# Group-wise Automatic Music Transcription

MPhil Thesis

William White

Spetember 2018

**DECLARATION**

This work has not been submitted in substance for any other degree or award at this or any other university or place of learning, nor is being submitted concurrently in candidature for any degree or other award.

Signed: _____

Date: _____

**STATEMENT 1**

This thesis is being submitted in partial fulfillment of the requirements for the degree of MPhil.

Signed: _____

Date: _____

**STATEMENT 2**

This thesis is the result of my own independent work/investigation, except where otherwise stated, and the thesis has not been edited by a third party beyond what is permitted by Cardiff University's Policy on the Use of Third Party Editors by Research Degree Students. Other sources are acknowledged by explicit references. The views expressed are my own.

Signed: _____

Date: _____

**STATEMENT 3**

I hereby give consent for my thesis, if accepted, to be available online in the University's Open Access repository and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed: _____

Date: _____

**STATEMENT 4: PREVIOUSLY APPROVED BAR ON ACCESS**

I hereby give consent for my thesis, if accepted, to be available online in the University's Open Access repository and for inter-library loans after expiry of a bar on access previously approved by the Academic Standards & Quality Committee.

Signed: _____

Date: _____

# Abstract

**Background:** Music transcription is the conversion of musical audio into notation such that a musician can recreate the piece. Automatic music transcription (AMT) is the automation of this process. Current AMT algorithms produce a less musically meaningful transcription than human transcribers. However, AMT performs better at predicting notes present in a short time frame. Group-wise Automatic Music Transcription, (GWAMT) is when several renditions of a piece are used to give a single transcription.

**Aims:** The main aim was to perform investigations into GWAMT.
Secondary aims included: Comparing methods for GWAMT on the frame level; Considering the impact of GWAMT on the broader field of AMT.

**Method(s)/Procedures:** GWAMT transcription is split into three stages: transcription, alignment and combination. Transcription is performed by splitting the piece into frames, and using a classifier to identify the notes present. Convolutional Neural Networks (CNNs) are used with a novel training methodology and architecture.
Different renditions of the same piece have corresponding notes occurring at different times. In order to match corresponding frames, methods for the alignment of multiple renditions are used. Several methods were compared, pairwise alignment, progressive alignment and a new method, iterative alignment. The effect of when the aligned features are combined (early/late), and how (majority vote, linear opinion pool, logarithmic opinion pool, max, median), is investigated.

**Results:** The developed method for frame-level transcription achieves state-of-the-art transcription accuracy on the MAPS database with an F1-score of `76.67%`. Experiments on GWAMT show that the F1-score can be improved by between 0.005 to 0.01 using the majority vote and logarithmic pool combination methods.

**Conclusions/Implications:** These experiments show that group-wise frame-level transcription can improve the transcription when there are different tempos, noise levels, dynamic ranges and reverbs between the clips. They also demonstrate a future application of GWAMT to individual pieces with repeated segments.

# Dedication

I'd like to thank all three of my supervisors (David Marshall, Kirill Siderov and Andrew Jones) for their guidance during my year at Cardiff, and for their support during the year afterwards. I have been grateful for every bit of criticism, positive and negative. I'd particularly like to thank Kirill who organised the hackathon where I first developed an interest in music transcription, without his initial support I would not have pursued this MPhil.

My partner Ellis has really helped me through the two years. I'd like to thank her and am glad to have to have the chance to return the favour.

Finally, I'd like to thank my running club, MDC. Running has been a welcome distraction. Tuesday night runs have let me blow off steam running up and down hills in the welsh valleys. Giving me some great friends and a fantastic hobby.

# Contents

# Chapter 1

# Introduction

This introduction is meant to identify some of the fundamental considerations of any music transcription system, such as the feature representation and its justification in relation to the human hearing system. It provides some of the background knowledge that will help in the understanding of this thesis, and introduces some concepts that will be discussed later. Not all of the background knowledge needed to understand this thesis is covered, as much of this will be covered in the related chapters. The hypothesis of this thesis is presented followed by the scope, which outlines the content of each chapter along with the contributions of the thesis as a whole.

## 1.1 Theory

The verb "to transcribe" is defined by the Oxford English Dictionary as follows: "To make a copy of (something) in writing" [1], so by extension, music transcription is to copy and represent played music in writing so that it might be recreated at a later date. Traditionally music is written as a score which is a symbolic representation where the pitch, duration and loudness of each note are shown sequentially, see Fig. 1.1. In this type of representation the height of the markings on the score indicates the pitch of the note [2], and the progression from left to right indicates the order of playing. The relative duration of each of the notes is determined by its note head and the note tails [2]. Other markings include the dynamics (loudness) which is shown by the letters and symbol below the notes and hairpins, and the accidentals which are the "♯" and "♭" symbols, the accidentals shift the pitch up or down by a semi-tone [2].

Other methods for representing music exist. These are usually specific to a single instrument, such as guitar tablature, which instead of indicating the pitch of the note to play, shows the position to place the fingers on the guitar so as to play

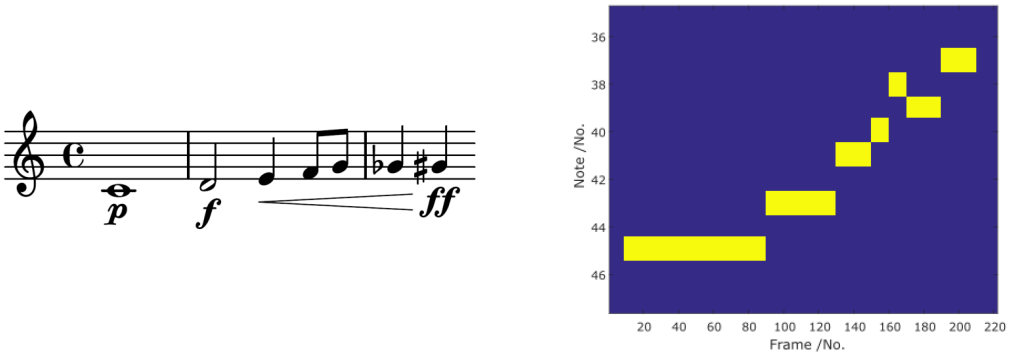


Figure 1.1: Left: An example of a score based representation of musical audio. Right: a piano-roll representation of the score on the left. The yellow horizontal regions are valued as 1 and indicate a note being present, and the blue regions as 0 and the absence of a note. The vertical axis represent the MIDI note number and the horizontal axis the frame (small period of time) number.

the intended note or chord. What both of these methods have in common is that it is easy for the musician to create, but more importantly, it allows the musician to understand how to play the piece.

Some of the instructions provided in a score are open to interpretation by the musician, e.g. the dynamics, global and local tempo. This means that if two musicians were to play from the same score, they are likely to sound quite different. Therefore, both score and tablature notation are better defined as a performance instruction rather than a representation. A less abstract representation for musical audio is the piano-roll notation, see Fig. 1.1, which is a two dimensional, time-pitch, representation that indicates which notes are present at any point during the piece, this removes the temporal ambiguity that the high level, score based representations have. The name piano-roll comes from the pianola, which is a piano that when loaded with a "piano-roll" (a roll of paper perforated with holes that indicate when to play the notes) will play the piece of music on its own. This representation is equivalent to the modern MIDI (Musical Instrument Digital Interface), which can be given to a virtual instrument as instructions.

Automatic Music transcription (AMT) is the extraction of such a representation from audio via signal processing and/or machine learning methods. For this thesis, the piano-roll notation will be used, following [3]. The reasoning for this is that piano-roll to score conversion is a complex subject area in its own right [4] with difficulties like the arrangement of staves on the score and the extraction of accidentals, dynamics and key signature.

The most apparent application of automatic music transcription is to be able to

write down instantaneously composed music (improvisation) which is common in genres such as Jazz, Flamenco and Funk. AMT would also be a valuable training tool, providing the musician with the ability to analyze his/her performance in hindsight to spot mistakes and make note of areas needing work, similarly for conductors, being able to know when a musician might be out of tune, or have played incorrectly is a valuable skill that would be aided by AMT. There will be cases where no score at all is available, in which case AMT can be used directly to other interactive tools include: Music analysis, analysis of improvisations by shortest grammar and other such approaches [5]; Real-time accompaniment of a singer or soloist [6]; Music search and retrieval, the ability to identify a song given a short extract; Music synchronization, synchronizing lighting or special effects to music.

## 1.2 Terminology

Before continuing on to the field of pitch perception and automatic music transcription some terms will be defined that allow a better description of musical attributes.

The *Pitch* of a musical sound describes where it lies on a perceptual scale that is determined by how we hear and process sound: A rudimentary definition is how 'high' or 'low' the tone is. If a target sound can be reliably matched to a sine wave at a set frequency by a group of human listeners then it is said to have pitch [7]. The pitch of that note would then be the frequency of the sine-wave. The fundamental frequency of an acoustic signal, or just the fundamental (F0), is the lowest of the prominent frequencies present, if ambiguous, then the frequency of the corresponding pitch is its fundamental. This form of pitch is termed absolute pitch because it has a corresponding fundamental at a fixed frequency. Relative pitch is similar but is defined by the difference between two pitches. Humans are generally better at transcribing relative pitch to absolute pitch, however people with perfect pitch are able to distinguish the absolute pitch of a heard sound.

The term *Rhythm* encapsulates the temporal aspects of music, played and perceived. In written score, rhythm can be broken down into the beat and the rhythmic structure of the notes. The beat marks out fixed intervals through the piece. In a score the beat dictates how fast or slow the musician should play the marked notes. Several beats combine to create a bar. These are represented on the score as the region between consecutive vertical horizontal lines, see Fig. 1.1. In a sequence of notes, the rhythmic structure is the length of time each note is played for, and the relative durations between each note. These times and durations are relative and so are dependent on the locations of the beats. The onset and offset refer to the perceptual start and end of notes, which is quite different to the start

of the note on a score, as for some instruments it can take several milli-seconds for the sound to be created, and then for the sound to be perceived.

The loudness of a musical signal is a perceived quantity and an ongoing investigation in the field of psycho-acoustics [8]. It is often described as a characteristic of sound ranging from very soft to very loud. Dealing with a perceptual scale is inconvenient for music processing and so usually a logarithmic (deciBel) scale is used. Dynamics are how musicians represent the loudness, and hence timbre, of a note on a score. Note that the perceived duration of a note is generally the same as the measured duration. This is not true for some instruments like the piano. The note can be marked on the score to last longer or shorter than the decay of the piano strings.

The tempo of a piece, or an extract of, is a continuous scalar quantity that measures the speed of progression through a piece of music. Although often measured in BPM (beats per minute), scores sometimes only give a loose description of the tempo e.g. *Andante* (walking pace). Typical values range between 60-120, subject to local variation depending on the artist's interpretation of a piece.

Timbre is the perceptual characteristic by which we distinguish sounds of equal pitch, duration and loudness [9]. It is how we would separate the sounds of two instruments playing the same note, and is often referred to as the sound's "colour", perhaps due to the emotion that timbre adds, for example, a simple synthesizer that produces sinusoids at each note's fundamental frequency playing Mozart might sound less "colourful" than a piano playing the same piece. The timbre of a sound is a spectral energy distribution and can change over time, usually in a more complex fashion than that of the loudness or the pitch.

## 1.3 Pitch Perception, Harmonics and the Human Hearing System

### 1.3.0 Pitch Perception

Pitch is fundamental to communication; in infant humans pitch and timbre distinction is learned from a much earlier age than spoken language [10]. Communication through pitch is also found in other primates, and thought to be important in the evolution and development of language [11]. In speech, inflections in conjunction with body language are used to communicate additional information such as emotion or intent that otherwise might not be communicated. The use is even more apparent in tonal languages such as Mandarin or Chinese, where both inflections and the absolute pitch (not relative pitch) are used to change the meaning of the words. For communication in music, pitch is used in sequences to create melodies

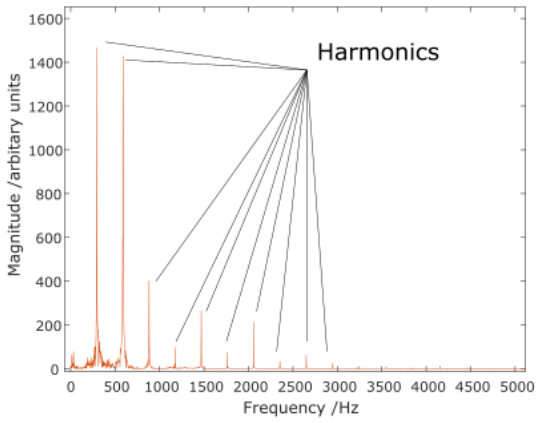


Figure 1.2: The frequency content for a single piano note shows that the harmonics occur at multiples of the fundamental frequency which in this case is the lowest frequency peak

and in combination with notes of different pitches to create harmony, it is the collaboration between harmony and melody that creates music.

Pitch is a subjective quality corresponding to the periodicity of an acoustic waveform [12]. A single musical note with a well defined pitch can be broken down into constituent harmonics and overtones. The harmonics are integer multiples of the note's fundamental frequency $f_0$

$$f_n = nf_0. \tag{1.1}$$

Overtones are resonant frequencies that are present at any other frequencies. Any single resonant frequency in the note can be called a partial. An example of harmonics can be seen in Fig. 1.2. The $f_0$ was defined as the lowest of the frequency components of a pitched sound, this is only partly correct. Generally this is the case, however, if the $f_0$ is removed from the signal or masked in some way, then the perceived pitch of the signal remains unchanged [13].

Throughout this thesis, the notes on the piano are referred to numerically using the numbers from 1 to 88. The corresponding frequency $f$of the $n$-th note on a piano is

$$f(n) \approx 2^{\frac{n-49}{12}} \times 440 \text{ Hz}, \tag{1.2}$$

This means that notes equally spaced along the piano will have logarithmically spaced frequencies. We perceive pitch differences in a similar way to how notes are

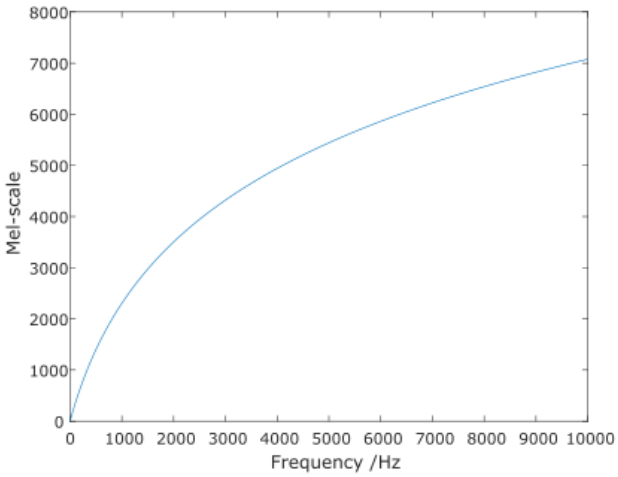


Figure 1.3: The Mel-scale is calculated by measuring the pitch resolution of human subjects.

mapped on the piano, logarithmically. As such it makes sense to construct a scale that is better modeled with how we perceive pitch. The Mel-scale is exactly that, a perceptual, subjectively determined frequency scale. This Mel-scale can be seen plotted relative to linear axis in Fig. 1.3.

### 1.3.0 Manual Music Transcription

Before considering the problem of automatic music transcription, it is important to first understand the method by which professional musicians transcribe, but also learn to transcribe, music. This may provide some understanding that can be applied to the task of Automatic Music Transcription.

Professional musicians (classically trained) generally begin their education at an early age, learning to read and write music initially, in conjunction with learning to play their instrument. It is likely that it is much later, at the start of their professional development, for example at a conservatoire, that they will begin formal training in the practice of transcription, otherwise known as music dictation to musicians. Typically the musician will be given examples of notes or harmonies, such as individual intervals, rhythms or melodies, and then be asked to transcribe the result. The musician will be given simple isolated cases first, then after mastering these, progress onto higher polyphony, more complex examples. In total it's hypothesized that it takes roughly 10000 hours to become a professional musician [14].

The next stage of understanding how musicians transcribe music is to consider

the process by which a final result is achieved. How is the problem split into stages to best apply their knowledge and training? Taken as the problem of determining the notes that are playing at any one time, Klapuri et al. [15] presented professional musicians with chords containing between two to five notes and asked them to list the intervals. The percentage of chords guessed correctly by the best in the sample set was still below 50% for high polyphony. One result to draw from this is that it is difficult to ascertain a reliable transcription when a sample is taken out of context, from which the musician can draw upon global information to aid his transcription. This conclusion that utilising temporal and global information aids musicians in transcription is confirmed by Hainsworth [16], who performed a survey of 19 professional musicians asking them each about the process that they use. What was found is that there is a process that was fairly universal: they iterate over the piece of music to be transcribed, first noting the global structure, then transcribing dominant themes using repeated sections to aid the transcription of other sections, and continuing to transcribe finer details, using the knowledge from previous iterations to aid the next. This is a direct application for group transcription, it could be used to improve the transcription of a repeated section or melody within a piece, as each repeated phrase gives more information about the notes present. Detecting repeated sections like chords or melodies is a simpler problem than transcription and shows how group-wise transcription is applicable not just to a situation where multiple renditions of a piece are available, but to an individual piece with repeated elements.

Other finer details of music dictation are pitch spelling and score arrangement. Pitch spelling is the selection of accidentals so that the score is not cluttered, whilst score arrangement is how the notes are arranged on the score in piano music for example, this determines which notes are contained on the bass (left-hand) stave and which are contained on the higher (right-hand) stave. Professional musicians can use their musical experience to perform these tasks easily, however, they are much more difficult to perform automatically [4, 17].

### 1.3.0 Human Auditory System

Before considering how to represent audio suitable for AMT, it is useful to understand how it is transmitted to the brain via the ear, as it could provide hints as to why some humans are better at music transcription. For instance, some savants have an incredible ability to recognize concurrent pitches [18]. When sound enters the human ear it passes through the three stages of the human auditory system before being converted into an electric signal that is transmitted to the brain via the auditory nerve. The outer ear first gathers and funnels sound into the ear canal. As the ear canal is sensitive to direction it acts to "filter" the incoming sounds so as to encode some spatial information, this helps the owner of the ear to determine the

location of the incoming sound. In the second stage of the ear, the sound waves are converted into mechanical vibrations via the eardrum, the mechanical vibrations are transmitted through three bones called the ossicles into to the final stage, the inner ear. The vibrations are passed on to the basilar membrane which extends through the two water filled sacks of the cochlea. The basilar membrane has a varying stiffness along its length which causes the wavelength of each frequency component of the incoming signal to decrease, and amplitude to increase, until it peaks at a position determined by its frequency. The result of this is that each position in the cochlea is stimulated by a different frequency. The stimulating frequency is converted into an electrical impulse via the stereocilia, which are small hair cells covering the inside of the cochlea, there are roughly 15,000 in total. Understanding the inner-workings of the human hearing system is important in the field of AMT as the state of the art is still way below that of a human expert, by modelling how the human ear represents audio for processing by the brain it could aid the development of biology inspired audio representations as in [19].

## 1.4 Representation of Audio

When musicians are transcribing music they are presented with the raw audio, however the human ear has a way of pre-processing this audio via the ear so that the brain is presented with a higher-level representation. This leads us onto the next section, where some of the different methods for representing audio, their applications to this thesis and their advantages for music informatics as a whole are explored.

When audio is recorded, the physical sound vibrations are recorded as one dimensional time-series data. This is a poor representation for music as it is difficult to discern any information such as the pitch content or instrumentation of the piece. Much better, two dimensional representations of audio can be derived using frequency analysis, as we will see.

Any representation of audio that is appropriate to describe audio before it reaches the cochela is termed “low-level”, whilst any representation suitable for describing audio in a cognitively straightforward way is termed “high-level” (score, pianoroll). At some point in between these representations there exists a collection of audio representations defined as mid-level, Ellis et al. [20] define the ideal characteristics of what are termed mid-level audio representations: they should be of high enough detail to discern the independent contributing sources; they should be invertible (the audio should be reconstructable). This is important as you want the mid-level representation to contain the same quantity of information as the low-level representation; they should be physiologically plausible, although much about what happens to audio after it has been passed into the auditory

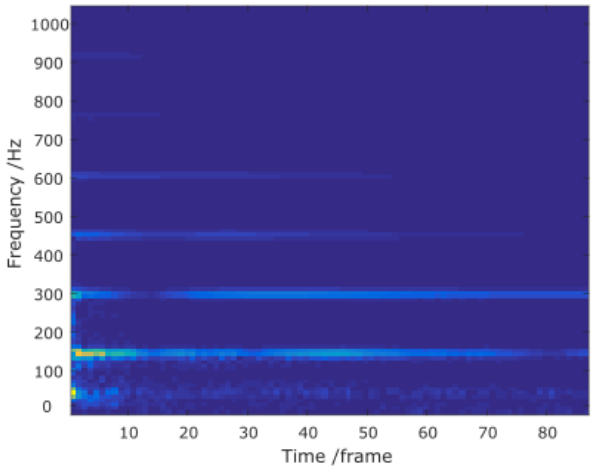


Figure 1.4: An example of the absolute magnitude of the complex Short Time Fourier Transform (STFT) for a single note. Consecutively higher harmonics decrease in magnitude.

nerve is still unknown, it is important to consider physiology when developing any representations. Thus, studying the workings of the human hearing system is important for AMT.

Often when calculating a mid-level audio representation, a "windowing" function is applied. This means that a one dimensional function of appropriate length (usually short - 10 ms) is multiplied by the audio time-series at regularly spaced intervals. Example By far the most used mid-level audio representation is the short-time Fourier transform (STFT) [21] which applies a windowing function to successive audio frames, before applying the fast Fourier transform (FFT) and transforming the windowed segment into the frequency domain. The result is a two dimensional complex time-frequency representation as seen in Fig. 1.4.

Whilst the STFT is an effective way to represent audio, the frequency scale is linear, and although it can be logarithmically binned, it would lead to a loss in information and resolution, no longer satisfying the requirement of reversibility required from any mid-level audio representation. As the pitches of consecutive notes on the western music scale are logarithmically spaced in pitch, it will mean that the distance between corresponding harmonics for different notes will be different, for example the distance between the first and second harmonic for an E will be less than the distance between the first and second harmonic for an E#. This is an undesirable quality for any pitch detection tasks as the majority of algorithms are based on pattern recognition and so an intermediary step, such as applying logarithmic frequency bands would be needed. Mel-scale spectrograms do exactly

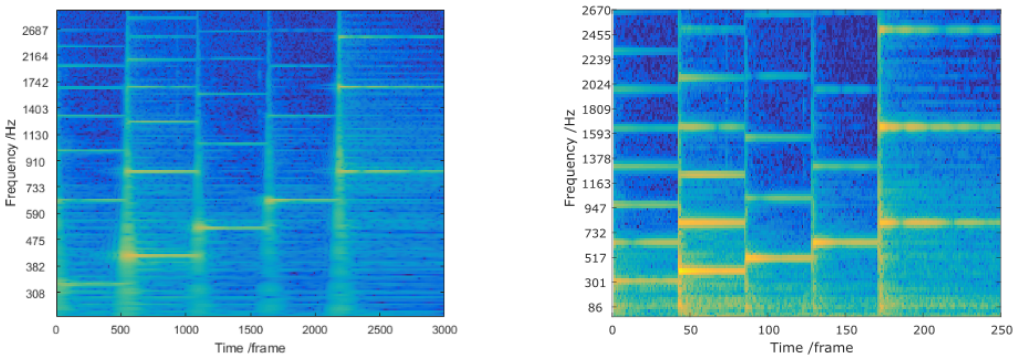


Figure 1.5: The CQT (Left) and STFT (Right) of a series of notes separated by equal intervals. These plots were created by stitching together individual note audio samples and performing the respective transforms. This highlights the translational properties of the CQT, the relative positions of the harmonic peaks from the fundamental are the same for each note. That is, you could translate the first note vertically up the frequency axis until all of the note's partials overlapped with the second note. In the fft this is not the case due to a linearly spaced frequency bins.

this, they apply the mel-scale that was mentioned earlier [22] to high resolution spectrograms. Triangular windowing functions are applied at frequencies specified by the Mel scale to give a spectrogram with logarithmically spaced frequency bins, see Fig. 1.3. However, by binning the Fourier transform the ability to reconstruct the original signal is removed, no longer satisfying the requirements in [20] for an ideal mid-level feature representation. Following this logic that both a logarithmic frequency scale, and invertibility are needed in a mid-level feature representation suitable for music informatics, the constant-Q transform (CQT) was developed in [23].

Similarly to the STFT, the CQT is a complex time-frequency representation, however, the method for obtaining the transform is quite different. The filter-bank used in the obtaining of a CQT is modelled off the human hearing system [24], as opposed to the linearly spaced frequency bins of the Fourier transform. The CQT also has geometrically spaced frequency bins, meaning that a region containing a played note can be translated up the frequency axis with the result being a linear change in pitch, see Section 1.4. For more details concerning the CQT and its variants, see [23].

On the subject of musically meaningful audio representations, a group of high level features called "chroma-features" were introduced by Mueller et al. [25]. Chroma-features are calculated by binning the signals STFT into 12 bins, with each bin corresponding to the total pitch content of the notes A-G. These features have

the advantage of being very compact, and more suitable for any applications that have a heavy computational overhead such as methods for audio alignment [26, 27], or for audio matching of music by searching through databases [28]. Another advantage of these features is that they are highly robust to changes in timbre, dynamics, articulations and tempo differences. The lack of these variations means that the features correlate to the underlying harmonic content, such as the melody or the progressing harmony. This makes Chroma-features suitable for tasks such as chord recognition [29] and music classification [30].

## 1.5 Scope of this thesis

This section summarizes the contribution of this work and why its important for music transcription as a whole. It outlines the structure of this thesis, where results are contained and summarizes the main points of each chapter and section.

Stated below is the Group-Wise Automatic Music Transcription (GWAMT) hypothesis that is tested in this thesis. How the hypothesis is tested is covered briefly in this scope and in more depth in Chapter 5.

**Hypothesis** Multiple renditions of the same piece of music improves the automatic transcription performance beyond that of the average individual piece

This thesis develops a pipeline for performing Group-Wise Automatic Transcription (GWAMT), which tests the hypothesis stated above. The chapters are focused on elements of this pipeline. After performing a literature review in Chapter 2, Chapter 3 introduces a method for frame-level transcription, Chapter 4 discusses methods for the aligning of musical audio, and Chapter 5 details the experiments carried out to test the group-wise transcription hypothesis using the described pipeline. Chapter 6 contains the conclusions of the work, and discusses areas where future work could be carried out.

The literature review summarizes the research in the field of AMT in a chronological style. This is directly relevant to the third chapter which develops an AMT system as it shows what previous authors have found successful, and also relevant to the chapter on group-wise AMT as it shows that AMT is not yet performing well enough for widespread use, a performance that could be improved by GWAMT. The difficulties related to the field are discussed, and how these affect the research carried out here. In general this chapter provides the context for this research, which supports the methodology and justifies some of the recommended future work.

The third chapter focuses on frame level transcription, how deep learning and convolutional neural networks are used to classify a frame of musical audio from a

piano to determine the notes present. The mechanics of classification using CNNs are detailed, and results compared against the public MAPS database [31], achieving state of the art F1-measure of 76.67%, higher than the previous best[32] at 70.60%. The architecture and training methodology, where the network is trained on synthetic random data, are novel. It also shows that the entanglement problem described [33] can be avoided by training the network on randomly distributed data, however by doing so the network becomes less capable at recognizing chords, this trade-off between generalization and over-fitting is called bias-variance decomposition. A main limitation of this chapter is that its focus is frame-level transcription, which is suitable for this definition of transcription and testing the hypothesis, however lacks the ability to take into account high level musical phenomena (trills, note dynamics etc.), and also the ability to consider a larger context such as the note level transcription system in [34].

For the fourth chapter the performance of 3 methods for multiple audio alignment are compared, this is the process of achieving a frame-to-frame correspondence between each piece. Two methods are taken from the literature: progressive alignment [35] and pairwise alignment [36] and one new method outlined in this thesis called iterative alignment. This new method achieves an alignment by iteratively aligning the pieces to an evolving average template. This method shares a principle with progressive alignment in that there is a template that contains the combined information for the mapping between each of the pieces. This method does not perform as well as pairwise alignment for a small number of pieces, however if developed further it could prove useful for aligning a large number of pieces more accurately than pairwise alignment and more quickly than progressive alignment. The method used for the group-wise transcription experiments is pairwise alignment as the implementation of progressive alignment did not achieve a high enough standard. A minor novelty here is the use of windows for comparing sequence elements, as this yielded a better result than otherwise for pairwise alignment. For comparing the alignment methods the average note onset deviation is used as in [27]. The two main limitations of this chapter are the failure to compare progressive alignment to the other methods, and that synthetic datasets are used for testing, meaning the results cannot be compared to any in the literature.

The final chapter before the conclusion is on group-wise automatic music transcription and details the experiments testing the overall hypothesis. Using the methods described and experimented with in the first two chapters two strategies are described for achieving a group-wise transcription, early and late feature combination. Features are combined either before or after the transcription. In general late feature combination performed better, two methods for averaging the features are recommended as they beat the transcription score without the group-wise method by 1%. These two methods are the majority vote and Logarithmic

pool. The logarithmic pool method was shown to perform well when a larger number of pieces were available, whilst the majority vote method performed better with less pieces available. The experiments support the group-wise hypothesis however as they are carried out on synthetic data, further research using real data is needed. The chapter also highlights how group-wise music transcription has a place not just in piano transcription but for the field of music transcription in general.

The concluding chapter summarizes the key points of the preceding chapters, highlights areas for future work and reflects on the research methodology.

The conclusion explains that the work performed could have been more focused on performing experiments on GWAMT from the start, as opposed to first working on transcription. Whilst the work will show some success for transcription, there is enough active research in this area and as such increased novelty could have been found in GWAMT. Rather than aligning multiple pieces to test GWAMT, it is suggested that the hypothesis could have been tested by identifying repeated sections within a piece, akin to human transcription. Areas of future work are then discussed, a clear extension of the following work would be to extend beyond the piano to other instruments, however, studying the cases where GWAMT is and is not successful on the frame level could provide more detail about why GWAMT is possible, and how it should be applied in practice.

### 1.5.0 Contributions

This contributions section draws attention to the novel aspects of this work, highlighting the sections which the reader should turn to for more information.

- The investigation into the GWAMT hypothesis (see Section 1.5) and subsequent experimentation is novel and forms the main contribution of this work. It was found that GWAMT can improve the transcription accuracy in some cases, see Chapter 5.

- A novel machine learning architecture and training methodology is developed that achieves state of the art performance on the MAPS [31] database. The architecture achieved an F1-measure of 76.67%, surpassing the previous best by Kelz et. al [32] at 70.60%. See Chapter 3 for further details.

- A new method for the alignment of multiple musical renditions was developed, with performance comparable to pairwise alignment. If developed further it could prove useful for aligning a large number of pieces more accurately than pairwise alignment and more quickly than progressive alignment. See Section 4.2 for details.

- A minor novelty was the use of multiple spectrogram frames in the cost metric for Dynamic Time Warping (DTW) as this was found to be more reliable in practice. See Section 4.2 for further details.

# Chapter 2

# Literature Review

## 2.1 Introduction

Research into automatic music transcription was first published by Moorer in 1977 [37] with his work on the transcription of simple duets. In the same year the Computer Music Journal was started. Since then, the problem has been well studied and has separated into several sub-tasks:

**Multiple-F0 Detection** This is one of the predominant tasks in AMT and has received the largest attention of the sub tasks. It is detection of all of the notes present in a selected window, with no limit on polyphony.

**Note onset and offset detection** This is the localization of the positions of the onsets and offsets of each note, called note tracking; it is usually tackled separately.

**Loudness estimation** As in the section on pitch perception, how "loud" a note is heard is a perceptual quantity and in itself difficult to quantify directly from audio.

**Instrument recognition** Given an unlabeled sample of music, an instrument recognition algorithm attempts to identify how many, or more specifically, which instruments are present.

**Rhythm transcription and beat tracking** The challenge here is to extract the rhythm from a musical sample and present it in an interpretable way, such as a score. Beat tracking just aims to gauge the beat locations, and is more relevant to the unpitched transcription of percussion.

We can categorize the overall problem of AMT into different, context specific problems, relevant to genres of music, and different musical applications. We call

the transcription of a single melody, without accompaniment *monophonic transcription*, and the transcription of multiple concurrent notes *polyphonic transcription*. Monophonic transcription is widely considered a solved task [38], where as the problem of polyphonic transcription is still very much open. Polyphonic transcription can also be split into pitched and unpitched transcription. Pitched transcription is the estimation of pitch tracks or notes from musical audio signals [39], and unpitched is the transcription of drums or other unpitched instruments. Only the problem of pitched transcription will be discussed, but for more information regarding unpitched transcription a good summary is presented here [40].

After Moorer's pioneering work in the 70's, Chafe was the first to attempt piano transcription by utilizing prior knowledge of the instrument [41]. He used the knowledge that pianos typically produce significant energy at the first or second partial. In his work, Chafe ranks pitch hypotheses, then removes the hypotheses that do not fit the piano model. This top down approach of identifying all of the possible pitches, then culling the false positives is seen later in the literature and achieved good results at the MIREX competition from 2008 to 2011, see Table 2.1. One of the reasons that this method worked well is due to its application of global and local information to make an informed frame-level prediction, akin to how humans transcribe music.

Work on the transcription of duets was continued by Maher, who first proposed a method for the source separation of two instruments [42]. He highlights the applications to other domains, such as removal of noise, or the separation of multiple speakers. He later built on his original work by attempting to transcribe the sources separated by his method [43]. Most of these early attempts suffered from the same problem of a low polyphony and pitch range.

The MIREX competition is held every year where entrants compete in a variety of computer music tasks to achieve the highest score on a database designed to be challenging for that sub-task. Table 2.1 contains the results from the Multiple-F0 task at the MIREX annual competition for the past 10 years. Submissions are evaluated on a range of different instruments and genres. Prior to the first entry using deep learning in 2014 the results for the past 5 years had failed to improve significantly and in most cases fell. A possible reason for this could be due to the limitations of hand-designed features for a model fitting for the entire task of AMT.

In the 90's there were many different approaches to improve the polyphony of AMT systems, with a large potion of these methods termed "statistical methods" [44]. The number of methods developed are quite substantial so only a few will be highlighted. Hawley [45] attempted polyphonic transcription by identifying the prominent pitch and subtracting it at the estimated velocity (scalar variable of the speed at which the piano key is struck, is related to loudness). Kashino et al. [46] designed a system based on perceptual sound source separation; this is different

to sound source separation which only separates sources based on their location. Perceptual sound source separation attempts to simulate the human perceptual system. For instance, if a loudspeaker was playing a piece of orchestral music, a human would hear multiple instrument sources, although technically it would be from a single source, the loudspeaker. After separating out the perceptual sources Kashino then transcribes them individually. For cases with high polyphony he first finds the lowest of the notes present, then models its spectral shape to find the best candidates for the rest of the signal.

In the late 90's, the development in the field of artificial intellignece (AI) was gaining momentum once again, with events such as the defeat of the chess world champion Garry Kasparov, by IBM's AI, Deep Blue. AI was also applied for the problem of AMT. Martin et al. [47] and Godsmark et al. [48] used a blackboard AI system to combine a top-down, knowledge based approach, with a bottom up, data driven approach. A "blackboard" system takes its name from the idea that multiple professors can solve a problem on a black board by all contributing their knowledge and altering others' input, for computers and AMT this means different sources of information regarding the possible transcription are all tied together and considered using the AI. A log-lag correlogram is used to extract onset information and pitch information that form hypotheses on the "blackboard".

Up until this point in time, most approaches to AMT had been knowledge based, where information such as the instrument, the polyphony and our fundamental physical understanding of audio, were used to build models suitable for specific cases of music transcription. Another way to approach the problem is learning based methods, some of which were unfeasible before the 90's due to limited computing power and data. In these learning methods, data examples are used to train machine learning algorithms. The algorithm then tries to predict the transcription of a piece of music not used for training.

In the 2000's, the focus shifted onto sparse coding based methods, such as Independent Component Analysis (ICA) and Non-Negative Matrix Factorisation (NMF). Sparse coding methods can be defined as follows: "a sparse code is one which represents information in terms of a small number of descriptors out of a whole set" [49]. This principle was first applied to music by Abdallah et al. who extended the popular method of Independent Component Analysis (ICA) to decomposing spectrograms into note constituents. Virtanen [50] tried a method similar to Maher [43], after first separating out the contributing musical sources, he approximates the spectrums of the sources by assuming that their shape is constant over time. He then attempts to transcribe the tracks by minimizing the signal reconstruction error from its constituent parts.

Non-Negative-Matrix-Factorisation is a widely used method for transcription. It was born out of ICA and can be thought of a sparse coding with non-negativity

constraints [51]. Sometimes known as spectrogram factorisation, NMF attempts to use provided spectral data to learn a series of pitch bases corresponding to notes played by the instruments present in the training data. It was used initially by Brown [51] for AMT in an unsupervised approach, but was limited to instruments that exhibit a static harmonic profile (unlike piano as it is partially percussive and higher partials decay much faster than lower ones). NMF was improved by Abdallah et al. [52], by modelling the spectrogram of a piano as a Gaussian process, with noise that is multiplicative and non-Gaussian. This allowed NMF to be trained on entire recordings, as opposed to individual notes as in [51].

NMF is based on the assumption that the bases learned will correspond to pitch spectra. However, this is not always the case when training on polyphonic music. Constrained NMF is a variant of NMF where the bases are constrained in a smaller region of entire frequency range, making the likelihood of NMF learning meaningful pitch spectra higher. This means that the bases can only contain just a few adjacent harmonic or in-harmonic partials. This was first proposed by Vincent et al. [53] where they use these constraints on NMF applied to piano transcription. Vincent et al. later extend their work [54] by allowing for in-harmonic constraints and adaptive tuning.

A more recent, novel application of NMF to AMT was developed by Kirchoff et al. [55]. In their work they allow prior information about the location and pitch of some of the notes to be used to iteratively improve the bases approximation that NMF performs. Cheng et al. [56] proposed a model for piano transcription based on NMF, which models the attack and decay of the piano spectra to improve the learned bases. It is a promising work as it shows state of the art note level results, without incorporating a musical language model.

A whole separate class of Learning-based AMT approaches are defined here as being classification based approaches. Where classification techniques such as support vector machines (SVMs), artificial neural networks (ANNs) and convolutional neural networks (CNNs) are used to classify each frame, assigning a probability to each note for its likelihood of being present. This class of methods has received the most attention lately with all the entries to the 2017 MIREX frame-level transcription competition being classification based.

Poliner and Ellis [57] were the first to use support vector machines (SVMs) to try and classify piano notes. In their approach no prior knowledge was used. They trained 88 individual SVMs for piano transcription with each corresponding to a different note. Nam et al. [58] also used SVMs, a deep belief network was used to learn a representation suitable for piano transcription. Nam et al. also introduced a method for jointly training all 88 SVMs, which they concluded was better and faster than single note training. Multiple note training is used from this point on-wards for classification based algorithms.

The MAPS database is a collection of piano recordings played from corresponding ground-truths and was created by Emiya et al. [31]. It has since been the benchmark for any proposed piano transcription methods and will be used for the purpose of training and testing in this thesis. The database comprises of a total of 7 pianos and 10000 piano recordings. Two of the pianos are real pianos where as the rest are virtual. The samples include a mixture of chords and complete pieces with corresponding MIDI files. The configurations used for splitting the data into train-test splits can be found [59], and will be covered in the next chapter.

Deep neural networks are powerful classifiers that can learn to generalize well, and have seen success in many areas including computer vision and other Music Information Retrieval tasks. The specifics of deep learning will be covered in the next chapter. Neural Networks were applied to the problem of AMT by Sigtia et al. [34], who trained a deep network to classify frames of piano music. They train and evaluate their network on the MAPS piano database. They improve on their work by using CNNs which are a group of classification models similar to deep networks well suited to music transcription due to their ability to identify repeating patterns in noisy data. The specifics of training CNNs for music transcription, and some of the challenges that arise will be covered in the next chapter.

The best frame level piano transcription system to the author's knowledge is detailed in [32] where Kelz et al. experiment with simple CNN based frameworks for piano transcription, with excellent results. They specifically experiment with the best frequency based audio representations for deep learning. They conclude that the STFT with logarithmically spaced frequency bins and logarithmically scaled magnitudes is the best input representation for CNNs. They were surprised with the CQT failing to perform as well as it has in the past (Although they have not experimented with a CQT with logarithmically scaled magnitudes. This seems to be an unfair comparison of the STFT to the CQT.).

On the topic of piano transcription there is one final model that is worth mentioning. The model developed by Cheng et al. [56] uses NMF and an attack/decay model for transcribing piano music at the note-level. This constitutes the state of the art for note-level piano transcription.

## 2.2 Language Models and Note Tracking

Frame transcription suffers from a large flaw that is inherent to the problem: the limitation of not being able to consider information outside the frame of interest. As discussed in Section 1.3 on how humans transcribe music, humans are quite poor at frame-level transcription, and instead use multiple passes to accumulate enough global and local information to make an accurate note predication. In this section some attempts to utilize temporal information for AMT will be highlighted.

| Year | Accuracy | Transcription Method | L/K |
|---|---|---|---|
| 2007 | 0.605 | Identifying and removing the prominent pitch using a computational model of the human auditory system [60] | K |
| 2008 | 0.665 | Generate and iteratively select note candidates by minimizing the reconstruction error [61]. | K |
| 2009 | 0.688 | Generate and iterativly select note candidates by minimizing the reconstruction error [62]. | K |
| 2010 | 0.692 | Generate and iterativly select note candidates by minimizing the reconstruction error [63]. | K |
| 2011 | 0.683 | Generate and iterativly select note candidates by minimizing the reconstruction error [64]. | K |
| 2012 | 0.642 | Pair-wise analysis of spectral peaks to generate pitch hypotheses with tracking of high tone note objects [65], | |
| 2013 | 0.662 | Probabilistic latent component analysis for learning instrument spectral models [66] | L and K |
| 2014 | 0.723 | Deep learning and perceptual features [67]. | L |
| 2015 | 0.654 | probabilistic latent component analysis for learning instrument spectral models [68]. | L and K |
| 2016 | 0.537 | Combination of an auditory model, adaptive oscillator networks and neural networks [69] (note that the Convolutional Neural Network model for this year from the corresponding paper [32] had bugs and didn't perform). | L |
| 2017 | 0.720 | Convolutional Neural Network trained on log-spaced frequency data generated from a filter bank, and augmented by pitch shifting [70] | L |

Table 2.1: The results from the MIREX competition for frame level transcription from 2007. The column on the right is the method of approach, L (Learning) or K (Knowledge)

A direct comparison can be made between the problem of automatic speech recognition and automatic music transcription, though speech transcription has recieved far more research attention than AMT due to larger commercial applications. Both speech and music are generative: they are made up of much smaller constituent parts that can be drawn from the same dictionary. For speech these would be phonemes that collectively make words and in music this dictionary would be notes that form melodies, chords and rhythm. The main difference between the two is that speech is mainly monophonic and music is polyphonic, however speech signals generally change much more quickly as the phonemes vary in pitch and time. In contrast, because notes usually vary in only pitch it makes them hard to identify when containing overlapping harmonics. Generally speech generation follows a set of rules. This set of rules can be modelled by a language model that provides the acoustic frame-level model with a set of prior probabilities based on the previous words and phonemes. These language models are vital for the success in speech recognition as the acoustic model often does not have enough information in a single frame to separate differences between possible outputs [71]. The same principle can be applied to music transcription, a language model that provides prior probabilities to each frame level note hypothesis can be used to improve the performance of the AMT system. There has been less research into musical language models than into frame level transcription. This is potentially because it is believed that language models should not be necessary for frame level transcription. Another reason for this could be the difficulty in modelling such a large output space for polyphonic music, whereas in speech recognition there are a relatively small number of words to consider. To give an example, the English language has about 170000 words and a vocabulary of 3000 words covers about 95% of all occurrences [72]. whereas on the piano with a polyphony of 6 there are $24 \times 23 \times ... \times 19 = 96909120$ combinations for 6 notes in a 2 octave range.

A common fairly simple way to process the outputs of a frame-level model is to use hidden Markov models (HMMs). These were first employed by Box et al. [73] who use a 3-state HMM combined with a silence model and an auditory based acoustic model. Badeau et al. [74] instead use a 4-state HMM where an additional state is dedicated to silence. HMMs were applied to the task of piano transcription and evaluated on the MAPS database by Cheng et al. [75], who used a 4-state HMM with the states corresponding to the attack, decay, release and sustain of the note. It is interesting to note that all of the models have assumed that an underlying HMM has states that correspond to the production of the notes, and do not model the musical language. So whilst this method may serve to improve the results of a frame level model, it cannot be considered a musical language model.

Recurrent neural networks (RNNs) are powerful models for sequential data, Kaparthy et al. [76] demonstrated their prowess at language modelling by train-

ing an RNN on the works of Shakespeare, with the goal being to generate new Shakespearean text. This showed surprisingly good results. RNNs suffered from a problem caused by not knowing the length of the sequence being evaluated, however this was eliminated by Graves et al. [77] who first managed to train RNNs in an end-to-end fashion. Graves et al. then applied RNNs to speech recognition [78] where they achieved state of the art results.

RNNs have also shown success when applied to AMT. Bock et al. [79] used bi-directional RNNs to transcribe piano music by identifying both the onsets and the notes simultaneously, hence providing some context. Transcribing music in this way also makes it more practical for musicians and the number of false positives and negatives when evaluated at the note level is much lower.

It is difficult to use RNNs as a language model because of the high computational requirement in computing all of the prior probabilities for combinations of notes. To get around this issue, Boulanger-Lewandowski et al. [80] improve upon frame-level transcription by combining RNNs with restricted Boltzmann machines (RBNs), which have proved to be able to represent complicated distributions at one time step, with parameters that depend on the previous time step [80]. They conclude that the best method for using RNNs as a language model is a variant called RNN-NADE which stands for Neural Auto-regressive Distribution Estimators, which is a variant of RBNs specialized to representing high dimensional variables [81].

Language models being used to improve the transcription accuracy is demonstrated again by Sigtia et al. [82], who use an RNN-NADE to provide Dirichlet priors for a PLCA frame-level acoustic model. They observe an improvement of 3%, similar to the improvement seen in [80]. In later work by Sigtia et al. [34] they improve upon the Hybrid model by combining the results of the two models using a hashed beam search. Further to this, they then train the hybrid model in an end-to-end fashion and publish their results in a very well written paper [59], their configurations for evaluation on the MAPS database of piano music are now used as the benchmark for piano transcription.

## 2.3 Other AMT sub-tasks

Aside from pitch detection, there are other tasks that are important for AMT. Of these, onset detection and instrument recognition are done almost sub-consciously when manually transcribed, however, these tasks are a challenge to automate when the music contains multiple voices, or contains instruments without any clear attack or decay. These tasks are often used in conjunction with pitch detection algorithms, and the information they provide goes towards a more musically meaningful transcription where mistakes are less of an issue. These tasks are worth

studying due to their close relationship to pitch detection, although investigating their relationship to group-wise transcription is beyond the scope of this thesis.

### 2.3.0 Onset detection

Onset detection is defined as the automatic detection of the start of note events, characterized by a sharp change in intensity, timbre or pitch of a sound [83]. For instruments like the piano, an increase in intensity at any pitch indicates the start of a new note, whereas for woodwind and stringed instruments, it is possible to change pitch with negligible intensity rise (known as *legato*). One of the main challenges that onset detection systems face is differentiating onsets from modulations caused by overlapping tones. This is more common in cases where more than one instrument is present, and so whilst success in onset detection in limited cases has seen success, a general purpose onset detector is still an open and challenging problem. As onsets occur over a number of audio oscillations, they can not be detected by differentiating the audio time-series. For example, the oscillation of the sound wave must pass through peaks and troughs at which point the time differential will be equal to 0, even during a period of intensity. This means that to detect onsets we must first transform the data into a representation that captures changes in intensity, the STFT for instance. From here an onset detection function or model can be applied that aims to extract the onset likelihood. The position of the onsets can then be determined by identifying and thresholding the peaks to obtain the position of the onsets.

Early onset detection functions split the STFT into bands tailored to onset detection. Klapuri et al. [83] used an auditory model to collect the STFT into bands before combining the results using intensity coding. Duxbury et al. [84] use a hybrid approach with two different detection functions tailored to the upper and lower frequency bands. Bello et al. [85] give a good review and tutorial of onset detection algorithms before 2005.

More recently, CNNs have proved effective in onset detection [86]. Schluter et al. trains a CNN on 10000 annotated onsets from a variety of instruments, and achieved an overall F1-score of 0.885% which constitutes the state of the art in the field. Their method uses multiple time resolutions of a mel-scaled STFT as the input to the network, and the network attempts to only predict the onset when centred in the frame.

For the instrument specific case of the piano, onset detection is considered solved, with methods such as NMF being able to detect onsets with an accuracy of 98.6% [87]. The reason for this is that the onsets are well defined and follow a clear timbre profile caused by the striking of the strings in the piano.

### 2.3.0 Instrument Identification

Instrument recognition, or musical instrument identification (MII), is typically done by matching instrument specific timbre templates to various acoustic features. This is a trivial task for isolated instruments, however is a challenging task in the case of polyphonic music. Several methods aim to use sound-source separation to separate out the contributions from each instrument first [88]. Other attempts include Kashino [89], who used adaptive template matching based on matched filtering for sound source identification, which is a more general field than MII. Kashino notes that it is difficult for humans to identify the instrument in a short window and a language model is needed to improve performance. As such, Kashino uses a Bayesian language model in combination with a frame-level model with a large increase in classification accuracy (60.8% to 88.5%). This confirms the conclusion made in the last section on language models, that a language model is needed for a fully functioning music transcription system.

$K$-nn classifiers have been used for instrument identification, by Fujinaga et al. [90] for real-time timbre recognition, and Eronen et al. [91] for instrument identification with cepstral coefficients and temporal features. Kitahara et al. [92] identifies the three main problems facing MII, and tackles them individually: feature variations caused by sound mixtures, the pitch dependency of timbres and the use of musical context.

As with the other fields of Music transcription, CNNs have had success in MII, see [93, 94]. These architectures will be looked at in more detail in Chapter 3.

### 2.3.0 Rhythm Transcription

Rhythm can be represented simply as the positions of the note onsets and offsets. However, this representation lacks the abstraction needed to be interpreted. It does not encode information such as the tempo or score arrangement that are necessary for rhythm [95]. Beat tracking is the extraction of the tempo and beat locations. The beat can be extracted directly from audio as in [96] where multiple agents are used for different beat hypotheses, which when combined give the beat locations. Scheirer [97] uses a small number of band-pass filters in conjunction with a bank of parallel comb filters to extract the beat from polyphonic music. His results show that the system performs as well as humans at guessing the time of the next beat. For a complete history of rhythm transcription pre-2005 see Gouyon et. al [98]. A variant of musically constrained NMF is used to to detect note onsets and offsets by Ochiai et. al [99].

## 2.4 Challenges For Transcription

In this section some of the challenges facing AMT will be highlighted, specifically challenges facing AMT from a data driven perspective. How these problems can be tackled will be suggested, and attempts to do just that will be mentioned.

Overlapping harmonics are the single largest issue for music transcription, and are what makes the problem non-trivial. Fourier series tell us that for two notes separated by an octave, the higher of the two will share all of its harmonic peaks with the lower note. This means there is no clear way to transcribe the notes separately. To accurately transcribe both notes, the transcription system must disentangle the individual contributing harmonic content.

Current transcription methods rely on frequency representations such as the STFT or the CQT with a number of bins that are spaced either linearly (STFT), logarithmically (CQT) or quasi-logarithmically (mel-scaled STFT or CQT). This is good for a global representation of any audio sample, however for a musical chord, the majority of bins in the representation will be empty, and the areas where the information is contained, will be low resolution. This leads to spilling across bins and inexact locations of harmonics, which without increasing the dimensionality of the STFT, is difficult to avoid. This is a difficult problem to solve as hand-crafted features are limited by design, and learned features can sometimes be uninterpretable. This is especially true when dealing with neural networks, a hand-designed feature representation, or an auditory modelled representation might not be the optimum way to represent data for deep learning. If the best way to represent audio for neural networks could itself be learned, such as Bengio et al. [100] propose, then this might alleviate this problem.

In the case of neural network based automatic transcription systems there is a core problem, termed the "entanglement" problem. This is when a model "abuses its power" for learning by remembering the harmonic power distributions of combinations of notes, instead of generalizing to unseen combinations. Kelz et al. [33] experiment with the issue, they remove certain individual notes, and combinations from the training data, and observe the results on those notes in the test set. What they confirm is that instead of learning a method for disentangling the input spectrum into notes that contribute different spectra, the network has just remembered combinations it has seen previously, and is unable to generalize to unseen data. What they suggest to alleviate this problem is the design of a custom loss function, one that can force the network to disentangle note combinations explicitly.

Generally all of the neural network based methods take in a spectrogram and no other prior knowledge. If they could be modified to take local contextual information such as the chord that is being played, or global information such as

the key-signature and time-signature of the piece, then the network could perhaps learn prior probabilities of its own, without the need for a separate language model (although Sigtia et al. [34] train a language model and frame model end-to-end, its not the same as a single model).

Kahino et al. [46] use both the phase and the magnitude in order to separate out notes with overlapping harmonics, stressing that "the relative phase of each frequency component should be taken into account". Considering the importance of phase in separating out overlapping harmonics, it is surprising that most approaches today, such as NMF and using deep learning and CNNs, discard the phase information. It is also unsurprising that the entanglement problem exists, given that the most important entangled information, the phase, has not been represented in a way that deep networks can understand. Perhaps this has been considered and attempted previously with no positive results, however there is a strong argument for the inclusion of phase is any frame-level model.

## 2.5 Summary

This chapter has attempted to provide a summary of past and present literature in the field of AMT, which provides the context required to investigate GWAMT in Chapter 5.

It was seen that the task of AMT can be separated into a number of sub-tasks including pitch-detection, onset detection and rhythm detection. The problem of AMT was split into context specific problems, which in the case of this thesis is polyphonic piano transcription. A similarity was then drawn between speech recognition and polyphonic music transcription where phonemes can be strung together to make sentences as notes can for melodies. Speech recognition can also be split into context-specific problems dependent on language and dialect. However, the success in the field of speech recognition has not translated across to AMT, due partially to a lack of interest and also differences in the spectral characteristics.

A clear distinction was drawn between bottom up, data-driven approaches to AMT, and top-down, knowledge driven approaches to AMT. Data-driven approaches, specifically learning based approaches to the problem of AMT have seen more success recently due to the advent of SVMs and deep learning. This thesis follows this trend and uses deep learning for the task of multiple F0-detection. Although it was shown recently that neural network based learning methods do not generalize well to unseen note combinations, they are still incredibly powerful classifiers and are suited to the problem. How this class of methods can be made to generalize better is partially covered in this thesis, with pre-training on generated data used to initialize the weights. This question is likely to be covered soon in

the field, as now the vast majority of methods competing at the 2017 MIREX competition are deep learning based methods.

Two key approaches to AMT, frame-level and note-level, were discussed, concluding that often note level systems produce more musically meaningful transcriptions, but perform worse on the frame level than dedicated frame-level models. In this thesis only frame-level transcription and group-wise transcription are considered. However, there is scope for work in this area, or for the post processing of frame-level transcriptions from multiple pieces using methods like HMM smoothing, as each piece provides another observation of the hidden state (note played). The MAPS database introduced in this chapter will be used for training and evaluating the AMT model developed in Chapter 3, so that the results can be compared to those in the literature.

# Chapter 3

# Deep Learning for Automatic Music Transcription

## 3.1 Introduction

The history of deep learning begins with the development of one of the first machine learning models, the perceptrons. They were developed in 1959 [101], inspired by the neuron connections in the brain, with the idea that a single "neuron" can have multiple inputs and produce only a single output. The parameters of the neuron were learned from data examples via a simple iterative procedure. The problem with the perceptron however, is that it can only learn to discriminate between linearly separated data. This means that in practice they are not particularly useful.

30 years later this issue was addressed with Artificial Neural Networks (ANNs) [102], which also follow the analogy with the human brain. The human brain consists of many neurons connected to each other and not a single unit. Following this, Rumelhart et al. [102] connected the output of multiple neurons to several other neurons, with each set of neurons termed a "layer". A series of linear transformations cannot resemble a non-linear function, so to add non-linearity an "activation" function was applied to the output of each neuron. A sigmoid function was used which also scaled the output between minus one and one. ANNs were limited in their usefulness at this time due to the "vanishing gradient" problem. This problem occurred when the networks were increased in their size beyond a few layers and is caused partially by the activation function used at the time, the hyperbolic tangent. Now, the ReLU activation function is used, which will be covered in greater depth later in this chapter. These ANNs are in essence the same as what are used today for deep learning.

Popularity for ANNs declined after Support Vector Machines (SVMs) became

usable for a wide range of machine learning tasks, such as the recognition of hand-written digits [103]. It was not until 1998 that neural networks were once again being considered with the continued development of CNNs by LeCun et al. [104]. They were designed to be scale/rotation invariant, suited perfectly for vision computing tasks. They demonstrated their CNN's performance on the recognition of hand-written digits, beating all other machine learning models of the time, including SVMs. The dataset used was called the MNIST database, and is still used today to benchmark new architectures.

In 2002 Restricted Boltzman Machines (RBMs) [105], which are a form of 2 layer network, became feasible to train [106]. From these RBMs, deep auto-encoders emerged [107], which are stacked RBMS that are trained in a greedy layer-wise manner to recreate their own input. This is done by training each layer individually to recreate the input, then fixing the weights on these layers and increasing the depth. It is termed "greedy" as the model is not trained end-to-end. This means they can learn to reduce the dimensionality of input data by finding a higher level feature representations in the intermediary layers. These were applied to machine learning tasks with unsupervised pre-training [108], where after learning a higher level feature representation the network is altered and then "fine-tuned" to classify the input samples.

It was after 2010, when the training of much deeper ANNs was proved possible, that the deep learning "boom" happened. Cirean et al. [109], showed that a deep network could be trained to recognize digits on the MINST database, achieving state of the art performance by using a GPU to speed up training time. The reason why it was not possible before was mainly due to the limited processing power and availability of large structured databases, as the methods of training, back-propagation of errors and gradient descent, are the same as was originally proposed in 1986 [102].

Since then the field has exploded in many directions, with applications in a variety of fields: reinforcement learning, where the world Go champion was beaten by a program, partly trained using deep learning [110]; computer vision, the automatic colourization of black and white images [111]; drug discovery, a DNN names Atomnet was able to predict candidate molecules for treating the Ebola virus, with one of candidates as the lead molecule awaiting animal trials; bioinformatics, a deep neural auto-encoder was used to predict gene ontology annotations [112] etc. The dominant field that deep learning has excelled in is of course computer vision, as we will see in the section on CNNs. As well as classfication tasks, deep networks are able to generate plausible unseen data with a large enough training set. These types of networks are called generative adversarial networks, and have recieved attention recently, see [113] for examples.

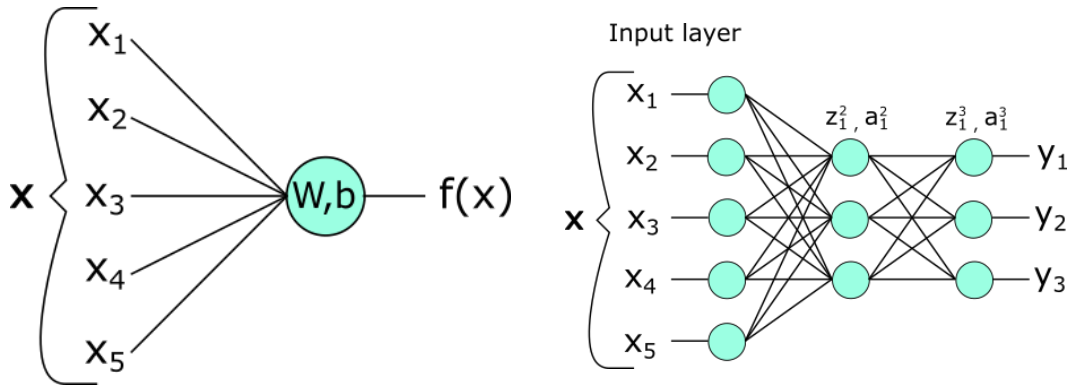


Figure 3.1: Left: A single neuron unit with parameters $\boldsymbol{W}$ and $b$. The lines connecting the inputs to the neurons are can represent the weights. Right: A two-layer ANN that transforms an array of inputs $\boldsymbol{x}$ into an array of outputs $\boldsymbol{y}$. An example for a neuron's outputs $z_i^l$ and activations $a_i^l$ is given above the top neuron in each layer.

## 3.2 Theory

In this section the mathematical theory underpinning deep learning and CNNs will be covered, including their training and how they are constructed. Common architectures for music related tasks will also be investigated, followed by detailing the architecture used to test the group-wise transcription hypothesis. How the network's performance is measured is described and compared it to other AMT networks.

### 3.2.0 Neural Networks

The raw building blocks of ANNs and CNNs are neurons. For a vector input $\boldsymbol{x}$, they take each element $x_i$, multiply it by a fixed weight $W_i$ and add a bias $b$. The weights and bias are learned during the training process, and are different for each neuron. The process that a single neuron applies to its input can be given as

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} W_i x_i + b, \tag{3.1}$$

where $f(\boldsymbol{x})$ is the neuron's output and $n$ is the number of inputs to the neuron. This can be seen graphically in Fig. 3.1

If several of these neurons are connected in parallel, so that the inputs go to several neurons, then this is called a layer. If the output of one layer is fed into another layer then this constitutes a basic ANN, this can be seen in Fig. 3.1. For a

multi-layer ANN, the pre-activation output of the $i$th neuron in layer $l$ is $z_i^l$, and the post activation output is given by $a_i^l$. The outputs of an ANN are $y_i$. The weights and biases for the neurons in each layer are given as $W_{i,j}^l$ and $b_i^l$. Now that the inputs, outputs and weights across the network can be represented, layer-wise operations can be performed on the layers input as

$$z^l = \boldsymbol{W}^l a^{l-1} + \boldsymbol{b}. \tag{3.2}$$

As mentioned in the introduction, ANNs cannot capture non-linear relationships without taking the output of each neuron and passing it through an activation function. So the function for our layer now becomes

$$a^l = \sigma(z^l), \tag{3.3}$$

where $\sigma$ is the activation function.

When training an ANN, such as a one layer network, similar to the one in Fig. 3.1, the goal is to determine a set of parameters $\boldsymbol{b}$ and $\boldsymbol{W}$ from a set of $N$ input-output data pairs $\{x_1, y_1, x_2, y_2 ... x_N, y_N\} = \{X, Y\}$ that minimize a loss function such as the mean squared error $\frac{1}{N}\Sigma_i \left\| y_i - \sigma(x_i W + b) \right\|^2$. Given enough data, and a network with a sufficient number of parameters, the network should be able to approximate to unseen data. The loss function itself is $l$ which when evaluated over the data, can be expressed as

$$L^{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{N}\Sigma_i [l(f_{\boldsymbol{W},\boldsymbol{b}}(x_i), y_i)]. \tag{3.4}$$

Previously the mean-squared-error loss function was given as an example, however for regression a more common loss function to be used is the Euclidean loss

$$L^{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{X}, \boldsymbol{Y}) = \frac{1}{2N}\Sigma_{i=1}^{N} \left\| y_i - \hat{y_i} \right\|, \tag{3.5}$$

which is generalized to an arbitary model with predicted outputs $\{\hat{y_1}, \hat{y_2}, ... \hat{y_N}\}$ from inputs $\{x_1, y_1, x_2, y_2 ... x_N, y_N\} = \boldsymbol{X}$.

Often a loss function alone will overfit to the training data. Overfitting means that the model will simply memorize the input data and not generalize towards the inference function mapping any input to any output $f : X \to Y$, in other words, the loss over the training data $\boldsymbol{X}, \boldsymbol{Y}$ increases but the loss on the validation data $\boldsymbol{X_{val}}, \boldsymbol{Y_{val}}$ decreases or remains unchanged. The validation data is kept separate from the training data to observe the performance of the network as it learns. One method to counter over-fitting by the loss-function is to add $L_2$ regularization, otherwise known as weight decay. This is done by adding an additional term to the loss function dependent on the weights

$$L^{\boldsymbol{W},\boldsymbol{b}}(\boldsymbol{X},\boldsymbol{Y}) = \frac{1}{N}\Sigma_i[l(f_{\boldsymbol{W},\boldsymbol{b}}(x_i), y_i)] + \frac{\lambda}{2n}\Sigma_{i,j,l}W^2_{i,j,l}. \tag{3.6}$$

The effect of this is to force the network to prefer small weights when learning, as large weights would increase the cost function. This helps to reduce over-fitting because the weights are limited to being small, so the effect of the gradient on the relative size of the weights is always meaningful, it reduces the search space and stops the cost function getting trapped in local minima. From this point the loss function refers to the calculation of error between the input-output samples, and the cost function refers to the regularized loss function.

Having a loss function that tells us how the network has performed over a particular subset of samples is useful for evaluation, however knowing this alone cannot improve the network. The original proposal for ANNs in 1986 [102] proposed that these networks were trained via the backpropagation of error. The basic idea is that the gradient of the cost function can be found with respect to the each weight $\frac{\partial L}{\partial W^l_{i,j}}$, therefore obtaining the direction and relative magnitude to alter the weights so as to improve the result of the cost function on the training data, and hopefully the network's performance on the validation data.

Before going into more detail about the back-propagation algorithm, how the weights are updated is explained, so as to understand why the gradient is needed. Gradient descent is the method by which neural networks are optimized, by following the gradient of the weights "downhill" we hope to reach a global minimum of the cost function. The variant of gradient descent most commonly used is Stochastic Gradient Descent (SGD), the difference being that gradient descent finds the gradient over all of the training examples before updating the weights, where as SGD finds the gradient over a single example and iterates over the entire training set. The reason SGD is preferred is twofold, firstly it is more convenient to process a single example at a time due to the computational requirements of storing the data for processing, and secondly it adds random noise into the training process, which tends to help the network reach a minimum quicker than otherwise. A single iteration of gradient descent to update the weights and biases is given by

$$W^l_{i,j} = W^l_{i,j} - \eta\frac{\partial L^{\boldsymbol{W},\boldsymbol{b}}(x_i, y_i)}{\partial W^l_{i,j}}, \tag{3.7}$$

$$b^l_{i,j} = b^l_{i,j} - \eta\frac{\partial L^{\boldsymbol{W},\boldsymbol{b}}(x_i, y_i)}{\partial b^l_{i,j}}, \tag{3.8}$$

where $\eta$ is the learning rate which is a global network parameter that changes the rate at which the network learns. Set this too high and the network will fail to reach a decent minimum, too low and the network will take a long time to train.

The gradients in Eq. 3.7 are given with respect to a single example $\{x_i, y_i\}$, and are unknown, requiring back-propagation to be found.

A measure for how much each neuron is responsible for errors in the the output is needed to calculate the gradients of the weights and biases. This measure is called the error term and is given as $\delta_i^l$ for each neuron $i$ in layer $l$. For the output neurons this is simple enough to define in relation to the difference between the predicted output and the target output

$$\delta_i^{n_l} = \frac{\partial}{\partial z_i^{n_l}} L(x_i, y_i), \tag{3.9}$$

where the derivative of the loss function with respect to each output unit is calculated.

For propagating the error back further, there will be multiple neurons connected to one neuron on the previous layer. To combine the error from each of those connected neurons a weighted average is calculated as

$$\delta_i^l = (\sum_{j=1}^{p_{l+1}} W_{j,i}^l \delta_j^{l+1}) f'(z_i^l), \tag{3.10}$$

where the function $f'(z_i^l)$ is the derivative of the activation function applied to the outputs of the neurons in layer $l$. Eq. 3.10 takes the error from the layer $l+1$ and propagates it back to the layer $l$ using the weights to calculate the new error, this process is repeated after the error for the output units is calculated using Eq. 3.9 until the error terms have been calculated for neuron in each layer. Next the partial derivatives needed for calculating the updates to the weights and biases can be calculated using

$$\frac{\partial L^{\boldsymbol{W},\boldsymbol{b}}(x_i, y_i)}{\partial W_{i,j}^l} = a_j^l(\boldsymbol{x}) \delta_i^{l+1}, \tag{3.11}$$

$$\frac{\partial L^{\boldsymbol{W},\boldsymbol{b}}(x_i, y_i)}{\partial b_{i,j}^l} = \delta_i^{l+1}. \tag{3.12}$$

The weights and biases for the network can now be updated using Eq. 3.7 thereby reducing the value of our cost function on that example. The process of updating the parameters using a single input-output example constitutes one iteration of SGD. All of the training data examples would be iterated over, updating the weights in each case hoping that the network will learn a more meaningful representation each time so it can improve its performance on the unseen test data.

If the parameter space is represented as a smooth two dimensional surface, that has peaks and troughs, like a landscape with hills and valleys, then it becomes easier

to understand what the optimization procedure is doing in its attempt to reach the global minimum. In the case of SGD, the algorithm is approximating the gradient around its parameters using the current training example, then following this gradient "downhill", asymptotically approaching a local minimum or saddle point. This means that the algorithm is prone to getting trapped in one of the infinite local-minima that all networks with greater than two layers suffer from. There have been several attempts to rectify this problem by altering the method by which the weights are updated. The most common of these is SGD with momentum [114], where an additional term is included in the update rule that is a fraction of the previous update. The result in the path followed by the optimization is similar to a ball rolling down our parameter space. This allows the optimization to escape local minima, assuming that there is enough momentum to escape.

There are many other variants of SGD, each of which has advantages in particular applications, for information on gradient optimization algorithms see the article by Ruder [115]. The variant that has shown success recently is ADAM (ADAptive Moment estimation) [116] and is used throughout this thesis. ADAM stores an exponentially decaying average of the previous gradients and squared gradients

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\frac{\partial L^{\boldsymbol{W},\boldsymbol{b}}(x_i, y_i)}{\partial \boldsymbol{\theta}}, \tag{3.13}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)\left(\frac{\partial L^{\boldsymbol{W},\boldsymbol{b}}(x_i, y_i)}{\partial \boldsymbol{\theta}}\right)^2, \tag{3.14}$$

where $\boldsymbol{\theta_t}$ are the parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ for the entire network at a timestep $t$, $m_t$ and $v_t$ are the first and second moments which correspond to the mean and the variance respectively, $\beta_1$ and $\beta_2$ are decay factors for each moment, and are recommended to be kept at 0.9 and 0.999 [116]. ADAM uses these to weight the learning rate for each update. This allows each neuron to have a different learning rate, one that is customized to its previous history. When initialised, $m_t$ and $v_t$ are zeros and are biased towards zeros in the early stages of trianing, to counteract these, ADAM calculates the bias-corrected first and second moments:

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{3.15}$$

$$\hat{v_t} = \frac{v_t}{1 - \beta_2^t} \tag{3.16}$$

These can then be used to update the parameters of the network using the ADAM update rule

$$\boldsymbol{\theta_{t+1}} = \boldsymbol{\theta_t} - \frac{\eta}{\sqrt{\hat{v_t}} + \epsilon}\hat{m}_t. \tag{3.17}$$

### 3.2.0 Convolutional Neural Networks

Convolutions neural networks, or CNNs, are a form of constrained artificial neural network. The neurons in a layer are constrained to seeing a smaller part of the input at any one time. The weights are shared across the inputs by common *filters* being applied via convolution. This constraint causes deep networks to progressively build up more complex representations of the input as layers are staked together. CNNs are well suited to computer vision related tasks due to their ability to detect patterns, and being translational and rotational invariant, akin to human vision. Partly biologically inspired, CNNs behave similarly to how we believe the human visual cortex processes inputs, simple cells detect edges, which in turn pass this information onto complex cells that are only locally connected [117]. This process is similar to how a CNN processes the input image. Convolutional layers detect simple patterns in each layer that are compounded into more complex patterns in each successive layer.

They were originally introduced by LeCun et al. [118] for handwritten digit recognition, who used a shallow network capable of being trained by backpropagation at the time. His later work in the same field [104] proposes a deeper network with 7 layers of tunable parameters to recognise handwritten numbers in 32 by 32 pixel images.

The explosion in the use of CNNs was aided by the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [119], which is a competition where teams compete to achieve the best classification accuracy using a set of training and test data provided by image-net. The competition started in 2010 however it was not until 2012 that the first CNN entry was seen with AlexNet [120]. This pioneering work demonstrated that very deep convolutional networks could be trained for the first time and with astonishing accuracy, their top-5 error at the image-net 2012 competition was 15.4%, compared to the next best entry at 26.2%.

Some other successes since then include GoogLeNet [121], which won the ILSVRC in 2014 with their 22 layer network. Their network contained multiple parallel layers, and showed that it was possible to have several convolutional layers in parallel instead of sequential with its inception module. This paved the way for many much deeper architectures, and proved that deeper was indeed better however due to its complexity and depth, it was not as practical as some other networks. Another entry to the competition that year was VGG-net [122], which was smaller than GoogLeNet with both less layers and smaller filters, and much simpler with no complex inception module. This made it practical and transferable to other areas, therefore it received far more attention than the winner GoogleNet.

One more piece of work in computer vision is described as it has influenced the design of the network architecture developed in this thesis. He et al. [123] developed the architecture called res-net. The novelty of this work was in the use

of "residual" connections between layers, which allow the layers to retain part of their input. This allowed much deeper networks than had been seen before. They successfully trained a 1000 layer network using these connections and won 1st place at the ILSVRC 2015 classification competition.

As we saw in Chapter 2 deep learning and more specifically CNNs, have seen success in most areas of music transcription, including instrument recognition [93, 94], frame-level polyphonic music transcription [32] and onset detection [86]. Success has also been achieved in other areas of music informatics such as genre recognition [124] and mood classification [125]. For more information on deep learning in music informatics consult [126].

CNNs are particularly suited to the field of music informatics for the same reason they are to vision computing: their ability to recognize patterns globally; being invariant to translation, scaling and rotation. What we define as genres have different power distributions in their frequency representation, what we hear as different instruments too have clear timbre profiles. Patterns are seen in all aspects of music, both locally in the structure of notes and harmony, and globally in the structure of melody and its progression.

The following sections contain how CNNs are designed and built, and the constituent building blocks that are necessary for a CNN. The back-propagation algorithm tailored to CNNs is not covered, however the principle is the same as above, to find the error term for each neuron (which are called filters for CNNs as we will see) then to calculate the weight updates for that neuron.

### Basic Building Blocks

Two terms are defined, *forward pass* and *backward pass*. A forward pass through a network is the calculation of each of the layers' activations in order to calculate the output. The backward pass is the back-propagation of error through the network in order to calculate the derivative and the weight updates for each neuron.

#### *Convolutional Layers*

Convolutional layers are based on the convolution operation outlined in Fig. 3.2. The neurons in CNNs are three dimensional, and consist of stacked two dimensional "filters" containing the learned weights and biases. The input and output of convolutional layers are generally 3 dimensional, with two spatial dimensions and one for channels. For each channel, the spatial dimensions contain the result of the convolution operation performed by a single filter. The filters have a much smaller receptive field than in ANNs, however by stacking convolutional layers the receptive field of the original input is increased.

Each output channel can be thought of as an activation map for one filter, with the magnitude at the points in the spatial dimension being a measure of how much the input correlates with that filter. This is one reason why CNNs are excellent at classification.

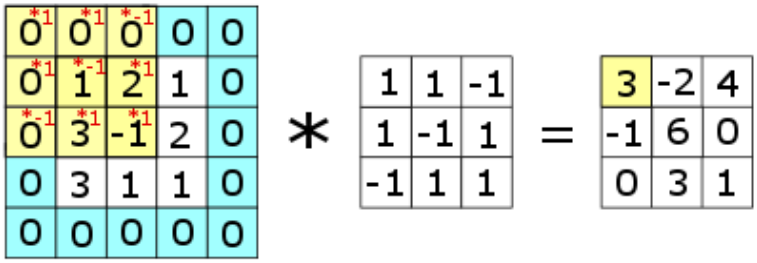


Figure 3.2: The convolution operation is applied to the array on the left with the $3 \times 3$ filter in the centre. In this case the padding is shown in blue, and an example of the results of one of the multiplications is shown in yellow. To retain the same dimensionality, generally odd-sized filters are used in conjunction with padding of the input with zeros, as seen in this example.

As an example, consider the case where the input for the convolutional layer is an $N \times N$ image $\boldsymbol{x}$. This is analogous to the first layer in a CNN, or when considering a single channel from a previous layer. For an $m \times m$ filter $w$ where $m$ is odd, the input is padded with $(m-1)/2$ zeros in each dimension, resulting in its size being $(N + (m-1)/2) \times (N + (m-1)/2)$. For each output neuron the convolution operation with the input and the filter is expressed as

$$z_{ij}^{l} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} w_{ab} x_{(i+a),(j+b)}. \tag{3.18}$$

Convolutional layers have a number of parameters that are determined either manually or experimentally and can often be quite different depending on the application of the network and the position of the layer in the network. These parameters are:

- The size of the filters, these are usually small for image recognition, but larger for music applications
- The number of filters in the layer, the more filters the higher the number of parameters in the network
- The padding, usually the padding is determined by the size of the filter as in the example in Fig. 3.2, in some cases using no or little padding can be used to reduce the size of the input in conjunction with pooling layers

- The stride, the stride is the "shift" that the filter moves between adjacent convolution operations, with a stride of one, the dimensions of the input are preserved, however the stride can be increased to reduce the size of the input

*ReLU*

As with ANNs, after convolutional layers an activation function is applied. Previously the sigmoid function was used, however recently the Rectified Linear Unit (ReLU) activation function has been used to speed up the training process, as it helps to avoid the vanishing gradient problem [127].

The ReLU function can be expressed simply as

$$f(x) = \begin{cases} x & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases} \tag{3.19}$$

and can be seen in Fig. 3.3.

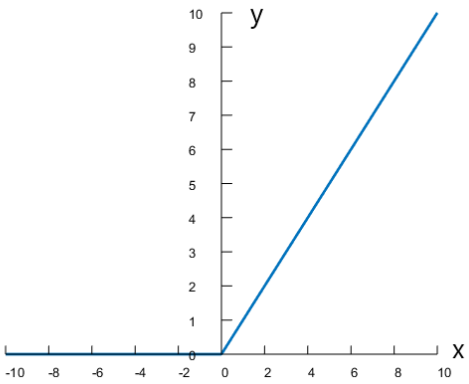


Figure 3.3: The ReLU function is displayed above.

An issue that ReLU units can encounter is the "dying ReLU" problem. When this happens, the majority of the inputs to the unit are negative, meaning there is an insufficient propagation of the input, and no gradient to allow the network to recover. A variant of ReLU exists that counters this problem called "leaky ReLU". This variant only changes what happens to negative inputs, instead of forcing all negative inputs to 0, the gradient is drastically reduced. The reformulated ReLU function is now

$$f(x) = \begin{cases} x & \text{for } x > 0 \\ \frac{x}{a} & \text{for } x \leq 0 \end{cases}, \tag{3.20}$$

where $a$ is the factor by which the gradient is reduced below zero. This was originally suggested to be set to a high number, however recently has been shown to be more effective at a lower value such as 5.5. [128].

*Pooling*

The outputs of CNNs are generally much smaller than the number of inputs, for example a spectrogram could have a size of $400 \times 50$ with the output being 88 note classes. This reduction in the size of the input happens in the pooling layers. The pooling layers take the maximum or average value within a sliding window. To reduce the size, the pooling layers use striding. Striding is when the pooling window is only applied to evenly spaced points in each dimension. Using a value for the stride larger than 1, the size will always be reduced. This is illustrated using a similar example as Fig. 3.2 in Fig. 3.4.

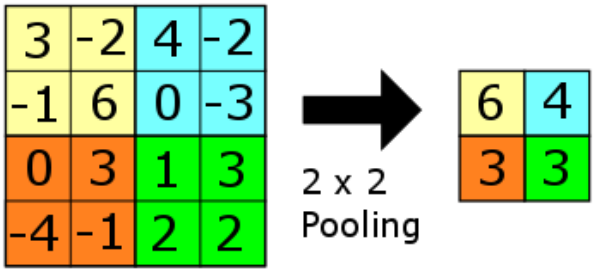


Figure 3.4: The max-pool operation is displayed above, with $2 \times 2$ pooling applied to a $4 \times 4$ input. The colours on correspond to the different pooling regions.

The parameters that can be set for pooling layers include the pooling size, the stride and the type (either max or average pooling). For this thesis max-pooling is used, this makes sense as it is the regions of maximal activation (peaks in the spectral domain) that contain more information about the presence of notes in the signal.

*Batch Normalization*

For much deeper networks that have many more parameters, not only are the data volume requirements and training time much higher, but they are prone to over-fitting, meaning they can learn to remember the training data, providing a shortcut to minimizing the cost function. Batch normalization [129] is a method for adding regularization and also helps to reduce the training time. The inputs

to each layer are normalized across each batch, which consists of several training examples, therefore making the distribution of the inputs to each layer constant. This has shown to be highly effective when skip connections are used to learn residual mappings [123]. The batch normalization function is given as

$$(BN)_{\gamma,\beta}(x_i) = \gamma\hat{x_i} + \beta \tag{3.21}$$

Where $\hat{x_i}$ is the normalized $i$th input to the layer, and $\gamma$ and $\beta$ are learned scale and shift parameters. The learned parameters allow the layer to become more effective during training. Batch normalization is usually applied after activation

*Loss Layers*

In Neural networks and indeed CNNs, the loss layer is a Layer after the final output of the network that computes the value of the cost function on the forward pass through the network, and on the backward pass calculates the derivatives of the cost function with respect to the input, for each output neuron.

In the previous example, the Euclidean loss was used as a loss function for the network. However when this is used with the sigmoid non-linearity it leads to a very slow training when the values of the outputs are close to one, this is simply because the gradient of the sigmoid activation function plateaus towards one. An alternative to the Euclidean loss function is the cross-entropy loss function

$$L(\boldsymbol{X}, \boldsymbol{Y}) = -\frac{1}{n}\sum_{x}\sum_{j}(y_j \ln a_j^L + (1 - y_j)\ln(1 - a_j^L)) \tag{3.22}$$

which is the loss summed across all training examples and all output neurons.

Even when the sigmoid function is not used as an activation function, the sigmoid function can be applied to the outputs and the cross-entropy used. This, it turns out, is still a better alternative to the Euclidean loss function. For more information about the cross-entropy loss function including its derivation and how the gradient is calculated for use in backpropagation, consult [130].

*Other Layers*

An alternative to batch normalization for regularization is "Dropout". Dropout adds noise into the training of deep networks by randomly setting some of the inputs to a layer to 0. The number of neurons which are "turned off" is determined by a manually assigned parameter.

In most convolutional architectures it is common to reduce the size of the input throughout the convolutional and pooling layers, and to maintain dimensionality increase the number of filters. When the propagated input has become sufficiently

| Layer-Type | Filter/Pooling Size |
|---|---|
| Convolution | $3 \times 3$ |
| Pooling | $3 \times 3$ |
| Convolution | $3 \times 3$ |
| Pooling | $3 \times 3$ |
| ⋮ | ⋮ |
| Fully Connected | 256-1000 |
| Loss Layer | Global |

Table 3.1: Skeleton architecture for models in [93, 33]

small to reduce the number of parameters required, "fully-connected" layers are used to connect all of the filter neurons to every neuron in a layer identical to one seen in an ANN. This connects parts of the input that may not have been due to a small receptive field. Fully connected layers also convert the feature representation learned by the network into a abstract representation, helping the model separate categories with only subtle differences.

## 3.3 Deep Learning for AMT

### 3.3.0 Architectures

When considering an architecture to use for frame level transcription, other architectures that had seen success in AMT tasks, as well as other music informatics tasks were studied. In general most of the architectures followed the same skeletal structure as illustrated in table 3.3 [93, 33], with features like small $3 \times 3$ filters and increasing numbers of filters similar to architectures in the field of vision computing, such as VGG net [122]. Whilst it makes sense to exploit the success seen in other research areas, and guarantees that the architecture should work reasonably well, it assumes that the properties of spectrograms displaying different notes are similar to what might be seen in a computer vision classification task. A better approach to the problem is to consider the differences between the tasks being attempted in AMT and those in computer vision, then from this understanding, adapt architectures that have shown success elsewhere. This is what is attempted in developing the architecture in this thesis.

There are three notable architectures used specifically for piano transcription in the literature. The first was introduced by Kelz et al. [32], where he used convolutional and pooling layers to reduce the size of the input, whilst increasing the numbers of filters to maintain the dimensionality until the fully connected layers. Another model designed by Kelz et al. [33], is named AUNet. This is a much

deeper network based on a network used for medical image segmentation. The advantages of this architecture are that information can skip across layers of the network, a feature that is employed here in a different fashion. In his network he concatenates feature maps from earlier layers to the current feature map, which he claims is beneficial for smoothing in the time dimension. The final architecture to be mentioned is one available commercially called AnthemScore [131], it is difficult to compare this architecture directly to any other methods as it has not been evaluated on any public databases yet, however some points on the architecture are given in an accompanying blog [131]. The prominent features are the use of "skip" connections over every other layer, as introduced in Resnet [123], and the use of separate time and frequency convolutions. One final feature to note is that no fully connected layers are used. These features are shared by the architecture used here, but implemented differently, they are as follows

**Skip Connections** Skip connections are a method of propagating the representation at one layer in the network $x^l$ to another layer $x^{l+n}$. They were first proposed in [123], but have since been researched widely. How a single convolutional "block" uses these connections is illustrated in Fig. 3.5.

**Time and Frequency Convolutions** In the majority of deep learning applications two dimensional filters are used as the input dimensions are highly correlated. For music transcription this is not necessarily the case. For the piano, each note is produced by a separate mechanism, therefore the pitch and time dimensions are much less correlated. Other instruments sometimes exhibit chirps during their onset or offset and so this architecture might not be as effective.

Therefore convolutions are performed in the time domain in separate layers to frequency convolutions, also the size of the convolutions in the time domain is limited, it was found that large filters here lead to a higher training error than otherwise.

**No Fully Connected layers** If the goal of deeper convolutional layers is to develop an increasingly higher feature level representation then fully connected layers are not needed for piano transcription. Excluding onsets, given a piano transcription, a spectrogram could be constructed that strongly resembles the original. This is not the case in computer vision when classifying into abstract categories, i.e. cats do not all look the same, but all middle C notes happen with partials at similiar frequencies. This is demonstrated in this thesis with the architecture's performance in the results section of this Chapter.

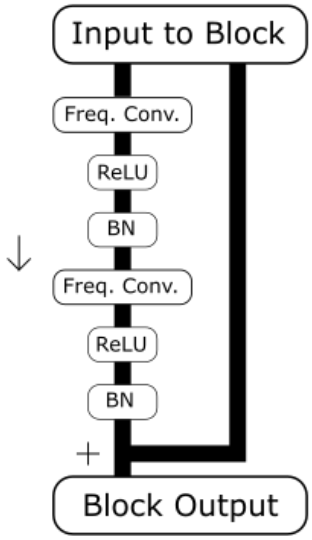


Figure 3.5: A block is a section of a ANN or CNN that consists of several layers. The block in this figure is repeated a total of 10 times in the network architecture, repeated 5 times consecutively at two points. The label "BN" indicates a batch normalization layer, and the label "ReLU" indicated a ReLU activation being applied. The input feeds the channel on he left, and is added to the output of the left channel after the second BN layer. This figure highlights how skip connections are used in the network. Where the lines connect the input is added to the transformed input to give the output of the block. The frequency convolutional layers are indicated by "Freq. Conv." Skip 2 connections are used as in [123].

The full architecture is listed in Section 3.3 and visually represented in Fig. 3.6. The network is trained using multiple note training, similar to [58], and will be detailed in this section along with how the multiple hyper-parameters are optimized.

### 3.3.0 Evaluation and Dataset

The output from any frame-level model is an array of note probabilities for the input frame. After thresholding, which will be covered later, the result is a binary array indicating the notes detected within that frame. For evaluating the performance of the network on a given selection of frames with corresponding ground-truth labels, following [132], the metrics used are:

**Precision** The precision is a measure of how precise the system is, i.e. what

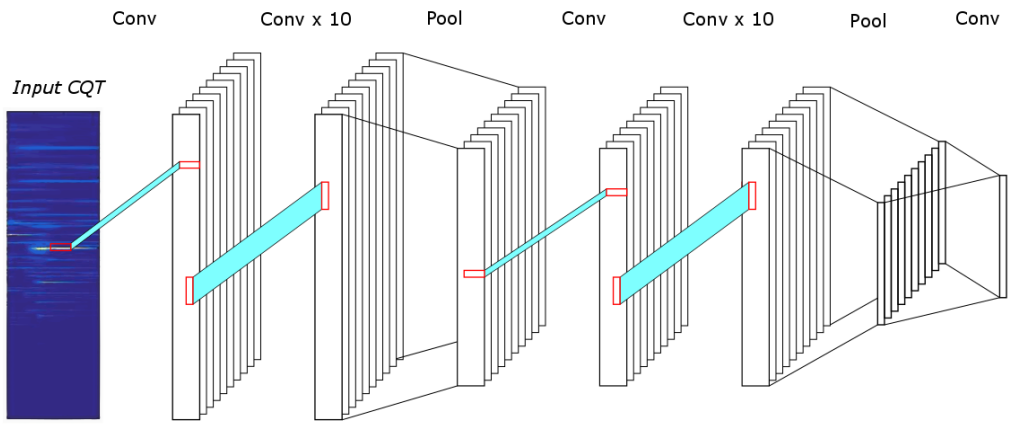


Figure 3.6: The above figure illustrates the stacked filter maps for each layer of the network, and the process of obtaining the final output. The labels at the top of the figure indicate the type of layer convolutional (Conv) or pooling (Pool), see Figure Section 3.3 for more detail on the architecture. For the vertical convolutions there are many stacked layers, connected with skip connections which are omitted in this diagram for simplicity, see Fig. 3.5 for how the frequency convolutional layers are built.

percentage of the predicted notes are correct, ignoring false negatives

$$\text{Precision} = \frac{\Sigma_{t=1}^{T} TP(t)}{\Sigma_{t=1}^{T}(TP(t) + FP(t))}. \tag{3.23}$$

**Recall** The recall is a measure of the scope of the system i.e. what percentage of the ground truth notes were predicted, ignoring false positives

$$\text{Recall} = \frac{\Sigma_{t=1}^{T} TP(t)}{\Sigma_{t=1}^{T}(TP(t) + FN(t))}. \tag{3.24}$$

**F1-measure** The F1-measure is a combined measure of the precision and recall

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}, \tag{3.25}$$

and is generally used to compare frame-level models, however at the MIREX competition it is the accuracy (below) that is used.

**Accuracy** The accuracy is an overall measure of the performance, taking into account the false positives and false negatives

$$\text{Accuracy} = \frac{\Sigma_{t=1}^{T} TP(t)}{\Sigma_{t=1}^{T}(TP(t) + FP(t) + FN(t))}. \tag{3.26}$$

| Layer | Parameters | Input Size |
|---|---|---|
| Time convolution | $1 \times 5$ convolution, 10 filters, pad $[1, 2]$ | $352 \times 55 \times 1 \times 1$ |
| ReLU | - | $352 \times 55 \times 10 \times 1$ |
| Batch normalization | - | $352 \times 55 \times 10 \times 1$ |
| Convolutional block, repeated 5× | see below | $352 \times 55 \times 10 \times 1$ |
| Pooling | $2 \times 5$ max-pooling | $352 \times 55 \times 10 \times 1$ |
| Time convolution | $1 \times 5$ convolution, 10 filters, pad $[1, 2]$ | $176 \times 11 \times 10 \times 1$ |
| ReLU | - | $176 \times 11 \times 10 \times 1$ |
| Batch normalization | - | $176 \times 11 \times 10 \times 1$ |
| Convolutional block, repeated 5× | see below | $176 \times 11 \times 10 \times 1$ |
| Pooling | $2 \times 11$ max-pooling | $176 \times 11 \times 10 \times 1$ |
| Convolution | $1 \times 1$, 1 filter | $88 \times 1 \times 10 \times 1$ |
| Convolutional block | | |
| Skip start | - | - |
| Frequency convolution | $13 \times 1$ convolution, 10 filters | - |
| ReLU | 0.3 leak | - |
| Batch normalization | - | - |
| Frequency convolution | $13 \times 1$ convolution, 10 filters, pad $[7, 1]$ | - |
| ReLU | 0.3 leak | - |
| Batch normalization | - | - |
| Skip end | - | - |

Table 3.2: The network architecture, and the parameters used throughout is shown in this figure. The bottom half of the table defines a convolutional block (see Figure Fig. 3.5) that is repeated a total of 10 times throughout the network. The start and end of the block are bookended by skip layers, which indicate where the input to the block is connected to the output of the block, in the skip end layer the representation at the last skip start layer is added to the current representation. The column on the right gives the size of the input into that layer (and the size of the output from the previous layer)

The number of true positives $TP$, false negatives $FN$ and false positives $FP$, are given as functions of the frame number $t$ out of the total number of frames $T$. Therefore their sums are the total number over the entire piece being evaluated.

All of the above metrics are used to give a score between 0 and 1 for a given transcription, with 1 being a perfect transcription. Whilst the accuracy and F1-measure perform a similar job of measuring the effect of false positives and negatives, the F1-measure prefers there to be a similar number of both, and penalizes a system that gives more false positives than false negatives, whereas the accuracy does not discriminate between false positives and false negatives.

The evaluation method follows the configuration set-up by Sigtia et al. [34]. They detail two evaluation strategies for automatic piano transcription systems.

In both cases they use the MAPS data base. In the first they split the entire database into 4 train/test splits, this means that the test data will contain pianos that have been used to train the systems, they propose this method for specifically training language models. They devise the second evaluation strategy to be more realistic, and to evaluate frame-level models. For training they use the 210 tracks from synthesized pianos, and for evaluation they use the remaining 60 tracks obtained from Yamaha Disklavier piano recordings. They call this second method of evaluation "Configuration 2". It is this method of evaluation that will be used throughout this thesis.

Of the 210 pieces from synthesized pianos, 30 are used for validation and the remainder used for training. Frames for classification by the network are sampled from the spectrograms with a size of 100 ms (55 time-steps) and a hop size of 66 ms (36 time-steps), which corresponds to a 74% overlap. The corresponding labels are obtained in a similar fashion from the provided MIDI files, selecting the notes played in the MIDI track at the corresponding times to each frame. Across the entire test set the total number of false positives, true positives and false negatives are used to calculate the precision, recall, accuracy and F1-measure using Eq. 3.23 to Eq. 3.26.

### 3.3.0 Training

Typically deep networks are trained on large amounts of data, data that is usually too large to store on the RAM memory in its entirety and as such has to be loaded in for each training batch. One method that can be used to speed up the training time is by using dynamic training. The idea is that if a smaller subset of data can be stored on the RAM that can be sufficiently augmented so that it can cover a much larger volume of the input space, then the network can fabricate its own data from the smaller subset. This training strategy has several advantages: increased training speed, by removing the need to continuously transfer data to and from the RAM the bottleneck is removed and allows much faster training; unlimited data, if the data subset can be sufficiently and effectively augmented then this overcomes the issue of having little available data; more adaptable, if the training algorithm is sophisticated then the function used to fabricate data can adjust the distribution of data examples generated based on how the network is performing, e.g. if the network is struggling to detect the note G6 then the network could be altered so the probability of G6 occurring in each training example is marginally higher. A reason why dynamic training is well suited to music related problems is because music in fundamentally generative in nature; if the generation can be mimicked then so can the recorded data.

However, although dynamic training has its advantages is also suffers from several problems. Due to the limited size of the data subset, over-fitting is highly

| Parameter | Value |
|---|---|
| Bins per octave | 48 |
| Minimum Frequency | 31 Hz |
| Maximum Frequency | 19 kHz |
| Gamma | 4 |

Table 3.3: Parameters for the CQT transform used to transform the training examples.

likely in some cases. In music classification this is less of a problem as the harmonics are overlapping, however a problem that was found during training is the over-fitting to artifacts in the upper range of the training data. Another issue is the inability to mimic the variation in data found in true example cases. This would be a problem in the case of vision computing or a task where the main difficulty is capturing the variation in the input. This was not found to be much of an issue here.

The note dataset consists of recordings of a single piano, with each note played at various dynamics. Each clip is 2 seconds long and converted to mono after being recorded in stereo at a frame-rate of 44100 Hz. The CQT is applied to each clip individually using the parameters shown in Table 3.3. This results in a complex valued array of 445 frequency bins by 1098 time bins, with each time bin corresponding to 1.8 ms of audio. Due to the nature of the CQT there is spilling across frequency bins that causes each to affect the majority of the spectrum, such that each frequency bin has some information about its surrounding context. More information on the CQT and its properties can be found here [23]. The CQT is used because of its note translation properties as discussed in Chapter 1. The higher frequency bins above bin 352 (4927 Hz) are discarded. This means that there are no second harmonics for the final octave on the piano, however for the highest notes on the piano it is the fundamental that has the highest power content. The motivation for doing this is twofold. Firstly there are non-harmonic artifacts above this frequency that allow the network to over-fit to the training data, and secondly a cutoff at this frequency means that the range of frequencies almost matches the range of the fundamental frequencies in the output notes, therefore little compression of the frequency axis is needed throughout the network. Specifically 352 bins were used because this is exactly 4 times the number of output note classes, allowing two pooling layers to map the input frequency range onto the output note frequency range.

To generate training samples, the network randomizes the number of notes to be generated, $n$, up to a maximum of 6 then chooses $n$ random integers from between 8 and 88 which correspond to notes on the piano, not allowing duplicates. For each note, the volume is randomized above the lowest 4 volume samples. From

**Algorithm 1** The method for generating training samples is outlined. The function *RandInt* generates a random integer between the two arguments inclusively. The function *PianoLibrary* returns the CQT of a 2 second note sample from the pre-computed piano library, the argument specifies the MIDI number of the note to be returned. The function *RandomRange* returns a random range of 55 CQT time-steps from the input spectrogram. The output of the algorithm, *sample*, is a real-valued array that is then used to train the frame-level transcription model developed in this thesis.

$n \leftarrow \textbf{RandInt}(1, 6)$
$sample \leftarrow$ *Initialise empty array to hold the output sample*
$i \leftarrow 0$
**while** $i <= n$ **do**
    $note \leftarrow \textbf{RandInt}(8, 88)$
    $noteRecording \leftarrow \textbf{PianoLibrary}(note)$
    $sample \leftarrow sample + \textbf{RandomRange}(noteRecording)$
    $i \leftarrow i + 1$
$sample \leftarrow \textbf{abs}(sample)$
$sample \leftarrow \textbf{log}(sample/\textbf{std}(sample))$
$sample \leftarrow sample - \textbf{mean}(sample)$

the corresponding CQT clip a 55 frame segment (74 ms) is sampled. The random samples for each note are summed to simulate a clip of a person playing several notes of a chord sequentially. The absolute value of the segment is taken, discarding phase information due to the inability of unaltered deep networks to interpret complex valued inputs. The segment is then divided by the standard deviation of all its values. The reason for doing this is make the input to the network more consistent as it does not have the opportunity to capture the variance in real data. Next a logarithm is applied to allow much quieter harmonics to be more easily detected as can be seen in Fig. 3.7. Finally the mean of the sample is subtracted, this appears to improve the training speed without affecting performance. One reason for this is potentially due to the randomly initialized weights which are able to be both positive and negative. This procedure for generating samples is outlined in Algorithm 1. The described procedure specifies how to generate the input to the network, the output label for the data is represented by a binary array of length 88, with each value representing the presence of a note in the generated sample.

The network is initialized using random weights sampled from the following distribution recommended in [133] for when using ReLU activations

$$\frac{1}{2} n_l Var[W^l] = 1, \tag{3.27}$$

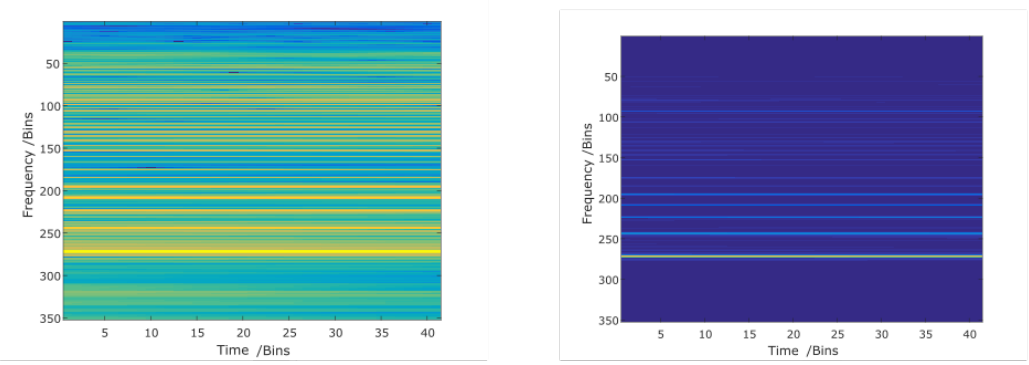


Figure 3.7: The above figure shows a single generated training example, before the logarithm is applied (right) and after (left). Higher harmonics of lower notes are more pronounced after the log has been applied.

where $n_l$ is the number of input connections to the layer. The biases are set to 0. The network is then trained with the ADAM optimizer described earlier in this chapter. The hyper-parameters are selected based on the parameter optimization experiments in the following section, with some reasoning why these parameters worked especially well. The network is trained until there is no or negligible improvement in the objective function.

In order to maximize the advantages of dynamic training whilst reducing the limitation of variation in the training data, after the parameters are optimized the network is trained further on real data from the MAPS database. This is similar to a method called pretext training in the field of image recognition [134, 134, 135]. Whilst the main aim of pretext training is usually to learn an unsupervised representation, pretext training is used here to reduce training time and reach a region of parameter space that will give a lower overall error than otherwise.

The MAPS data is split into train-validation-test splits as in the previous section and used to continue training the same network that was trained dynamically.

### 3.3.0 Parameter Optimization

Part of the difficulty in training deep networks lies in hyper-parameter optimization, with parameters often optimized via trial and improvement. In this section several hyper-parameter optimization experiments are performed, with the results used to determine the network parameters. The parameters optimized are:

**Filter size** The size of the filter used to perform frequency convolutions.

**Number of filters per layer** A low amount of filters will improve generalization but limit the model capacity.

| Parameter | Value |
|---|---|
| Time filter size | 5 |
| Frequency filter size | 13 |
| Number of stacked convolutional layers | 10 |
| Batch-size | 60 |
| Number of filters | 10 |
| Frame size | 55 |
| Number of epochs | 3000 |

Table 3.4: The network parameters used for training..

**Number of layers** Increasing the model depth generally always increases the performance of the network up until a point, so long as there is enough data to train it. In the case of skip connections, a model with 1000 layers was trained to demonstrate that it is possible to train much larger networks [123], however, the 1000 layer network decreased in performance when compared to their 50 layer network.

**Batch-size** The batch-size should not effect the final result of the training, however using smaller batches adds some noise that can help the network reach a minimum more quickly.

**Frame size** A smaller frame means less temporal information but potentially a more precise transcription.

For each of the following parameter experiments the parameter in question is varied and all other parameters kept as in Table 3.4. The learning rate for all experiments is kept fixed at 0.01, this value is chosen from experience and is also recommended to be kept constant by the creators of the ADAM optimizer, Kingma et al. [116]. The value for the L2 regularization term (weight decay) is also not varied and kept at 0.0005.

The size of the frequency filters used in the convolutional blocks is varied. It was expected to have a large effect on the networks performance as it affects the receptive field size. This is confirmed by the results of the experiment in Fig. 3.8. The performance of the network increases with an increased filter size until it plateaus at a filter size of 13. For improved training speed and reduced parameters filters with a size of 13 are selected as the optimum parameter to use.

There are two parts of the network which consist of stacked frequency convolution blocks as in Fig. 3.5. The number of these stacked blocks is determined by sweeping across a limited range and recording the network's performance. The results of this experiment are shown in Fig. 3.9. Ten layers are selected as the final parameter, as part of the reason that more layers are effective is due to an increased receptive

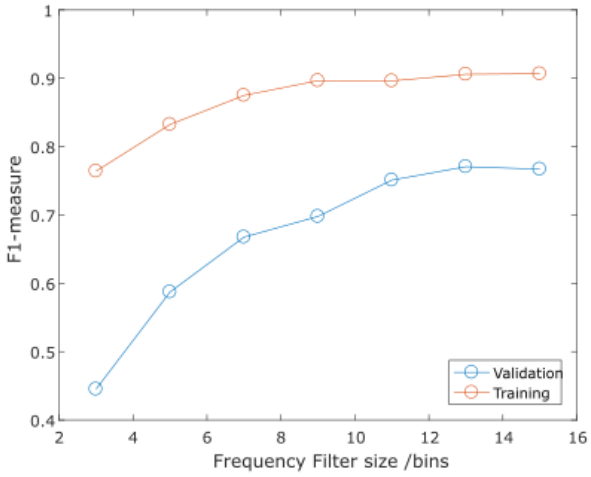


Figure 3.8: The filter-size of the frequency convolutions is determined by measuring the F1-measure on the validation set for a range of filter sizes. A filter size of 13 is used as the parameter for the network as it gives the best result on the validation set, and an additional increase would needlessly increase the number of parameters.

field size. As the receptive field will be increased by using larger frequency filters, a smaller amount of layers will be just as effective.

A parameter that was expected to have a clear effect on the network's performance is the frame size (the number of CQT time-steps taken as the network's input), as you would expect that with a smaller frame size the network would have less information to come to a conclusion about the note in its centre. As with the other parameters the frame size was varied with the size of the time convolutions adjusted accordingly, keeping the other parameters constant, see Fig. 3.10. What was observed is that the frame size has a little effect on the performance of the network, this was thought to be due to the small receptive field in the time domain and so the size of the filters in the time convolutional layers was increased drastically up to 20 for the first layer and 15 for the second. However this led to an increased training time and a higher training and validation error.

It was found that the optimum frame size was 55 frames, see Fig. 3.10. When varying the batch-size, see Fig. 3.11, the result that gave the lowest value on the training set did not perform as well as expected on a second run of the same experiment. Therefore the batch-size is selected to be 60 as this value has consistently given better results in other undocumented attempts. The batch-size changes the number of training examples used to update the weights in each case, a lower value will increase the amount of noise, but perhaps make it impossible to train. It also changes the size of the epoch, which affects how the Adam optimizer

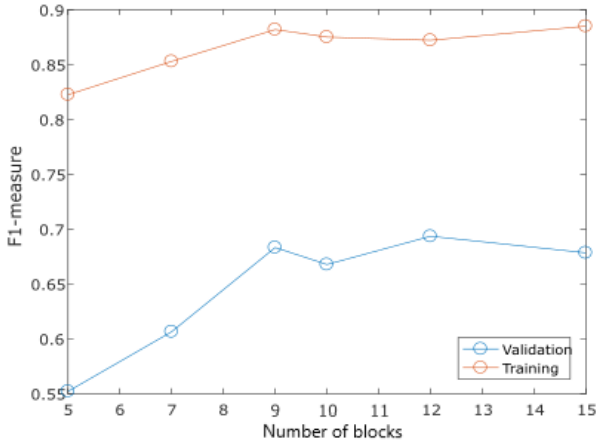


Figure 3.9: The total number of convolutional blocks (see Figure Fig. 3.5) used in the network is varied and the network retrained each time. The F1 score is quoted on the y axis. There is usually a trade off between training time and depth. In this case 10 blocks are selected for use in the final architecture.

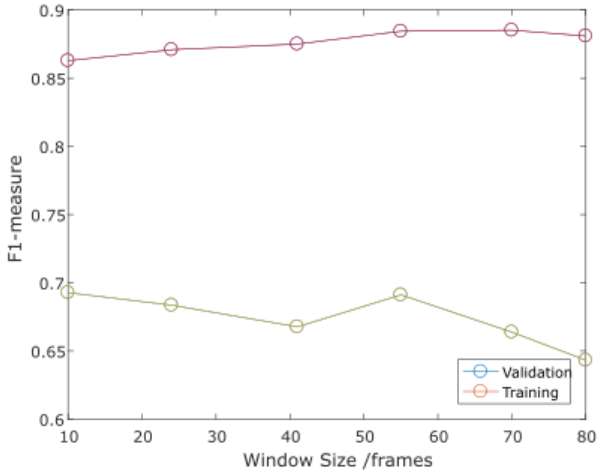


Figure 3.10: The frame size is varied and the network trained fully in each case. The best result with a size of 55 is selected.

calculates the consequent updates, this could have been controlled separately by adjusting the size of the epoch to make the total number of samples per epoch constant however performing these experiments is time consuming and this was deemed less important.

As a final experiment the number of filters was varied, see Fig. 3.12, the largest

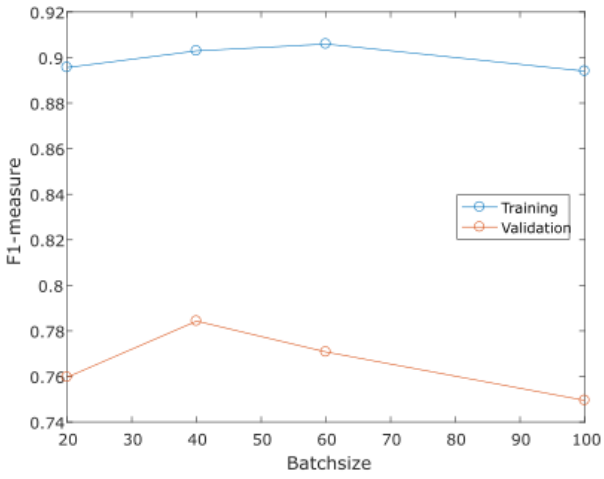


Figure 3.11: The batch-size is varied between 20 and 100.

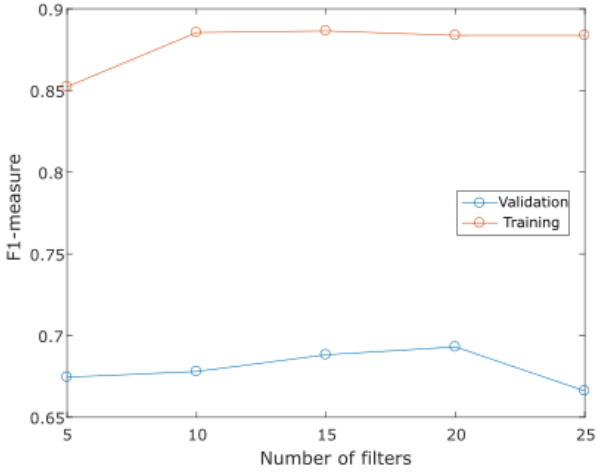


Figure 3.12: The number of filters used in each of the convolutional layers was varied between 5 and 25.

effect this had was on the training time, a large amount of filters drastically slowed down training, to the point where the experiment with 30 filters was cancelled as it took over 2 weeks. The number of filters seemed to have very little effect on the training and validation accuracy compared to the other parameters, for speed of training on MAPS data, 10 filters were used.

Due to the randomness of initialization, it is difficult to judge the parameters of the network in a single pass through the parameters, therefore some human

| Parameter | Value |
|---|---|
| Frequency filter size | 13 |
| Time filter size | 5 |
| Number of stacked convolutional layers | 10 |
| Batch-size | 60 |
| Number of filters | 10 |
| Frame size | 41 |
| Number of epochs | 3000 |
| Learning rate | 0.001 |
| $\beta_1$ | 0.9 |
| $\beta_2$ | 0.999 |

Table 3.5: The parameters kept constant in the parameter experiments.

judgment was used when selecting the final parameters for the network. Judgment was used to alter some parameters that were predicted to perform well by these experiments, but did not perform well in the final parameter selection. The parameters used for both the dynamic pre-training and then training on the train set of the MAPS database are show in Table 3.5. If training this network from scratch, then the parameters would be optimized by training the network with many different sets of randomized parameters between reasonable values determined as in the above experiments. The parameters with the best resulting score would then be selected. This is reported to be a more effective way to optimize the parameters for deep networks [136].

### 3.3.0 Output Quantisation

For thresholding, the goal is to set the network's outputs above a given threshold to 1 and all other outputs to 0. It might be the case that the overall transcription score can be improved by simply setting this level. Specifically, setting different thresholds for each note is investigated. One reason this could improve the transcription score is because of the frequency cutoff above 4927 Hz, the higher notes might have a smaller activation than the lower notes due to the higher harmonic partials not being present. This experiment is also useful to inspect the relationship between precision and recall, as in some cases having a higher precision or recall may be preferable. The threshold for each note is varied between 0.1 and 0.9, and the threshold with the highest accuracy is chosen. The values for accuracy against threshold can be seen in Fig. 3.13, examples are chosen to highlight the range in optimal thresholds.

The improvement seen using this method was up to 3.5% on the frame level for previous generations of the network, dependent on the parameters, the results from the final network can be seen in the next section. The network has a small

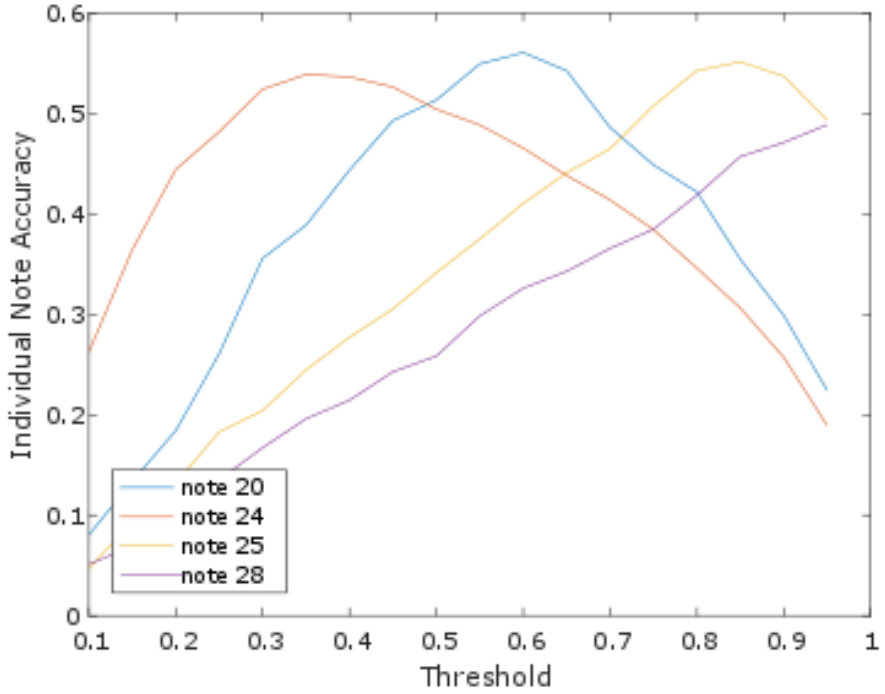


Figure 3.13: The threshold varied for each note and evaluated on the MAPS training data. The optimal threshold that is found is different for each note, this could be because differences in the timbre of the piano. The note numbers given in the legend are MIDI numbers.

number of filters compared to other networks of a similar size. A smaller number of parameters usually leads to better generalization at the cost of losing the ability to deal with variation in the input. This method for quantising the outputs of the network works well because each note has a slightly different timbre profile that the network cannot learn due to the low number of filters and no fully connected layers, so by thresholding the output, each note can be considered individually.

To inspect the relationship between precision and recall, they are plotted for middle C on the piano in Fig. 3.14. Whilst this carries no bearing on experiments in this thesis, it is useful for context, understanding the trade off between higher precision and recall, for instance if a musically meaningful pruning method or some physical constants could be applied at the frame level then a much higher recall would be preferred. Likewise if the outputs from this method were to be combined

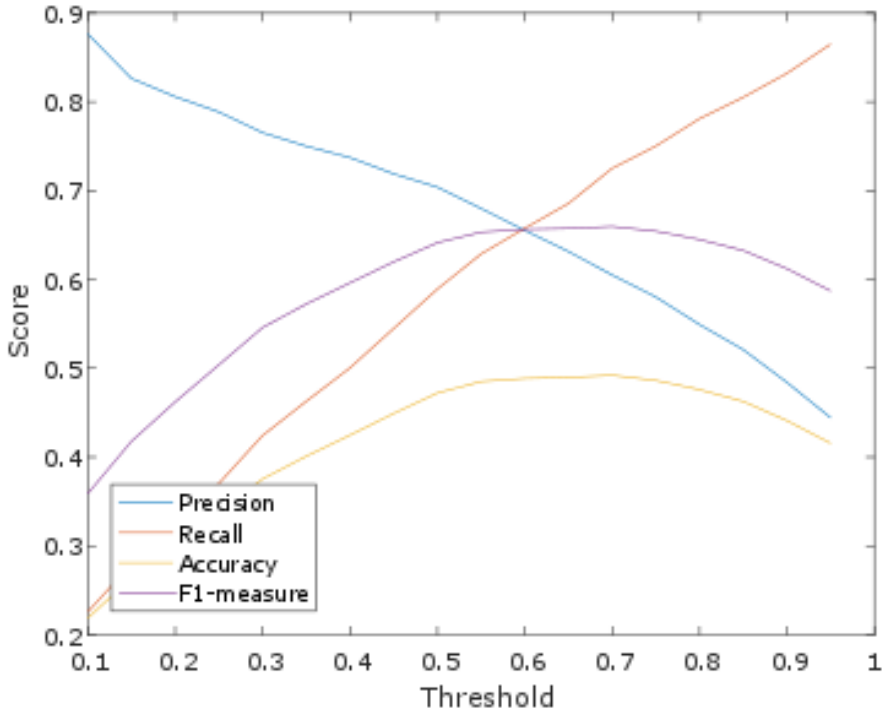


Figure 3.14: The precision, recall, F1-measure, and accuracy are plotted for middle C whilst the threshold is varied. Depending on the application, the precision and recall could be selected by altering the threshold.

with that of another frame-level method, then a higher precision might be preferred.

## 3.4 Critical analysis

A CNN is trained for the task of frame-level piano transcription by first training the network on synthetic chord examples, then training this network on the MAPS database. The results that follow are calculated using the evaluation set-up termed "configuration 2", described in [59]. The epoch with the best score on the validation portion of the MAPS database is selected for training on real data. The complete training graph and explanation can be seen in Fig. 3.15. During pre-training the validation data is taken from the MAPS database; synthetic data that contains onsets, offsets and is musically meaningful. The validation data used for training

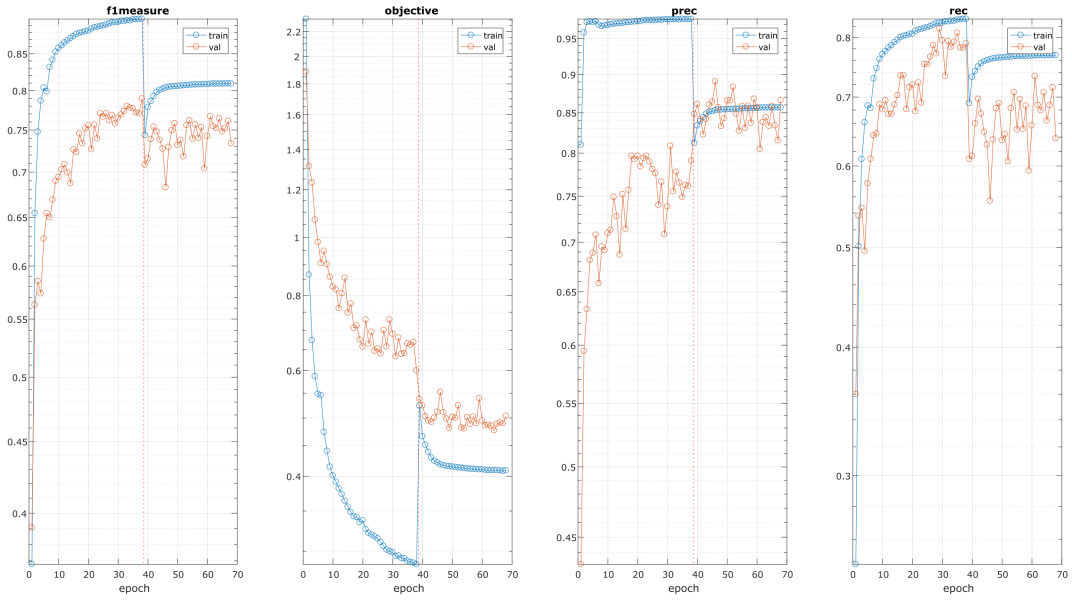


Figure 3.15: The complete training graph for both the pre-training stage and then the training on the MAPS database, these two stages are separated by a dotted line.

on the MAPS database is the portion of the MAPS database that is created from real pianos in the folders "ENSTDkCl" and "ENSTDkAm". The reason that the test data is shown here is because there was no selecting of parameters in the final run of the network and therefore the test data is shown as it better represents the network's performance whilst training. After training on real data the network achieves state of the art results on this database, outperforming the previous best F1-measure at 0.7060 [32] to 0.7667.

This method of training has worked particularly well, as it is thought to allow the network to reach a point in parameter space that is closer to a particularly good local minimum. Evidence of this is shown by experimenting with the entanglement problem in [33]. The authors find that instead of learning to detect individual notes, the network is instead learning to detect chords and note combinations. By first training the note on random notes initially, this forces the network to be unbiased towards chords and build a more general representation.

The network was trained using the CQT with the magnitude logarithimically scaled, this is in contrast to [32] where Kelz et al. use a logarithmically scaled FFT (both magnitude and scale) as the input to their network. He concludes this input representation is better than the CQT, however, does not experiment with scaling

| Model | Prec | Rec | $F_1$ |
|---|---|---|---|
| CNN - After Training | 0.8048 | 0.7335 | 0.7667 |
| CNN - After Thresholding | 0.8379 | 0.6729 | 0.7462 |
| CNN[32] | 0.7450 | 0.6710 | 0.7060 |
| Allconv[32] | 0.7653 | 0.6346 | 0.6938 |
| DNN[32] | 0.7551 | 0.5730 | 0.6515 |
| RNN[34] | - | - | 0.5767 |

Table 3.6: The results for the developed frame-level piano transcription system on the MAPS database. The results are compared to other networks trained on the same database. This generation of network performed worse when each note's threshold is optimized on a hold out set, a part of the MAPS data set kept separate from the training and validation sets.

of the CQT magnitude. Input representations have not been investigated here so no conclusions on which is better can be made. It would be surprising if the FFT outperformed the CQT in this setting as you would expect the CNN to perform particularly well with distances between adjacent harmonics that are translatable along the frequency axis (CQT).

Whilst training this network the number of parameters was kept small, this is partly due to the depth of the network, as a larger number of parameters would lead to a greatly increased training time, and also because the generalization is better with fewer parameters. This is also a limiting factor, as deeper and larger networks are always better, assuming that you have an infinite amount of data for training. If the amount of training data could be drastically increased then so could the size of the network.

As the network has no fully connected layers, the ability of the network to easily scale the outputs is partially reduced due to there being no processing past the last convolutional layer, this may be why thresholding helped in this instance.

It was observed that the performance of the network trained on generated data was identical on chords as it was on random notes. This is particularly interesting as you would expect the network to find it more difficult to separate out notes that share harmonics than otherwise. Another observation was that the final network performed worse on generated random notes than it did on generated chords, which is to be expected as it was only shown musically meaningful training data during the second stage of the training process.

The next step in building an effective system for piano transcription, would be to combine the output of this frame-level model with the output of a language model as in [82]. This note-level model could then be used to experiment with

note-level group-wise music transcription. However this is beyond the scope of this thesis.

## 3.5 Summary

This chapter has presented the approach taken in this thesis to the task of AMT. A CNN architecture was designed to work well with the properties of the CQT. The model accepts a short time frame from the CQT of the audio, and outputs the predicted notes. The architecture was trained dynamically initially and then further trained and evaluated on the MAPS piano database, which is the benchmark used for piano transcription. The results show that the new architecture developed here outperformed previous state of the art, in the realistic case where pianos contained in the training data were not contained within the test data. Currently the model is limited to the piano, as this is a good test casefor the transcription hypothesis. However, the model could be extended to work with multiple instruments by increasing the number of filters and the training data. Although this is not guaranteed to work as the strength on this model partially comes from its ability to generalize using a small number of filters. The dynamic training approach used here could be built upon, as currently it is only used as a pre-training stage before training on real data. If the variation in the samples generated during the dynamic training were improved, by modelling the acoustics of multiple pianos for instance, this should surely improve the model's ability to generalize to unseen data, and reduce the need to train on real data. Such an approach could provide a way to train a much larger model on multiple instruments, without the need for arduous data collection.

# Chapter 4

# Multiple Audio Alignment

In order to test the GWAMT hypothesis with frame-wise transcription transcription, in order to consider information across several pieces of music, the frames passed to the transcription algorithm must contain the same part of the piece in each rendition. That is, every frame in one piece of music must have a corresponding frame in every other piece. This is a problem, as when musicians play/interpret a piece of music they will likely play every section at differing speeds, loudness and articulations compared to other musicians, resulting in note events occurring at different locations in each piece. How to achieve a mapping between several musical audio samples is the problem of multiple-audio alignment. Throughout this chapter several methods in the literature for multiple audio alignment are compared, plus a new variant called iterative multiple audio alignment is proposed. This new method achieves an average alignment, as opposed to pairwise which achieves only an alignment to one recording, and is faster than progressive alignment for a large number of pieces. Whilst this has no direct relevance to the group-wise transcription hypothesis, it could prove useful for large scale alignment of musical samples for transcription and other analysis, such as from online resources like Youtube.

For group-wise transcription, there are two options for aligning features, either by using the locations of onsets in music to align the piece at the note level, which for piano at least are easily detectable [87], or by aligning each piece of music at the frame level. Since we have opted for frame-wise transcription over note-wise transcription it seems logical to go for the second option, and align features at the frame level. Another avenue of research could concern alignment and transcription at the note level. However, this would be less transferable to instruments other than the piano as their note onsets are harder to detect. Although theoretically, the onset detection could be improved if there are multiple renditions of the same piece available, this is beyond the scope of this thesis.

In the field of computer music, there are many applications for the alignment of music. The chief of these is automatic score following, which is the real-time alignment of a musical score to played audio. An example of such a musical tool was created by Artz et al. [137], a program was developed that will track your position in a piece for classical music, by constant re-evaluation. This solved an age old problem for musicians called page turning, which is the issue of having to stop playing your instrument to go from one page of sheet music to another.

In the age of big-data, interacting with large musical collections to find the piece, or group of pieces that you want can be difficult when you know little about the music you are looking for, or have only a fragment of audio. Recent work on audio alignment and audio matching has made navigating through these large collections more practical for the end-user [138, 139], with programs like "Shazam" [140] offering to search and find music based on a given musical fragment. The direct matching of exact fragments is one task, but searching for the original piece of music given a cover song is more difficult [141].

For score-following, several methods have given good results including hidden markov models (HMM) [142], conditional random fields [143] and dynamic time warping (DTW) [144]. HMM are generally preferred for aligning to a score, since the score gives the information about the order of the hidden states, assuming that they correspond to regions during a note event.

Early methods for music synchronization came about in the 80's [145], which were simplistic methods based on using DTW to match the played notes to the notes on the score (MIDI). Since then the advent of increased computing power means that other approaches have become feasible, mid-level audio features have been used in conjunction with peak-picking in [146], as well as the use of high-level chroma features in [144]. More recently chroma features have been combined with onset information to create more robust Decaying Locally-adaptive Normalized Chroma Onset (DLNCO) features [147], which greatly improve the quality of alignment. Chroma features have also demonstrated effectiveness in alignment with MIDI [148].

For comparative studies between performance styles [149], or expressiveness [150], achieving a precise and accurate alignment is necessary, this can occasionally not be achieved by simply aligning two recordings in a pair-wise fashion using traditional methods. A more robust alignment can be achieved using multiple audio alignment, if multiple renditions of the same piece of music are available, then each rendition provides another example of how the piece can be realized, therefore increasing the likelihood that a successful alignment path will be found. This helps to overcome some of the difficulties suffered by pair-wise alignment including the recording quality, location, and the playing style variation between musicians.

To achieve an alignment path between multiple pieces of music, it is possible to

extend the work on alignment of two pieces. For DTW this has been attempted in domains such as gesture recognition [151] first. However, in practice, this is unfeasible, as the computational requirements scale by a factor of $N^k$ where $N$ is the number of frames and $k$ the number of pieces to be aligned. It is possible to reduce the search space for application to multiple audio alignment as achieved by Wang et al. [35], but even then the number of pieces must be kept small. In order to use DTW and reduce the computational requirement, the alignment between all the recordings and one reference recording can be used to calculate a global alignment [36], this method is called pair-wise alignment. Although pair-wise alignment achieves a good alignment, it fails to compute the correct alignment in some cases and does not aim to use each alignment to improve future alignments, making it unpractical for a large $k$ as the chance of a piece not being aligned successfully increases linearly. Wang et al. [27] develop two methods of alignment based on the principle that a consensus must be built upon and improved with each piece. The first method for alignment is called progressive alignment, this methos uses DTW to align each piece to a template that grows as more pieces are added. Progressive alignment achieves state of the art alignment accuracy, and also improves upon cases that fail to align using traditional methods. Progressive alignment will be described more thoroughly later in this chapter. Later Wang et al. [27] applies the same logic to a HMM based alignment method and compares the two, although as of today progressive alignment is still the best available method for multiple musical audio alignment.

## 4.1 Dynamic Time Warping

Originally developed in the 70's for comparing speech patterns in speech recognition, dynamic time warping (DTW) is a method for finding the optimal alignment between two time series. Given series $U = u_1, u_2...u_m$ and $V = v_1, v_2...v_n$, DTW finds the minimum cost path $p = p_1, p_2...p_l$ where each point $p_k$ is an ordered pair $(i_k, j_k)$ indicating that element $u_{j_k}$ and $v_{j_k}$ occur at the same time. The path is drawn through a two dimensional array with each dimension representing the sequences $U$ and $V$, from the corner where $p_1 = (1, 1)$ to $p_l = (m, n)$. A distance matrix is calculated between the two sequences where element $D(i, j)$ in the matrix is the result of calculating a similarity measure between $u_i$ and $v_i$. The similarity measure or distance function $d(u_i, v_j)$ is small when elements are similar and large when elements are different, an example of this distance measure can be seen in Fig. 4.2. The choice of function for calculating the distance between elements is task dependent. Common functions include the Euclidean distance and the cosine distance. After a full distance matrix is available, a cost matrix is calculated, where each element $C(i, j)$ is the sum of the elements from $D$ up to $D(i, j)$ that give the

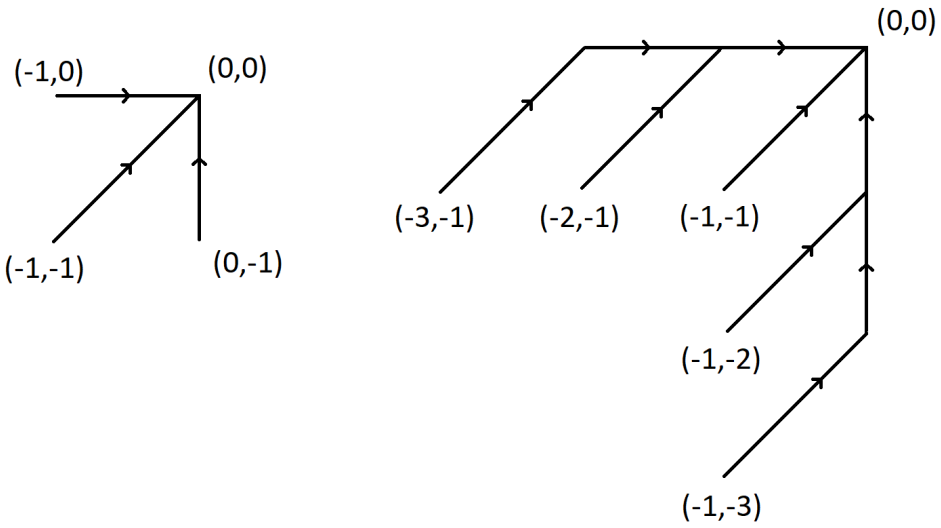


Figure 4.1: This shows steps that different DTW algorithms can take when they move from two aligned sequence elements to another two aligned elements. The dimensions represent the relative position of the potential matched sequence elements Left: An example of the path constraints used for traditional DTW, the possible steps are from (-1,0), (-1,-1) or (0,-1) to (0,0) Right: The path constraints for constrained DTW, all of the steps here must move in the direction of both sequences.

lowest cost path,

$$C(i,j) = \sum_{k=1}^{l} d(i_k, j_k), \tag{4.1}$$

where $l$ is the length of the minimum cost path up to that point. The path monotonically increases and is continuous, with subsequent path elements constrained by the step sizes. Typically the step size is limited to within one sequence element, however, by altering the step size [152] the path can be forced to have more desirable properties, such as being not allowing multiple elements in one sequence to align to one in the other. This comes at the cost of having some elements that do not have directly corresponding elements in the other sequence. The step size is the "jump" that the path can take, see Fig. 4.1, which shows the typical step-size and how it can be altered. The search space can also be reduced with classic methods such as the Sakoe and Chiba band [153], which constrains the path to lie within a distance from the diagonal. The cost matrix is determined from the distance matrix using dynamic programming, by recursively calculating

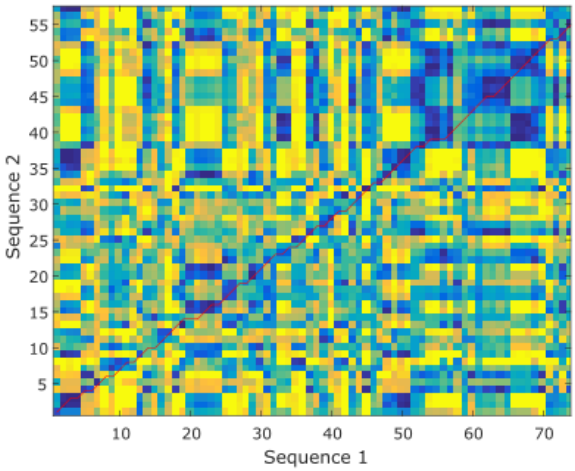


Figure 4.2: The distance matrix between two sequences, where blue indicates a small distance and yellow a large distance. The optimum path through the distance matrix is shown in red. The sequences used to create this figure are sampled from two renditions of the same piece of music.

$$C(i,j) = D(i,j) + \min \begin{cases} w_1 C(i, j-1) \\ w_2 C(i-1, j) \\ w_3 C(i-1, j-1) \end{cases} \tag{4.2}$$

for all elements $(i,j)\epsilon(m,n)$, where $w_1, w_2, w_3$ are the relative weights for each direction.

The cost matrix is initialized by setting $C(i,1) = \sum_{i=1}^{m} D(i,1)$ and $C(1,j) = \sum_{j=1}^{n} D(1,j)$. To obtain the lowest cost path it is then simply a matter of tracing the cost backwards from the last element $C(m,n)$, by using the recursion

$$p_{k-1} = \min \begin{cases} C(p_{k,1} - 1, p_{k,2}) & \text{set } p_{k-1} = (p_{k,1} - 1, p_{k,2}) \\ C(p_{k,1}, p_{k,2} - 1) & \text{set } p_{k-1} = (p_{k,1}, p_{k,2} - 1) \\ C(p_{k,1} - 1, p_{k,2} - 1) & \text{set } p_{k-1} = (p_{k,1}, p_{k,2} - 1) \end{cases} \tag{4.3}$$

An example of a calculated cost matrix can be seen in Fig. 4.3. Later in Section 4.2.3, the step size is altered for part of the iterative alignment method, the recursion used in this case for calculating the cost matrix and then the path is

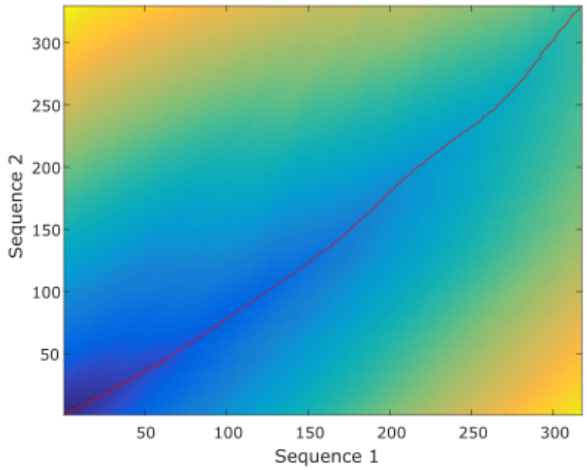


Figure 4.3: A cost matrix found using Eq. 4.4 after calculating the distance matrix, blue indicates a low cost path, and yellow a high cost path. The optimum path is shown in red, found using Eq. 4.5.

$$C(i,j) = min \begin{cases} C(i-1,j-1) + D(i,j) \\ C(i-2,j-1) + D(i-1,j) + D(i,j) \\ C(i-1,j-2) + D(i,j-1) + D(i,j) \\ C(i-3,j-1) + D(i-2,j) + D(i-1,j) + D(i,j) \\ C(i-1,j-3) + D(i,j-1) + D(i,j-2) + D(i,j) \end{cases} \tag{4.4}$$

$$p_{k-1} = min \begin{cases} C(i_k-1,j_k-1) & \text{set } p_{k-1} = (i_k-1, j_k-1) \\ C(i_k-2,j_k-1) & \text{set } p_{k-1} = [(i_k-1, j_k), (i_k-2, j_k-1)] \\ C(i_k-3,j_k-1) & \text{set } p_{k-1} = [(i_k-1, j_k), (i_k-2, j_k), (i_k-3, j_k-1)] \\ C(i_k-1,j_k-2) & \text{set } p_{k-1} = [(i_k, j_k-1), (i_k-1, j_k-2)] \\ C(i_k-1,j_k-3) & \text{set } p_{k-1} = [(i_k, j_k-1), (i_k, j_k-3), (i_k-1, j_k-3)] \end{cases} \tag{4.5}$$

Sakoe et al. [153] showed that weighting the diagonals paths with the same cost as the horizontal and vertical paths leads the final path to prefer the diagonals, as such they propose a weight for the diagonal direction of $w = 2$. However, each of the path directions can be weighted differently to achieve a path that prefers certain directions. The directional weights are set to $w_1 = 1.5, w_2 = 1.5, w_3 = 2$

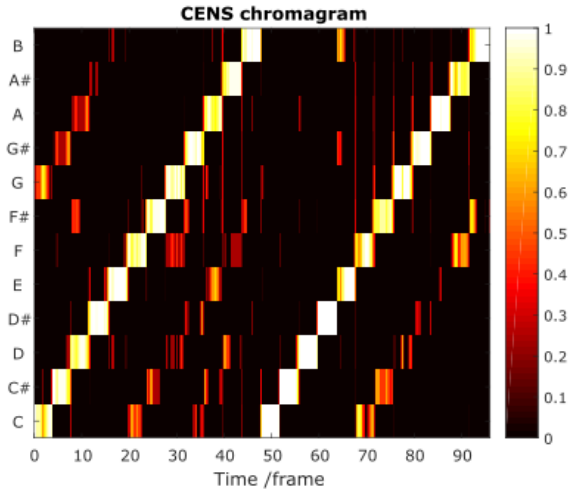


Figure 4.4: An example of the CENS features used for alignment. This is the CENS representation for a chromatic scale spanning two octaves. The vertical axis is circular, and if the captured fundamental frequency moves up from the final bin, it appears in the lowest bin.

as a weight of 2 in the diagonal is too strong, a diagonal path is preferable if it is available.

For aligning audio using DTW, CENS features are used following [26, 27, 147]. CENS features are a variant of chroma features which were mentioned in the introduction, they have 12 features per time step with each feature corresponding to the total harmonic content of each of the 12 notes on the western music scale. Short-time statistics are also used in producing the CENS features, in order to mitigate the effect of noise and instantaneous harmonic anomalies such as onsets. An example of this feature representation can be seen in Fig. 4.4. The two advantages of using this representation are that firstly it is small, which is necessary if DTW is to be computationally efficient, and also that they are robust to timbre changes and dynamic variation. The CENS features are calculated using the toolbox provided in [154]. For computing the distances between the CENS features the cosine distance is used as in [26, 27], which is given as

$$d(A,B) = 1 - \frac{\sum_{i=1}^{12} A_i B_i}{\sqrt{\sum_{i=1}^{12} A_i^2}\sqrt{\sum_{i=1}^{12} B_i^2}} \tag{4.6}$$

where $A$ and $B$ are the 12 dimensional feature vectors being compared. An example of what the distance matrix might look like between two pieces can be seen in Fig. 4.4. It was found that this distance metric is more effective when taken over a

short time window, perhaps as it improves the continuity of the final path, as it is less likely to deviate down shortcuts that provide a slight cost reduction due to random chance. This adapted distance is given as

$$d(A,B) = 1 - \frac{\sum_{j=-w}^{w}\sum_{i=1}^{12} A_{i,j}B_{i,j}}{\sqrt{\sum_{j=-w}^{w}\sum_{i=1}^{12} A_{i,j}^2}\sqrt{\sum_{j=-w}^{w}\sum_{i=1}^{12} B_{i,j}^2}} \tag{4.7}$$

where $w$ is the width of the window and $j$ is the time index.

The final traced out path provides a mapping from each element of $U$ to each element of $V$, and can be used as a lookup table to see where a location in one audio file corresponds to the other. This bi-directional mapping can be used to warp either sequence to the same length as the other, where each frame in both pieces are the corresponding aligned frames. The process of stretching the original sequences will be explored more later in this chapter.

### 4.1.0 Multi-scale Dynamic Time Warping

Dynamic time warping will always find the optimum alignment between two sequences for a given distance function. However, it is computationally expensive for long sequences, for two sequences of length $N$, the total number of calculations required is proportional to $N \times N$. As mentioned previously, this can be reduced by using path constraints such as the Sakoe Chiba band [153]. However, applying a hard constraint comes with the risk of the path not being accurate if the ground truth path falls outside of the confined region. For example, in the case of aligning musical sequences some pieces are highly open to interpretation, and as such would not necessarily fall within the Sakoe Chiba band. More efficient adaptations of DTW exist that keep the computational cost small, whilst still achieving an excellent alignment, the best example of this is multi-scale DTW (MSDTW), developed originally in [155]. The idea behind this is to downsample the original sequences to a much coarser resolution, where calculating the entire cost matrix is feasible, then once a path has been found it can be iteratively projected onto a finer resolution of the sequences, see Fig. 4.5. By using this method, only distance cells within a certain distance of the projected path need calculating. For aligning musical sequences this works well due to their temporal continuity [156], when a piece is downsampled, the remaining features are still representative of the original piece.

For any piece to be aligned, the CENS features are calculated with a time resolution of 20 ms. The features are down-sampled using decimation rather than average as averaging may cause temporal ambiguity. The down-sampling factors are 50,25 and 5, which results in resolutions of 1s, 500 ms and 100 ms respectively. These down-sampling steps are called stage 1, 2 and 3 respectively. The final stage, Stage 4, uses the maximum resolution CENS features at 20 ms. In order to ensure

| Stage | Number of Elementary Operations for MSDTW | Number of Elementary Operations for DTW |
|---|---|---|
| 1 | 67334 | - |
| 2 | 32226 | - |
| 3 | 209421 | - |
| 4 | 1184324 | 168335000 |
| Total | 1493305 | 168335000 |

Table 4.1: An example of the number of elementary operations required for finding the optimal alignment between two musical performances. MSDTW gives a speed up of $\approx 100\times$.

that there is not a dimension mismatch when projecting the path to a higher resolution, the end of the features are padded with zeros before down-sampling, until the length is divisible by 50. To demonstrate the increase in speed from traditional DTW the number of calculations at each stage in the MSDTW process are tabulated in Table 4.1, comparing the total number of calculations needed to achieve an alignment in each case. When projecting the path onto higher resolutions, the number of distance elements calculated are determined by the width of the path for each projection. Widths of 15, 30 and 30 for stages 2, 3 and 4 are used, following the recommendation by Salvador et al.[155], see Fig. 4.5.

## 4.2 Multiple Audio Alignment

This section describes the alignment method used to test the group-wise transcription hypothesis, pairwise alignment. Another method for multiple audio alignment, progressive alignment, will be detailed and one final method of alignment, iterative alignment proposed. Throughout all of the methods, CENS features are used as the feature representation for alignment. The path weights are set to $w_1 = 1.5, w_2 = 1.5, w_3 = 2$.

### 4.2.0 Pairwise Alignment

The principle of pairwise alignment is that if two pieces can be aligned, then all the pieces can be aligned to one piece. This method was first implemented by Dixon et al. [36], where a downloadable toolbox was created that will align several pieces of music to one track. A slightly different variant of MSDTW which is based on online-DTW was used. In pair-wise alignment, one track is chosen randomly for the reference track, then all pieces are aligned to that track using MSDTW. The

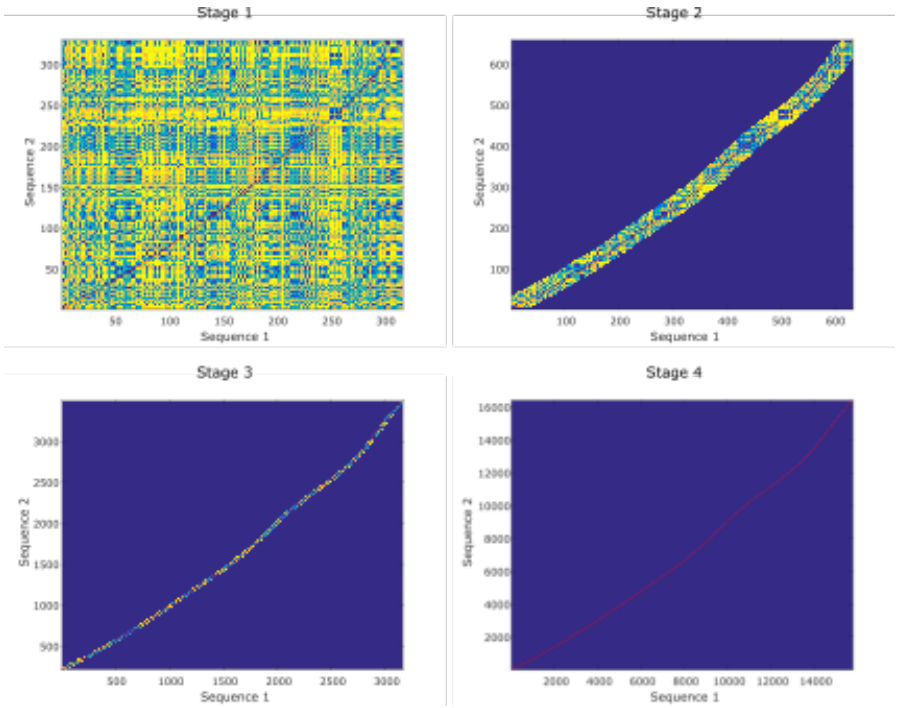


Figure 4.5: The 4 stages in MSDTW for CENS features, clockwise from the top left the resolutions are: 1 s, 50 ms, 100 ms, 20 ms. The path that is projected onto the next stage is shown in red. The width of the band is so small in the final stage that the cells where the distance has been calculated are not visble. CENS features from two renditions of the same piece of music were used to generate these plots. The x-axis represents sequence elements.

features used can be either CENS features or DLNCO features which are similar to CENS but use onset information. DeBruyn et al. [157] found that a combination of the two features was found to work best. However, for simplicity only CENS are used here. For an outline of this method, see the pseudo-code provided in Algorithm 2.

This is a good method and is surprisingly effective. However, it suffers from several systematic errors. If the piece chosen to be the reference track is drastically different to the remaining recordings then the alignment might suffer, it also cannot use information contained in other recordings that might aid the group alignment. One variant of this method might include calculating the pairwise alignments between all of the recordings, then by using a tailored cost function the piece best suited to being the reference piece can be chosen.

When using this method, the cosine distance with a window size of 5 frames is used as the distance function. Instead of comparing individual sequence elements

with elements in the other sequence, several sequence elements are compared at each DTW step. The cosine distance function was used to compare each window of elements. Using a time window to improve the alignment quality is novel for pairwise alignment, and although not experimented with in this thesis in great depth, it was found practically to improve the alignment quality and robustness.

**Algorithm 2** The pairwise alignment method is described. The **Rand** function returns a random track from the input list of tracks. The **CENS** function computes and returns the CENS features for the input track. The **MSDTW** function computes an alignment path between the two input CENS features using the MSDTW method. The **append** function appends the second argument to the list provided in the first argument. The whole algorithm returns a list of alignment paths between one track and a randomly selected reference track.

**procedure** PAIRWISE ALIGNMENT
    $n \leftarrow$ *Number of tracks*
    $tracks \leftarrow$ *List of the tracks to be aligned*
    $ref \leftarrow \textbf{Rand}(tracks)$
    $to\_align \leftarrow tracks - ref$
    $ref\_cens \leftarrow \textbf{CENS}(ref)$
    $paths \leftarrow$ *An empty list*
    $i \leftarrow 1$
    **while** $i < n$ **do**
        $track \leftarrow to\_align[i]$
        $track\_cens \leftarrow \textbf{CENS}(track)$
        $path \leftarrow \textbf{MSDTW}(ref\_cens, track\_cens)$
        $paths \leftarrow \textbf{append}(paths, path)$
        $i \leftarrow i + 1$
    **return** *paths*

### 4.2.0 Progressive Alignment

At the fundamental level, multiple audio alignment can be considered analogous to multiple sequence alignment, which is a well-studied field in bio-informatics. Wang et al. [27] make this comparison, and apply some of the methods for alignment to musical performances. Specifically they compare two classes of method, progressive alignment and probabilistic profile methods. They conclude that progressive alignment is both faster and more effective at alignment that any other methods. As its name suggests, instead of aligning all of the pieces simultaneously, each piece is aligned individually. A template is built up, which contains all of the sequences that have been aligned up to that point. However the sequences are stretched, so

that they each contain the same number of elements, with each element having a corresponding element in the other sequence.

The method for progressive alignment is now explained, for more detail consult its origin [27]. To aid the explanation, see the pseudo-code of the method given in Algorithm 3. To start with, assume that there are $K$ renditions to be aligned, with $k$ indicating any piece up to $K$. Each of the renditions are feature sequences denoted as $X^k = (x_1^k, ..., x_N^k)$ where each sequence element exists in a suitable feature space $x_n^k \in F$. The template structure that grows as pieces are added is denoted as $Z$ and initially is set as $X_1$. Afterwards the other sequences are aligned to $Z$, updating $Z$ in each case. At any point in this iteration the sequence being aligned to $Z$ is $X^k$, and the sequence $Z = (z_1, ..., z_M)$ where $M$ is the length of the template. Each element of $Z$ contains the aligned feature vectors for one timestep from pieces already added to the template. As in the section on DTW, when warping one sequence to map linearly onto another, elements that are mapped to multiple positions in the other piece are usually repeated. However, when pieces are stretched to be added to the template this would insert temporal ambiguity and as such repeated elements are skipped and replaced with a gap symbol $G$.

When calculating an alignment path $p$ between the next piece to be added $X^k$ and $Z$, a modified version of MSDTW is used to calculate a distance matrix between $X^k$ and every piece in $Z$. When a gap symbol is encountered in calculating the distance matrices, that distance element is replaced with a constant termed the gap penalty $\mathcal{C}$. The result of calculating these distances is $k-1$ matrices $D^r$, one for each piece in $Z$. To combine the distance elements between $X^k$ and each piece in $Z$, Wang et al. [27] attempted several strategies but found the most effective to be a simple average:

$$D(m,n) = \frac{1}{k-1} \sum_{r=1}^{k-1} D^r(m,n). \tag{4.8}$$

Once a distance matrix has been determined, the cost matrix can be calculated by using the recursion in Eq. 4.4, and then the alignment path $p$ using the second recursion in Eq. 4.5. It is the alignment path that is used to stretch both $X^k$ and $Z$, so that each sequence element become aligned. The corresponding sequence elements are then combined and becomes a part of a single element $z^l$ in $Z$. Both sequences are stretched so that they are the same length, gap symbols are inserted where any element is aligned to multiple locations in the other sequence. Specifically the gap symbols are inserted at all the locations, except where the cost between the repeated element and the corresponding element in the other sequence is lowest. Now this will be defined more formally. Firstly, the following functions $E_1$ and $E_2$

can be used to calculate the positions of the lowest cost elements:

$$E_1(m) := \underset{\{\tilde{n}|(m,\tilde{n})\epsilon p\}}{\operatorname{argmin}} C(m,\tilde{n}), \tag{4.9}$$

$$E_2(n) := \underset{\{\tilde{m}|(\tilde{m},n)\epsilon p\}}{\operatorname{argmin}} C(\tilde{m},n), \tag{4.10}$$

Then the stretching of each new element $\tilde{z}_l$ is given as

$$\tilde{z}_l = \begin{cases} (z^1_{m_l}, ..., z^{k-1}_{m_l}, x^k_{n_l}), & \text{if } (m_l, n_l) = E_1(m_l) = E_2(n_l) \\ (z^1_{m_l}, ..., z^{k-1}_{m_l}, G), & \text{if } (m_l, n_l) \neq E_2(n_l) \\ (G, ..., G, x^k_{n_l}), & \text{if } (m_l, n_l) \neq E_1(m_l) \end{cases} \tag{4.11}$$

This method of inserting gap symbols where repeated elements occur was found to improve the alignment quality, as it removes any ambiguity about the location of a particular element. This means there will always be a one to one mapping from the original features to the stretched features. The best value for the gap penalty was determined experimentally in [27] to be 3.6.

### 4.2.0 Iterative Alignment

There were difficulties in implementing the method for progressive alignment described above, mainly, the features used were a combination of DLNCO features and CENS features. The CENS features were calculated using a freely available toolbox, however, no such resource is available for DLNCO features, and so due to time limitations these were not implemented. At this time another method for the group alignment of audio was developed, called iterative alignment. This method is similar in principle to the method used by Sidorov et al. [5] for aligning three dimensional textured meshes of a set of faces. The meshes are first averaged which gives a blurred combination of all the meshes. Then the mapping to the mean are warped to reduce a defined cost function. This method treats the problem as an optimization problem. This is different to the approach here. When applying this to audio, 0.2 s CENS features for each piece are initially averaged in an attempt to give a combined feature representation, then the pieces are iteratively aligned to the combined average, stretched according to the alignment path, and averaged to create the combined features for use in the next iteration. However, what was observed is that after stretching each piece and averaging, the features from each individual piece were still distinct. This would allow each piece to align to a single prominent piece in the combined CENS features. This is a problem as the aim of group-wise alignment is to align feature sequences from all of the pieces collectively, using all the information available. To remedy this the CENS features are blurred in the time domain before alignment in each iteration. This blurring helps to

**Algorithm 3** Below is an outline of how the progressive alignment method aligns multiple pieces. The function **CENS** computes the CENS features for the input sequences and returns them in an indexed list. The **MSDTW** function uses a modified version of MSDTW to calculate the distance matrix between two sequences. The function **append** returns a list containing the second argument appended to the list in the first argument. The **average** function computes an average over the input list of matrices, as in Eq. 4.8. The **cost** function computes the cost matrix for the input distance matrix, see Eq. 4.4. The **traceback** function computes the lowest cost path through the input cost matrix, and hence the alignment path between the respective sequences, see Eq. 4.5. The **stretch** function stretches the input sequence based on the alignment path provided, inserting gap symbols when required, see Eq. 4.12. The output is a list of all of the aligned sequences.

**procedure** PROGRESSIVE ALIGNMENT
    $tracks \leftarrow$ *Pieces to be aligned*
    $n \leftarrow$ *Number of pieces*
    $X \leftarrow \textbf{CENS}(tracks)$
    $Z \leftarrow X^1$
    $i \leftarrow 2$
    **while** $i \leq n$ **do**
        $j \leftarrow 1$
        $l \leftarrow$ *Current length of* $Z$
        $distances \leftarrow$ *Empty List*
        **while** $j \leq l$ **do**
            $distanceMatix \leftarrow \textbf{MSDTW}(Z^j, X^i)$
            $distances \leftarrow \textbf{append}(distances, distanceMatrix)$
            $j \leftarrow j + 1$
        $averageDist \leftarrow \textbf{average}(distances)$
        $costMatrix \leftarrow \textbf{cost}(averageDist)$
        $path \leftarrow \textbf{traceback}(costMatrix)$
        $Z_{stretch} \leftarrow \textbf{stretch}(Z, path)$
        $X_{stretch} \leftarrow \textbf{stretch}(X^i, path$
        $Z_{stretch} \leftarrow \textbf{append}(Z_{stretch}, X_{stretch})$
        $Z \leftarrow Z_{stretch}$
        $i \leftarrow i + 1$
    **return** $Z$

remove the individuality of each piece in the combined feature representation. Even then, in some cases, this caused the alignment to prefer to align to an individual recording and not find a combined average. So as a final step to aid this, a coarse alignment is computed by constraining the DTW path as in Fig. 4.1. This stops the DTW path from taking horizontal and vertical paths through the cost matrix, and ensures continuity. This was successful in producing a group alignment as shown in Fig. 4.10.

An outline of the iterative alignment method is now given in more detail, for a higher level approach see the pseudo-code provided in Algorithm 4. Assume that there are $N$ pieces requiring alignment $X^k$, each of length $S^k$. Initially the pieces are mapped linearly onto the longest piece, this mapping is defined as $y = \frac{S^k}{S_max}x$, by sampling from this mapping at integer position and rounding, a discrete frame by frame correspondence is achieved, called an alignment path $p = ((m_1, n_1), ..., (m_L, n_L)$. Each piece is now stretched to the length of the longest piece, using the alignment paths

$$X^k = (x_{n1}, x_{n2}, ...x_{nl}), \tag{4.12}$$

and then combined by averaging to give $Z^1$ the result after the first iteration. For the first three iterations, constrained DTW is used.

$$Z^1 = \frac{1}{N}\sum_{k=1}^{N} X^k. \tag{4.13}$$

The combined feature representation $Z^l$ is then padded with zeros prior to convolution to maintain its dimensionality. The convolution is with a Hann window which blurs the features in the time direction

$$Z^I = Z^I * \text{hann}(n^I). \tag{4.14}$$

The size of the Hann window $n^I$ is determined by the current iteration $I$ as in Table 4.2. A blurry average of all the pieces to be aligned is now obtained, and the iterative procedure can begin. Each piece is aligned to $Z^I$ using MSDTW with a constrained path for the first 3 iterations, and without a constrained path afterwards. This gives an alignment path for each piece $p^k$, which is used to stretch the original CENS features for each piece as in Eq. 4.12, which are averaged as in Eq. 4.13 to give $Z^I$ for the next iteration. This process is repeated 25 times with the blur reduced on each iteration. The blur sizes listed in Table 4.2 were determined based on what worked, and could be further optimised by being determined experimentally. The size of the blur could even be determined dynamically depending on the CENS variation of the pieces being aligned. In any case there is much room for further optimization of this method, including in the final stretching of pieces, if the

| Iteration, $I$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Width | 4000 | 3000 | 2300 | 1900 | 1700 | 1500 | 1300 | 1100 | 1000 | 900 |

| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 800 | 700 | 600 | 500 | 400 | 350 | 300 | 250 | 200 | 100 | 50 | 30 | 10 | 2 | 0 |

Table 4.2: The size of the Hann window used for blurring on each iteration of the iterative alignment method.

position of the optimum match between elements was factored into the process in a similar way to the use of gap penalties in progressive alignment, then this could improve the accuracy.

An issue with this method in its current state is that the paths of all the pieces to the mean will sometimes all travel horizontally, giving no information about the group-alignment, and making the alignment more difficult on future iterations. The problem is caused by the high number of iterations allowing minor errors, where frames in one piece are matched to multiple frames in the other piece, to stretch. Without tackling the root of the problem in the methodology, the paths are pruned after the method is complete by removing repeated sections in the path. Other attempts to solve this problem included varying parameters such as the step-direction weights, and the number of iterations that constrained DTW was used for, however, these had little effect. Should this problem be solved then it should improve the accuracy of this method.

A large number of iterations are used to ensure accuracy, however, for a much larger N far few iterations would be needed. An advantage of this method is that the accuracy can be increased by increasing the size of the template that all the pieces are aligned to initially, the effect of this would be an increased path resolution. The gap penalty used for the progressive method could also be incorporated into this method.

Initially the features were averaged as chroma features, prior to the short time statistics being calculated for the CENS features. However, this worsened the alignment, but perhaps made it more robust. This could potentially be used during the coarse alignment stage, but the sharpness of each of the piece's features is needed in the later stages to give a high accuracy.

### 4.2.0 Evaluation

When deciding on how to evaluate and compare the three alignment methods, methods for both qualitatively and quantitatively comparing and checking the alignments are useful. The reason why qualitative evaluation is useful is twofold,

**Algorithm 4** This outlines the procedure for performing an iterative alignment of multiple pieces. The **CENS** function computes the CENS features for each piece. The **longestStretch** function stretches each of the input CENS features to be the length of the longest one, see Eq. 4.12. The function **HannBlur** applies the Hann windowing function to the input CENS features, see Eq. 4.14. The first argument is the CENS feature to apply the blurring to and the second argument is the iteration number which determines the window size, see Table 4.2. The **MSDTW** function aligns the two input sequences and returns the alignment path. The **stretch** function stretches the sequence provided in the first argument according to the alignment path in the second argument. The output of the **stretch** function is always the same length as the longest sequence that was used when calculating the alignment path.

**procedure** ITERATIVE ALIGNMENT
    $n \leftarrow$ *Number of pieces*
    $tracks \leftarrow$ *Pieces of be aligned*
    $X \leftarrow \textbf{CENS}(tracks)$
    $X_{stretch} \leftarrow \textbf{longestStretch}(X)$
    $i \leftarrow 1$
    **while** $i \leq 25$ **do**
        $Z \leftarrow \textbf{average}(X_{stretch})$
        $Z_{blurred} \leftarrow \textbf{HannBlur}(Z, i)$
        $k \leftarrow 1$
        **while** $k \leq n$ **do**
            $path \leftarrow \textbf{MSDTW}(Z_{blurred}, X^k)$
            $X^k_{stretch} \leftarrow \textbf{stretch}(X^k, path)$
            $k \leftarrow k + 1$
        $i \leftarrow i + 1$
    **return** $X_{stretch}$

in the case of when accurate onset information is not available for quantitative analysis, qualitative analysis can confirm that an alignment has been found, it is also useful for identifying features of the alignment method that may be undesirable, for example the bunching of note events in the finalized alignment. A phase vocoder was used for qualitative analysis, where alignment paths between the recordings are used to stretch the raw audio. The stretched raw audio from each of the pieces is summed to produce a chorus effect. The alignment can then be manually verified by listening to the chorus.

As highlighted previously, accurate onset information is difficult to obtain in many cases. There are projects such as the mazurka project [158] where onsets have been manually labelled in renditions of Chopin mazurkas played by famous pianists.

They give explicit information about where to obtain the correct recordings. Ideally these pieces and onset information would be used as this would allow a direct comparison to result in the literature [27], however, obtaining the recordings is time consuming and expensive, and so a different approach for quantitative evaluation was used. Working the problem backwards, if onset information for a specific piece of music is known, renditions could be generated that try to mimic the playing style variations of professional pianists. This is what was done, from a MIDI file of the piece to be used for alignment a synthesizer in MATLAB is used to create "renditions" of that piece. The global and local tempo and dynamic range are varied and the dynamics, relative onset positions and durations, reverb, noise are slightly randomized, the minute chance to play a completely incorrect note is also added.

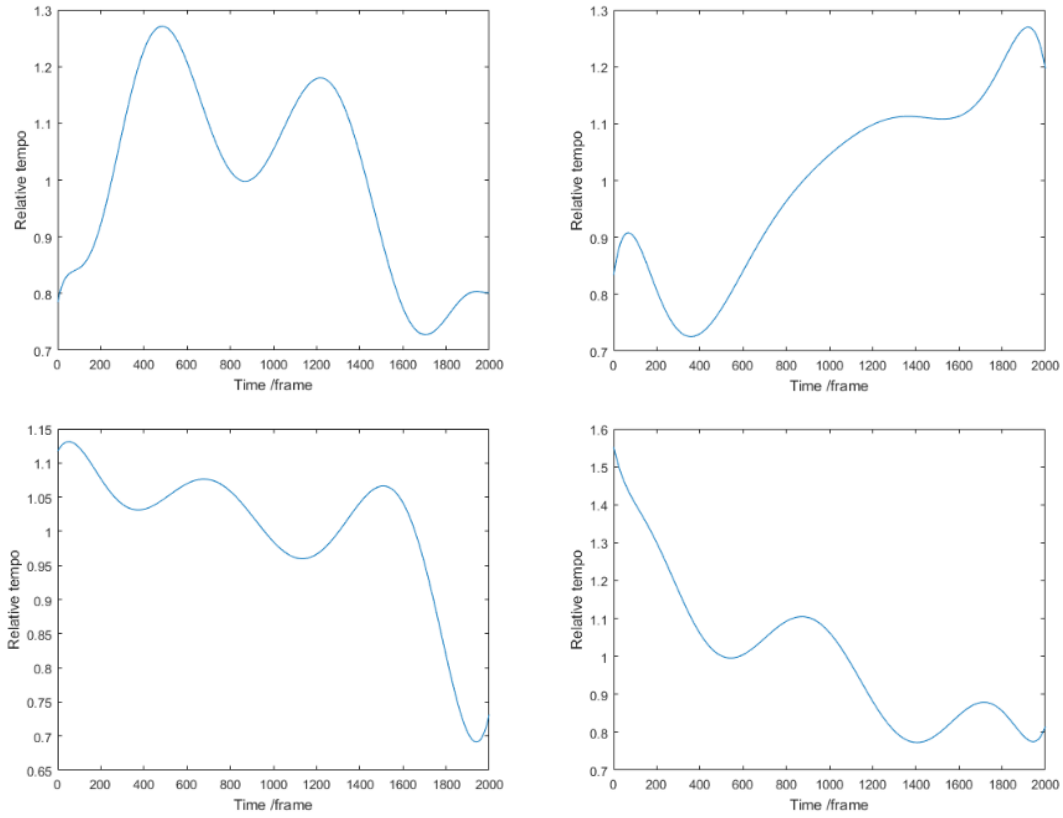


Figure 4.6: Example tempo graphs that show how the tempo varies over the generated renditions.

The renditions are generated from an initial MIDI file containing the onset time, duration, velocity and pitch. The following features of the MIDI track are adjusted for each rendition to be generated:

**Piece Length** The relative piece length is randomized by sampling from a normal deviation with a mean of 1 and a standard deviation of 0.15, leading to a range of between 0.775 — 1.225.

**Local Tempo** The change to the local tempo of the piece is modelled using a random walk that is updated at each time step. The random walk for the entire piece is fitted by a polynomial function, and centered around 1 so the average tempo is still the same. Some examples of these random tempos can be seen in Fig. 4.6. This method of adjusting the local tempo has not attempted to take into account how the tempo might vary between performers, only that any relative tempo change from the MIDI file is likely to be continuous, and not impulsive. This might not necessarily be true, but provides enough variation to test the alignment methods in this chapter.

**Onset Deviation** Differences in local tempo changes the time between the onsets, however, does not add more random effects that are not tempo dependent. Two examples of this are notes in a chord not occurring exactly simultaneously or rests that are extended at a cost to the length of the proceeding note. These are simulated by adding an additional value to the note onset sampled from a normal distribution centered around zero, with the standard deviation determined from a uniform distribution between 0 and 0.007. The reason the standard deviation is varied is to try and account for different styles, as some pianists might add more variation than others and/or be at a different skill level.

**Dynamic range** The dynamic range is related to the expressiveness of the piece, and as such is randomly varied for each rendition. The relative dynamic range is varied uniformly between 0.4 and 1.4.

**Impulse volume changes** The volume of each note has a 5% chance to be shifted up or down by one volume level (the volume level corresponds to the number of piano samples for that particular note). This additional randomness is applied to correspond with the pianist occasionally deviating from the dynamic contours of the piece. It also makes it more difficult for the DTW based methods to find an alignment as the value of the distance metric between the features of the same note played at a different volume will be larger. Therefore an instantaneous change in the volume might make the DTW determined path deviate around the impulses.

**Incorrect notes** It is unlikely that a professional pianist will make an error, however by adding randomly incorrect notes, it will make the testing procedure more robust. Randomly incorrect notes are played 0.1% of the time (Roughly one or two notes per piece).

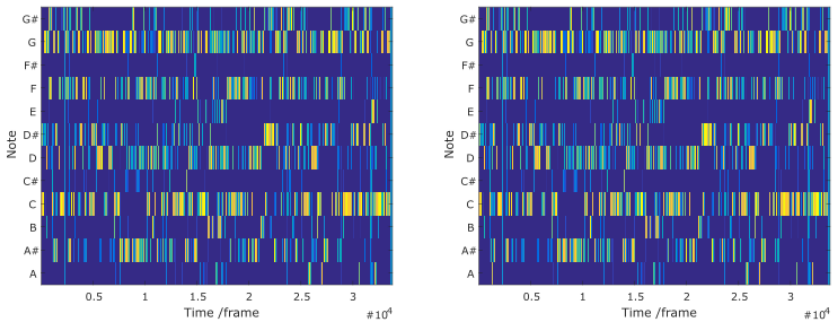


Figure 4.7: Two examples of the generated renditions, they have similar topographies, however, are different enough to test the alignment methods.

The above procedure used to generate renditions from a MIDI track has been developed based on what seems reasonable. Ideally the procedure could be more methodically determined by analyzing the statistics of MIDI files from real recordings and pianists. However, the primary goal of producing these renditions was to add enough variation to ensure the testing procedure was robust, for which the above method sufficed.

After the renditions have been aligned by the method in question, the onsets are warped according to the alignment paths. For each onset, the pairwise differences in onset time are calculated for all of the renditions. Averaging across all of the pieces and onsets gives an overall score called the average note onset deviation. This method of evaluation provides a good way to explicitly compare alignment methods, however, it means that no comparison to a baseline in the literature can be calculated. For a comparison of the CENS features from two of these generated renditions, see Fig. 4.7.

To compare the effectiveness of each alignment method 10 "renditions" are generated using the method described above, and then their average note onset deviation for each alignment method are compared for an increasing number of pieces. A graph showing the change in alignment accuracy with an increasing number of pieces for each of the methods can be seen in Fig. 4.8. Generally, iterative alignment is slightly worse than pairwise and progressive alignment gets progressively worse as the number of pieces being aligned increases, this is likely due to the implementation and not the method itself. Histogram plots to show the frequency of errors can be seen in Fig. 4.9. The method for pair-wise alignment was better in all but one case where iterative alignment worked better. Why progressive alignment did not work as well as expected is unclear, there are perhaps some subtleties in its implementation that were missed, or perhaps by using randomly generated renditions the problem has been made harder than by using

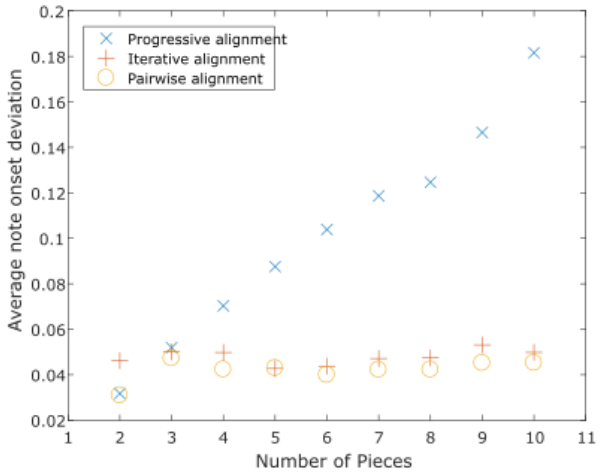


Figure 4.8: Change in alignment accuracy for the different methods with an increasing number of pieces.

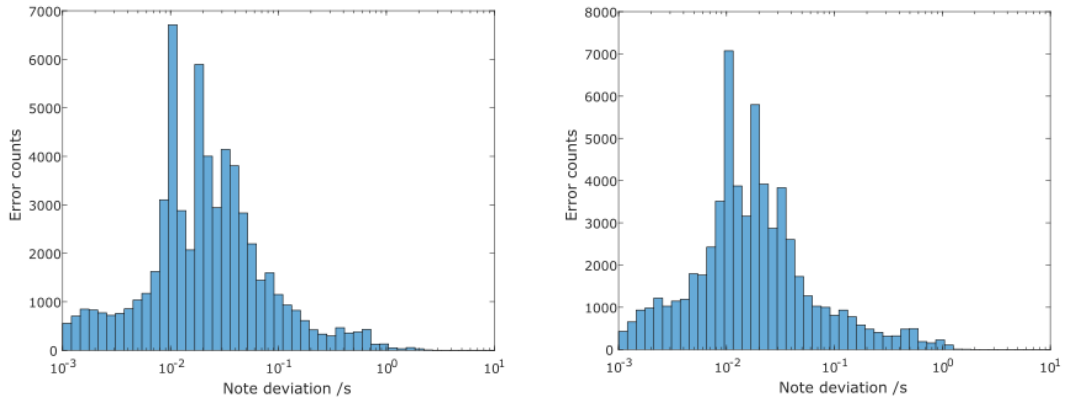


Figure 4.9: Histogram plots to show the onset deviation error frequency between each of the recordings for the pairwise (right) and iterative (left) alignment methods.

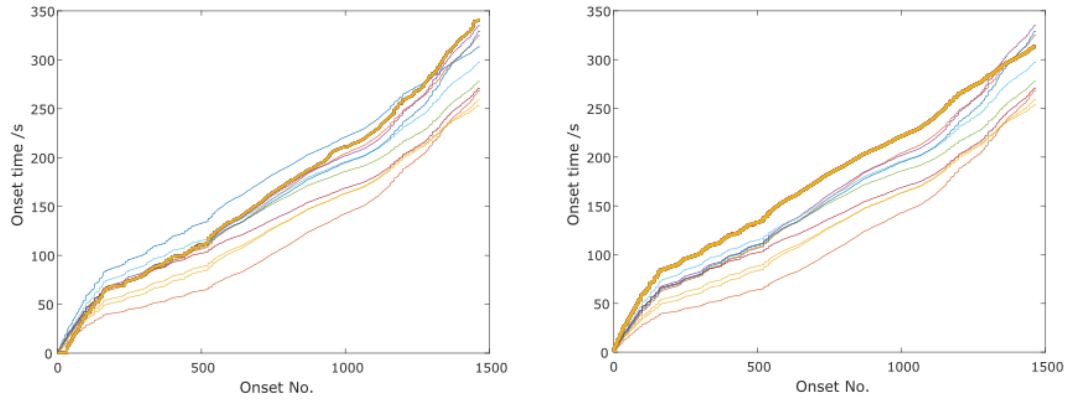


Figure 4.10: The onset locations for each of the generated renditions are shown before and after alignment. The thick yellow path in both figures shows the final alignment path. The figure on the right is pairwise alignment, and the figure on the left is iterative alignment.

real recordings. This might lead to excessive gap symbols being inserted if the variation in the length or the local tempo of the pieces is too large, although this is unlikely to be the case, as the difference between the shortest recording at 4:17 and the longest at 5:39 is only just over a minute. The reason why iterative alignment did not work as well as pairwise alignment is likely because the pieces are not being directly aligned to one another, they are being aligned to an average, which leads to slightly less exact locations of features and hence a higher error. However, by doing this it could make the alignment more robust to errors that can occur in pairwise alignment with a large number of pieces, although more experimentation would be required to confirm this. Given that progressive alignment is not working to a high enough standard, and that iterative alignment did not perform as well as pairwise alignment for a small number of pieces, pairwise alignment is used for the experiments in the next chapter on group-wise music transcription.

The onsets of pieces both before and after alignment are plotted for the pairwise and iterative methods in Fig. 4.10. For the pairwise method it is clear to see that it has aligned to a single reference track, and in the case of iterative alignment it appears to have found an "average" alignment path through the original onsets that is more representative of the collection of pieces.

# Chapter 5

# Group-wise Automatic Music Transcription (GWAMT)

The work in the preceding chapters has focused on developing methods that can be used to test the GWAMT hypothesis. Chapter 3 on frame-wise transcription outlines the system for performing automatic transcription which is a CNN that takes a segment of the CQT as input, and outputs the likelihood of a note being present in the frame. This system was trained on a combination of synthetic, generated data and real data taken from the MAPS database. As the network has a low number of parameters, it is unable to learn note specific filters, so note specific thresholds were learned on the MAPS database. This however did not perform as well on the final generation of the network as it did previously. In Chapter 2, several methods for the group alignment of audio using DTW are outlined and compared using synthetic recordings in order to select the best method for testing the hypothesis: pairwise alignment. This was selected as iterative alignment did not perform as well overall and progressive alignment was unable to be implemented to a high enough standard.

In this chapter, the group-wise transcription hypothesis will be tested, that multiple renditions of the same piece of music can improve the transcription accuracy. Since this is quite a broad hypothesis, a select number of small, simple tests are performed in one area that provide evidence in favour of the hypothesis. Group-wise automatic music transcription on the frame level is looked at, if multiple corresponding frames from renditions of the same piece of music can be used to improve the average frame level score for the entire piece. Approaches to combine the frame level output of each piece are considered and compared, determining which methods work the best. What is not considered is how to select the best piece for transcription. This as it turns out might be more useful for group-wise AMT as, quite often, a single piece out of the group will give a significantly higher

transcription score overall. So ideally, if a piece has characteristics that work especially well with the method of transcription being used, then this piece should be selected. This has not been considered due to the difficulty in knowing if one frame or piece will be more accurate than any other. The outliers could be removed, however this would likely only hinder the approach as it is could be the outlier that would give more accurate results, for instance a rendition that has remarkably low noise. In the case of selecting the best frames, this could extenuate any errors caused by a slight misalignment. It is also a preferable characteristic of any method to try and use all of the sources of information available, therefore improving with the number of pieces available. To my best knowledge, group-wise transcription has not be attempted previously and as such the methods and experiments that follow are a part of this work's novelty. Additionally, areas of potential research are identified and outlined where it was not possible to carry out research due to time constraints.

The instant application of GWAMT is for giving a transcription when multiple renditions are available. This, however, is not often the case in the transcription of a single piece or melody. What is more frequent in individual pieces is that small sections of the piece are repeated. These small sections can be:

**Phrases, melodies and trills** These are a string of notes that are usually repeated straight away in the piece

**Musical sections** Such as a chorus or a much longer repeated section in a piece

**Fugues/Themes** A fugue in classical music where usually each instrument will play a similar if not identical melody sequentially

**Chords** Chords of the same harmonic distribution could be identified across the piece

We saw in Chapter 1, that the way humans transcribe music is to use multiple passes to first transcribe the structure of the piece, then dominant themes and then finally minor details [16]. From this we can infer that they use multiple examples of the same melody, phrase etc. to ensure that they have the correct transcription. This is a more practical application, where group-wise transcription could have an application for the wider field of music transcription, not limited to piano transcription. The same musical phrases can occur at any time in the piece, and a transposed version of the phrase can also occur at multiple pitch realizations. This could decrease the amount of mutual information that two examples of a melody give, therefore improving the group-wise transcription result. Several methods exist for detecting repeated phrases and sections, such as cross-correlation of timbre features [159] and non-negative matrix factorization of chroma

features [159]. Such methods could be combined with this research to improve the frame level transcription of a piece of music, prior to language models being applied.

Some theory associated with achieving a consensus from multiple sources of information will be introduced, before detailing the experiments carried out to test the hypothesis. The method found that works best is a late combination of features, where the network prediction of corresponding frames in each piece are combined. Experimenting with the method for combination shows that a logarithmic opinion poll was the best performing but the majority vote the most reliable. This improves upon the average individual transcription score of a group of pieces, where the same piano is used to generate each recording.

## 5.1 Theory

A comparison could be drawn between GWAMT, where different realizations of the same ground-truth are shown to a classifier, which in this case is a CNN, to the problem of combining multiple classifiers, where the same example is shown to several classifiers, and the output of each combined. In this case, if the classifiers are correlated, so that they have the same architecture and are trained on the same data, then the average error of the classifiers will be equal to the error of result of combining the classifiers [160, 161] . However this is not quite true for GWAMT as it is the data that adds variation and not the different initial configurations of multiple classifiers.

Directly relevant to this thesis is the field of consensus theory, which given the individual "opinions" of a group of "experts" aims to find the best decision or concensus for the problem being solved. In our case the "opinions" are the frame level outputs, and the "experts" are the separate renditions/realizations. This is know as the "expert problem" where the decision maker is separate to the group of experts contributing the information. One of these methods is a linear opinion poll or a weighted average $C(p_1, ..., p_n)$, where each source is weighted corresponding to its reliability,

$$C(p_1, ..., p_n)(X) = \sum_{i=1}^{n} \alpha_i p_i(X) \tag{5.1}$$

where $p_i$ are the probabilities of each of the note classes, $X$ is the data, or frames in question from each of the sources, and $\alpha_i$ are the source weights. The weights can either all be set to the same value $1/n$ or they can be set separately for each source based on an estimate of the source's reliability by the decision maker. Another

method for forming a consensus is the logarithmic opinion poll

$$C(p_1, ..., p_n) = \frac{\prod_{i=1}^{n} p^{\alpha_i}}{\int \prod_{i=1}^{n} p_i^{\alpha_i} d\mu} \tag{5.2}$$

where the denominator is an integral with respect to a reference measure $\mu$. In this case, we will assume that the denominator is fixed, in which case it takes the form of a threshold applied to the numerator. The advantage of this method is that if the transcription of each piece predicted the correct collection of notes but with a probability of 0.4, because there is consensus at a lower probability threshold, the logarithmic pool method would give the correct prediction. In other methods such as the majority vote, the prediction from one piece would not be considered a vote unless the probability of that note being present was above 0.5. This helps to filter out unwanted noise in reaching a consensus. Note that in this method, zero probabilities act as vetoes.

## 5.2 Late Feature Combination

### 5.2.0 Method

Late feature combination means that the features obtained from each separate piece, i.e. CQT, network output, are combined after the transcription process. The reason why this method should improve the transcription accuracy is as follows. Assuming a limitless number of renditions of a particular piece and an accurate alignment mapping each piece to a mean, the resulting error from combining the classifier output from each piece can be attributed to the classifier and not to noise, unusual harmonic signatures from the piano or variations in interpretation between pianists. A limitless number of pieces is not available, however it is expected that the error attributed to sources other than the classifier should decrease as the number of pieces increase. This would be expected as the amount of total information about the "hidden" piece (hidden behind noise, rendition style) increases. The total new information added to the collection from each new piece will decrease as the number of pieces increases due to there being more mutual information between the new piece and the existing collection.

Assuming each piece has been accurately aligned prior to transcription, the alignment path between the piece and the reference piece are used to sample from the CQT so each frame has a corresponding frame in every other piece. This process is explained more formally, if there is an alignment path $p = (p_i, p_j) = ((i_1, ..., i_k), (j, ..., j_k))$ between each piece and a reference track, that maps the positions in the piece, $i$, to the positions in the reference track, $j$, then the positions in $p_i$ that correspond to unique positions in the reference piece $unique(p_j)$ are the

**Algorithm 5** An outline of the late feature combination method for performing GWAMT. The **pairwise** function computes a group alignment between one randomly selected reference track and the rest of the pieces. It returns the index of the reference track, $ref$ and an array of alignment paths, $p$. The *scalePaths* function selects path elements relative to the reference track, and scales the paths to be of the same magnitude as the length of the CQT for each piece. The *selectFrames* function selects a range of frames from the first argument centered on the index passed in the second argument. The **transcribe** function evaluates the input frames using the developed transcription algorithm. The **average** function combines the transcriptions from each of the renditions and gives the group transcription.

**procedure** LATE FEATURE COMBINATION
  $tracks \leftarrow$ *The renditions to be transcribed*
  $N \leftarrow$ *The number of renditions*
  $X_{CQT} \leftarrow \mathbf{CQT}(tracks)$
  $X_{CENS} \leftarrow \mathbf{CENS}(tracks)$
  $p, ref \leftarrow \mathbf{pairwise}(X_{CENS})$
  $length^{ref} \leftarrow$ *The length of the reference track used for alignment* $X^{ref}$
  $p \leftarrow \mathbf{scalePaths}(p, X_{CQT})$
  $i \leftarrow 1$
  $framePreds \leftarrow$ *An empty array to hold predictions*
  **while** $i \leq length^{ref}$ **do**
    $k \leftarrow 1$
    **while** $k < N$ **do**
      $frames \leftarrow \mathbf{selectFrames}(X^k_{CQT}, p^k_i)$
      $framePreds^k_i \leftarrow \mathbf{transcribe}(frames)$
      $k \leftarrow k + 1$
    $i \leftarrow i + 1$
  $output \leftarrow \mathbf{average}(framePreds)$

positions from which to sample the CQT in each piece. The magnitude of the alignment path must be scaled to the number of frames in the CQT. After this the frames should be combined with a suitable method. For an overview of the late feature combination method for GWAMT, see the pseudo-code provided in Algorithm 5. Several averaging methods are tested:

**Linear opinion poll** This is a simple average as the weights are chosen to be equal. Weighting the pieces based on their estimated signal-to-noise ratio was attempted but found to be worse than equal weighting where you assume no knowledge about the pieces reliability. See Eq. 5.1.

**Logarithmic opinion poll** This is the product of all the frame-level transcrip-

tions, as shown in Eq. 5.2. The denominator of this equation, or threshold, is investigated. The results from the investigation are shown later in this chapter.

**Majority Vote** Here a majority vote is carried out on the quantised transcriptions. For each piece, if a note was predicted present in one frame, that is equivalent to a "vote" for that note. Across all the pieces if there is a majority that consider a note present then the note is predicted by the majority vote method. All votes count equally, and no pieces get more than one vote. Although if knowledge about the reliability of the pieces was known in advance then this could be changed.

**Max** The maximum network output for each note is taken in each frame. This method does not remove any of the positive note predictions (true and false) from any of the pieces. This could be unfeasible for a large number of pieces where the number of false positives would drastically increase.

**Median** The median value of the network outputs is taken in each frame.

For evaluating the described method, synthetic data is used as with the evaluation in the previous chapter. This is due to the difficulty of obtaining labeled ground-truth data. Initially a single MIDI track is used to generate 10 renditions by the method outlined in Chapter 4. These 10 renditions are used to compare the proposed methods for group transcription. The pieces are aligned using the pair-wise alignment method to give an alignment path between one of the pieces and all the others. For each of the generated renditions a base-line is calculated to compare the late combination method against. The pieces are transcribed individually and their corresponding F1 scores on the frame level calculated, these scores are calculated by using the ground-truth piano-rolls provided by augmenting the MIDI track in the same way as the renditions were generated. For evaluating the produced group transcription the ground-truth piano-roll is warped for each of the pieces according to the alignment path calculated using pair-wise alignment and a new F1 score is calculated. This gives N scores before and after group-wise transcription that can be directly compared. The scores before alignment will be called BA (before alignment), and afterwards PA (post alignment). The condition for testing the hypothesis is that if the best and the average PA score are greater than the average BA score then GWAMT can be said to have been successful in improving the transcription. If only the best PA score is greater than the average BA score then it is still likely to have improved the transcription. However, due to slight inaccuracies in the alignment, the warped ground-truths used for evaluation are different for each piece, therefore it makes sense to compare the BA and PA scores on a piece by piece basis.

### 5.2.0 Results

The results for the preliminary experiment for each of the combination methods are shown in a combined table in Table 5.1. The only methods to improve upon the BA average score are the max and logarithmic opinion poll methods. However these methods have some shortcomings. For the max method, if one piece is particularly noisy then this could ruin the groupwise transcription. Although having a single prediction containing all the TPs from each piece is always good for the transcription, this would be accompanied by having all the FPs. This makes this method unsuitable for a large N. The reason why the max method worked particularly well is because generally the noise and number of false positives is quite low, and therefore the gain should outweigh the additional number of false positives added to the combined transcription. This does not show to be the case in the following larger experiment.

From these preliminary experiments, the majority vote method is the best candidate for taking into the next experiment. Although the max and Logarithmic methods performed better in this preliminary experiment, they have shortcomings. The logarithmic method relies of having to know the correct threshold, and the Max method will not scale to large numbers of pieces as it will generally select the frame with the most noise. The results show that this method, majority vote, warrants further experimentation, and as such a larger experiment is designed and performed in Section 5.4.

## 5.3 Early Feature Averaging

### 5.3.0 Method

This second attempted method for GWAMT is called early feature averaging as the point when data from each of the sources are combined, happens earlier than in the previous method. In this method, the points at which frames are selected is the same as with the previous method, however before passing the frames to the CNN for classification, they are averaged across all of the sources, so that a single, average frame is presented to the CNN. The reasoning for this if it was tested on real data, would be that the harmonic power distribution could become more uniform, as harmonic distributions that are particularly different to the pianos used for training could lead to poor results. A widening of the frequency peaks should also happen if the test data contained multiple pianos, although it is unclear if this would help or hinder the result of the classification. Another motivation is that the effect of noise in the CQT would be reduced. However as the methods for testing are only across a single piano, this was not expected to give any major advantage. This method could potentially reduce the effect of any misalignment between the

| Piece | ITS | Linear opinion poll | Log. opinion poll | Maj. Vote | Max | Median |
|---|---|---|---|---|---|---|
| 1 | 0.684 | 0.660 | 0.702 | 0.671 | 0.684 | 0.664 |
| 2 | 0.684 | 0.676 | 0.722 | **0.693** | 0.711 | **0.684** |
| 3 | 0.668 | 0.670 | 0.717 | 0.687 | **0.713** | 0.678 |
| 4 | 0.712 | 0.676 | 0.722 | 0.691 | 0.705 | 0.683 |
| 5 | 0.676 | 0.675 | 0.718 | 0.687 | 0.696 | 0.680 |
| 6 | **0.717** | **0.680** | **0.723** | 0.692 | 0.698 | **0.684** |
| 7 | 0.697 | 0.678 | 0.719 | 0.688 | 0.690 | 0.681 |
| 8 | 0.673 | 0.671 | 0.717 | 0.686 | 0.703 | 0.677 |
| 9 | 0.679 | 0.633 | 0.680 | 0.649 | 0.689 | 0.640 |
| 10 | 0.647 | 0.625 | 0.692 | 0.647 | 0.711 | 0.635 |
| Best | 0.717 | 0.680 | 0.723 | 0.693 | 0.713 | 0.684 |
| Average | 0.684 | 0.665 | 0.711 | 0.679 | 0.700 | 0.671 |

Table 5.1: Several different methods for combining results from 10 renditions of Clare de lune are tested. ITS is "Individual Transcription Score". The quoted results are the F1-measure of the piece in question. Taking the maximum activation for each frame and each note gives the best combined transcription score. However all of the above methods except for the median improve upon the average individual transcription result.

pieces, as with a large number of pieces that are partially misaligned, the resulting average would give a steady increase and decrease in the power of the onset and offset.

Two averaging techniques are used to test this method, a straightforward average of all of the frames, and selecting the frame with the maximum total power content. The max method might cause the output to contain an excessive number of false positives.

### 5.3.0 Results

This GWAMT method is evaluated in the same fashion as the late feature averaging method, with a single MIDI track being used to generate 10 renditions.

The results can be seen in Table 5.2. This method fails to improve either the best or average transcription score. This could be because the network is trained on data with well defined onset, offset and band widths, so when the spectrograms are averaged it removes clarity and adds ambiguity. Perhaps there is an improvement

| Piece | ITS | Early feature averaging | Maximum frame activation |
|---|---|---|---|
| 1 | 0.684 | 0.659 | 0.631 |
| 2 | 0.684 | **0.682** | 0.647 |
| 3 | 0.668 | 0.679 | **0.648** |
| 4 | 0.712 | 0.678 | 0.644 |
| 5 | 0.676 | 0.676 | 0.646 |
| 6 | **0.717** | 0.679 | 0.645 |
| 7 | 0.697 | 0.675 | 0.641 |
| 8 | 0.673 | 0.673 | 0.642 |
| 9 | 0.679 | 0.642 | 0.620 |
| 10 | 0.647 | 0.640 | 0.618 |
| Best | 0.717 | 0.682 | 0.648 |
| Average | 0.684 | 0.668 | 0.638 |

Table 5.2: Results for the early feature averaging method. The rows are the F1 scores using the warped ground-truth for that piece, and the method from the corresponding column. ITS stands for Individual Transcription score

but its effect is masked due to the reduction in F1 score caused by this blurring, however further experimentation is required to test this. however, if this was the case then the CQT signal could be cleaned up prior to being fed into the classifier. Taking the maximally activated frame across all the pieces using the Early feature averaging method was attempted and experimented with, however this worsened the results drastically. This could have happened for a number of reasons, by taking the maximum it could have more frequently selected the frame that had the worse alignment, and therefore was closer to an onset than the other frames, it also could have selected the most noisy frame, or have selected the frame with the most power in the lower harmonics, reducing the effectiveness of classifying high notes.

Given that this initial experiment shows that the method performs poorly in all cases, it is discarded from this point on-wards in favour of the late feature averaging method.

## 5.4 Late Feature Averaging: Larger experiment

For the larger experiment 15 MIDI tracks are used, some taken from the MAPS database and others from various online resources. All of which are classical piano

pieces. The method used is as described in Section 5.2, with some differences, the results are averaged across all of the MIDI tracks, and 20 renditions are generated as opposed to the 10 that were generated in the preliminary experiment.

The goal of this experiment is to confirm the results of the preliminary late feature averaging experiment, and also to test how the improvement changes with an increasing number of pieces. The number of pieces was varied between 2 and 20, with the improvement from the mean and best BA score recorded for each method. These results can be seen in Fig. 5.1.

The results show that given a number of candidate transcriptions, the average frame level accuracy can be improved by either the majority vote, median or logarithmic opinion poll method. However at this point the result from the logarithmic opinion poll is unreliable as it is based on being able to determine the best threshold on a piece by piece basis. This is unreasonable, and is experimented with further later on. A natural expectation is that the transcription would improve with the number of pieces available as you then have more examples to discern a transcription from. This however was not seen so much, the improvement quickly plateaus after 5 or so pieces. This is not particularly desirable, as then the problem of choosing which pieces to use for the group transcription arises. It is likely that the point at which it plateaus is due to the added noise and ambiguity outweighing the additional gain. However where this happens I expect would vary depending on the piece and recording situation, as this experiment is too limited to make another conclusion.

The results from this experiment are quite different to the preliminary experiment which showed the max method performing well. The max method has performed much poorer this time, which was expected. For the majority vote method, it performs better when presented with an even number of samples. It is not obvious why this would be the case, as it would be expected that an odd number of samples makes a majority more likely. Whilst the improvement does not change much past 7 pieces, the error reduces greatly, making the method more robust. This can be seen in Fig. 5.1 by the narrowing of the error bars as the number of pieces increases.

As mentioned, the results from the logarithmic poll method are dependent on being able to identify the optimum threshold. In reality this is not practical, as you would have to already have part of the piece transcribed for each rendition. However, it was useful to see that the method can be used to give an improvement. To investigate this method further the threshold for the method (the denominator in Eq. 5.2) was changed globally for all of the pieces and renditions used in the previous experiment. This is more of a feasible approach as the method can be set without any further calculation. The global threshold is also varied, the results from this experiment are shown in Fig. 5.2.

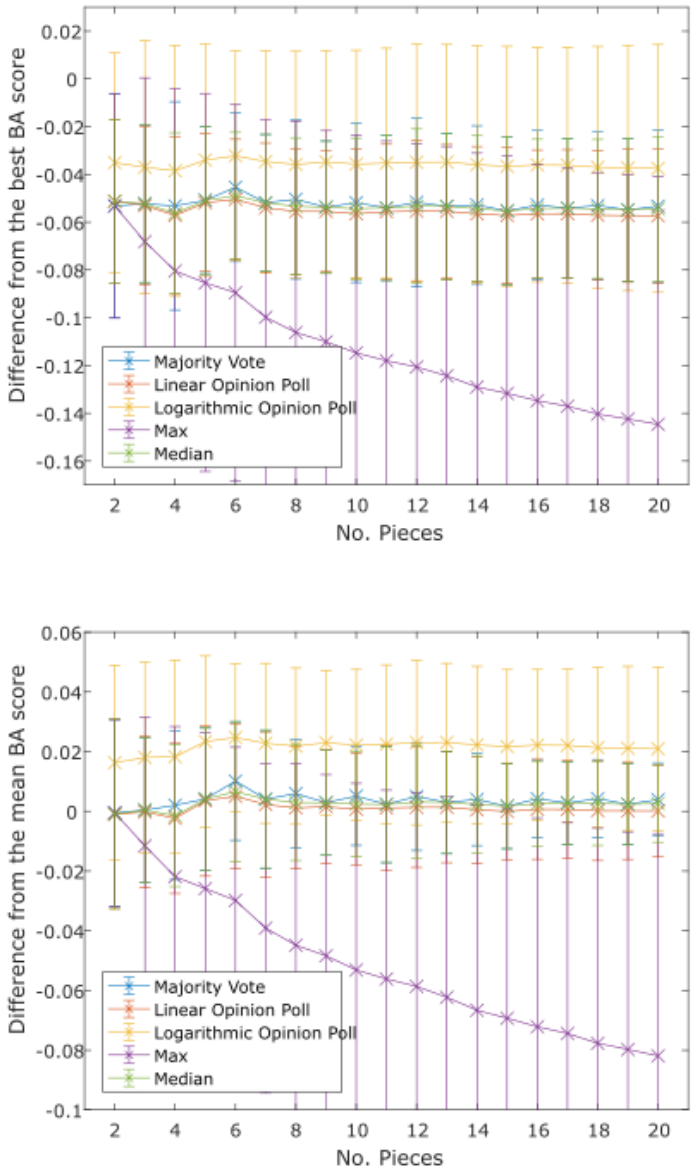


Figure 5.1: The differences between the BA scores and the PA scores are shown when comparing against the best overall transcription (top), and the mean transcription score (bottom) for an increasing number of pieces. A positive difference indicates that the method has been successful in improving the score from either the best or mean BA score. The bottom figure shows all methods apart from the Max method giving some improvement over the average score.

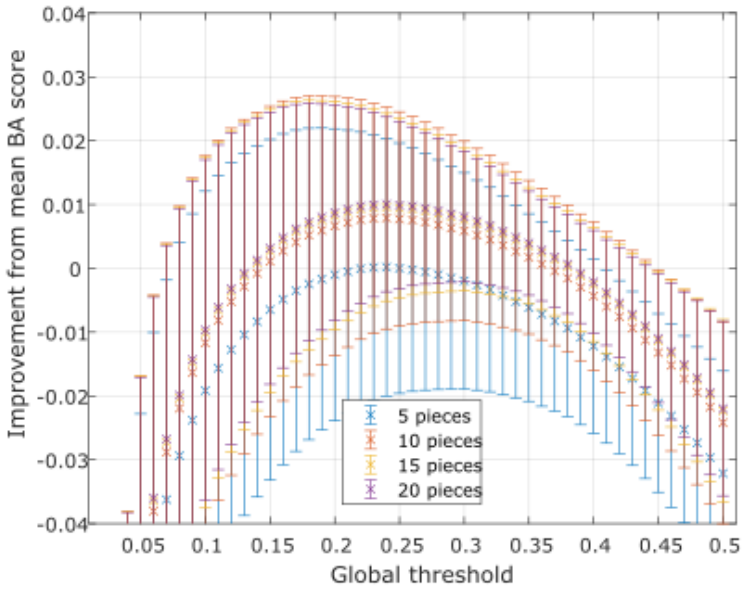


Figure 5.2: The threshold for the logarithmic opinion poll method is varied and applied to all of the pieces and renditions from the larger experiment. The error is the standard deviation across all of the pieces.

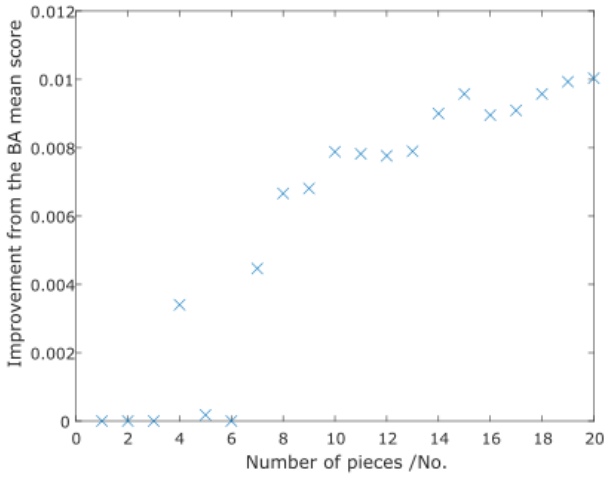


Figure 5.3: The improvement using the optimum global threshold for the logarithmic opinion poll is shown for an increasing number of pieces. The improvement increases with the number of pieces, plateauing at about a 1% improvement.

What was found is that naturally the logarithmic opinion poll performs much worse now that the threshold is set globally. However, it does still improve upon the mean when more than 5 pieces are available. This method also improves with the number of pieces available, see Fig. 5.3, which is different from all of the methods tested previously. Another advantage of this method is that the range of thresholds that still give an improvement is quite large, and as such even if the threshold was chosen poorly the method would still perform reasonably well. This experiment suggests that the logarithmic opinion poll is the best method when more than 5 pieces are available, however to confirm this the same experiment would have to be performed with real data to confirm that this is not just due to how the experiment was carried out.

## 5.5 Summary

This chapter has investigated and experimented with, methods for performing GWAMT. It combined the AMT method described in chapter 3, with the pairwise alignment method described in chapter 4. Late feature averaging, where the classifier output was combined was tested initially, with a preliminary experiment showing that the max and logarithmic opinion poll averages improved upon the individual baseline. A larger experiment was performed using the same setup however with 10 different MIDI tracks, and 20 pieces generated for each. This experiment confirmed that the late feature averaging method improved upon the baseline results, however, the max method performed worse than the preliminary experiment, and the other methods performed better. This might be due to the particular MIDI track chosen for the preliminary experiment.

The logarithmic opinion poll was chosen for further investigation, and it was shown that a global threshold can be used to give an improvement that scales with the number of pieces available. This method and the majority vote, improved the transcription score up to 1%. Whilst this improvement is minor, it capitalizes on differences between each rendition, which the usual AMT approach would not be able to. As such it complements AMT, and could potentially give an improvement regardless of the AMT method, although this is not tested. This improvement is also in the case where the same piano is used to generate each of the pieces. In the more realistic situation where each piece was from a different piano, the improvement is likely to increase, as the amount of mutual information between pieces will be lower.

A GWAMT method where the CQT spectrograms are average prior to transcription, called early feature averaging, was experimented with using a single MIDI track. However, this method performed poorly and was discarded in favour of late feature averaging.

Ideally real data would be used for testing and this is a major limitation to this work, however, the experiments in this chapter have demonstrated that group-wise transcription can be used to improve the transcription score.

# Chapter 6

# Conclusion

The Chapters in this thesis have been stages towards understanding and developing a method for group-wise transcription. The first two Chapters set out the context for this research, the applications for AMT, the research that has been done to that end, and how group-wise transcription could aid AMT. In the third Chapter a method for frame-level piano transcription was developed and tested against the MAPS database, achieving state of the art results. The fourth Chapter investigated different methods for the group alignment of audio, introducing one method for group alignment, and the use of time windows for distance comparison to make the alignment more robust. The final Chapter then tied together the methods from Chapters 3 and 4, to test the group-transcription hypothesis, showing that group-transcription is possible and warrants further investigation. This structure has allowed the research to be performed sequentially, with the transcription system being fully developed, then methods for audio alignment, then finally group-transcription. However if structured in a more dynamic way, focused wholly around testing the main hypothesis, it would have given more control of the research direction as promising avenues would have been discovered earlier on in the project. This is not necessarily a bad outcome, as the additional time spent on frame-level transcription has produced good results, but if repeated the project would have been more focused on group-wise transcription.

In Chapter 1 the way that human experts transcribe music was discussed as they still perform music transcription better than AMT, concluding that human experts transcribe music in an iterative way, accumulating additional information on each successive pass. Annotating global information such as the key signature and musical structure first and finer details last. It was also seen that humans are apparently bad at discerning the notes present when only given a short sample (40ms) of music, this is the one area that computer algorithms perform well, and therefore should be exploited. This is the other way around for musical language

models, they are generally worse than human transcription performance. This leads on to a large problem with AMT algorithms, normally when they make errors they are not musically plausible, however when humans make a transcription error, they are usually musically plausible, and as such affect the resulting transcription very little when it comes to performance. This is because humans can draw upon a wealth of experience playing and transcribing music, whereas most transcription algorithms rely on having learned good representations during training. This provides a good argument for group-wise transcription, which has the goal of using multiple renditions of the same piece or melody to improve the transcription performance. As when humans aim to transcribe a piece of music, they will use repeated melodies and rhythms within the same piece of music to aid each transcription, ensuring consistency and making the transcription more musically plausible.

Chapter 2 conducts a literature review into music transcription, identifying that generally music transcription systems have been made up of separate frame-level and note-level musical language models. A comparison is also drawn between music transcription and speech recognition, where language models are relied upon to provide priori probabilities. Whilst this is not possible for music transcription due to to the large output space, several other attempts have been made at incorporating musical language models that have shown good success. This literature review identified that frame-level transcription was the best route to testing the group-wise transcription hypothesis, as often the language models are complex and would be too time consuming given the length of the project. However, the potential of using note-level transcription for group-wise transcription is discussed later in this conclusion.

A frame-level transcription system was developed in Chapter 3. The model used CNNs to classify frames of musical audio from a piano. The input to the model is a two dimensional slice of the CQT centered around the frame being classified. A CNN in conjunction with the CQT was chosen due to the pattern recognition prowess of the CNN and the note-translational properties of the CNN. The architecture used for the CNN was based around architectures that had worked well in the field of image recognition [123], and altered for the problem of music transcription based on the differences between the problems. The key features of the architecture were the use of skip connections, separate time and frequency convolutions and the lack of fully connected layers. The training methodology used was novel, dynamic training on synthetic randomly generated chords was used so that the network learns to generalize far better than only training on musically meaningful data. This was found as the network performed just as well on musically meaningful synthetic data as it did otherwise. Then after this training had completed the network was trained on the MAPS database in the same

configuration as used for testing in the literature [32, 34] with the resulting network outperforming the previous state of the art in frame-level piano transcription. This shows that generated data can be used to improve the training of CNNs for AMT. This principle could be extended to a much larger network that can transcribe instruments other than the piano however in this case the filters would have to be adapted. This would be to allow for instruments that have timbres that have a more complex time-frequency relationship, such as woodwind instruments. A useful analysis would be to investigate the learned representations of the network. Kelz et al. [33] performed experiments on how well neural network based transcription systems generalize to unseen data, showing that instead of learning to recognize and separate out overlapping note content, then network learns to recognize note combinations, but if the inner-workings of the network could be analyzed further to ascertain what each filter is contributing to the overall transcription then this would help in the design of larger scale networks. De-convolution, where the filter activations can be visualized, or occlusion (masking parts of the input spectrogram) to see how important parts of the spectrogram to the final transcription, could be used in a potential analysis.

Chapter 4 on audio alignment was needed due to the variation in note event location between renditions performed by different musicians. As separate pieces cannot be used to form a consensus (group-wise transcription) if there is no knowledge about when in the pieces that corresponding notes occur. A distinction between two types of group-wise transcription was drawn, frame-level group-wise transcription, where information is combined at each individual frame, and note-level group-wise transcription, where the notes are identified and all frames in each note across all renditions are used to form a consensus. If the transcription was being performed on the note level, then only the onset and offset locations in each piece would need to be known, and as such an onset and offset detector could be used (which are well defined for the piano), with the transcription being performed not on the frame level but on the note level, perhaps as done by Cheng et al. [56]. However as it was frame-level transcription that was investigated in this thesis, a frame-frame correspondence was needed between each pair of pieces. To achieve this correspondence, methods for multiple audio alignment were investigated. Three methods were compared, pairwise alignment, progressive alignment and a new method, iterative alignment. To compare the three alignment methods, multiple renditions were created from a single MIDI track by emulating playing style variations, this was the easiest way to get ground-truth information for testing given the time constraints, however with more time, beat information for performances of Chopins mazurkas, provided by the mazurka project [158], would have been used. These renditions were aligned using each method and the average note onset deviation compared. Pair-wise alignment performed the best

out of the three methods, however, some difficulties with progressive alignment were encountered due to the time constraints. The combined feature representation as used by Wang et al. [27] (DLNCO + CENS), could not be used so just CENS features were used. this led to the method not performing as well as reported. The iterative alignment method was only experimented with briefly, however the method showed promising results, performing almost as well as pair-wise alignment. Should the method be developed and optimized further then it could provide a robust method for the multiple audio alignment. Specifically it could provide a method that has a linearly scaling computational cost, as opposed to progressive alignment. This means that it is feasible to align vast numbers of musical pieces, which could make the performance analysis of renditions taken from online resources like Youtube possible.

Chapter 5 used methods developed in the previous two chapters to test the group-wise transcription hypothesis. As in the chapter on audio alignment, multiple synthetic renditions were generated from MIDI tracks, providing ground-truth information. Early feature averaging and late feature averaging were compared in an initial experiment using only a single MIDI track and 10 renditions. From this initial experiment it was clear that late feature averaging improved the transcription and early feature averaging did not. A larger experiment using 15 MIDI tracks was then performed, they were used to generate 20 renditions each, all of the renditions for each piece were aligned using pairwise alignment providing a frame-level correspondence between each rendition. Several methods for combining the aligned and transcribed frames were investigated. From this investigation two methods are worth mentioning, the majority vote method and the logarithmic pool method. The majority vote method improved the score from before alignment (BA score) when any number of renditions is available, however a lower number was generally preferred. What was unexpected is that increasing the number of renditions used in the combination did not always improve the results of the transcription, and in fact it appeared to drop after 6. Using this method the improvement from the BA average F1-score was around 0.005 or 0.5%. The second method that worked well was the logarithmic pool method, this method however only improved the transcription score for when there are more than 6 pieces available for alignment. However the improvement from the BA baseline did improve with the number of pieces, with an improvement of 0.01 or 1% when al 20 renditions where used. However after all of these experiments no method could improve upon the BA best score, which is the upper outlier of the renditions. This indicated that certain performance characteristics, including the tempo, noise level and reverb could determine the baseline transcription. If the outlier could be determined then this would be more valuable for group-wise transcription than trying to combine information from every piece, as a single noisy piece could ruin

or affect the group-wise transcription score.

## 6.1 Future Work

Testing the hypothesis on synthetic data generated from the same piano will have made the problem harder as there is less variation between pieces. It has also made the problem more constrained, and hence conclusions about the results of the experiments easier. The improvement seen using the BA average method is likely due to this small variation caused by noise and differences caused during the generation and alignment. As the problem is harder and positive results were seen, further improvement should be seen in future work. However, as the improvement was minor and no statistical analysis has been performed on the results, there is not enough evidence to support the hypothesis.

To test the hypothesis of group-wise transcription, only experiments on the frame level and on piano music were performed. The positive results here means that in the future, further research could investigate more general scenarios with multiple different instruments, and also with note-level transcription. This would be done by training a network that has many more filters on data that contains multiple instruments. An additional piece of further work is using phase in neural network based transcription systems. This was noticed when performing the literature review, that very few methods had tried to incorporate phase, with the method here being no different. Since the literature review was performed, this exact problem has been solved by Trabelsi et al. [162], who achieve state of the art frame level transcription results using a deep complex network. Using both the complex network outlined and by adapting the architecture detailed here, the results for frame level piano transcription could be improved.

Given more time the mazurka project would provide a better way to evaluate the method for group-wise transcription introduced and also test the hypothesis. The evaluation could have been improved by using more MIDI tracks and also by using multiple pianos to generate the recordings. This would increase the variation between the individual renditions and make the evaluation more robust.

# Bibliography


[1] Oxford University. Oxford English dictionary. www.oed.com/oed2/00256125, 2017. Accessed: 2017-09-30.

[2] M Benward, B. Saker. *Music in Theory and Practice*. McGraw-Hill, 2003.

[3] A.T. Cemgil. *Bayesian music transcription.* PhD thesis, Radboud University, 2004.

[4] Emilios Cambouropoulos. From MIDI to Traditional Musical Notation. *Working Notes of the AAAI Workshop on Artificial Intelligence and Music: Towards Formal Models for Composition, Performance and Analysis*, pages 19–23, 2000.

[5] Kirill Sidorov, Andrew Jones, and David Marshall. Music Analysis as a Smallest Grammar Problem. *International Society of Music Information Retrieval Conference (ISMIR)*, pages 301 – 306, 2014.

[6] Robert. Rowe. *Machine musicianship*. MIT, 2004.

[7] S S Stevens. *Psychophysics: Introduction to its perceptual, neural, and social prospects*, volume 329. Transaction Books, 1975.

[8] Michael Epstein and Jeremy Marozeau. Loudness and intensity coding. In *Oxford Handbook of Auditory Science Hearing*. Oxford University, 2012.

[9] Anne Caclin, Stephen McAdams, Bennett K Smith, and Suzanne Winsberg. Acoustic correlates of timbre space dimensions: A confirmatory study using synthetic tones. *The Journal of the Acoustical Society of America*, 118(1):471–482, 2005.

[10] Nobuo Masataka. Music, evolution and language. *Developmental Science*, 10:35–39, 2007.

[11] E.H. Haimoff. Convergence in the duetting of monogamous old world primates. *Journal of Human Evolution*, 15:51–59, 1986.

[12] Andrew J Oxenham. Pitch Perception. *Journal of Neuroscience*, 32(39):13335–13338, 2012.

[13] J F Schouten. The residue and the mechanism of hearing. In *Koninklijke Nederlandsche Akademie von Wetenschappen*, volume 43, pages 991–999, 1940.

[14] John C. M Brust. This Is Your Brain on Music: The Science of a Human Obsession. *Neurology Today*, 8:41, 2008.

[15] Anssi Klapuri, Tuomas Virtanen, and Jan Markus Holm. Robust Multipitch Estimation for the Analysis and Manipulation of Polyphonic Musical Signals. In *Conference on Digital Audio Effects*, pages 233–236, 2000.

[16] Stephen Hainsworth. *Techniques for the automated analysis of musical audio.* PhD thesis, University of Cambridge, UK, 2003.

[17] Emilios Cambouropoulos. Automatic pitch spelling: From numbers to sharps and flats. In *VIII Brazilian Symposium on Computer Music (SBC&M)*, pages 2001–2012, 2001.

[18] Pring Linda. Savant talent. *Developmental Medicine & Child Neurology*, 47(7):500–503, 2005.

[19] Ramin Pichevar, Hossein Najaf-Zadeh, Louis Thibault, and Hassan Lahdili. Auditory-inspired sparse representation of audio signals. *Speech Communication*, 53(5):643 – 657, 2011.

[20] Daniel Patrick Whittlesey Ellis and David Felix Rosenthal. Mid-level representations for computational auditory scene analysis. *Computational auditory scene analysis*, pages 257 – 272, 1995.

[21] Jont B. Allen. Short-term spectral analysis, and modification by discrete fourier transform. *IEEE Transactions on Acoustics Speech and Signal Processing*, pages 235–238, 1977.

[22] S. S. Stevens, J. Volkmann, and E. B. Newman. A Scale for the Measurement of the Psychological Magnitude Pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190, 1937.

[23] Judith C. Brown. Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, 1991.

[24] John Paul Stautner. *Analysis and synthesis of music using the auditory transform.* PhD thesis, Massachusetts Institute of Technology, 1983.

[25] Meinard Müller, Sebastian Ewert, and Sebastian Kreuzer. Making chroma features more robust to timbre changes. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1877–1880, 2009.

[26] M. Mueller, Frank Kurth, and Michael Clausen. Audio matching via chroma-based statistical features. In *International Society of Music Information Retrieval Conference (ISMIR)*, pages 288–295, 2005.

[27] Siying Wang, Sebastian Ewert, and Simon Dixon. Robust and efficient joint alignment of multiple musical performances. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 24(11):2132–2145, 2016.

[28] Frank Kurth and Meinard Müler. Efficient index-based audio matching. *IEEE Transactions on Audio, Speech and Language Processing*, 16(2):382–395, 2008.

[29] T. Fujishima. Realtime chord recognition of musical sound: A system using common lisp music. In *International Computer Music Conference*, pages 464–467, 1999.

[30] Daniel P W Ellis. Classifying Music Audio with Timbral and Chroma Features. In *International Society of Music Information Retrieval Conference (ISMIR)*, volume 199, pages 339–340, 2007.

[31] Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch Estimation of Piano Sounds Using a New Probabilistic Spectral Smoothness Principle. In *European Signal Processing Conference (EUSIPCO)*, pages 1643–1654. IEEE, Aug 2010.

[32] Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. *CoRR*, abs/1612.05153, 2016.

[33] Rainer Kelz and Gerhard Widmer. An experimental analysis of the entanglement problem in neural-network-based music transcription systems. *CoRR*, abs/1702.00025, 2017.

[34] Siddharth Sigtia, Emmanouil Benetos, Nicolas Boulanger-Lewandowski, Tillman Weyde, Artur d'Avila Garcez, and Simon Dixon. A Hybrid Recurrent Neural Network for Music Transcription. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2061–2065, 2015.

[35] Siying Wang, Sebastian Ewert, and Simon Dixon. Compensating for asynchronies between musical voices in score-performance alignment. In *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 589–593. IEEE, 2015.

[36] Simon Dixon and Gerhard Widmer. MATCH: A Music Alignment Tool Chest. In *International Symposium on Music Information Retrieval (ISMIR)*, pages 492–497, 2005.

[37] J. A. Moorer. On the transcription of musical sound by computer. *Computer Music Journal*, 1(4):32–38, 1977.

[38] Alain de Cheveigné and Hideki Kawahara. YIN, a fundamental frequency estimator for speech and music. *The Journal of the Acoustical Society of America*, 111(4):1917–1930, 2002.

[39] M Ryynänen. *Automatic Transcription of Pitch Content in Music and Selected Applications*. PhD thesis, Tampere University of Technology, 2008.

[40] Emmanouil Benetos, Sebastian Ewert, and Tillman Weyde. Automatic transcription of pitched and unpitched sounds from polyphonic music. In *IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings (ICASSP)*, pages 3107–3111, 2014.

[41] Chris Chafe and David Jaffe. Source separation and note identification in polyphonic music. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 11, pages 1289–1292, 1986.

[42] Robert C. Maher. Evaluation of a method for separating digitized duet signals. *Journal of the Audio Engineering Society*, 38(12):956–979, 1990.

[43] Robert C Maher and James W Beauchamp. Fundamental frequency estimation of musical signals using a two-way mismatch procedure. *Journal of the Acoustic Society of America*, 95:2254–2263, 1994.

[44] Anssi Klapuri. Introduction to music transcription. *Signal Processing Methods for Music Transcription*, pages 1–28, 2006.

[45] Michael Jerome Hawley. *Structure out of Sound*. PhD thesis, Massachusetts Institute of Technology, 1994.

[46] Kunio Kashino and Hidehiko Tanaka. A Sound Source Separation System with the Ability of Automatic Tone Modelling. In *International Computer Music Conference*, volume 1993, pages 249–257, 1993.

[47] Keith D Martin. A Blackboard System for Automatic Transcription of Simple Polyphonic Music. *M.I.T Media Laboratory Perceptual Computing Section Technical Report No. 385*, 385:1–13, 1996.

[48] Darryl Godsmark and Guy J Brown. Blackboard architecture for computational auditory scene analysis. *Speech Communication*, 27(3):351–366, 1999.

[49] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[50] T. Virtanen. Sound source separation using sparse coding with temporal continuity objective. In H. C. Kong and B. T. G. Tan, editors, *International Computer Music Conference (ICMC).*, pages 231–234, 2003.

[51] Judith C Brown. Non-Negative Matrix Factorization for Polyphonic Music Transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, number 3, pages 177–180, 2003.

[52] Samer A Abdallah and Mark D Plumbley. Polyphonic music transcription by non-negative sparse coding of power spectra. In *International Society for Music Information Retrieval Conference (ISMIR)*, volume 510, pages 10–14, 2004.

[53] Emmanuel Vincent, Nancy Berting, and Roland Badeau. Two nonnegative matrix factorization methods for polyphonic pitch transcription. *2007 Music Information Retrieval Evaluation eXchange (MIREX)*, 2007.

[54] Emmanuel Vincent, Nancy Benin, and Roland Badeau. Harmonic and inharmonic Nonnegative Matrix Factorization for polyphonic pitch transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings (ICASSP)*, pages 109–112. IEEE, 2008.

[55] Holger Kirchhoff, Simon Dixon, and Anssi Klapuri. Multi-Template Shift-Variant Non-Negative Matrix Deconvolution for Semi-Automatic Music Transcription. pages 415–420, 2012.

[56] Tian Cheng, Matthias Mauch, Emmanouil Benetos, and Simon Dixon. an Attack / Decay Model for Piano Transcription. *International Society for Music Information Retrieval Conference (ISMIR)*, pages 584–590, 2016.

[57] Graham E Poliner and Daniel P W Ellis. Improving generalization for classification-based polyphonic piano transcription. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 86–89, 2007.

[58] Juhan Nam, Jiquan Ngiam, Honglak Lee, and Malcolm Slaney. A Classification-Based Polyphonic Piano Transcription Approach Using Learned

Feature Representations. In *International Society of Music Information Retrieval Conference (ISMIR)*, pages 175–180, 2011.

[59] Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An End-to-End Neural Network for Polyphonic Music Transcription. *IEEE Transactions on Audio, Speech, and Language Processing*, 24(5):1–13, 2016.

[60] Klapuri Anssi. A Perceptually Motivated Multiple-F0 Estimation Method. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 291–294, 2005.

[61] Chunghsin Yeh, Axel Roebel, and Wei-Chen Chang. Mutiple-F0 Estimation for MIREX 2008, 2008.

[62] Chunghsin Yeh, Axel Roebel, and Wei-Chen Chang. Mutiple-F0 Estimation for MIREX 2009, 2009.

[63] Chunghsin Yeh and Axel Roebel. Multiple-F0 Estimation for MIREX 2010. *MIREX*, 2010.

[64] Chunghsin Yeh and Axel Roebel. Multiple F0-estimation for MIREX 2011. *MIREX*, 2011.

[65] Karin Dressler. Multiple fundamental frequency extraction for MIREX 2012. *MIREX*, 2012.

[66] Emmanouil Benetos and Tillman Weyde. Multiple F0-estimation and note tracking for MIREX 2013 using a sound-state based spectrogram factorisation model. *MIREX*, 2013.

[67] Anders Elowsson and Anders Friberg. Polyphonic Transcription with Deep Layered Learning for MIREX 2014. *MIREX*, 2014.

[68] Emmanouil Benetos and Tillman Weyde. Multiple-F0 estimation and note tracking for MIREX 2015 using a sound state-based spectrogram factorization model. *MIREX*, 2015.

[69] Matija Marolt. Multiple fundamental frequency estimation & tracking submission for mirex 2016. *MIREX*, 2016.

[70] J Thickstun. Frequency Domain Convolutions for Multiple F0 Estimation. *MIREX*, 2017.

[71] Lawrence R. Rabiner and B. H. (Biing-Hwang) Juang. *Fundamentals of speech recognition*. PTR Prentice Hall, 1993.

[72] Lingholic. How many words do I need to know? The 95/5 rule in language learning, part 2/2. https://www.lingholic.com/how-many-words-do-i-need-to-know-the-955-rule-in-language-learning-part-2/, 2013. Accessed: 2018-05-15.

[73] P O Box and Fi Tampere. Polyphonic Music Transcription using Note Event Modelling. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, pages 319–322, 2005.

[74] Roland Badeau, Bertrand David, and Valentin Emiya. Automatic transcription of piano music based on HMM tracking of jointly-estimated pitches. In *European Signal Processing Conference*, pages 1–5, 2008.

[75] Tian Cheng, Simon Dixon, and Matthias Mauch. Improving piano note tracking by HMM smoothing. In *European Signal Processing Conference (EUSIPCO)*, pages 2009–2013. IEEE, 2015.

[76] Andrej Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks, 2015.

[77] Alex Graves. Sequence transduction with recurrent neural networks. *CoRR*, abs/1211.3711, 2012.

[78] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 38, pages 6645–6649, 2013.

[79] Sebastian Bock and Markus Schedl. Polyphonic Piano Note Transription with Recurrent Neural Networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–124, 2012.

[80] Nicolas Boulanger-Lewandowski, Pascal Vincent, and Yoshua Bengio. Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription. In *International Conference on Machine Learning (ICML)*, pages 1159–1166, 2012.

[81] Hugo Larochelle and Iain Murray. The Neural Autoregressive Distribution Estimator. In *International Conference on Machine Learning*, volume 15, pages 29–37, 2011.

[82] Siddharth Sigtia, Emmanouil Benetos, Srikanth Cherla, Tillman Weyde, Artur S. d'Avila Garcez, and Simon Dixon. An rnn-based music language model for improving automatic music transcription. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 53 – 58, 2014.

[83] A. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3089–3092 vol.6. IEEE, 1999.

[84] Chris Duxbury, Mark Sandler, and Mike Davies. A hybrid approach to musical note onset detection. In *Digital Audio Effects Workshop*, pages 33–38, 2002.

[85] Juan Pablo Bello, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark B Sandler. A tutorial on onset detection in music signals. *IEEE Transactions on Speech and Audio Processing*, 13(5):1035–1046, 2005.

[86] Jan Schlüter and Sebastian Böck. Musical onset detection with Convolutional Neural Networks. In *International Workshop on Machine Learning and Music (MML)*, 2013.

[87] Giovanni Costantini, Massimiliano Todisco, and Giovanni Saggio. A new method for musical onset detection in polyphonic piano music. In *International Conference on Computers - Proceedings*, volume 1, pages 545–548, 2010.

[88] Perfecto Herrera-Boyer, Anssi Klapuri, and Manuel Davy. *Automatic Classification of Pitched Musical Instrument Sounds*. Springer US, 2006.

[89] Sound source identification system for ensemble music based on template adaptation and music stream extraction. *Speech Communication*, 27(3):337–349, 1999.

[90] Realtime recognition of orchestral instruments. In *International Computer Music Conference (ICMC)*, pages 141–143, 2000.

[91] A. Eronen and A. Klapuri. Musical instrument recognition using cepstral coefficients and temporal features. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 2, pages 753–756. IEEE.

[92] Tetsuro Kitahara, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G Okuno. Instrument Identification in Polyphonic Music: Feature Weighting with Mixed Sounds, Pitch-Dependent Timbre Modeling, and Use of Musical Context. In *International Music Information Retrieval Conference (ISMIR)*, pages 558–563, 2005.

[93] Yoonchang Han, Jaehun Kim, and Kyogu Lee. Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music. *IEEE/ACM Transactions on Audio Speech and Language Processing*, 25(1):208–221, 2017.

[94] Vincent Lostanlen and Carmine-Emanuele Cella. Deep convolutional networks on the pitch spiral for music instrument recognition. In *International Society of Music Information Retrieval (ISMIR)*, 2016.

[95] H Honing. From Time to Time: The Representation of Timing and Tempo. *Computer Music Journal*, 25(3):50–61, 2001.

[96] Masataka Goto and Yoichi Muraoka. Beat tracking based on multiple-agent architecture-a real-time beat tracking system for audio signals. In *International Conference on Multiagent Systems (ICMS)*, pages 103–110, 1996.

[97] Eric D. Scheirer. Tempo and beat analysis of acoustic music. *The Journal of the Acoustical Society of America*, 103:588–601, 1998.

[98] Gouyon Fabien and Simon Dixon. A Review of Automatic Rhythm Description Systems. *Computer Music Journal*, 29(1):34–54, 2005.

[99] K. Ochiai, H. Kameoka, and S. Sagayama. Explicit beat structure modeling for non-negative matrix factorization-based multipitch analysis. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 133–136, 2012.

[100] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation Learning: A Review and New Perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828, 2013.

[101] F. Rosenblatt. The perceptron–a perceiving and recognizing automaton. *Cornell Report*, 1:85–460, 1957.

[102] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[103] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20:273–297, 1995.

[104] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.

[105] Yoav Freund and David Haussler. Unsupervised learning of distributions on binary vectors using two layer networks. In *Advances in Neural Information Processing Systems 4*, pages 912–919. Morgan-Kaufmann, 1992.

[106] Geoffrey E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[107] G. E. Hinton and R R Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, 2006.

[108] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. *Advances in Neural Information Processing Systems*, 19(1):153, 2007.

[109] Dan Claudiu Cirean, Ueli Meier, Luca Maria Gambardella, and Urgen Schmidhuber. Deep Big Simple Neural Nets Excel on Hand-written Digit Recognition. *Neural Computation*, 22, 2010.

[110] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, and Ioannis et al. Antonoglou. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[111] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016.

[112] Davide Chicco, Peter Sadowski, and Pierre Baldi. Deep autoencoder neural networks for gene ontology annotation predictions. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics - BCB '14*, pages 533–540, 2014.

[113] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[114] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145 – 151, 1999.

[115] Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016.

[116] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[117] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1):106–154, 1962.

[118] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551, 1989.

[119] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[120] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[121] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[122] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, pages 1–14, 2015.

[123] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[124] Keunwoo Choi, George Fazekas, Mark B. Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. *CoRR*, abs/1609.04243, 2016.

[125] Thomas Lidy and Alexander Schindler. Parallel convolutional neural networks for music genre and mood classification. *MIREX 2016*, 2016.

[126] Eric J Humphrey. *An exploration of deep learning in content-based music informatics*. PhD thesis, New York University, 2015.

[127] Vinod Nair and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. Number 3, pages 807–814, 2010.

[128] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical Evaluation of Rectified Activations in Convolutional Network. *CoRR*, abs/1505.00853, 2015.

[129] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *CoRR*, abs/1502.03167, 2015.

[130] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015.

[131] Anthemscore. https://www.lunaverus.com/cnn. Accessed: 2017-06-26.

[132] Mert Bay, Andreas F. Ehmann, and J. Stephen Downie. Evaluation of multiple-f0 estimation and tracking systems. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR)*, pages 315–320, 2009.

[133] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[134] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pages 69–84, 2016.

[135] Pulkit Agrawal, João Carreira, and Jitendra Malik. Learning to see by moving. *CoRR*, abs/1505.01596, 2015.

[136] James Bergstra and Yoshua Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13:281–305, 2012.

[137] Andreas Arzt, Sebastian Böck, Sebastian Flossmann, and Harald Frostel. The Complete Classical Music Companion V0. 9. In *Proceedings of the AES International Conference on Semantic Audio*, number 1, pages 133–137, 2014.

[138] Meinard Müller, Verena Konz, Michael Clausen, Sebastian Ewert, and Christia Fremerey. A Multimodal Way of Experiencing and Exploring Music. *Interdisciplinary Science Reviews*, 35(2):138–153, 2010.

[139] M.A. Casey, R. Veltkamp, M. Goto, M. Leman, C. Rhodes, and M. Slaney. Content-Based Music Information Retrieval: Current Directions and Future Challenges. volume 96, pages 668–696, 2008.

[140] Shazam. https://www.shazam.com/gb. Accessed: 2018-07-01.

[141] Emilia Gómez, Perfecto Herrera, and Xavier Serra. Chroma binary similarity and local alignment applied to cover song identification. *IEEE Transactions on Audio, Speech and Language Processing*, 16(6):1138–1151, 2008.

[142] Nicola Orio and François Déchelle. Score Following Using Spectral Analysis and Hidden Markov Models. In *International Computer Music Conference*, pages 1–1, La Havane, Cuba, 2001.

[143] C. Joder, S. Essid, and G. Richard. A conditional random field framework for robust and scalable audio-to-score matching. *IEEE Transactions on Audio, Speech and Language.*, 19(8):2385–2397, 2011.

[144] Ning Hu, R.B. Dannenberg, and George Tzanetakis. Polyphonic audio matching and alignment for music retrieval. *Signal Processing*, pages 185–188, 2003.

[145] Roger B Dannenberg. An On-Line Algorithm for Real-Time Accompaniment. In *International Computer Music Conference*, pages 193–198, 1984.

[146] Nicola Orio and Diemo Schwarz. Alignment of monophonic and polyphonic music to a score. In *International Computer Music Conference*, pages 155–158, 2001.

[147] S. Ewert, M. Muller, and P. Grosche. High resolution audio synchronization using chroma onset features. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1869–1872, 2009.

[148] Roger B Dannenberg and Ning Hu. Polyphonic Audio Matching for Score Following and Intelligent Audio Editors. In *International Computer Music Association*, pages 27–33, 2003.

[149] Gerhard Widmer, Simon Dixon, Werner Goebl, Elias Pampalk, and Asmir Tobudic. In search of the horowitz factor. *AI Mag.*, 24(3):111–130, September 2003.

[150] C.C.S. Liem and Alan Hanjalic. Expressive timing from cross-performance and audio-based alignment patterns: An extended case study. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 519–524, 2011.

[151] Gineke A. ten Holt, Marcel J. T. Reinders, and Emile A. Hendriks. Multi-dimensional dynamic time warping for gesture recognition. In *13th annual conference of the Advanced School for Computing and Imaging*, volume 119, 2007.

[152] Dynamic Time Warping. In *Information Retrieval for Music and Motion*, pages 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[153] Hiroaki Sakoe and Seibi Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, 1978.

[154] Meinard Müller and Sebastian Ewert. Chroma Toolbox: Matlab Implementations for Extracting Variants of Chroma-Based Audio Features. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 215–220, 2011.

[155] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analytics*, 11(5):561–580, 2007.

[156] Meinard Müller, Henning Mattes, and Frank Kurth. An efficient multiscale approach to audio synchronization. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 192–197, 2006.

[157] Leen De Bruyn, Marc Leman, Dirk Moelants, Michiel Demey, Sølvi. Ystad, Richard. Kronland-Martinet, Kristoffer. Jensen, and Leen De Bruyn. Computer Music Modeling and Retrieval. Genesis of Meaning in Sound and Music. *in Sound and Music*, 5493:93–106, 2009.

[158] The Mazurka Project. http://www.mazurka.org.uk/. Accessed: 2017-07-10.

[159] Jean-julien Aucouturier and Mark Sandler. Finding repeating patterns in acoustic musical signals. In *Virtual, Synthetic, and Entertainment Audio*, pages 412–421. Sony, 2002.

[160] Kagan Tumer and Joydeep Ghosh. Error Correlation and Error Reduction in Ensemble Classifiers. *Connection Science*, 8(3-4):385–404, 1996.

[161] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(3):226–239, March 1998.

[162] Chiheb Trabelsi, Olexa Bilaniuk, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J. Pal. Deep complex networks. *CoRR*, abs/1705.09792, 2017.