontal axis, subtracting the sum area below the axis from the areas above

it. Thus, positive values for this integral indicate similarity while negative values for indicate dissimilarity.

To apply this concept in signal analysis, let $f(t) : \mathbb{R} \to \mathbb{R}$ represent a signal of a periodic nature and $g(t) = A\cos(2\pi(\omega t - \phi))$ be a set of standard cosine functions with parameters $\omega \in \mathbb{R}$ and $\phi \in \mathbb{R}$ representing their frequency and phase. We assume that the amplitude $A = \sqrt{2}$ and write these functions as $\cos_{\omega,\phi}(t)$. Then Eq. (3.2) gives the definition of the magnitude coefficient $d_\omega \in \mathbb{R}$ in the Fourier transform, which represents the similarity between the signal and the sinusoid with frequency ω. The integral in this definition is used to compute the similarity as in Eq. (3.1). The optimal phase $\phi_\omega \in \mathbb{R}$ for a frequency ω may be found by maximizing for the argument ϕ over this definition, as shown in Eq. (3.3).

$$d_\omega = \max_{\phi \in [0,1)} \left(\int_{t \in \mathbb{R}} f(t) \cos_{\omega,\phi}(t) \, \mathrm{d}t \right) \tag{3.2}$$

$$\phi_\omega = \operatorname*{argmax}_{\phi \in [0,1)} \left(\int_{t \in \mathbb{R}} f(t) \cos_{\omega,\phi}(t) \, \mathrm{d}t \right) \tag{3.3}$$

Thus, for any analog audio signal represented by $f(t)$, constructing its Fourier transform gives the activity of various frequencies ω in the original signal by computing the similarity d_ω for each of those frequencies. This representation gives the similarity between $f(t)$ and the set of sinusoidal wave functions with parameters ω and ϕ, but no longer contains temporal information once the integral over t is computed.

In practice, the Fourier transform is often represented as a complex function $\hat{f} : \mathbb{R} \to \mathbb{C}$ which maps the magnitude and phase information of $f(t)$ for frequency ω to a single complex number.

$$\hat{f}(\omega) = \frac{d_\omega}{\sqrt{2}} \cdot \exp(2\pi i(-\phi_\omega)) \tag{3.4}$$

$$\hat{f}(\omega) = \int_{t \in \mathbb{R}} f(t) \exp(-2\pi i \omega t) \mathrm{d}t \tag{3.5}$$

By rewriting Eq. (3.4) with the definitions for d_ω and ϕ_ω given in Eqs. (3.2) and (3.3), we obtain the commonly used, compact definition of the complex Fourier transform in Eq. (3.5). The magnitude and phase information, as well as the coefficients d_ω and ϕ_ω, may be retrieved from $\hat{f}(\omega)$ by computing its magnitude and phase.

Since the optimization step to compute d_ω and ϕ_ω is performed over a continuous frequency spectrum ω and an infinite, continuous time-span t, further steps are needed for practical application of the Fourier transform to digital signals. First, the transform is computed for a range of discrete values of ω in the case of a digital input signal. Second, we may assume a finite number of samples N over which to compute the Fourier transform. To simplify the notation of a discrete series of integers, we use the notation $[a : b]$ to denote a range of integers from a to b. For a digital signal $x(n) : \mathbb{Z} \to \mathbb{R}$ consisting of real-valued

samples, Eq. (3.5) may be rewritten to compute the discrete Fourier transform $\mathbf{DFT} : \mathbb{R} \to \mathbb{C}$ of this signal. Instead of a frequency parameter ω, this equation uses a parameter $k \in [0 : N-1]$ where $N \in \mathbb{N}$ represents the total number of samples in $x(n)$. Thus, we consider only the frequencies $\omega = k/N$.

$$\begin{aligned} \mathbf{DFT}\{x(n)\}(k) &= \sum_{n=0}^{N-1} x(n) \exp(-2\pi i k n/N) \\ &= \sum_{n=0}^{N-1} x(n) \exp(-2\pi i \omega n) \end{aligned} \tag{3.6}$$

In many signal analysis tasks, the temporal information discarded by the Fourier transform is relevant, such as in detecting the timing of musical events. To recover the time information, the Short-Time Fourier Transform (STFT) was introduced [40], which multiplies the input signal $x(n)$ with a window function $w(n) : [0 : N-1] \to \mathbb{R}$ to isolate short sections of the audio signal before computing the DFT. The window function weights the original signal with a function (often bell-shaped) and sets it to 0 for all but a section of the signal to be considered. By shifting the input signal $m \in \mathbb{N}$ times by the *hop size* parameter $H \in \mathbb{Z}$, we can compute $\mathbf{STFT} : \mathbb{R} \to \mathbb{C}$ over a section of $x(n)$ at the m^{th} hop. Unlike for the DFT, N represents the number of samples of the signal being considered (e.g. the samples where $w(n)$ is non-zero).

$$\mathbf{STFT}\{x(n)\}(m,k) = \sum_{n=0}^{N-1} w(n+mH)x(n) \exp(-2\pi i k n/N) \tag{3.7}$$

By repeatedly computing the Fourier transform over the windowed signal at every step, the frequency spectrum of the input signal may be evaluated over discrete steps in the time dimension. This calculation allows a time-frequency representation to be constructed from the output, which produces a useful data format for temporal signal analysis.

3.1.2 Constant-Q Transform

Discrete Fourier transforms such as the STFT use a linear spacing of ω, since $\omega = k/N$ and $k \in [0 : N-1]$. Although N may be selected arbitrarily for the STFT, digital systems can handle a finite number of parameters, which necessitates choosing a finite number of model sinusoid functions to compute the Fourier transform of the audio signal. However, given a fixed sampling rate for $x(n)$, there is a trade-off between temporal and spectral resolution. Lower fundamental frequencies have longer wavelengths and therefore require a larger amount of samples to be detected. However, computing the STFT over a larger window of samples per hop reduces the accuracy with which musical events will be detected in time, such as note onsets or percussive sounds with higher frequencies. While one could potentially perform multiple iterations of STFT

with various window sizes to detect different parts of the spectrum, this can vastly increase the number of frequency bins needed to calculate for linearly spaced values of ω.

The constant-Q transform (CQT) was introduced to solve this problem [23]. Similar to the STFT, CQT also utilizes a set of filter banks to store frequency activations at each hop. However, unlike STFT, it uses a geometric spacing for ω in order to retain a good frequency resolution on all parts of the spectrum. To solve the trade-off between frequency and time and make the filters adjectant, both the bandwidth and the window size $N(k) \in \mathbb{N}$ of each frequency filter depend on the frequency being considered. The quality factor, which gives the ratio between the frequency of a filter and its bandwidth, remains constant across the entire spectrum. The parameters of the constant-Q transform are the sample rate $f_s \in \mathbb{R}$, desired number of frequencies per octave $b \in \mathbb{N}$, and a chosen fundamental frequency $f_0 \in \mathbb{R}$ from which to start the computation. Equations (3.8) to (3.11) give the expressions used to compute the center frequency $f_k \in \mathbb{R}$ and filter width $\delta f_k \in \mathbb{R}$ for the k^{th} filter, the quality factor $Q \in \mathbb{R}$ and the window length $N(k)$.

$$f_k = f_0 * 2^{\frac{k}{b}} \tag{3.8}$$

$$\delta f_k = f_k \left(2^{\frac{1}{b}} - 1\right) \tag{3.9}$$

$$Q = \frac{f_k}{\delta f_k} \tag{3.10}$$

$$N(k) = \frac{f_s}{\delta f_k} = \frac{f_s}{f_k} Q \tag{3.11}$$

Note that the filter frequency f_k increases logarithmically with a power of two for every integer fraction of k and b. Thus, the frequency of the k^{th} filter doubles every octave, which corresponds to the fact that partials of musical notes are distributed as integer multiples of the fundamental. b is often chosen to be an integer multiple of 12 in the context of musical signals, as there are 12 tones in an octave in an equal-tempered system. A sensible choice for f_0 depends on the task at hand. Choosing 8.18 Hz, for example, corresponds to the MIDI[8] note number 0, a common choice in the context of musical signal analysis. The quality factor Q expresses the ratio between the center frequency f_k and the bin width δf_k, and denotes the frequency resolution of the constant-Q transform.

Let $x(n) : \mathbb{N} \to \mathbb{R}$ be a discrete input signal with a sample rate f_s. Eq. (3.12) gives the calculation of the Constant-Q Transform $\mathbf{CQT} : \mathbb{R} \to \mathbb{C}$ for each spectral bin k [23] at the m^{th} hop. Calculating the CQT for $k \in [0 : K]$ where K

[8] `https://www.midi.org/`

is an integer multiple of b and represents the desired number of bins, we obtain a complex-valued spectral vector of size $K + 1$.

$$\mathbf{CQT}\{x(n)\}(k,m) = \frac{1}{N(k)} \sum_{n=0}^{N(k)-1} w(k, n + mH)x(n) \exp\left(\frac{-i2\pi Qn}{N(k)}\right) \quad (3.12)$$

Unlike in STFT, the window function is scaled depending on which spectral component is being analysed, which solves the trade-off between time and frequency resolution. The complex-valued matrix produced by computing the CQT of signal $x(n)$ for all values of k and m may be transformed into a real-valued matrix by computing the element-wise magnitude. The result of this transformation is called a *magnitude spectrogram*, and may be plotted to display the activity of frequencies over time. Examples of magnitude spectrograms are shown in Fig. 5.

3.1.3 Time-Frequency Representations for Musical Audio

Besides addressing the previously outlined trade-off between frequency and time, CQT offers a number of additional advantages over STFT for musical signal analysis. CQT greatly reduces the number of bins in the resulting spectrogram compared to the linear bins of STFT given its logarithmic spacing of bins along the frequency axis, and fast computation is possible through the use of spectral

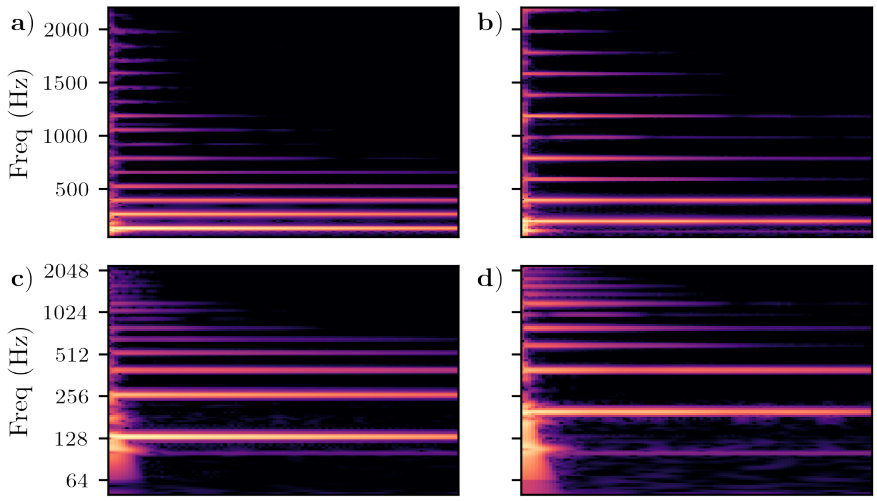


Fig. 5: Magnitude spectrograms of the notes C3 (**a** & **c**) and G3 (**b** & **d**) played on an acoustic guitar. **a** & **b** show a linear pitch axis (STFT), while **c** & **d** show a logarithmically scaled axis (CQT).

kernels [24]. Furthermore, the human ear perceives pitch in a logarithmic fashion rather than a linear one. For instance, if the frequencies of the tones of a melody are translated by a certain frequency value, the melody will no longer sound the same. However, if these frequencies are scaled by a factor, the logical sequence of the melody would sound the same to a human listener. This provides an intuitive reason to use CQT to generate input for AMT applications, as it reflects the way the ear perceives pitch.

To illustrate the difference between a linear and logarithmic pitch axis in a magnitude spectrogram, consider the example shown in Fig. 5, which shows four magnitude spectrograms of two recorded guitar notes (C3 and G3). Fig. 5a and Fig. 5b show the STFT spectrograms computed from the recordings with a linearly spaced frequency axis, while Fig. 5c and Fig. 5d show CQT spectrograms of the same recordings with a log-scaled axis. When a note is produced on an acoustic instrument, the vibrations emitted follow a characteristic pattern visible in Fig. 5. As mentioned in Section 2.1.1, the spectral lines appear in integer multiples of the fundamental frequency, which is clearly visible in the linear axis spectrograms. In music, this pattern of pitches is known as the *overtone series* [77,101]. We may denote this property as $f_n = n \cdot f_0$, where f_n is the n^{th} partial of fundamental frequency f_0.

In the linear domain, since $f_n - f_0 = (n-1) \cdot f_0$, the difference $f_n - f_0$ depends on the fundamental frequency f_0. Thus, in a spectrogram with a linear frequency axis, the gap between the spectral lines differs between notes with different fundamental frequencies. The spectral patterns for different fundamentals appear as scaled variants of each other relative to the zero position. For example, the ratio between C3 and G3 is roughly $2:3$ ($130.81\text{Hz} \,/\, 196.00\text{Hz} \approx \frac{2}{3}$). However, if we log-transform the frequencies to obtain $\log f_n = \log(n \cdot f_0)$, we derive that $\log f_n = \log(n) + \log(f_0)$, and thus $\log f_n - \log f_0 = \log(n)$. Thus, in the logarithmic domain, the difference between f_n and f_0 is always $\log(n)$, which means that in a logarithmic frequency spectrogram, the gaps between the lines are always equal to $\log(n)$, independent of the fundamental frequency. They appear as a shifted versions of each other on a magnitude spectrogram computed with CQT. Since each representation offers its respective advantages for the analysis of signals, the preferred representation depends on the context in which the spectrogram is analysed. While certain systems may benefit from different scaling between fundamentals, others are more easily able to recognize a pattern between shifted variants of the same spectral profile.

3.2 Machine Learning

In this section, we discuss the mathematical background for machine learning (ML) models for classification. First, we give a global description of how machine learning models learn to model relationships from observation in Section 3.2.1. In Section 3.2.2 we specify the mathematical optimization techniques used to enable the ML model described in this thesis to learn. Lastly, Section 3.2.3 specifies the mathematical formulation of the optimization objective.

3.2.1 Function Approximation

The training of a machine learning model is based on the principle of *function approximation*, where a model is trained to approximate an unknown underlying function from real-world observations. To illustrate this, let $x \sim X$ be an independent random variable and let $y \sim Y$ be a variable that is, with some error ϵ, dependent on X. We denote the causal relationship between these variables as $y(x) = f(x) + \epsilon$, where $y(x)$ denotes the relationship by which y is dependent on x, f represents an unknown function and ϵ is an unknown error term that varies between samples drawn from X and Y. Thus, y is dependent on x by some unknown relationship f, plus some randomly distributed noise. Due to the unknown error, f cannot be directly inferred from observation, and may be assumed to be a complex nonlinear function for most cases in which ML is applied. The goal is to predict the value of y based on an observation x. In *classification*, values to predict may either be one of a set of discrete labels, while in *regression*, the aim is to estimate a continuous target value. In this thesis, we focus on the classification case.

To be able to make predictions about y given observations of x, the aim is to fit a model, which we denote as $\hat{f}$, that approximates the true function f as closely as possible. We find $\hat{f}$ by means of a learning algorithm and a data set of observations sampled from X and Y. In this context, samples from X are referred to as *features* and samples from y as *labels*. To illustrate, let $\mathcal{D}$ denote a data set of n feature-label pairs constructed by sampling from distributions X and Y, such that $\mathcal{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. To model f, we use a machine learning model that has parameters θ, and denote it as $\hat{f}_\theta$. The model takes observations from X as its input and returns predictions for the associated values from Y. Denoting the predictions as $\hat{y}$, we write that $\hat{y}(x) = f_\theta(x)$. In classification, the predictions usually take the form of probability estimates for each outcome such that $\hat{y}(x) \in \mathbb{R} \cap [0, 1]$ and $\sum \hat{y}(x) = 1$.

Data Partitioning In order to obtain accurate predictions from our model, $\hat{f}_\theta$ should approximate f as closely as possible. We may approximate the function f by fitting the model to a subset of $\mathcal{D}$, which we call the *training set*. The remainder of the data is withheld for evaluation after training, and is called the *test set*. For instance, if we choose to use m samples for training and we withhold $n - m$ samples for evaluation, then $\mathcal{D}_{train} = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ and $\mathcal{D}_{test} = \{(x_{m+1}, y_{m+1}), \ldots, (x_n, y_n)\}$. The training set is used to optimize the model parameters θ, such that the error between the predictions issued by the model and the true values for y is minimal. Having optimized the model parameters, the resulting model $\hat{f}_\theta$ may then be evaluated by computing predictions for the sample pairs in $\mathcal{D}_{test}$. It is crucial that the test set be kept separate during training to allow for a fair evaluation of the model.

It is often beneficial to perform multiple evaluations of the model over different parts of the data set. This procedure is known as *cross-validation*, and ensures that the results are not biased by the data partitioning, as evaluations on different partitions may produce different results. In this context, $\mathcal{D}_{test}$ is

often referred to as a *validation set*. In the case of *k-fold cross-validation*, the data is divided into k folds, where for each fold, $\mathcal{D}_{train}$ has $n - n/k$ samples and $\mathcal{D}_{test}$ has n/k samples. A model is trained and evaluated over the training and test set for each fold, providing a more robust evaluation than a single split.

3.2.2 Iterative Optimization

In machine learning, the optimization of the model parameters θ is carried out using iterative optimization algorithms. These algorithms optimize the model parameters by computing a *loss metric* over samples of the data set and gradually adjusting the parameters until an optimum is found.

Stochastic Gradient Descent To restrict the discussion to optimization methods relevant to the thesis, we discuss *stochastic gradient descent* (SGD) [20,17]. SGD is an optimization algorithm that optimizes parameters by computing the gradient until an optimal value is found. After randomly initializing n model parameters $\theta \in \mathbb{R}^n$, the model is used to compute a prediction for the i^{th} instance of the data set. Assuming we have a method to quantify the accuracy of this prediction given the true label of the i^{th} instance, the loss over the i^{th} instance may be computed. To quantify the loss, let $L_i(\theta) : \mathbb{R}^n \to \mathbb{R}$ be a loss function that computes the loss for the i^{th} instance of a data set dependent on model parameters θ. The update is weighted by a step size parameter $\eta \in \mathbb{R}$ called the *learning rate*, which prevents the algorithm from (repeatedly) overshooting its descent target with large updates. In stochastic gradient descent, the parameters are then updated after every step via the update rule given in Eq. (3.13).

$$\theta \leftarrow \theta - \eta \nabla L_i(\theta) \tag{3.13}$$

This update rule relies on the fact that the gradient can be computed with respect to the model parameters θ.

The loss may be imagined as a surface in the parameter space of the model (e.g. a space containing all possible combinations of the parameters θ). Gradient descent finds a local minimum on that surface. By subtracting the parameter update term $\eta \nabla L_i(\theta)$, the parameter search is oriented in the opposite direction to the steepest ascent on the loss surface. Convergence for SGD is only guaranteed for a convex loss surface that is continuously differentiable [17,32]. However, since loss functions are usually chosen to be convex and differentiable, the algorithm has proven a useful tool for the optimization of machine learning algorithms [58,57,17].

Batch Optimization Calculating the loss over a single instance before updating the parameters introduces a random component to the optimization process. Each update points the algorithm towards a minimum, but outlier samples that significantly deviate from the trend of the function being approximated may misdirect the gradient descent process. In the other extreme, the gradient may be computed over the entire data set. However, this approach is computationally

intensive. In practice, the gradient is usually computed over a set number of sample pairs before updating θ. This is known as *batch gradient descent*. Eq. (3.14) gives the update rule for batch gradient descent, where $m \in \mathbb{N}$ represents the number of samples included in the batch.

$$\theta \leftarrow \theta - \frac{\eta}{m} \sum_{i=1}^{m} \nabla L_i(\theta) \tag{3.14}$$

3.2.3 Maximum Likelihood Estimation

To find the optimal model parameters to fit f, the objective is to maximize the probability of the observations in data set $\mathcal{D}$ given parameters θ. Maximizing this probability is equivalent to maximizing the likelihood of the parameters θ given the observations [17]. This approach to model optimization is called *Maximum Likelihood Estimation*. Since the natural logarithm is a monotone transformation, this is also equivalent to maximizing the log likelihood. Denoting the finite Euclidean space of all possible model parameters as Θ, the optimization objective is to find the model parameters $\theta \in \Theta$ that maximize the log likelihood. Eq. (3.15) expresses the optimization objective in terms of both probability and log-likelihood maximization, where $\mathcal{L}(\theta|\mathcal{D})$ denotes the likelihood of parameters θ given the data set $\mathcal{D}$.

$$\underset{\theta \in \Theta}{\operatorname{argmax}}\, p(\mathcal{D}|\theta) = \underset{\theta \in \Theta}{\operatorname{argmax}} \log \mathcal{L}(\theta|\mathcal{D}) \tag{3.15}$$

To define a clear formulation for the optimization objective, the likelihood may be expressed in the form of a *loss function*. The loss function expresses the model misfit as a function of the model parameters θ. The loss function used to express the log likelihood when optimizing a machine learning model over a training set is usually selected based on the problem at hand. In classification problems, a commonly used loss function is the *cross-entropy loss* [41]. The cross-entropy can be derived from the likelihood, such that maximizing the (log) likelihood is equivalent to minimizing the cross-entropy [17]. Eq. (3.16) gives the expression of the binary cross-entropy loss function $L(\theta)$ for a binary classification, where the label $y_i \in \{0, 1\}$ of the i^{th} instance of the data set is either 1 or 0, and $\hat{y}_i \in \mathbb{R} \cap [0, 1]$ is the probability of a positive label for i estimated by the model.

$$L(\theta) = -\frac{1}{N} \sum_{i}^{N} y_i \log \hat{y}_i + (1 - y_i) \log (1 - \hat{y}_i) \tag{3.16}$$

Eq. (3.16) may be extended to a multi-class classification setting, where the label y_i may take one of a set of labels. This loss function is called the *categorical cross-entropy loss*. Eq. (3.17) gives the categorical cross-entropy loss for a classifi-

cation problem with C classes. The term $y_{i,k} \log \hat{y}_{i,k}$ computes the cross-entropy for the k^{th} class label and the i^{th} instance.

$$L(\theta) = -\frac{1}{N} \sum_{i}^{N} \sum_{k}^{C} y_{i,k} \log \hat{y}_{i,k} \tag{3.17}$$

Since the use of two subscripts in Eq. (3.17) becomes cumbersome in later sections, we use a vector notation for the categorical cross-entropy loss hereafter. In this notation, for a classification problem with C classes, $\mathrm{y}_i \in \{0,1\}^C$ is a binary vector of length C that denotes label k with a nonzero value in its k^{th} entry. Using binary sparse vectors to indicate a positional value is known as *one-hot encoding*. We use $\hat{\mathrm{y}}_i \in \mathbb{R}^C$ to denote a vector of predicted probabilities for instance i. Eq. (3.18) shows the vector notation formulation of the loss function in Eq. (3.17). The dot product computes the pairwise product and sum of all entries of vectors y_i and $\hat{\mathrm{y}}_i$.

$$L(\theta) = -\frac{1}{N} \sum_{i}^{N} \mathrm{y}_i \cdot \log \hat{\mathrm{y}}_i \tag{3.18}$$

3.3 Neural Networks

This section describes the relevant theoretical background regarding the convolutional neural network model applied in this thesis. A mathematical specification of the layers of the network is also given here.

Neural networks were first introduced as perceptrons in 1958 [90], but saw little attention for a long period due to unsolved problems with parameter optimization and scaling. The development of faster methods of computation and better approaches to neural network design and optimization led to their resurgence as *deep neural networks* [96,58].

3.3.1 Deep Neural Networks

Deep Neural Networks (DNNs) are neural networks that contain multiple layers between the input and the output [14,93]. The application of large data sets has made it feasible to train DNNs with millions of parameters that are capable of learning complex, nonlinear relations [97]. Each layer of a neural network contains a set of *units*, which compute a linear combination of inputs from the units of the previous layer, and pass this through an *activation function*. The neural network takes a set of numerical inputs as its features (x) and returns one or more numerical predictions ($\hat{y}(x)$). Neural networks can be used for both classification and regression. During a *forward pass*, the activations of the network for a batch of samples is computed. Building on the notations introduced in Section 3.2, predictions $\hat{y}(x) = f_\theta(x)$ are computed for each sample in a batch. Here, the parameters θ represent the weights and biases used for the linear combinations carried out by each unit of the network. Subsequently, the weights

and biases are updated in a *backward pass* by computing the loss $L(\theta)$ and its gradient with respect to θ. The development of algorithms that enable fast computation of the gradient allow θ to be optimized using batch gradient descent [59]. In the context of multi-class classification, the final layer has a *softmax* activation function which returns the probability density over all possible class labels. The parameters are updated between each cycle to minimize the loss. To prevent overfitting, *dropout* may be applied to ignore connections to units in a given layer.

3.3.2 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a type of DNN that are especially well suited to processing image data. Since image data typically has a large feature space, CNNs learn shared parameters for a set of *kernels* (also referred to as *filters*) that are *convolved* with their input to compute a set of *feature maps*. These filters are inspired by traditional edge detection algorithms, and consist of a small map of weights that are multiplied with a section of the input to produce a numerical component of the next feature map. The output represents how closely the pattern on the input matches that of the filter. Thus, these filters can learn to detect relevant patterns in the input data, and are useful detecting edges and objects [57]. Each filter is applied to various positions on the input feature map, meaning the same pattern can be detected in different locations. This is known as the *shift-invariance* property of CNNs, and implies that a filter can learn to detect a spectral pattern at various locations on a feature map [34]. The general assumption is that filters in the early layers of the network will learn to recognize small elements in the input image, while later filters should aggregate these observations to detect larger frequent patterns in the data.

CNNs are made up of a number of different layer types. *Convolutional layers* compute a sliding dot product with a kernel for each feature map, subsequently passing the result through an activation function to construct the output. The filter slides across the entire input feature map during computation. Typically, multiple filters are used in each layer to produce additional channels in the output. The number of parameters for each convolutional layer is independent of the size of the input, and is only determined by the chosen dimensions of the filter and the channel depth of the input or feature map being computed. This form of parameter sharing greatly limits the required number of parameters to be learnt by the network through backpropagation and reduces overfitting compared to fully connected layers [56]. *Pooling layers* are used to allow computationally efficient downsampling between hidden layers. Once the dimensionality of the input has been sufficiently reduced by convolution and pooling, 'regular' neural network layers may be used, which are referred to as fully connected layers in the context of CNNs.

An important reason for applying CNNs to machine learning problems that involve image data is the amount of network parameters used. While feed-forward neural networks offer significant advantages in processing features in various ML tasks, direct application to computer vision problems introduces

some complications. Using a gray-scale image of size 125×125 as input for a feed-forward neural network, an input vector of length 15,625 would be required. Given an equal number of nodes in the first layer of the network, this would require $15{,}625^2$ ($\approx$ 244M) connection weights to be learned, rendering a feed-forward neural network architecture with fully connected layers impractical for the task of raw image data processing. However, CNNs are able to handle such large input spaces by downsizing the input in the convolution and pooling layers before passing it to the fully connected layers.

CNNs for AMT By transforming a digital audio signal to a spectral representation like those discussion in Section 3.1, AMT tasks such as multi-F0 estimation may be treated as computer vision tasks, allowing for the application of CNNs. Various such approaches have been applied in piano transcription [112,97,31] and guitar transcription [47,114], which have been shown to outperform the preceding state-of-the-art. These implementations utilize a series of log-scaled time-frequency spectrograms produced by CQT (musical 'frames') with a small window of preceding and following time-steps as input. The inclusion of time-domain information allows the CNN to make predictions about temporal musical events and their likelihood of belonging to a given class when given a training set with ground truths for each temporal event. CQT is posited to be superior to STFT in the context of CNNs for three main reasons: the aforementioned implicit advantages for music analysis offered by logarithmic spacing of frequency filters, the small input dimension resulting in a smaller input layer compared to STFT, and the pitch invariance property of CQT which can be exploited by the shift invariance property of a CNN.

While CNNs offer significant advantages in the context of AMT, they incur similar limitations to traditional neural networks in terms of explainability. While neural networks may approximate a function that produces the desired output with an arbitrary precision, the architecture and parameters of the network give no direct insight into the network's performance on edge cases, nor into its reasoning for reaching a conclusion on a given input [79]. The performance of the model on examples not contained in the training data is relevant in the context of AMT, since practical application of such models relies on their ability to process new information. Subtle differences in recording quality, conditions and instrument timbre necessitate good generalization to new cases. While some methods do exist to visualize or otherwise represent a CNN's response to a given input [79,121], empirical testing remains an important tool for exploring model response to novel input.

3.3.3 Layer Specifications

In this section, bold-font symbols represent real-valued matrices in capital case and real-valued vectors in letter case.

Fully Connected Layers Fully connected layers are equivalent to the layers used in a standard neural network, and compute the activation function on a

linear combination of the input. Equation (3.19) gives the computation of the activations $\mathbf{a}^{[l]}$ of fully connected layer l in linear algebra notation. $\mathbf{W}^{[l]}$ denotes the weights matrix for layer l, $\mathbf{b}^{[l]}$ the bias vector and and $\mathbf{a}^{[l-1]}$ the activation of the previous layer and $h^{[l]}$ the activation function. If the preceding activation is not 1-D, the input matrix is flattened to a 1-D vector before computation. The activations $\mathbf{a}^{[l]}$ of each layer are computed as a linear combination of the weights and biases on the previous layer, plus an activation function $h^{[l]}$.

$$\mathbf{a}^{[l]} = h^{[l]}(\mathbf{W}_{\top}^{[l]}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}) \tag{3.19}$$

Convolution Layers In convolution layers, each kernel matrix is *convolved* with the input and maps its activation to an output feature map. The convolutional layer computation is specified in Eq. (3.20). $\mathbf{W}_{i,j}^{[l]}$ and $\mathbf{B}_{i}^{[l]}$ denote the weight and bias matrices for the i^{th} kernel of layer l and the j^{th} feature map in the previous layer activation tensor $\mathbf{A}^{[l-1]}$. $h^{[l]}(\cdot)$ denotes the activation function of layer l.

$$\mathbf{A}_{i}^{[l]} = h^{[l]}\Big(\mathbf{B}_{i}^{[l]} + \sum_{j} \mathbf{W}_{i,j}^{[l]} * \mathbf{A}_{j}^{[l-1]}\Big) \tag{3.20}$$

The $*$ operator in Eq. (3.20) indicates a *convolution* operation. $\mathbf{W} * \mathbf{A}$ repeatedly computes the sum of the point-wise product while shifting $\mathbf{W}$ over $\mathbf{A}$, first horizontally until the end of the row is reached and then vertically and to the left side of the next row. The edge of matrix $\mathbf{A}$ may be padded with zeros to preserve its dimensions after convolution with $\mathbf{W}$. Thus, for a convolution operation of a kernel on an input matrix with dimensions $n_1 \times n_2$ and padding p, convolved with filters with dimensions $f_1 \times f_2$, the output matrix has dimensions $n_1 + 2p - f_1 + 1 \times n_2 + 2p - f_2 + 1$.

Pooling Layers Pooling layers are used to subsample the feature maps after convolution. Max pooling retains the maximum value in a section of non-overlapping cells of a feature map, while average pooling computes the average. Maximum and average pooling layers are convolutional operations that do not have weights to optimize, and do not have an activation function. They are fast to compute, and can help downsize feature maps to reduce the number of parameters required later on in the network.

Activation Functions Activation functions are applied to the linear combination of inputs computed for each unit of the neural network.

Most layers use a *Rectified Linear Unit* (ReLU) activation function, given by $h(\cdot) = max(0, \cdot)$. First introduced in [72], this activation function leaves the output of the linear unit unchanged for positive input, but sets negative values to zero. This allows networks with ReLU activations to model nonlinear relations for negative inputs. Modeling nonlinear relationships is beneficial for neural network classification because nonlinear relationships may be necessary to learn complex dependencies in the data.

The *softmax function* is used to transform the output of the hidden layers of the network to a probability estimate for each classification label. The output of a softmax activation function is a probability distribution that represents the probability of each label. For a classification problem with C classes, the softmax function computes the exponential function for the k^{th} unit $\mathbf{a}_k$ of $\mathbf{a}$, normalized over the sum of the exponential function for $\mathbf{a}_k$ for all outputs k. The expression for the softmax activation function is given in Eq. (3.21). As with any probability distribution, the resulting output values $\mathbf{a}_k$ always sum to 1.

$$h(\mathbf{a}_k) = \frac{\exp(\mathbf{a}_k)}{\sum_{k=1}^{C} \exp(\mathbf{a}_k)} \tag{3.21}$$

Chapter 4

Methods

This section describes the method used to train a model to estimate tablature, as well as the proposed adaptations to this approach. Section 4.1 gives a mathematical formulation of the problem and introduces the representations for the features and labels. A mathematical description of the method used for modeling and optimization is given in Section 4.2. Section 4.3 describes the proposed adaptations and their motivation.

4.1 Task Description

To describe the task of estimating string-fret combinations, we specify the problem setting and describe the representations used for the features and labels. The feature and label representations and problem formulation described are the same as in the approach used by Wiggins & Kim [114]. The notation $[a : b]$ is used to denote a range of integers from a to b.

4.1.1 Problem Specification

The problem of guitar tablature estimation may be formulated as a set of multi-class classification problems, where the aim in each classification problem is to predict the correct fretting on a string on the guitar for time-steps of an audio signal. Because of the large combinatorial output space produced by all possible string-fret combinations across strings, predicting the fretting for each string is treated as an individual classification problem. Each training instance represents a segment of the input signal obtained by time-frequency transformation. The classes to predict for each training instance is the fret class for every string at that instance. The fret classes are defined as all playable positions on the string, plus an additional class for a string not being played. A pitch may be produced by multiple string-fret combinations on the guitar.

Feature Representation Given is a digitized audio signal vector $\mathrm{x} \in \mathbb{R}$ of variable length consisting of amplitude samples at a fixed sample rate. We compute the constant-Q transform described in Section 3.1.2 of this signal to produce a time-frequency spectrogram $\mathrm{X} \in \mathbb{C}^{K \times N}$. K represents the chosen number of spectral bins for the CQT and N the number of time-steps after applying the CQT. N is derived from the proportion of the number of samples in x and the

hop length used for the CQT transform. Computing the element-wise magnitude of X produces a magnitude spectrogram $\mathrm{X} \in \mathbb{R}^{K \times N}$. An example of a CQT magnitude spectrogram from the data set is shown in Fig. 6a.

The inclusion of temporal context can help the model predict the correct string-fret combinations [31,112]. Therefore, sections of X centred around each time-step are obtained using a sliding context window and used as input. Let $i \in [0 : N]$ and W be the width of the context window, and the input to the model. Then the input spectrograms $\mathrm{X}_i \in \mathbb{R}^{K \times W}$ are created by extracting the columns of spectrogram X in the index range of $[\, i - (W-1)/2 : i + (W-1)/2 \,]$. To allow $i \in [0 : N]$, X may be padded with columns of zeros at the start and end, ensuring that each input has the same dimensions and the full sequence of data in X can be used. Thus, X_i represents a slice of the spectrogram of the recording centred around instance i. This step is visualized in Fig. 6a and Fig. 6b, where Fig. 6b shows a conceptual representation of the time-frequency spectrograms X_i resulting from the aggregation step. The window allows some of the temporal context of spectral information in the preceding and following time-steps to be input into the model.

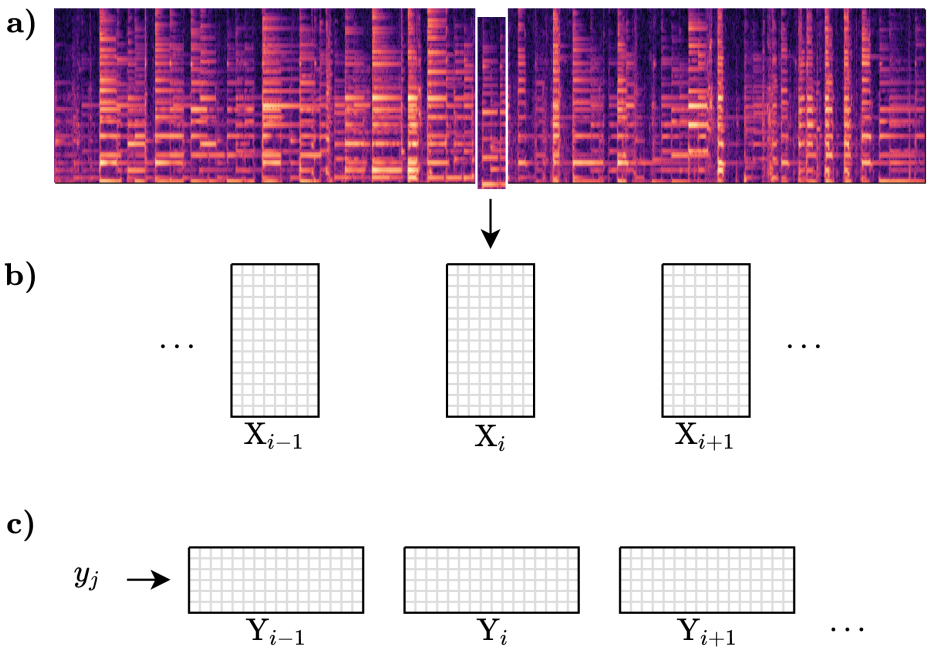


Fig. 6: Process diagram showing the extracted feature and label representations. A slice of the CQT spectrogram (a) is taken to create features X_i (b). c) shows a conceptual representation of the matrix label representation Y_i.

Label Representation The ground-truth annotations used to train the model are continuous in the time dimension and contain a separate fret annotation for each string. Therefore, a straightforward approach to obtaining labeled pairs of training data is to sample the annotations at the time-steps at which the constant-Q transform was calculated to produce spectrogram X. Sampling these annotations produces a set of MIDI pitch annotations that give the annotation for every guitar string $j \in [0:5]$ at i. Each annotation may be formatted as a sparse binary vector $\mathrm{y}_{i,j} \in \{0,1\}^{21}$ that stores the fret position being played on string j at i. Using one-hot encoding and assuming 20 playable positions on each string, each unit of $\mathrm{y}_{i,j}$ denotes a playable fret position, plus an additional unit for string j not being played at i. Thus, $\mathrm{y}_{i,j}$ denotes the ground-truth label for string j at the i^{th} instance of X_i. Only one value may be active per row for each instance, as only a single note may be played on each string at any time. Although the amount of playable frets varies between strings and different guitars, a reasonable restriction is to assume 20 playable fret positions, plus one 'non-playing' position. The vectors $\mathrm{y}_{i,j}$ at instance i may also be visualized as rows of a binary matrix $\mathrm{Y}_i \in \{0,1\}^{6\times 21}$, where the rows represent the label for each string and the columns the possible fret classes. This representation matches the visualization used in Fig. 2 in the Introduction and is shown in Fig. 6c.

Classification Problem Deriving the features and labels as described, we obtain a time-frequency representation matrix X_i and class label vectors $\mathrm{y}_{i,j}$ across strings j for each time-step i of a recording. The aim is to model the probability distributions $p(\mathrm{y}_{i,j}|\mathrm{X}_i)$, which describe the probability of each fret label of $\mathrm{y}_{i,j}$, conditional on X_i. We use $\hat{\mathrm{y}}_{i,j} \in \mathbb{R}^{21}$ to denote vectors of predicted probabilities output by the model for each fret class of string j at instance i. Thus, the aim is to train a model to predict probability vectors $\hat{\mathrm{y}}_{i,j}$ for each string j such that the prediction matches the ground-truth label $\mathrm{y}_{i,j}$. While the provided explanation assumes a single input sequence x to simplify notation, the features and labels are prepared across multiple audio tracks as described and then aggregated to produce a sufficiently large training set.

4.2 Model

The model proposed by Wiggins & Kim is a Convolutional Neural Network that takes a section of a spectrogram as its input and estimates the probability of each label for each string of the guitar for a given time-step. The same model is used as a baseline and adapted in this thesis. For an input X_i, the model computes a probability vector $\hat{\mathrm{y}}_{i,j}$ for each string j. To obtain the probability vectors, the final output layer of the network is computed, truncated into separate vectors and passed through a set of separate softmax activation functions, referred to as a *6-D Softmax* by the authors [114]. Thus, the model learns to simultaneously generate all string annotations, and is therefore able to learn complex dependencies between the strings and does not assume that each classification problem

is statistically independent. To achieve this, the categorical cross-entropy loss introduced in Section 3.2.3 is computed for each string j and summed to train the model to solve the classification problems in unison.

4.2.1 Loss Function

For each batch, the loss of the network is computed using the loss function introduced by Wiggins & Kim [114]. First, the elements of $\mathrm{y}_{i,j}$ and $\hat{\mathrm{y}}_{i,j}$ are multiplied and the sum of the resulting values is computed. The results are summed across all strings j, after which the normalized sum is computed across the entire batch. Eq. (4.1) shows the loss function $L(\theta)$, where the model parameters are denoted as θ and the batch size parameter m determines the accuracy of each gradient descent update [15]. For each string j, $\mathrm{y}_{i,j}$ denotes the ground-truth label vector and $\hat{\mathrm{y}}_{i,j}$ the vector of predicted probabilities for the i^{th} sample of the batch. The element-wise multiplication and sum are notated as a dot product in Eq. (4.1) so there is no need for an additional indexer to denote each entry of vectors $\mathrm{y}_{i,j}$ and $\hat{\mathrm{y}}_{i,j}$. The model parameters θ are updated before computing the loss over the next batch.

$$L(\theta) = -\frac{1}{m}\sum_{i}^{m}\sum_{j}\mathrm{y}_{i,j}\cdot\log\hat{\mathrm{y}}_{i,j} \tag{4.1}$$

4.3 Adaptations

This section describes the adaptations made by the author to the tablature estimation model proposed by Wiggins & Kim. The motivation and hypothesis for the application of each adaptation are discussed, as well as the expected outcomes of its application.

4.3.1 Musically Motivated Architectures

In the model proposed by Wiggins & Kim, square 3×3 kernels are used to detect patterns in the spectrogram input. Square kernels are standard in image recognition networks [58,57,29], as such kernels are able to recognize patterns along any orientation and make no *a priori* assumptions about the patterns they aim to detect. However, the use of different kernel shapes to improve the performance of CNNs in the domain of AMT has been investigated in the literature. Pons *et al.* adjusted the architecture of a CNN in a musical genre classification task to learn generalizable musical concepts [82,83,95]. They investigated the use of longer kernels to improve detection of spectral patterns along the frequency axis and the temporal axis. They found that such kernels improved the performance of a CNN for genre classification. To explain this result, they hypothesized that these kernels are able to detect more spread-out patterns in the time and frequency domain, capturing characteristics of pitch, timbre and rhythm more effectively than square kernels.

The upper harmonics of tones produced on acoustic instruments are spaced in integer multiples of the fundamental frequency, which results in a characteristic pattern of spectral lines. Figure 7 shows the log-scaled spectrogram for one of these notes with a pitch axis. The use of a log-scaled frequency axis makes this pattern symmetrical across all fundamentals that can be produced on the guitar. A CNN kernel of sufficient length along the frequency axis would be able to detect such a pattern in a CQT spectrogram, and may also be more easily able to disentangle several concurrent notes by their overtone pattern. Configuring a CNN to be able to specialize in detecting such patterns could improve model performance either in raw pitch prediction or in tablature estimation. Since the models are scored individually on pitch estimation and tablature transcription, the model's ability to capture characteristics of pitch, relating to pitch classification accuracy, and timbre, relating to string-fret classification accuracy, may be assessed separately.

The 3×3 kernels used in the original model proposed by Wiggins & Kim may be a limiting factor in the model's ability to detect patterns in the input. Thus, we evaluated model architectures with longer kernels along the frequency axis to allow the model to learn configurations of weights that detect longer spectral patterns. Several filter sizes used in the spectral kernel variations are shown in Fig. 7 to give an impression of the size of the pattern they are able to capture. Note the activation of spectral bins on the overtones (C3, C4, G4, C5, E5...) in the figure. While not explored in [82], we also test spectral kernels in layers after the input layer to detect patterns in broader sections of the input. Temporal filters are not explored here, as the temporal context window, which includes only a few time-steps of the CQT spectrogram, is presumed to be too short to detect relevant temporal patterns. This adaptation is expected to yield an improvement in pitch estimation. If longer frequency filters are able to help disentangle polyphonic signals or help differentiate between different strings, an improvement on tablature estimation may also be observed.

4.3.2 Fully Convolutional Architecture

The baseline architecture uses convolutional layers in the initial layers of the network and fully connected layers in the last two layers preceding the final softmax activation. Using fully connected layers towards the end of the network stack has been standard for CNNs in image recognition tasks [57]. Convolutional layers are used in the early layers of the network to handle the large input size, as they employ *parameter sharing* to reduce the number of parameters required to train a model and maintain training efficiency. This is useful because machine learning methods are subject to restrictions of time and computational power, meaning efficiency is an important factor in training models. Additionally, the initial layers of the network benefit strongly from this reduced parameter size and translation invariance property of convolutional layers. The fully connected layers towards the end of the network require many more parameters than convolutional ones, but are able to model more complex logical relationships and receive a downsized input from the preceding layers.

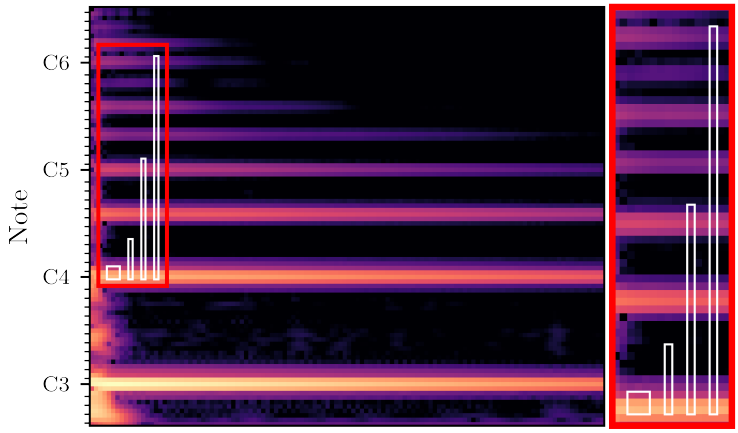


Fig. 7: Section of the CQT spectrogram of the note C3 played on a guitar shown in Fig. 5d. The white boxes show the dimensions of a 3×3, 1×9, 27×1 and 50×1 filter on the input spectrogram. The section within the red box is shown in a larger view on the right.

In [54], a fully convolutional architecture was used for a musical chord recognition task. By using convolutional layers to downsize the input to the required dimensionality for generating the class labels, the network does not require any fully connected layers to generate its predictions. Since this approach seems promising for a chord tracking task, it may have potential in a tablature estimation context as well. In order to adapt the CNN component of the classification architecture presented in [54], the quantities and dimensions of the layers were adjusted to fit the tablature transcription problem. Adapting the baseline architecture to a fully convolutional architecture allows for the investigation of whether such a fully convolutional architecture is sufficiently complex to capture in the tablature estimation task. Reducing the number of parameters without negatively affecting the predictive power of the model would result in a shorter training time, which can be beneficial for hyperparameter optimization procedures. Comparing the performance to the baseline also yields insight into whether the fully connected layers offer added value in the tablature estimation task.

4.3.3 Oracle Method

An additional adaptation of the model was given the additional input of the chord labels included with GuitarSet as an additional feature. While such an architecture serves no direct practical purpose for annotation (as a recording to be annotated is unlikely to have time-aligned chord labels prepared), the aim of this approach is to investigate the effect of including musical domain information, and to test to what degree the model can improve using additional

information. The fully connected layer should be able to model the influence on the probability of each string-fret combination, improving the performance by utilizing this information. This is especially true for solo recordings, where harmonic information is difficult to deduce without sufficient temporal context [62].

The dataset contained two sets of chord annotations. One of these contained basic chord annotations with major, minor, dominant of half-diminished variants used as instructions for the guitarists during recording. The other more complex representation was machine-annotated and manually checked, and included information on chord extensions played in the guitarists' improvisation. The former was chosen as input, as it was assumed that the chord fundamental and colour provide the model with additional information on the distribution of the target for a particular frame, without the need for extension-related information obtained using machine transcription. An important assumption that supports this approach is that the dense layers of the neural network can learn the harmonic relationships between chord symbols and note activations. Given that these relationships are usually dependent on consonance with the root and fit within the associated scale of a chord, this is assumed to be the case. Since musicians tend to improvise within the scale associated with the chord being played, even a shallow neural network should be able to estimate the probability distribution of the notes given a chord symbol as input.

4.3.4 Data Augmentation

The collection of annotated training data is a common bottleneck in deep learning-based AMT systems [11,16]. Since CNNs require a large amount of data to train and are sensitive to biases in training data [107,106,4], the investigation of methods of generating additional synthetic data for deep learning-based AMT systems is a valuable direction of research. Synthetic data refers to data generated by procedural means, such as adapting source data, rather than collected through measurement. A popular method of synthetic data generation in deep learning is data augmentation, which refers to generating additional data via transformations that preserve the ground-truth label [66,68]. Data augmentation is commonly applied to computer vision problems, such as by applying shifts, rotations and mirroring to image data to artificially generate additional instances [57]. In the context of music data, various techniques such as pitch shift and time stretching have been applied in transcription and instrument recognition tasks [16,68,92]. Data augmentation by pitch shift was found to be helpful in singing voice detection [92], instrument classification [16] and musical genre classification [68], but also in tasks involving pitch estimation, such as melody extraction [55,104]. These applications indicate that applying a label-preserving pitch shift-based data augmentation strategy could improve the performance of the Wiggins & Kim model.

Pitch shifting audio in semitones is referred to as *transposition*, and allows the classification label to be preserved. Applying a transposition-based data augmentation is expected to have a number of important effects on model performance.

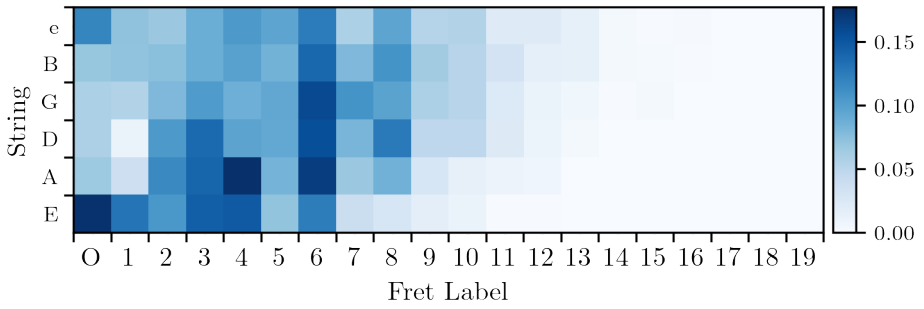


Fig. 8: Distribution of the string-fret combinations in the training data, normalized by row (string). The horizontal axis displays the fret labels with the closed position omitted, and the rows represent guitar strings.

The transposed instances should work to reinforce the model's ability to learn shift-invariant patterns in the input spectrograms. Detecting the same pattern in various positions should help the model learn a set of kernel configurations that are able to detect patterns associated with single notes or combinations of notes. The note combinations represented in the training data are limited compared to the full scope of note combinations that are commonly played on the guitar. Convolutional neural networks in AMT seem to have difficulty generalizing to unseen combinations of notes compared to combinations they have seen before [51]. Therefore, reinforcing the network's ability to detect specific combinations in various locations is an important aspect of the tablature estimation challenge. Additionally, the network's dense layer component is prone to overfitting to a certain part of the pitch spectrum. This was suggested to be an important reason to perform data augmentation for CNNs in AMT in [46]. Thus, the pitch shift data augmentation procedure is expected to improve generalization and reduce overfitting of the network.

An additional consideration is the distribution of class labels in the training data. Any data set for AMT with real guitar playing will likely contain imbalances in terms of the distribution of string-fret combinations played. Fig. 8 shows the aggregated distribution of the tablature annotations of sounding notes in the training set, visualized to resemble the layout of the fretboard. The fret positions are plotted along the horizontal axis, with 'O' representing the open playing position. The closed position, which represents a string not being played, was omitted to highlight the distribution of the other labels. The rows of each plot represent the six strings on the guitar. The values have been normalized across each row, and thus give the proportional frequency for each note by string in the training set. The plot shows that certain notes are played far less frequently than others, such as the 1st fret of the G string. In particular, notes that are high on the fretboard (far to the right) are played less frequently than lower notes, to the extent that a portion of the fretboard is almost never played. The distribution of notes also differ significantly between chord accompaniment recordings

and guitar solo recordings, although this is not visible in Fig. 8. This is particularly relevant given the aforementioned difficulty for CNNs to recognize the same notes in different combinations [51].

The data augmentation process involved two important steps: Transposing the audio and transposing the ground-truth annotations. The audio was transposed by changing the sample rate of the source audio by a factor associated with the desired transposition range before resampling to the original rate [103]. This produced transposed versions of the original recordings of the same length and number of samples as the original. Transposing the annotations proved more complicated. While shifting each fretting up by the desired number of semitones was trivial, many transpositions would produce frettings that were not playable on the guitar fretboard. To handle this limitation, only upward transpositions were used to augment the training data, as downward transpositions would make open string positions impossible to play without changing the string on which that note was played, producing too many invalid combinations. For transpositions that exceeded the upper bound of the fretboard, these frames were excluded from the training set. This meant that for large upward transpositions, only a subset of the transposed frames would be used in the training data.

Chapter 5

Evaluation

5.1 Experimental Setup

This section describes the experimental setup used to evaluate the proposed adaptations to the model. Table 4 at the end of the section gives a summary overview of the experiments.

5.1.1 Data Set

The models were trained on GuitarSet[9], the same data set that was used by Wiggins & Kim in [114]. GuitarSet consists of recordings of solo acoustic guitar performances by 6 guitarists, played on an acoustic guitar [115]. The guitarists were asked to improvise over 30 simple tune arrangements, created by combining three chord progressions, five styles and two tempi. The guitarists each recorded a solo and an accompaniment for every sheet, for a total of 360 recordings in the dataset. The datasets were annotated using a hexaphonic pickup, which enabled a fast, partially automated transcription procedure with a higher accuracy than manual transcription [115].

5.1.2 Data Preprocessing

Following the procedure in [114], the recordings were first downsampled from 44,100kHz to 22,050kHz and normalized. A 192-bin constant-Q transform with 24 bins per octave and a hop length of 512 samples was used to generate series of spectral vectors at around 43 samples per second from each recording. A window size of 9 was used for the temporal windowing used to generate the time-frequency spectrogram, meaning each instance contained information relating to the 4 preceding and 4 following time-steps. In cases where there were no preceding or following time-steps, the spectrograms were padded with zero-vectors. These conditions matched those described by Wiggins & Kim in order to enable a comparison [114]. The length of the recordings ranged from 14.4 to 45.7 seconds, producing between 620 and 1968 frames after applying CQT.

The ground-truth labels were retrieved from the `.jams`[10] files provided with GuitarSet to obtain 6 annotations for each time-step (one for each string). The `.jams` files give the MIDI pitch values and durations for each pitch played per

[9] `https://guitarset.weebly.com/`

[10] `https://jams.readthedocs.io/`

string. These were sampled to retrieve the ground-truth labels for the spectrograms produced by the CQT. To process the pitch annotations, the lowest playable MIDI pitch on each string is subtracted from its annotated values to translate them. There are 21 classes for each string: 20 to model the fret positions the guitarist can play, and one to denote a 'closed' string (a string not being played).

5.1.3 Data Augmentation Experiments

Four new data sets were created to run the data augmentation experiments. These contained added instances from transposing the training data upward by 1, 2, 3 and 11 semitones. Table 1 gives the number of instances in each augmented data set, as well as the number of instances that were dropped because they resulted in unplayable string-fret combinations after transposition. Transpositions of the audio recordings were performed using the librosa module[11] in Python.

Table 1: Number of instances and removals in each augmented data set.

	Base	**+1**	**+2**	**+3**	**+11**
# instances	472,559	943,631	1,415,513	1,887,105	5,370,130
# removed	0	1487	2164	3131	300,578

5.1.4 Model Adaptation Experiments

This section describes the parameters and topologies evaluated to test the approaches described in Section 4.

Baseline Model The model proposed by Wiggins & Kim [114] was used as a baseline for comparison. Table 2a describes the topology and parameters of the network. The convolutional layers use a rectified linear unit activation function (Conv-ReLU). The result is passed through a max pooling layer (Pool-Max) and two fully connected layers (Dense-FC) to connect to the final layer. The final layer (6-D Softmax) computes the probability of each fret class per string by reshaping output of the previous layer (126 units) to 6 vectors of 21 units. The connection dropout was set to occur with a probability of 0.25 and 0.5 for the max pooling layer and the fully connected layer respectively. The base model has a total of 833,982 parameters.

[11] `https://librosa.org/`

Musically Motivated Architectures Three adapted model architectures were tested that were inspired by the Musically Motivated Architectures approach taken in [82]. The design of these architectures are given in Tables 2b to 2d. Filters of sizes 50×1, 27×1 and 9×1 were added at various points in the baseline architecture. The span of the first two filter sizes is shown in Fig. 7, indicating the range of overtones they are able to capture from the fundamental. While longer filters along the temporal axis are explored in the aforementioned references, the temporal resolution of the CQT diagrams used as model input was considered too fine to expect characteristic musical patterns to be detectable. A single time-frequency spectrogram spans only 9 samples, or roughly 209ms. Thus, temporal filters were not investigated for this application.

Fully Convolutional Architecture The fully convolutional architecture was implemented with a similar architecture to the model presented in [54]. Table 3 shows the architecture of this network. Some adjustments were required to fit the data dimensions for CQT diagram input and fretboard label output for the tablature annotation task. Additional convolution layers and an average pooling layer (Pool-Avg) were added to obtain the desired output dimensions for the 6-D softmax. Omitting the fully connected layers reduces the number of parameters to 333,470, as convolutional layers usually require far less parameters than fully connected layers.

Oracle Method The chord annotations read by the guitarists during the recording of GuitarSet were encoded as one-hot vectors of length 48. Four types of each chord were used (major, minor, half-diminished and diminished), resulting in 48 possible chords across all 12 keys. These vectors were concatenated to the output of the first fully-connected layer of the baseline network (Table 2a), changing the output size of this layer from 128 to 176. This increased the number of parameters in the following dense layer from 16,254 to 22,352.

5.1.5 Data Subset Experiments

The initial results obtained from running the experiments deviated from expectation and the hypotheses stated in the Methods section, which warranted additional investigation of the effect of the size of the training set. To investigate this effect, additional training runs and evaluations were conducted by incrementally increasing the number of recordings used to train the model. The size of the training set was increased by increments of 30 recordings to investigate whether the trend would indicate a need for a larger data set to improve on pitch or tablature estimation.

5.1.6 Training Setup

Training runs were conducted under the same conditions as in [114], using the ADADELTA optimization algorithm [120] with an initial learning rate of 1.0

Table 2: Architecture of the baseline model, implemented as in [114], and three Musically Motivated Architectures. The horizontal table lines represent network dropout.

(a) Baseline model

Layer Type	**Params**	**Output Size**
Input		192×9
Conv-ReLU	$32 \times 3 \times 3$	$32 \times 190 \times 7$
Conv-ReLU	$32 \times 3 \times 3$	$64 \times 188 \times 5$
Conv-ReLU	$32 \times 3 \times 3$	$64 \times 186 \times 3$
Pool-Max	2×2	$64 \times 93 \times 1$
Dense-FC	761,984	128
Dense-FC	16,254	126
6-D Softmax		6×21

(b) Musically Motivated Architecture 1

Layer Type	**Params**	**Output Size**
Input		192×9
Conv-ReLU	$32 \times 27 \times 1$	$32 \times 166 \times 9$
Conv-ReLU	$64 \times 9 \times 1$	$64 \times 158 \times 9$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 156 \times 7$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 154 \times 5$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 152 \times 3$
Pool-Max	2×2	$64 \times 76 \times 1$
Dense-FC	622,720	128
Dense-FC	16,254	126
6-D Softmax		6×21

(c) Musically Motivated Architecture 2

Layer Type	**Params**	**Output Size**
Input		192×9
Conv-ReLU	$32 \times 3 \times 3$	$32 \times 190 \times 7$
Conv-ReLU	$64 \times 50 \times 1$	$64 \times 141 \times 7$
Conv-ReLU	$64 \times 9 \times 1$	$64 \times 133 \times 7$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 131 \times 5$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 129 \times 3$
Pool-Max	2×2	$64 \times 64 \times 1$
Dense-FC	524,416	128
Dense-FC	16,254	126
6-D Softmax		6×21

(d) Musically Motivated Architecture 3

Layer Type	**Params**	**Output Size**
Input		192×9
Conv-ReLU	$32 \times 3 \times 3$	$32 \times 190 \times 7$
Conv-ReLU	$64 \times 50 \times 1$	$64 \times 141 \times 7$
Conv-ReLU	$64 \times 27 \times 1$	$64 \times 115 \times 7$
Conv-ReLU	$64 \times 9 \times 1$	$64 \times 107 \times 7$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 105 \times 5$
Conv-ReLU	$64 \times 3 \times 3$	$64 \times 103 \times 3$
Pool-Max	2×2	$64 \times 51 \times 1$
Dense-FC	417,920	128
Dense-FC	16,254	126
6-D Softmax		6×21

Table 3: Fully convolutional model architecture based on [54]. The horizontal table lines represent network dropout.

Layer Type	**Params**	**Padding**	**Output Size**
Input			192×9
Conv-ReLU	$32 \times 3 \times 3$	Yes	$32 \times 192 \times 9$
Conv-ReLU	$32 \times 3 \times 3$	Yes	$32 \times 192 \times 9$
Conv-ReLU	$32 \times 3 \times 3$	Yes	$32 \times 192 \times 9$
Conv-ReLU	$32 \times 3 \times 3$	Yes	$32 \times 192 \times 9$
Pool-Max	2×1		$32 \times 96 \times 9$
Conv-ReLU	$64 \times 3 \times 3$	No	$64 \times 94 \times 7$
Conv-ReLU	$64 \times 3 \times 3$	No	$64 \times 92 \times 5$
Pool-Max	2×1		$64 \times 46 \times 5$
Conv-ReLU	$64 \times 12 \times 3$	No	$64 \times 35 \times 3$
Conv-ReLU	$64 \times 23 \times 1$	No	$64 \times 13 \times 3$
Conv-Linear	$126 \times 1 \times 1$	No	$126 \times 13 \times 3$
Pool-Avg	3×13		126
6-D Softmax			6×21

and a mini-batch size of 128 samples. Training was conducted over 8 epochs to provide sufficient training time without overfitting. Evaluations were conducted using 6-fold cross-validation. During each fold, the data from one guitarist was withheld for evaluation, and the scoring metrics were computed over this test set during training. The scores presented in the Results section are the average scores within each metric over all 6 folds. All experiments were conducted on a system with an AMD Ryzen 7 2700 processor and a Nvidia GeForce 1070Ti GPU. Training took around 8 hours for the base data set and roughly 24 to 80 hours for the augmented sets.

5.2 Evaluation Metrics

This section describes the metrics used to evaluate the models. These evaluation metrics were originally described in [114] and are calculated over the test set withheld during each cross-validation fold. For each fold, the data from one guitarist in the data set is withheld as a test set. All metrics are in the range of 0 and 1, with 1 being the best value and 0 being the worst.

5.2.1 Multi-pitch Estimation Metrics

The multi-pitch estimation metrics measure the correctness of the pitch predictions without considering the additional step of tablature estimation. The models are scored on their ability to assign the correct pitch to each note, but

Table 4: Summary table with experiments, abbreviations and descriptions.

Experiment	**Abbr.**	**Description**
Data Augmentation	DA	Training the baseline model on a data set augmented by transposition (4 variants)
Musically Motivated Architectures	MMA	Training models with longer kernels along the frequency axis to detect spectral patterns (3 variants)
Fully Convolutional Architecture	FCA	Training a model with only convolution and pooling layers to reduce the number of parameters
Oracle Method	OM	Inputting chord annotations to the dense layer component of the model
Subsets		Evaluating models trained on smaller subsets to investigate the effect of training set size

not on whether they assign each note to the correct string and fret position on the guitar.

In Eqs. (5.1) to (5.3), P denotes a binary matrix of size $N \times 44$, which has a row for each of the N instances in the test set and a column for each pitch that can be played on the guitar. P_{gt} denotes the matrix with the ground-truth labels and P_{pred} the labels predicted by the model. The $\odot$ symbol denotes the element-wise product of two matrices, and e a vector of all ones.

Multi-pitch Precision The multi-pitch precision (p_{pitch}) gives the proportion of correctly predicted pitches relative to the total number of predicted pitches. This metric indicates how often the model correctly identifies the pitch of a note played on the guitar.

$$p_{pitch} = \frac{e^T(\mathrm{P}_{gt} \odot \mathrm{P}_{pred})\, e}{e^T\, \mathrm{P}_{pred}\, e} \tag{5.1}$$

Multi-pitch Recall The multi-pitch recall (r_{pitch}) gives the total number of correctly predicted pitches relative to the total number of pitch annotations in the data set. This metric indicates how many of the pitch annotations in the test set are retrieved by the model.

$$r_{pitch} = \frac{e^T(\mathrm{P}_{gt} \odot \mathrm{P}_{pred})\, e}{e^T\, \mathrm{P}_{gt}\, e} \tag{5.2}$$

Multi-pitch F-measure The multi-pitch F-measure (f_{pitch}) is the harmonic mean of the multi-pitch precision and multi-pitch recall. It serves as a summary metric of a model's multi-pitch estimation performance.

$$f_{pitch} = \frac{2p_{pitch}\, r_{pitch}}{p_{pitch} + r_{pitch}} \tag{5.3}$$

5.2.2 Tablature Estimation Metrics

The tablature estimation metrics measure the correctness of the string-fret combinations predicted by the model. Annotations are considered correct if the predicted string-fret combination matches the provided ground-truth label. The tablature scores are strictly lower than pitch scores, as any correctly identified string-fret combination must also be correctly classified in terms of pitch.

In Eqs. (5.4) to (5.6), Q denotes a binary matrix of size $N \times 120$, which has a row for each of the N instances in the test set and a column for each possible fret annotation for each string (excluding the label that indicates a string not being played). Q_{gt} denotes the matrix with the ground-truth labels and Q_{pred} the labels predicted by the model.

Tablature Precision The tablature precision metric (p_{tab}) gives the proportion of correctly predicted string-fret combinations relative to the total number of string-fret combinations predicted by the model. This metric indicates how often the model correctly identifies a string-fret combination played on the guitar.

$$p_{tab} = \frac{e^T(\mathrm{Q}_{gt} \odot \mathrm{Q}_{pred})\, e}{e^T\, \mathrm{Q}_{pred}\, e} \tag{5.4}$$

Tablature Recall The tablature recall metric (r_{tab}) gives the proportion of correctly predicted string-fret combinations relative to the total number of string-fret combinations annotated in the data set. This metric indicates how many of the string-fret combinations in the test set are retrieved by the model.

$$r_{tab} = \frac{e^T(\mathrm{Q}_{gt} \odot \mathrm{Q}_{pred})\, e}{e^T\, \mathrm{Q}_{gt}\, e} \tag{5.5}$$

Tablature F-measure The tablature F-measure (f_{tab}) is the harmonic mean of the tablature precision and tablature recall metrics. This metric serves as a summary statistic for the model performance in the tablature estimation task.

$$f_{tab} = \frac{2p_{tab}\, r_{tab}}{p_{tab} + r_{tab}} \tag{5.6}$$

Tablature Disambiguation Rate The tablature disambiguation rate (TDR) indicates the proportion of correctly identified pitches that are also assigned to the correct string-fret combination. A high TDR indicates that the model is consistently able to correctly identify the string-fret combination used to produce a correctly identified pitch.

$$TDR = \frac{e^T(\mathrm{Q}_{gt} \odot \mathrm{Q}_{pred})\, e}{e^T(\mathrm{P}_{gt} \odot \mathrm{P}_{pred})\, e} \tag{5.7}$$

5.3 Results

The full results of the evaluations carried out to test the adaptations and strategies described in Sections 5.1.3 to 5.1.5 are presented in Tables 5 and 6. The tables give the mean and standard deviation of each experiment. Since one of the six guitarists was left out for each cross-validation fold, each experiment produced six values for the evaluation metrics from which the mean and standard deviation are derived. The withheld test sets per fold were identical between experiments, meaning any procedures that affected the number of instances used during training were applied only to the training data. Fig. 9 shows a visualization of the performance of the baseline, adapted model architectures and the large augmented data set experiment (11 transpositions). The plot gives a helpful overview of the results shown in Tables 5 and 6 and enables a quick visual comparison between each adaptation and the baseline. The results for each model are grouped per metric on the horizontal axis, where each point gives the mean and the vertical whiskers give the standard deviation.

5.3.1 Baseline Model

The scores of the baseline model did not deviate significantly from the scores reported in [114]. This indicates that the procedure used to train and test the baseline model presented in this thesis a good reproduction of the one used in the original paper, and that the experimental conditions were correctly maintained.

5.3.2 Data Augmentation

The results of the augmented data set experiments described in Section 5.1.3 are shown in Table 5. The best performing augmentation (in terms of F-score) was included in Fig. 9. The smaller augmentation strategies (+1, +2 and +3) did not affect model performance along any metric in the pitch or tablature domain, showing no significant deviation from the baseline. The largest augmentation (+11) had a pronounced negative impact on all metrics except pitch precision. In tablature estimation, the large augmentation by transposition produced negative results in both metrics for the large augmented set.

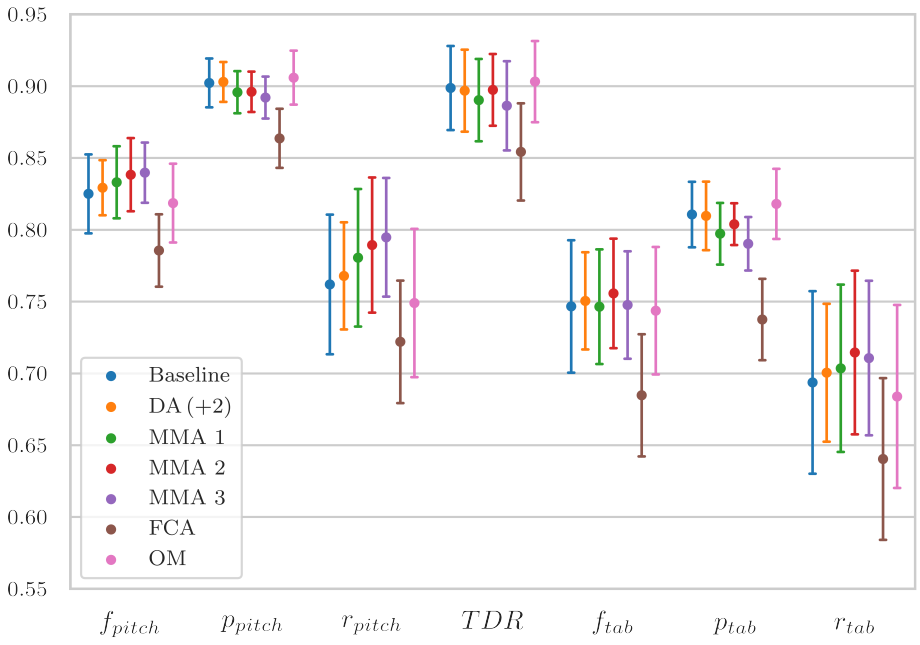


Fig. 9: Mean and standard deviation of the evaluation results for each experiment, grouped per metric along the horizontal axis.

Table 5: Results for augmented model experiments. The mean and standard deviation (between parentheses) are reported for each experiment. Bold values indicate the maximum value for a metric.

Experiment	f_{pitch}	p_{pitch}	r_{pitch}	TDR	f_{tab}	p_{tab}	r_{tab}
Baseline	0.825 (0.027)	0.902 (0.017)	0.762 (0.049)	0.899 (0.029)	0.747 (0.046)	0.811 (0.023)	0.694 (0.064)
DA (+1)	0.819 (0.031)	**0.909** (0.011)	0.747 (0.053)	**0.902** (0.027)	0.744 (0.405)	**0.819** (0.018)	0.683 (0.060)
DA (+2)	**0.829** (0.019)	0.902 (0.014)	**0.768** (0.037)	0.897 (0.029)	**0.750** (0.034)	0.810 (0.024)	**0.700** (0.048)
DA (+3)	0.819 (0.023)	0.907 (0.021)	0.748 (0.049)	0.895 (0.027)	0.739 (0.036)	0.812 (0.034)	0.681 (0.054)
DA (+11)	0.787 (0.039)	0.903 (0.015)	0.701 (0.064)	0.861 (0.027)	0.688 (0.044)	0.777 (0.022)	0.620 (0.066)

Table 6: Results for the adapted model architectures. The mean and standard deviation (between parentheses) are reported for each experiment. Bold values indicate the maximum value for a metric.

Experiment	f_{pitch}	p_{pitch}	r_{pitch}	TDR	f_{tab}	p_{tab}	r_{tab}
Baseline	0.825 (0.027)	0.902 (0.017)	0.762 (0.049)	0.899 (0.029)	0.747 (0.046)	0.811 (0.023)	0.694 (0.064)
MMA 1	0.833 (0.025)	0.896 (0.015)	0.781 (0.048)	0.890 (0.029)	0.746 (0.040)	0.797 (0.021)	0.704 (0.058)
MMA 2	0.838 (0.025)	0.896 (0.014)	0.789 (0.047)	0.897 (0.025)	**0.756** (0.038)	0.804 (0.015)	**0.715** (0.057)
MMA 3	**0.840** (0.021)	0.892 (0.015)	**0.795** (0.041)	0.886 (0.031)	0.748 (0.037)	0.79 (0.019)	0.711 (0.054)
FCA	0.786 (0.025)	0.864 (0.021)	0.722 (0.043)	0.854 (0.034)	0.685 (0.043)	0.738 (0.028)	0.640 (0.056)
OM	0.819 (0.027)	**0.906** (0.019)	0.749 (0.052)	**0.903** (0.028)	0.744 (0.044)	**0.818** (0.024)	0.684 (0.064)

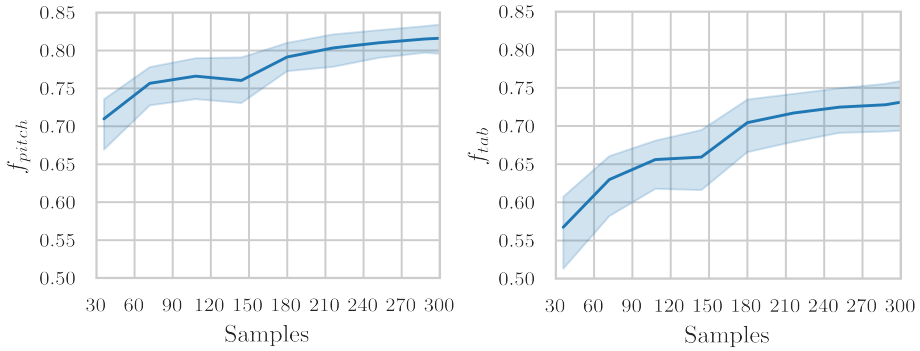
5.3.3 Musically Motivated Architectures

The three evaluated implementations of Musically Motivated Architectures given in Tables 2b to 2d offered a reasonable improvement over the baseline. In particular, the adaptations labelled MMA 2 and MMA 3 in Fig. 9 increased pitch recall and also increased pitch F-score, at the cost of only a small decrease in pitch precision.

Conducting a paired t-test between the measured F-scores of the Musically Motivated Architectures and the baseline over the six folds yielded p-values of 0.089, 0.012 and 0.024 for MMA 1, MMA 2 and MMA 3 respectively. For the tablature metrics, the p-values were 0.509, 0.115, and 0.461. The null hypothesis was that there was no significant difference between the samples, and the alternate hypothesis was that the mean score of the adaptations was higher than that of the baseline. Thus, at a significance level of $\alpha = 0.05$, MMA 2 and MMA 3 yielded a statistically significant improvement in raw pitch estimation. This indicates that the implementation of kernels that are longer along the frequency axis is beneficial to pitch estimation tasks.

5.3.4 Fully Convolutional Architecture

The fully convolutional architecture presented in Table 3 had a significant drop in performance across all metrics. Substituting dense layers in the model seems to have a detrimental effect in both pitch estimation and tablature estimation. This indicates that the dense layers play an important role in classification models for



(a) Pitch F-score

(b) Tablature F-score

Fig. 10: F-score for the baseline model over the number of training samples.

pitch and tablature estimation. Furthermore, it is unlikely that the reduction in parameters outweighs the loss in model performance of this approach.

5.3.5 Oracle Method

The model approach that included chord information did not outperform the original approach in any metric. A significant drop in pitch recall was observed, and observed improvements in other metrics were too small to be deemed significant. This is a surprising result, as the inclusion of additional harmonic information should make it easier for the model to identify the correct note. A small increase in pitch precision could indicate an improvement in misidentified notes in the pitch domain, but this result is too small to be deemed significant. Additionally, a slight drop in pitch recall was observed, which indicates that the inclusion of harmonic information does not improve the retrieval of notes present in the signal. Performance in terms of F-score did not significantly change based on this approach.

5.3.6 Subsets

The subset experiments discussed in Section 5.1.5 were conducted since most of the adaptations scored lower than expected. The adaptations were motivated by literature, and by observed limitations of the baseline model. The subset experiments were carried out to investigate whether the size of the training set could be the cause of the observed results.

The results obtained by varying the training set size for the baseline model are plotted in Fig. 10, which shows the F-score plotted over the number of recordings included for pitch (Fig. 10a) and tablature (Fig. 10b). A visual investigation of this trend shows that a significant improvement on both metrics would be possible given additional instances in the data set. In particular, the F-score in the tablature domain is still increasing when approaching 300 samples,

which indicates that a larger data set would provide a significant improvement in tablature estimation performance. The more pronounced slope for the tablature F-score indicates that tablature estimation is more sensitive to training set size limitations than performing pitch estimation.

Chapter 6

Discussion

The goal of this thesis was to investigate whether improvements in tablature estimation with a CNN could be achieved by adapting the structure of the network and the procedure used during training. Overall, the results obtained during the evaluations failed to provide a clear and consistent answer to this question. While the musically motivated architectures provided a modest improvement over square kernels, the other adaptations and augmentation strategy had either a neutral or adverse effect on predictions. An investigation into the effect of training set size showed that the number of training instances is an important limitation of this method. This section will discuss the results and implications of the experiments and present a rationale for the observed inconsistencies in the context of the observed limitations of data set size.

The musically motivated filters seemed to show promise for pitch estimation. The increase in recall and reduction in precision indicate that spectral kernels increase the model's sensitivity to patterns, reducing the number of missed detections at the cost of a small increase in false positives. The slight improvement in tablature estimation seems to be a consequence of the improved pitch estimation ability of the network rather than an improved ability to predict string-fret combinations. This notion is supported by the lower TDR of the musically motivated architectures, which indicates that while they classified more pitches correctly compared to the baseline, their ability to find correct string-fret combinations for these did not improve significantly. Furthermore, including chord annotation data, as implemented in the Oracle Method adaptation, also did not provide a significant boost to the model's ability to correctly estimate pitch or tablature. Given the results of these two adaptations, it is reasonable to assume that ability to detect spectral patterns is not a main limitation of the model, as adapting the network to improve this capacity offered only a modest benefit. Additionally, missed detections in the (polyphonic) context of guitar chords are also unlikely to be an important limitation of the model, since chord annotations provide a wealth of information about likely frettings used by a guitarist in the accompaniment recordings contained in GuitarSet. Thus, a failure to detect parts of the guitar chords played in the accompaniment tracks would be reflected by a significant increase in performance of the Oracle Method adaptation. These observations warranted further investigation into the importance of training set size and the possibility of creating additional synthetic instances to augment the training data.

To address limitations of data set size, a novel approach to data augmentation was applied. Data augmentation by pitch-shifting (transposition) was considered

a promising approach, as the translational nature of such an augmentation reinforces the architectural structure of the translation-invariant network when using log-scaled spectrograms as input [104]. Furthermore, it prevents overfitting of the network's dense layer component to a particular part of the pitch spectrum, as the dense layers are not shift-invariant [46]. This is particularly relevant in the context of the imbalance of string-fret combinations in GuitarSet described in Section 4 and in [27], since an unbalanced distribution of frettings may bias the network towards certain string-fret combinations. Data augmentation methods such as transposition are an important topic of investigation, since a successful method of creating additional synthetic data would be valuable in addressing limitations around data set size, since producing large data sets for AMT is costly and time-consuming.

The results of the data augmentation experiments carried out ranged from neutral to significantly negative. A number of explanations may be posited for this result. The broad transposition ranges may have introduced artefacts in the training data. This is particularly true of pitch shift which preserves the length of the input and the sample rate, requiring interpolation of samples for upward transposition [103]. The sharp increase in amount of training instances proportional to the number of model parameters may also have made it difficult for the model to improve its modeling of relationships in tablature estimation. The number of training epochs remained the same as for the baseline due to restrictions in computational resources, which may also have had an adverse effect given the increase in data set size. Additionally, the complex method by which training frames that could not be played on the guitar were removed may have introduced undesirable effects. However, the runs conducted with only smaller augmentations did not produce a better effect. The smaller sets did not require specific removal of frames, and was less prone to artefacts introduced by the transposition. Because it is difficult to ascertain the exact reasons for the observed results, no final conclusion can be drawn about its effectiveness as a strategy for Deep Learning in pitch or tablature estimation in general. Additional specific investigation would be required to assess whether data augmentation can be used to improve deep neural networks for AMT tasks. However, the results are consistent with the finding presented in the limited examples of transposition-based data augmentation for AMT in the literature, in that its potential seems limited compared to non-synthetic approaches to creating larger data sets.

The testing concerning the influence of data set size on performance yielded what is arguably the most important finding of this research. The size of the data set seems to be a key factor in the model's performance during evaluation, as limiting the amount of data showed that the model had not reached optimal performance with the amount of available training data. Hence, the findings of the other adaptations and strategies are also influenced by this limitation. The relationship between the number of parameters and training set size in Deep Learning is not fully understood, but the general trend is that a higher model complexity requires a larger training set [122]. Since many of the adaptations reduced the number of trainable parameters, this may also have been responsible

for the improvement in performance. In particular, two of the musically motivated architectures (MMA2 and MMA3) greatly reduced the parameter space and saw the largest increases in performance. Reducing the parameter count by substituting the network's dense layers appeared to have an adverse effect, meaning these are important for pitch and tablature estimation. In any case, it may be concluded from the training set size investigations that the amount of training data offered by GuitarSet is not enough to train a CNN of this complexity to optimal performance for tablature estimation.

The fact that GuitarSet is the only publicly available data set with high-precision string-fret annotations is likely to be a key limitation in any future implementations aimed at transcribing tablature. Furthermore, given that GuitarSet consists of recordings of six guitarists improvising over chord progressions, the range of styles and arrangements in its recordings is likely to be a limitation as well. This was illustrated by the model's inability to classify string-fret combinations that were not represented in the training data, such as in the example presented in Fig. 2 in Section 1. The generalizability of CNNs and their translation-invariant properties are unlikely to be sufficient in classifying every sequence of string-fret combinations that is playable on the guitar. Additionally, the challenge presented by requiring the model to transcribe string-fret combinations rather than pitches may not be entirely solvable by increasing the amount of training data or model complexity. The way in which human annotators apply their knowledge and intuition is difficult to define, and often involves a deep knowledge of the guitar, the possibilities and limitations involved in playing it and the historical tradition of guitar. While techniques like the inclusion of temporal context in CNNs attempt to emulate such intuitions about logical sequences on the guitar, modeling this knowledge remains an open challenge that likely requires further innovation to solve.

Chapter 7

Conclusion

The aim of this thesis was to yield insights into the functioning of CNNs for automatic music transcription, and to evaluate the performance of a set of adaptations to a state-of-the-art model. The baseline model, proposed by Wiggins & Kim [114], used a Convolutional Neural Network to predict string-fret combinations from log-frequency spectrograms, trained on annotated guitar recordings. While the system improved significantly on existing methods, several key shortfalls were identified by the authors and in the initial investigations described in Section 1. To improve on these shortfalls, four adaptations to the system were implemented and evaluated: A transposition-based data augmentation strategy, Musically Motivated Architectures, a fully convolutional architecture, and an oracle method. Additionally, the effect of training set size was investigated to see how this affected the performance of the model and the results of the evaluations of the adaptations.

The outcomes of the experiments yielded the following key findings:

- Data augmentation by transposition did not significantly improve pitch or tablature estimation, and is detrimental when carried out with large intervals
- The musically motivated architectures helped the model correctly estimate pitch, but did not significantly improve the model's capacity to predict the correct string for a pitch
- Using only convolutional layers had a significant negative impact
- Surprisingly, the Oracle Method adaptation, which gave the chord annotations used to produce the data set as input, did not benefit the model
- Incrementally increasing the amount of training data showed that more data would lead to an increase in performance

While the proposed adaptations did not perform as well as expected, the additional investigations into the limitation of data set size showed that this limitation likely had a significant effect on results of the other experiments. Therefore, the outcomes experiments should be placed in perspective of this data set size limitation. Nonetheless, the key findings had some important implications:

- The model is not limited by its ability to detect spectral patterns or analyze polyphony
- The potential of transposition-based data augmentation seems limited for tablature estimation
- Dense layers play an important role in tablature estimation CNNs of this nature

– The amount of training data offered by GuitarSet is not enough to train a CNN of this complexity to optimal performance for tablature estimation
– The size of the data set is a key factor in model performance

An additional limitation of the work was that evaluations were carried out on the original data set. Evaluating on samples outside of GuitarSet would have offered the opportunity to provide an evaluation of the model's ability to generalize to other data, especially given the proposed adaptations. However, no data set of sufficient precision and size was available at the time of writing this thesis, and manual annotation of training data would have been to labour-intensive for the author to feasibly perform as part of this project.

7.1 Future Work

The research presented in this thesis investigated a number of ways to improve the performance of a CNN for tablature transcription. However, there are many other ways in which this approach could be adapted that may yield benefits to the model. A notable example that has seen success in other AMT domains is the application of discriminative models that can take sequential information in the data set into account. These include recurrent variants of neural networks, such as Long-Short Term Networks (LSTMs) and Convolutional Recurrent Neural Networks (CRNNs). Such models have previously been applied to transcription, but have yet to be applied to the tablature transcription problem [62,123]. While the model proposed by Wiggins & Kim takes some temporal context into account, modeling temporal dependencies and musical language in sequences may bring benefits to the approach. State-of-the-art representations for audio data, such as the CQT representation used in this approach, may not be sufficient to capture the difference in timbre of the strings of the guitar. Therefore, investigating the modeling of musical language may prove more fruitful in approaching tablature estimation. The development of novel approaches to model the language of music has the potential to improve tablature estimation, but to yield broader insights into automatic music transcription and sequence analysis as well.

While the representation used audio data has been very consistent, and there are good reasons to use log-frequency spectrograms as input data, CNNs are very flexible in the type of input they can take. Therefore, a potential avenue of exploration for future neural network-based transcription approaches would be to compute additional features or representations to include in the CNN input.

The most key finding of the thesis may be regarded as that CNNs for tablature estimation require larger annotated data sets to improve their performance. Therefore, future research in tablature estimation should aim to tackle this limitation. The problem of data set size will require an original approach to solve, since manual annotation of large data sets is time-consuming and requires expertise, even when partially automating the process, and requires a high level of quality train fine granularity machine learning models. In addition, the effect of imbalances in the representation of chords, voicings and notes in a training set on

models for AMT remains an open problem. Future research could focus on ways to circumvent data set size limitations. For example, Few-Shot Learning (FSL) is a technique aimed at generalizing from a small number of examples, and is designed to help deal with situations where few training examples are available [113]. Alternatively, research could aim to facilitate the creation of larger data sets for AMT. An exemplary approach is presented by Byambatsogt *et al.*, who used robotics to apply a physical data augmentation by recording inversions and transpositions of a set of chords [27] . Such approaches to automation may have strong potential for creating larger data sets for AMT researchers to work with.

References

1. Abdallah, S.A., Plumbley, M.D.: Polyphonic Music Transcription by Non-negative Sparse Coding of Power Spectra. In: Proceedings of the 5th International Conference on Music Information Retrieval (ISMIR 2004). pp. 318–325. Barcelona, Spain (2004)
2. Abdallah, S.A., Plumbley, M.D.: Unsupervised Analysis of Polyphonic Music by Sparse Coding. IEEE Transactions on Neural Networks **17**(1), 179–196 (2006)
3. Akbari, M., Cheng, H.: Real-Time Piano Music Transcription Based on Computer Vision. IEEE Transactions on Multimedia **17**(12), 2113–2121 (Dec 2015). https://doi.org/10.1109/TMM.2015.2473702
4. Alvi, M., Zisserman, A., Nellåker, C.: Turning a blind eye: Explicit Removal of Biases and Variation from Deep Neural Network Embeddings. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops, ECCV 2018. pp. 556–572. Munich, Germany (2018)
5. Argenti, F., Nesi, P., Pantaleo, G.: Automatic Music Transcription: From Monophonic to Polyphonic. In: Musical Robots and Interactive Multimodal Systems, pp. 27–46. Springer (2011)
6. Ariga, S., Fukayama, S., Goto, M.: Song2Guitar: A Difficulty-Aware Arrangement System. In: Proceedings of the 18th International Conference on Music Information Retrieval (ISMIR 2017). pp. 568–574. Suzhou, China (2017)
7. Barbancho, A.M., Klapuri, A., Tardon, L.J., Barbancho, I.: Automatic Transcription of Guitar Chords and Fingering From Audio. IEEE Transactions on Audio, Speech, and Language Processing **20**(3), 915–921 (Mar 2012). https://doi.org/10.1109/TASL.2011.2174227
8. Basaran, D., Essid, S., Peeters, G.: Main Melody Extraction with Source-Filter NMF and CRNN. In: Proceedings of the 19th International Conference on Music Information Retrieval (ISMIR 2018). pp. 82–89. Paris, France (2018)
9. Bay, M., Ehmann, A.F., Downie, J.S.: Evaluation of Multiple-F0 Estimation and Tracking Systems. In: Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009). pp. 315–320. Kobe, Japan (2009)
10. Benetos, E., Dixon, S.: A Shift-invariant Latent Variable Model for Automatic Music Transcription. Computer Music Journal **36**(4), 81–94 (2012)
11. Benetos, E., Dixon, S., Duan, Z., Ewert, S.: Automatic Music Transcription: An Overview. IEEE Signal Processing Magazine **36**(1), 20–30 (Jan 2019). https://doi.org/10.1109/MSP.2018.2869928
12. Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H., Klapuri, A.: Automatic Music Transcription: Breaking the Glass Ceiling. In: Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012). pp. 379–384. Porto, Portugal (2012)
13. Benetos, E., Dixon, S., Giannoulis, D., Kirchhoff, H., Klapuri, A.: Automatic Music Transcription: Challenges and Future Directions. Journal of Intelligent Information Systems **41**(3), 407–434 (Dec 2013). https://doi.org/10.1007/s10844-013-0258-3
14. Bengio, Y.: Learning Deep Architectures for AI. Foundations and Trends® in Machine Learning **2**(1), 1–127 (2009). https://doi.org/10.1561/2200000006

15. Bengio, Y.: Practical Recommendations for Gradient-based Training of Deep Architectures. Neural Networks: Tricks of the Trade **7700**, 437–478 (Sep 2012). https://doi.org/10.1007/978-3-642-35289-8_26
16. Bhardwaj, S.: Audio Data Augmentation with respect to Musical Instrument Recognition. Doctoral dissertation, MA thesis, Universitat Pompeu Fabra, Barcelona, Spain (Aug 2017), https://doi.org/10.5281/zenodo.1066137
17. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006)
18. Bock, S., Schedl, M.: Polyphonic Piano Note Transcription with Recurrent Neural Networks. In: Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 121–124. IEEE, Kyoto, Japan (Mar 2012). https://doi.org/10.1109/ICASSP.2012.6287832
19. Boogaart, C.G., Lienhart, R.: Note Onset Detection for the Transcription of Polyphonic Piano Music. In: 2009 IEEE International Conference on Multimedia and Expo. pp. 446–449. IEEE (2009)
20. Bottou, L.: Stochastic Learning. In: Bousquet, O., von Luxburg, U. (eds.) Advanced Lectures on Machine Learning, pp. 146–168. Lecture Notes in Artificial Intelligence, LNAI 3176, Springer Verlag, Berlin (2004)
21. Boulanger-Lewandowski, N., Bengio, Y., Vincent, P.: Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. In: Proceedings of the 29th International Conference on Machine Learning (ICML 2012). pp. 1881–1888. Edinburgh, UK (2012)
22. Boulanger-Lewandowski, N., Bengio, Y., Vincent, P.: Audio Chord Recognition with Recurrent Neural Networks. ISMIR p. 6 (2013)
23. Brown, J.C.: Calculation of a constant-Q spectral transform. The Journal of the Acoustical Society of America **89**(1), 425–434 (Jan 1991). https://doi.org/10.1121/1.400476
24. Brown, J.C., Puckette, M.S.: An Efficient Algorithm for the Calculation of a Constant Q Transform. The Journal of the Acoustical Society of America **92**(5), 2698–2701 (Nov 1992). https://doi.org/10.1121/1.404385
25. Burlet, G., Fujinaga, I.: Robotaba Guitar Tablature Transcription Framework. In: Proceedings of the 14th International Conference on Music Information Retrieval (ISMIR 2013). Curitiba, Brazil (2013)
26. Burlet, G., Hindle, A.: Isolated Guitar Transcription Using a Deep Belief Network. PeerJ Computer Science **3** (Mar 2017). https://doi.org/10.7717/peerj-cs.109
27. Byambatsogt, G., Choimaa, L., Koutaki, G.: Guitar Chord Sensing and Recognition Using Multi-Task Learning and Physical Data Augmentation with Robotics. Sensors **20**(21), 6077 (Oct 2020). https://doi.org/10.3390/s20216077
28. Carvalho, R.G.C., Smaragdis, P.: Towards End-to-end Polyphonic Music Transcription: Transforming Music Audio Directly to a Score. In: 2017 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA). pp. 151–155. IEEE, New Paltz, NY (Oct 2017). https://doi.org/10.1109/WASPAA.2017.8170013
29. Chen, F., Chen, N., Mao, H., Hu, H.: Assessing four Neural Networks on Handwritten Digit Recognition Dataset (MNIST). arXiv preprint arXiv:1811.08278 (2018)
30. Cho, T., Weiss, R.J., Bello, J.P.: Exploring Common Variations in State of the Art Chord Recognition Systems. In: Proceedings of the 7th Sound and Music Computing Conference, 2010. pp. 1–8. Barcelona, Spain (2010)
31. Cong, F., Liu, S., Guo, L., Wiggins, G.A.: A Parallel Fusion Approach to Piano Music Transcription Based on Convolutional Neural Network. In: Proceed-

ings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 391–395. IEEE, Calgary, Canada (Apr 2018). https://doi.org/10.1109/ICASSP.2018.8461794
32. Davis, D., Drusvyatskiy, D., Kakade, S., Lee, J.D.: Stochastic Subgradient Method Converges on Tame Functions. Foundations of Computational Mathematics **20**, 119–154 (2020). https://doi.org/10.1007/s10208-018-09409-5
33. Dessein, A., Cont, A., Lemaitre, G.: Real-time Polyphonic Music Transcription with Non-negative Matrix Factorization and Beta-divergence. In: Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010). pp. 489–494. Utrecht, Netherlands (2010)
34. Dieleman, S., Brakel, P., Schrauwen, B.: Audio-based Music Classification with a Pretrained Convolutional Network. In: Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011). pp. 669–674. Miami, USA (2011)
35. Donoho, D., Stodden, V.: When Does Non-Negative Matrix Factorization Give a Correct Decomposition into Parts? In: Proceedings of the 16th International Conference on Neural Information Processing Systems (NIPS 2003). pp. 1141–1148. Vancouver, Canada (2003)
36. Downie, J.S.: The Music Information Retrieval Evaluation EXchange (2005–2007): A Window into Music Information Retrieval Research. Acoustical Science and Technology **29**(4), 247–255 (2008). https://doi.org/10.1250/ast.29.247
37. Elowsson, A., Friberg, A.: Polyphonic Transcription with Deep Layered Learning. The Journal of the Acoustical Society of America **148**(1) (2018). https://doi.org/10.1121/10.0001468
38. Emiya, V., Badeau, R., David, B.: Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle. IEEE Transactions on Audio, Speech, and Language Processing **18**(6), 1643–1654 (Aug 2010). https://doi.org/10.1109/TASL.2009.2038819
39. Frieler, K., Basaran, D., Höger, F., Crayencour, H.C., Peeters, G., Dixon, S.: Don't Hide in the Frames: Note- and Pattern-based Evaluation of Automated Melody Extraction Algorithms. In: 6th International Conference on Digital Libraries for Musicology. pp. 25–32. ACM, The Hague Netherlands (Nov 2019). https://doi.org/10.1145/3358664.3358672
40. Gabor, D.: Theory of Communication. Journal of the Institution of Electrical Engineers **93**(26), 429–441 (Sep 1945)
41. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
42. Goto, M.: A Real-time Music-scene Description System: Predominant-F0 Estimation for Detecting Melody and Bass Lines in Real-world Audio Signals. Speech Communication **43**(4), 311–329 (Sep 2004). https://doi.org/10.1016/j.specom.2004.07.001
43. Grindlay, G.C., Ellis, D.P.: A Probabilistic Subspace Model for Multi-instrument Polyphonic Transcription. In: Proceedings of the 11th International Society for Music Information Retrieval Conference (ISMIR 2010). pp. 21–26. Utrecht, Netherlands (2010)
44. Hochreiter, S., Schmidhuber, J.: Long Short-term Memory. Neural Computation **9**(8), 1735–1780 (1997)
45. Hsieh, T.H., Su, L., Yang, Y.H.: A Streamlined Encoder/Decoder Architecture for Melody Extraction. In: Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Brighton, UK (Feb 2019). https://doi.org/10.1109/ICASSP.2019.8682389

46. Humphrey, E.J., Bello, J.P.: Rethinking Automatic Chord Recognition with Convolutional Neural Networks. In: Proceedings of the 2012 11th International Conference on Machine Learning and Applications. pp. 357–362. IEEE, Boca Raton, FL, USA (Dec 2012). https://doi.org/10.1109/ICMLA.2012.220
47. Humphrey, E.J., Bello, J.P.: From Music Audio to Chord Tablature: Teaching Deep Convolutional Networks to Play Guitar. In: Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 6974–6978. IEEE, Florence, Italy (May 2014). https://doi.org/10.1109/ICASSP.2014.6854952
48. Kameoka, H., Nishimoto, T., Sagayama, S.: A Multipitch Analyzer Based on Harmonic Temporal Structured Clustering. IEEE Transactions on Audio, Speech, and Language Processing **15**(3), 982–994 (2007). https://doi.org/10.1109/TASL.2006.885248
49. Kehling, C., Abeer, J., Dittmar, C., Schuller, G.: Automatic Tablature Transcription of Electric Guitar Recordings by Estimation of Score- and Instrument-related Parameters. In: Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14). Erlangen, Germany (2014)
50. Kelz, R., Dorfer, M., Korzeniowski, F., Böck, S., Arzt, A., Widmer, G.: On the Potential of Simple Framewise Approaches to Piano Transcription. In: Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016). pp. 475–481. New York, USA (Dec 2016). https://doi.org/10.5281/zenodo.1416488
51. Kelz, R., Widmer, G.: An Experimental Analysis of the Entanglement Problem in Neural-Network-based Music Transcription Systems. arXiv preprint arXiv:1702.00025 (Jan 2017)
52. Klapuri, A., Davy, M. (eds.): Signal Processing Methods for Music Transcription. Springer, New York (2006)
53. Korzeniowski, F., Widmer, G.: Feature Learning for Chord Recognition: The Deep Chroma Extractor. In: Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016). pp. 37–43. New York, USA (Dec 2016)
54. Korzeniowski, F., Widmer, G.: A Fully Convolutional Deep Auditory Model for Musical Chord Recognition. In: Proceedings of the 2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP). pp. 1–6. Salerno, Italy (Sep 2016). https://doi.org/10.1109/MLSP.2016.7738895
55. Kum, S., Oh, C., Nam, J.: Melody Extraction on Vocal Segments Using Multi-Column Deep Neural Networks. In: Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016). pp. 819–825. New York, USA (2016)
56. LeCun, Y., Bengio, Y., Hinton, G.: Deep Learning. Nature **521**(7553), 436–444 (2015). https://doi.org/10.1038/nature14539
57. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE **86**(11), 2278 – 2324 (1998). https://doi.org/10.1109/5.726791
58. LeCun, Y., Jackel, L.D., Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Muller, U.A., Sackinger, E., Simard, P., others: Learning Algorithms for Classification: A Comparison on Handwritten Digit Recognition. Neural Networks: The Statistical Mechanics Perspective **261**(276) (1995)
59. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.R.: Efficient Backprop. In: Neural networks: Tricks of the trade, pp. 9–48. Springer (2012)

60. Lee, D.D., Seung, H.S.: Learning the Parts of Objects by Non-negative Matrix Factorization. Nature **401**(6755), 788–791 (Oct 1999). https://doi.org/10.1038/44565
61. Lee, K.: Automatic Chord Recognition from Audio Using Enhanced Pitch Class Profile. In: Proceedings of the 2006 International Computer Music Conference, ICMC 2006. New Orleans, USA (2006)
62. Lim, H., Rhyu, S., Lee, K.: Chord Generation from Symbolic Melody Using BLSTM Networks. In: arXiv preprint arXiv:1712.01011 (2017)
63. Lu, W.T., Su, L.: Vocal Melody Extraction with Semantic Segmentation and Audio-Symbolic Domain Transfer Learning. In: Proceedings of the 19th International Conference on Music Information Retrieval (ISMIR 2018). pp. 521–528. Paris, France (2018)
64. Macrae, R., Dixon, S.: Guitar Tab Mining, Analysis and Ranking. In: Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011). pp. 453–458. Miami, USA (2011)
65. McFee, B., Bello, J.P.: Structured Training for Large-Vocabulary Chord Recognition. In: Proceedings of the 18th International Conference on Music Information Retrieval (ISMIR 2017). pp. 188–194. Suzhou, China (2017)
66. McLaughlin, N., Del Rincon, J.M., Miller, P.: Data-augmentation for Reducing Dataset Bias in Person Re-identification. In: AVSS 2015 - 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS). IEEE, Karlsruhe, Germany (2015)
67. Michelson, J., Stern, R., Sullivan, T.: Automatic Guitar Tablature Transcription from Audio Using Inharmonicity Regression and Bayesian Classification. In: Audio Engineering Society Convention 145. Audio Engineering Society, New York, USA (2018)
68. Mignot, R., Peeters, G.: An Analysis of the Effect of Data Augmentation Methods: Experiments for a Musical Genre Classification Task. Transactions of the International Society for Music Information Retrieval **2**(1), 97–110 (Dec 2019). https://doi.org/10.5334/tismir.26
69. Mistler, E.: Generating Guitar Tablatures with Neural Networks. Master of Science Dissertation, The University of Edinburgh, Edinburgh, UK (2017)
70. Moorer, J.A.: On the Transcription of Musical Sound by Computer. Computer Music Journal pp. 32–38 (1977)
71. Müller, M.: Fundamentals of Music Processing. Springer International Publishing, Cham (2015), https://doi.org/10.1007/978-3-319-21945-5
72. Nair, V., Hinton, G.E.: Rectified Linear Units Improve Restricted Boltzmann Machines. In: ICML 2010: Proceedings of the 27th International Conference on Machine Learning. pp. 807–814. Haifa, Israel (2010)
73. O'Hanlon, K., Plumbley, M.D.: Polyphonic Piano Transcription Using Non-negative Matrix Factorisation with Group Sparsity. In: Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3112–3116. IEEE, Florence, Italy (May 2014). https://doi.org/10.1109/ICASSP.2014.6854173
74. O'Hanlon, K., Sandler, M., Plumbley, M.D.: Non-negative Matrix Factorisation Incorporating Greedy Hellinger Sparse Coding Applied to Polyphonic Music Transcription. In: Proceedings of the 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 2214–2218. IEEE, Brisbane, QLD (Apr 2015). https://doi.org/10.1109/ICASSP.2015.7178364
75. Patel, A.D.: Music, Language, and the Brain. Oxford University Press (2010)

76. Paulus, J.K., Klapuri, A.P.: Conventional and periodic N-grams in the transcription of drum sequences. In: 2003 International Conference on Multimedia and Expo. ICME'03. Proceedings. vol. 2, pp. II–737. IEEE (2003). https://doi.org/10.1109/ICME.2003.1221722
77. Persichetti, V.: Twentieth Century Harmony. WW Norton New York (1961)
78. Pertusa, A., Inesta, J.M.: Multiple Fundamental Frequency Estimation Using Gaussian Smoothness. In: 2008 IEEE International Conference on Acoustics, Speech and Signal Processing. pp. 105–108. IEEE (2008)
79. Petsiuk, V., Das, A., Saenko, K.: RISE: Randomized Input Sampling for Explanation of Black-box Models. In: 2018 British Machine Vision Conference (BMVC). Newcastle, UK (2018)
80. Piszczalski, M., Galler, B.A.: Automatic Music Transcription. Computer Music Journal **1**(4), 24–31 (1977)
81. Poliner, G.E., Ellis, D.P.W.: A Discriminative Model for Polyphonic Piano Transcription. EURASIP Journal on Advances in Signal Processing **2007**, 1–9 (2006). https://doi.org/10.1155/2007/48317
82. Pons, J., Lidy, T., Serra, X.: Experimenting with Musically Motivated Convolutional Neural Networks. In: 2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI). pp. 1–6. IEEE, Bucharest, Romania (Jun 2016). https://doi.org/10.1109/CBMI.2016.7500246
83. Pons, J., Serra, X.: musicnn: Pre-trained Convolutional Neural Networks for Music Audio Tagging. In: Proceedings of the 19th International Conference on Music Information Retrieval (ISMIR 2018). Paris, France (2019)
84. Raczyński, S., Ono, N., Sagayama, S.: Extending Nonnegative Matrix Factorization—a Discussion in the Context of Multiple Frequency Estimation of musical signals. In: 2009 17th European Signal Processing Conference. pp. 934–938. IEEE (2009)
85. Raczyński, S.A., Vincent, E., Sagayama, S.: Dynamic Bayesian Networks for Symbolic Polyphonic Pitch Modeling. IEEE Transactions on Audio, Speech, and Language Processing **21**(9), 1830–1840 (2013)
86. Radicioni, D.P., Lombardo, V.: Computational Modeling of Chord Fingering for String Instruments. In: Proceedings of the 27th Annual Conference of the Cognitive Science Society. pp. 1791–1796. Stresa, Italy (2005)
87. Radisavljevic, A., Driessen, P.F.: Path Difference Learning for Guitar Fingering Problem. In: Proceedings of ICMC 2004, the 30th Annual International Computer Music Conference. Miami, Florida (2004)
88. Ramos, J.V., Ramos, A.S., Silla, C.N., Sanches, D.S.: An Evaluation of Different Evolutionary Approaches Applied in the Process of Automatic Transcription of Music Scores into Tablatures. In: 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI). pp. 663–669. IEEE, San Jose, USA (2016)
89. Román, M.A., Pertusa, A., Calvo-Zaragoza, J.: Data Representations for Audio-to-score Monophonic Music Transcription. Expert Systems with Applications **162** (Dec 2020). https://doi.org/10.1016/j.eswa.2020.113769
90. Rosenblatt, F.: Principles of Neurodynamics. Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, USA (1961)
91. Rutherford, N.: FINGAR, a Genetic Algorithm Approach to Producing Playable Guitar Tablature with Fingering Instructions. Undergraduate Project Dissertation, Univ. of Sheffield, Sheffield, UK (2009)

92. Schlüter, J., Grill, T.: Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks. In: Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015). pp. 121–126. Málaga, Spain (2015)
93. Schmidhuber, J.: Deep Learning in Neural Networks: An Overview. Neural Networks **61**, 85–117 (Jan 2015). https://doi.org/10.1016/j.neunet.2014.09.003
94. Scholz, R., Vincent, E., Bimbot, F.: Robust modeling of musical chord sequences using probabilistic N-grams. In: Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 53–56. IEEE, Taipei, Taiwan (2009). https://doi.org/10.1109/ICASSP.2009.4959518
95. Schreiber, H., Müller, M.: Musical Tempo and Key Estimation using Convolutional Neural Networks with Directional Filters. In: Proceedings of the 16th Sound & Music Computing Conference (SMC 2019). pp. 47–54. Málaga, Spain (2019)
96. Siegelmann, H.T., Sontag, E.D.: On the Computational Power of Neural Nets. Journal of Computer and System Sciences **50**(1), 132–150 (1995)
97. Sigtia, S., Benetos, E., Dixon, S.: An End-to-End Neural Network for Polyphonic Piano Music Transcription. IEEE/ACM Transactions on Audio, Speech, and Language Processing **24**(5), 927–939 (Feb 2016). https://doi.org/10.1109/TASLP.2016.2533858
98. Smaragdis, P., Brown, J.C.: Non-negative Matrix Factorization for Polyphonic Music Transcription. In: 2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics. pp. 177–180. IEEE, New Paltz, NY, USA (2003). https://doi.org/10.1109/ASPAA.2003.1285860
99. Smaragdis, P., Raj, B., Shashanka, M.: A Probabilistic Latent Variable Model for Acoustic Modeling. In: Proceedings of the Neural Information Processing Systems Workshop on Advances in Models for Acoustic Processing (2006)
100. Su, L.: Vocal Melody Extraction Using Patch-based CNN. In: Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Calgary, Canada (Apr 2018)
101. Su, L., Yang, Y.H.: Combining Spectral and Temporal Representations for Multipitch Estimation of Polyphonic Music. IEEE/ACM Transactions on Audio, Speech, and Language Processing **23**(10), 1600–1612 (Oct 2015). https://doi.org/10.1109/TASLP.2015.2442411
102. Su, L., Yang, Y.H.: Escaping from the Abyss of Manual Annotation: New Methodology of Building Polyphonic Datasets for Automatic Music Transcription. In: Kronland-Martinet, R., Aramaki, M., Ystad, S. (eds.) Music, Mind, and Embodiment, Lecture Notes in Computer Science (LNCS), vol. 9617, pp. 309–321. Springer International Publishing, Cham (2016)
103. Tan, L., Jiang, J.: Digital Signal Processing: Fundamentals and Applications. Academic Press (2018)
104. Thickstun, J., Harchaoui, Z., Foster, D., Kakade, S.M.: Invariances and Data Augmentation for Supervised Music Transcription. In: Proceedings of the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Calgary, Canada (2018). https://doi.org/10.1109/ICASSP.2018.8461686
105. Thomé, C., Ahlbäck, S.: Polyphonic Pitch Detection with Convolutional Recurrent Neural Networks. MIREX (2017)
106. Tommasi, T., Patricia, N., Caputo, B., Tuytelaars, T.: A Deeper Look at Dataset Bias. In: Domain Adaptation in Computer Vision Applications, pp. 37–55. Springer (2017)
107. Torralba, A., Efros, A.A.: Unbiased Look at Dataset Bias. In: CVPR 2011. pp. 1521–1528. IEEE (2011)

108. Tuohy, D.R., Potter, W.D.: A Genetic Algorithm for the Automatic Generation of Playable Guitar Tablature. In: Proceedings of the 2005 International Computer Music Conference, ICMC 2005. pp. 499–502. Barcelona, Spain (2005)
109. Vincent, E., Bertin, N., Badeau, R.: Two Non-negative Matrix Factorization Methods for Polyphonic Pitch Transcription. MIREX (2007)
110. Vincent, E., Bertin, N., Badeau, R.: Harmonic and Inharmonic Non-negative Matrix Factorization for Polyphonic Pitch Transcription. In: Proceedings of the 2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 109–112. IEEE, Las Vegas, USA (2008). https://doi.org/10.1109/ICASSP.2008.4517558
111. Vincent, E., Bertin, N., Badeau, R.: Adaptive Harmonic Spectral Decomposition for Multiple Pitch Estimation. IEEE Transactions on Audio, Speech, and Language Processing **18**(3), 528–537 (2009). https://doi.org/10.1109/TASL.2009.2034186
112. Wang, Q., Zhou, R., Yan, Y.: A Two-Stage Approach to Note-Level Transcription of a Specific Piano. Applied Sciences **7**(9), 901 (Sep 2017). https://doi.org/10.3390/app7090901
113. Wang, Y., Yao, Q., Kwok, J., Ni, L.M.: Generalizing from a Few Examples: A Survey on Few-Shot Learning. ACM Computing Surveys **1**(1), 1:1–1:34 (Mar 2020). https://doi.org/10.1145/3386252
114. Wiggins, A., Kim, Y.: Guitar Tablature Estimation with a Convolutional Neural Network. In: Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR 2019). pp. 284–291. Delft, Netherlands (2019)
115. Xi, Q., Bittner, R.M., Pauwels, J., Ye, X., Bello, J.P.: GuitarSet: A Dataset for Guitar Transcription. In: Proceedings of the 19th International Conference on Music Information Retrieval (ISMIR 2018). pp. 453–460. Paris, France (2018)
116. Yazawa, K., Itoyama, K., Okuno, H.G.: Automatic Transcription of Guitar Tablature from Audio Signals in Accordance with Player's Proficiency. In: Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3122–3126. IEEE, Florence, Italy (May 2014). https://doi.org/10.1109/ICASSP.2014.6854175
117. Yazawa, K., Sakaue, D., Nagira, K., Itoyama, K., Okuno, H.G.: Audio-based Guitar Tablature Transcription Using Multipitch Analysis and Playability Constraints. In: Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 196–200. IEEE, Vancouver, Canada (May 2013). https://doi.org/10.1109/ICASSP.2013.6637636
118. Yeh, C.: Multiple Fundamental Frequency Estimation of Polyphonic Recordings. PhD Thesis, University of Paris, Paris, France (2008)
119. Yoshii, K., Goto, M.: A Nonparametric Bayesian Multipitch Analyzer Based on Infinite Latent Harmonic Allocation. IEEE Transactions on Audio, Speech, and Language Processing **20**(3), 717–730 (2011). https://doi.org/10.1109/TASL.2011.2164530
120. Zeiler, M.D.: ADADELTA: An Adaptive Learning Rate Method. arXiv preprint arXiv:1212.5701 (Dec 2012)
121. Zeiler, M.D., Fergus, R.: Visualizing and Understanding Convolutional Networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014, Lecture Notes in Computer Science (LNCS), vol. 8689, pp. 818–833. Springer International Publishing, Cham (2014), https://doi.org/10.1007/978-3-319-10590-1_53

122. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: Proceedings of the 5th International Conference on Learning Representations, ICLR 2017. Toulon, France (Feb 2017)
123. Çakır, E., Parascandolo, G., Heittola, T., Huttunen, H., Virtanen, T.: Convolutional Recurrent Neural Networks for Polyphonic Sound Event Detection. IEEE/ACM Transactions on Audio, Speech, and Language Processing **25**(6), 1291–1303 (Jun 2017). https://doi.org/10.1109/TASLP.2017.2690575